International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 02 Issue: 06 | Sep-2015 www.irjet.net p-ISSN: 2395-0072

OPEN AUTHENTICATION PROTOCOL USING HYBRID NETWORK IN CLOUD COMPUTING

***1Mrs. Kavitha S.K., *2 Ms. Siva Sankari A., *3 Mrs. Sangeetha Lakshmi G.,**

**1M.Phil Research Scholar, Department of Computer Science, D.K.M. College for Women (Autonomous), Vellore, TamilNadu, India.*

**2Head of the Department, Department of Computer Science, D.K.M. College for Women (Autonomous), Vellore, TamilNadu, India.*

**3Assistant Professor, Department of Computer Science, D.K.M. College for Women (Autonomous), Vellore, TamilNadu, India.*

Abstract - ***Cloud computing offers the vision of a virtually infinite pool of computing, storage and networking resources where application can be scalable and deployed. The security threats on cloud increases rapidly. The threat starts from login module to the core storage. In this project different levels of threat is handled efficiently, which enables the secured data communications between the server and client. The login is authenticated using Open Authentication Protocol (OAuth) 2.0 with enhanced security mechanism. The data theft and other masquerading attacks are prevented using channel API (Application based Program Interface) which sends the data in a secured channel establish among the sender and the receiver using MD5 (Message-Digest) Hashing. This is demonstrated by building a cloud based chat application which transfers the data from server to client and vice versa. It includes voice, non-voice and video chat communication. The Video Chat is demonstrated using WebRTC (Web Real-Time Communication).WebRTC provides page to page communication among HTML. Hence this project covers the prevention mechanism for several threats in cloud using different techniques.***

Key Words: Video Chat application, OAuthentication, WebRTC, Cloud computing, Web Engine, Security Mechanism

I. INTRODUCTION

IT infrastructures, including SOA systems, have traditionally been implemented on an actual hardware owned by the company. Virtualization alone has not shaken this tradition, because virtual servers still reside on servers back at the office. However, virtualization was a key ingredient in creating cloud computing, that is an elastic computing resource provided on-demand over the network. Cloud computing has set out to remove those dusty servers from the offices while lowering the expenses and increasing service quality. Growing interest in cloud computing has raised interest in moving existing and future IT-systems to the cloud, which means a single cloud computing environment Like many other companies, NSN has several projects with interest in reaching for the clouds. One of these projects involved a SOA cluster configuration, which is a multi node system that runs SOA technologies to run services. The goal of this project started to form around putting mat SOA cluster into Elastic Compute Cloud (EC2) [1] by Amazon. While explaining how this was done it will be necessary to unveil some of the mysteries of cloud computing.

Cloud computing offers the vision of a virtually infinite pool of computing, storage and networking resources where application can be scalable and deployed. In particular Google cloud service provides Google App Engine for Java! With App Engine, you can build web applications using standard Java technologies and run them on Google's scalable infrastructure. The Java environment provides a Java 7 JVM, a Java Servlets interface, and support for standard interfaces to the App Engine scalable data store and services, such as JDO, JPA, Java Mail, and JCache. Standards support makes developing

Your application easy and familiar, and also makes porting your application to and from your own servlet environment straightforward. The Google Plug-in for Eclipse adds new project wizards and debug configurations to your Eclipse IDE for App Engine projects. App Engine for Java makes it especially easy to develop and deploy world-class web applications using Google

Web Toolkit (GWT). The Eclipse plug-in comes bundled with the App Engine and GWT SDKs. Third-party plug-in is available for other Java IDEs as well.

The main uniqueness of this implies on its learning structure. We are enhancing the features of this application by adding more functionality. One of the functionality we are adding is the calls through browsers where browser to browser communication is achieved for video chat. Text chatting based on individual and group is also available. The goal of this thesis has two major parts. The first is to map the possibilities of cloud computing. This includes an explanation what cloud computing actually means. Also different cloud types need some clarification. After that there will be pondering on the possible benefits and drawbacks of cloud computing. The theoretical part will grasp the surface on most important cloud computing aspects, but it will focus on the issues surrounding the implementation. This knowledge is then utilized in the implementation part. The goal of the implementation part is to create a working Service Oriented Architecture (SOA) application cluster prototype and a Training environment in Amazon's EC2. The cluster configuration uses SOA Suite 1 lg backed up by Oracle 1 lg database. This cluster is configured m a way that it benefits from the features of cloud computing. The cluster configuration is a platform that allows the development and use of other SOA based enterprise systems. The training environment offers trainer an easy way to provide a high performance environment to the trainees.

II. RELATED WORK

A systematic literature review (also known as a systematic review) is a process of identifying (planning), interpreting (processing) and evaluating (analyzing) of all available research articles connected to a previously defined research question. Individual studies contributing to a systematic review are known as primary studies, what makes a systematic review as a form of secondary study. The need for a systematic review arises from the requirement of researches to gather all existing information about some topic in a thorough and unbiased manner so it has a scientific value. The reason for choosing systematic literature review as a research method for this thesis is to get an overview of the existing research in the field of Cloud Computing architecture and that there was no previous research related to this topic done by performing this research method. Our main goal was to identify, classify, and systematically compare the existing research articles focused on planning and providing cloud architecture or design. As mentioned, in the process of doing a systematic literature we aimed to answer the following main research question:

2.1 Data Plane Considerations

In Figure 1, a typical single domain service provider net- work is shown. In the figure, provider and provider edge routers are labeled as P and PE respectively. Similarly, customer edge routers are labeled in the figure as CE. In general, provider routers are MPLS switch routers and provider edge Routers are MPLS edge routers

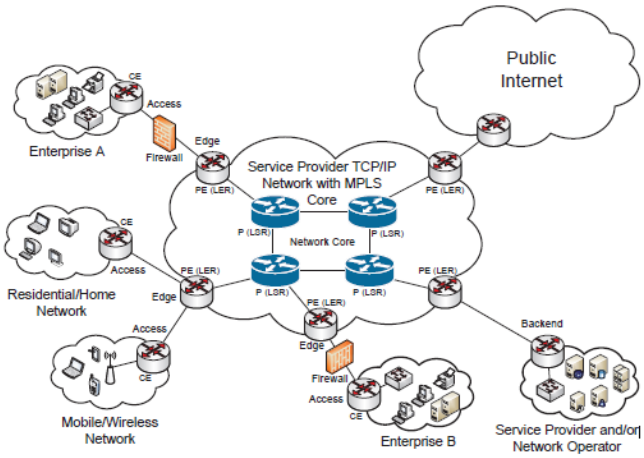


Fig1: A TYPICAL SINGLE DOMAIN SERVICE PROVIDER NETWORK.

Since MPLS is currently widely implemented in service provider networks, we propose to only replace or update the edge devices with OpenFlow-enabled network devices while keeping the network core unchanged. The latest versions of the OpenFlow protocol and their switch specifications support key MPLS operations. Our legacy network core involves proactive installation of full-mesh static LSPs, in contrast to dynamic LSPs built through signaling and routing protocols. As a result, the network edge performs all complex network operations and functions on the incoming network traffic and steers it across the simple and static legacy network core by Label switching.

In order to perform closed-loop network control and management, we propose to use sFlow at the network edge and SNMP at the network core. sFlow is chosen because most of the commercially available OpenFlow-enabled network devices, such as Open vSwitch, support sFlow. Similarity, we use SNMP because of its widespread adoption. The mentioned data plane considerations are depicted in Figure 2. This approach results in an intelligent and complex network edge that is decoupled from a simple and static legacy network core.

2.2 Control Plane Considerations

Our control plane considerations primarily aim to provision simple abstractions of the underlying network

resources, enabling network virtualization through SDN to the north-bound NaaS based network service orchestration platform as proposed in. For the complex network edge, the control plane involves

(1) An OpenFlow based SDN controller for applying dynamic and flow-level traffic control,

(2) An sFlow based network analyzer for flow-level traffic monitoring

(3) A network configuration system that configures the underlying resources. Furthermore, for NaaS based network service orchestration and closed-loop network control, the OpenFlow based SDN controller and sFlow based network analyzer must expose their northbound APIs. For the simple and static legacy network core, one can reuse the current Our Proof of Concept (PoC) design is depicted 3.

The PoC design does not involve any virtualized network functions being implemented on the underlying virtual servers. The NaaS based northbound service provisioning platform is accessed through CLI. The OpenDaylight Controller (Base Edition) is chosen as the OpenFlow based SDN controller because of its full-stack support for SDN and NFV, well documented northbound APIs and large developer community.

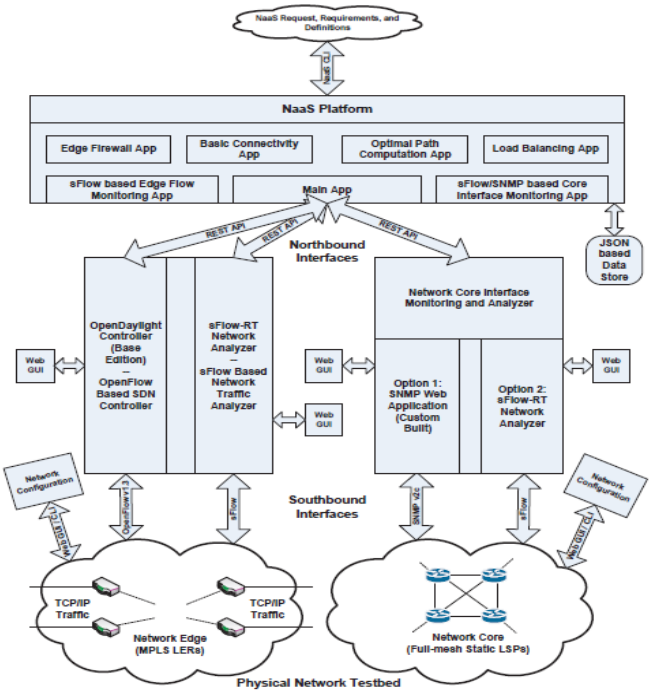


Fig 2: Proof-of-Concept design of our architecture.

FLOW DIAGRAM

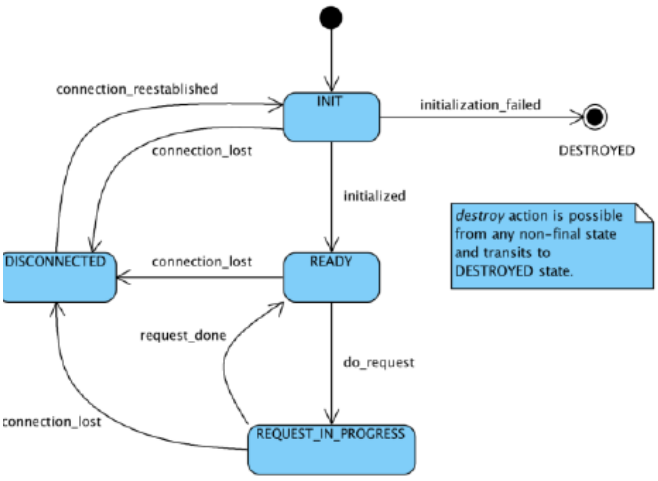


Fig 3: Data Flow Diagram

III .PROPOSED MODEL

3.1. Open Authentication

In general, this module deals with authenticating the user inside the application, this is considered to be the efficient and secured method in order to allow the user to be authenticated inside the application. An open authentication protocol is an open standard to authorization. It specifies a process for resource owners to authorize third-party access to their server resources without sharing their credentials. Designed specifically to work with Hypertext Transfer Protocol (HTTP), OAuth essentially allows access tokens to be issued to third-party clients by an authorization server, with the approval of the resource owner, or end-user. The client then uses the access token to access the protected resources hosted by the resource server.

Steps involved in OAuth:

(i). Obtain OAuth 2.0 credentials

Both Service Provider (SP) like Google and the application know OAuth 2.0 credentials such as a client ID and client secret. The set of values varies based on what type of application you are building. For example, a JavaScript application does not require a secret, but a web server application does.

(ii). Obtain an access token from the Google Authorization Server.

Before your application can access private data using a Google API, it must obtain an access token that grants access to that API. A single access token can grant varying degrees of access to multiple APIs. A variable parameter called scope controls the set of resources and operations that an access token permits. During the access-token request, your application sends one or more values in the scope parameter. There are several ways to

make this request, and they vary by type of application you are building.

For example, a JavaScript application might request an access token using a browser redirect to Google, while an application installed on a device that has no browser uses web service requests. Some requests require an authentication step where the user logs in with their Google account. After logging in, the user is asked whether they are willing to grant the permissions that your application is requesting. This process is called user consent. If the user grants the permission, the Google Authorization Server sends your application an access token (or an authorization code that your application can use to obtain an access token). If the user does not grant the permission, the server returns an error.

(iii). Send the access token to an API. After an application obtains an access token, it sends the token to a Google API in an HTTP authorization header. It is possible to send tokens as URI query-string parameters, but we don't recommend it, because URI parameters can end up in log files that are not completely secure. Also, it is good REST practice to avoid creating unnecessary URI parameter names. Access tokens are valid only for the set of operations and resources described in the scope of the token request.

(iv). Refresh the access token, if necessary.
Access tokens have limited lifetimes. If your application needs access to a Google API beyond the lifetime of a single access token, it can obtain a refresh token. A refresh token allows your application to obtain new access tokens. Final after getting the token from Google, the key is exchanged between the Google server and the web application, and then if the results match, then the user is allowed to enter in to the application.

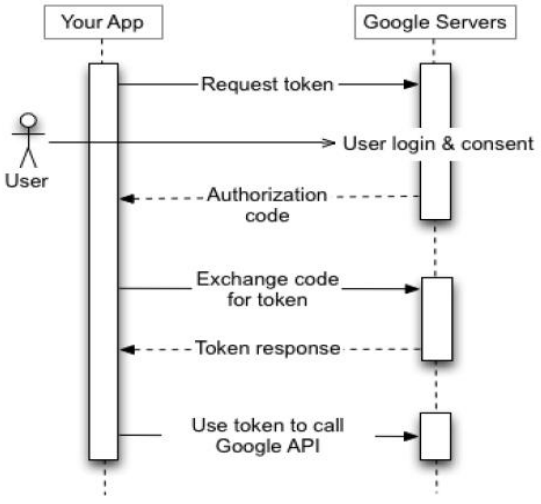


Fig 4: Open Authentication Process

2. Web Service Identification

In this module, the web services based on the functionalities required are generated. In general, each web services are intended to perform a separate operation. A web service repository is a set of disjoint services. We denote it as {w1, w2,}

3. Functionality Request

As the growth of web related requirements increases gradually, the users who interact with the web needs the system to system interaction, created the need for a structure which provided interaction not only with the user but also help application to application interaction. This problem called as Application Integration which is resolved by Web Services. Moreover, with the addition of the intelligence and autonomy of software agents, transactions may be equally automated for consumer-to-consumer, business-to-consumer, and business-to-business collaborations.

But it is a clear observation that number of web services is increased where a problem is raised to find an appropriate web service which satisfies user needs. The user wants an efficient way to find an appropriate web service which satisfied his needs in a short time. The functionality is mainly decided based on the input that is set for the web services and the output of the web services that the user needs. This is done in this module.

4. QoS Constraint

The objective of this module is to maximize an application-specific utility function under the end-to-end QoS constraints. Existing methods can be divided into two types: local selection method and global selection method. The local selection method is simple and efficient, but cannot meet the end-to-end QoS constraints. The global selection method can satisfy the global QoS constraints, but at the price of higher computational time, and it is not suitable for the dynamic environment To address this issue, an algorithm to combine global QoS constraints with local selection is proposed, which first splits the global QoS constraints into local constraints using heuristic method, and then uses local selection to find the optimal solution under the local constraints. It is intended to improve the performance by reducing the computation time greatly while achieving close-to-optimal results.

5. Operation Applicability

In this module, the web services are given an operation. After setting the operation the Web service is triggered. Then the implications evolved during the life time of the web service call are monitored. The variation is identified among the services where the operation differs one among the other as the time varies for each different web service calls.

5.1 Cross Domain Call

To make a cross domain call among web services, the Cross Origin Resource Sharing (CORS) is used which allows many resources from outside domain where the browser and the server are interact to determine whether or not to allow the cross-origin request.

Consider an example, a page from http://www.application1.com attempts to access a user's data in http://www.application2.com If the user's browser implements CORS, then request header (i.e.) Origin: http://www.application1.com sends to application2. If an application2 allows the request, then it sends Access-Control-Allow-Origin: http://www.application1.com header in its response. If it does not allow the cross origin request, then browser sends an error to application1 instead of application2 response by using the following header Access-Control-Allow-Origin: * this is applicable in the public content and it is intended to be accessible to everyone. It is related to JSONP technique for cross domain requests to make a cross domain call.

Service	Duration/Days/Hours/Minutes	Date
Facebook	1 h	Aug 10, 2011
Amazon	11 h	Apr 21, 2011
Foursquare	4 h	Aug 9,10,25, 2011
Microsoft Sidekick	6 d	Mar 13,2009
Google Gmail	30 h	Oct 16, 2008
Google Mail, Google Apps	24 h	Aug 15, 2008

Table 1.1, some of the outages in the cloud

In April, 2011, Amazon elastic compute cloud (EC2) experienced service disruption because of the incorrect network change performed few days before the outage occurred. The network change supposed to be a regular test of scalability. Amazons service was unable to read and write operations. Microsoft Sidekick experienced a massive outage [21] in 2009, which left his customers without access to their services. The data loss resulted from a system failure, this happened because of the lack of Microsoft disaster recovery policy. In 2008, Google Gmail experienced two outages. The problem was connected with availability concern. Since then, Google Apps offer a premiere edition for $50, in which customer gets 24*7 phone and email support in order to be able to access Google services at anytime (even in case of outages). Foursquare, in 2011, experienced several outages and their service was unavailable to the customers. This happened because of the lack of scalability and their servers could not manage to scale enough so they crashed. Face book also experienced an outage; customers were unable to log in. It was explained due to the site's experimental features which were being tested at the moment of the outage.

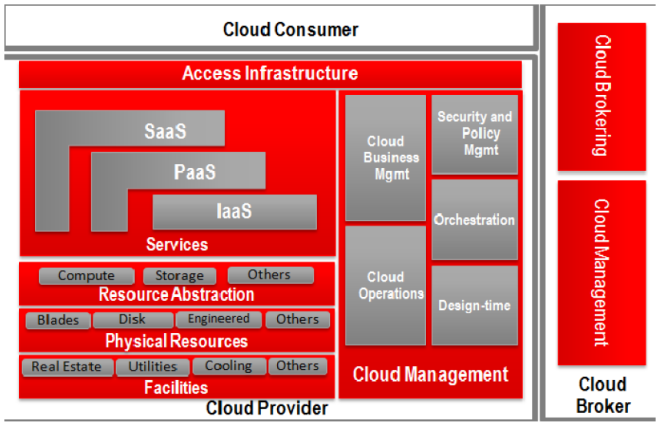


Fig5: CONCEPTUAL VIEW OF THE ARCHITECTURE

- A Cloud provider is a company or an individual that delivers cloud computing based services and solutions to consumers.
- A cloud consumer is a company or an individual that uses a cloud service provided by a
- Cloud service provider directly or through a broker.
- A cloud broker is an intercessor between cloud providers and cloud consumers.

IV. HTTP REDIRECT METHOD

The Http Redirect concept is used to make a cross domain call to third party web services. The Http Redirect property specifies the directory or URL to which a client is redirected when attempting to access a specific resource. For fast data retrieval this Http Redirect is used. In the simplest configuration, it needs only to set the enabled and destination attributes of the <http Redirect> element in order to redirect clients to a new location.

4.1 Aim of Http Redirect:

Firstly, transferring to another page using Redirect conserves server resources.

Instead of telling the browser to redirect, it simply changes the "focus" on the Web server and transfers the request. This means you don't get quite as many HTTP requests coming through, which therefore eases the pressure on your Web server and makes your applications run faster.

Secondly, Redirect maintains the original URL in the browser. This can really help streamline data entry techniques, although it may make for confusion when debugging.

4.2 Virtualization and Cloud Computing

Virtualization is a core technology for enabling cloud resource sharing. It enables abstraction of services and applications from the underlying IT infrastructure

[S20, S25, and S55]. Study [S25] gives an explanation of key cloud infrastructure evolution phases and architectural enablers for cloud data centres. The second phase [S25] is abstraction, data centre assets are abstracted from the services from which they are provided, enabled by virtualization. There are two basic approaches for enabling virtualization in the Cloud Computing environment [S14]: hardware virtualization and software virtualization. "Private clouds hold their own virtualization infrastructure where several virtual machines are hosted to provide service to their clients [S29] ". Studies [S31, S20] introduced the term server virtualization.

V. RESULT AND DISCUSSION

5.1 CHANNEL API FOR CHAT IMPLEMENTATION

The Channel API creates a persistent connection between your application and servers, allowing your application to send messages to JavaScript clients in real time without the use of polling. This is useful for applications designed to update users about new information immediately.

Using the HttpRedirect, the OAuth 2.0 is implemented to make the cross domain call and channel API sends the data in a secured channel establish among the sender and receiver using MD5(Message Digest) Hashing, apart from this the channel API based multiuser chat is implemented in order to show the efficiency of web service call made on every chat. Each channel is subjected to be a separate network connection and help to access quickly to update the information to users.

The following elements are used mainly in channel API.

i. JavaScript Client - the user interacts with a JavaScript client built into a webpage.

ii. The Server - creating a unique channel for individual JavaScript clients and send a token. Receive the update messages from clients via HTTP request. Finally, sending update messages to clients via their channels. **iii.** The client ID - The Client ID is responsible for identifying individual JavaScript clients on the server.

iv. Token - Tokens are responsible for allowing the JavaScript Client to connect and listen to the channel created for it. **v.** The channel - A channel is a one-way communication path through which the server sends updates to a specific JavaScript client identified by its Client ID. **vi.** The message - Messages are sent via HTTP requests from one client to the server. **vii.** Socket - The JavaScript client opens a socket using the token provided by the server. It uses the socket to listen for updates on the channel.**viii.** Presence - The server can register to receive a notification when a client connects to or disconnects from a channel.

The two diagrams illustrate the life of a typical example message sent via Channel API between two different clients using one possible implementation of Channel API.

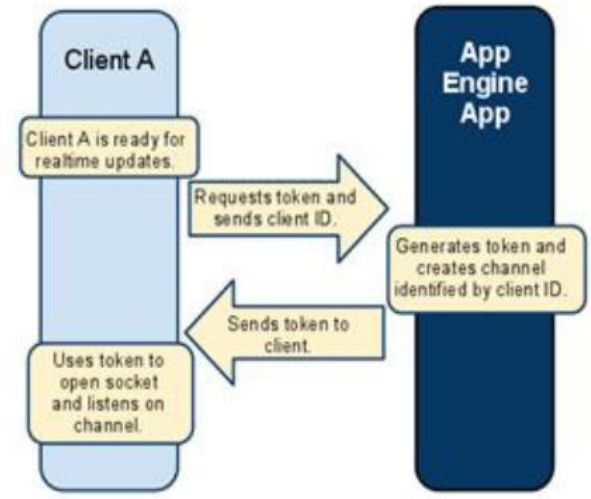


Fig 6: REQUEST TOKEN IN CHANNEL API

This diagram shows the creation of a channel on the server. In this example, it shows the JavaScript client explicitly requests a token and sends its Client ID to the server. In contrast, you could choose to design your application to inject the token into the client before the page loads in the browser, or some other implementation if preferred.

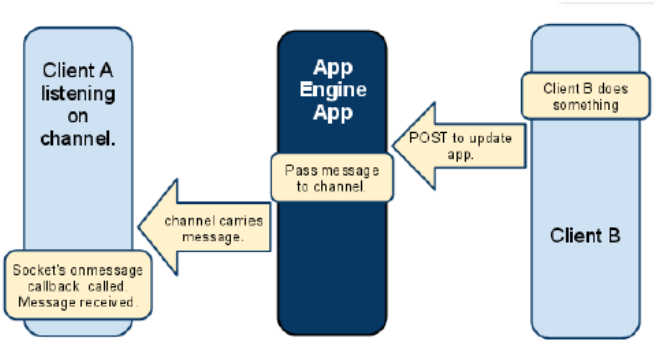


Fig 7: POSTING MSG IN CHAT API

Next, the server uses Client A's Client ID to create a channel and then sends the token for that channel back to Client A. Client A uses the token to open a socket and listen for updates on the channel.

VI. EVALUATION

Evaluation is the final phase of doing a systematic literature review in which we evaluate (analyze) articles after the full text screening and define our final selected studies. According to the results of a quality assessment and setting up the levels of maturity, we will thoroughly analyze and classify all the given and existing approaches

to the topic architecting for the cloud. It is important to follow guidelines for the each step to reduce chances for mistakes or excluding relevant materials. With the number of 240 articles after the full text screening we needed to do a quality assessment of materials to determine which article should be in the final list and which we could exclude. We took into consideration articles that had a pre-defined level of maturity which led to easier assessment of the maturity in general and future dimensions of cloud architecture.

Figure illustrates metro and backbone traffic growth based on video caching and DCs being located centrally within the metro network (case 1, metro centralized). According to the study, metro traffic will grow by a factor of 5.6 times: "Metro traffic driven by all applications will increase 560% by 2017". In comparison, "backbone traffic will grow by a factor of 3.2X" and "metro traffic will grow almost twice as fast as backbone traffic by 2017". Metro traffic includes broadcast TV, time-shifted TV, VoD, Internet video, web data; mobile video and audio, communication (video, VoIP, e-mail, immersive), file sharing, residential cloud; and DC-to-DC interconnect.

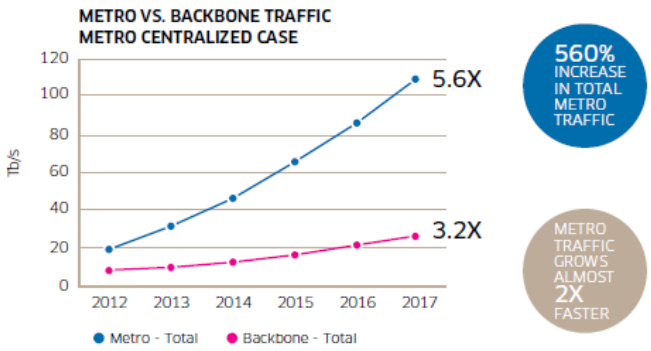


Fig 8: TRAFFIC GROWTH IN METRO AND BACKBONE NETWORK

The Bell Labs study forecasts that traffic derived from video (pay TV and Internet video) will skyrocket by as much as 720 percent. Data center (user-to-DC and DC-interconnect) traffic is forecast to increase more than 440% during the same time period. Combined, video and data center traffic are the key drivers to an overall forecast growth of 560% increase in traffic in the metro by 2017.

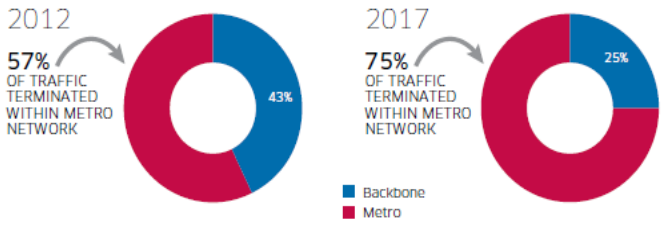
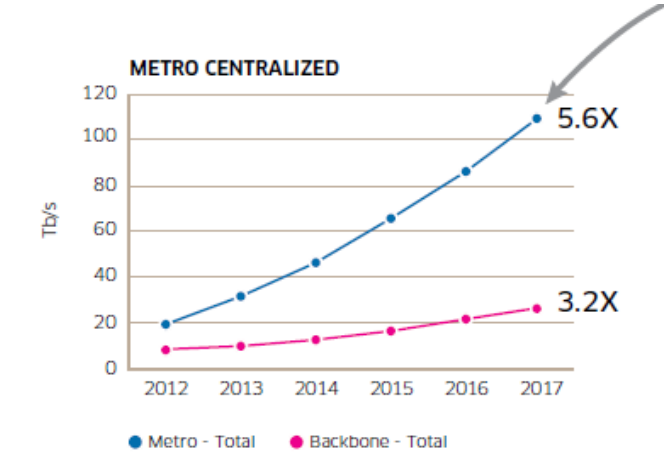


Fig 9: DUE TO THE INCREASED CONCENTRATION OF TRAFFIC SOURCES



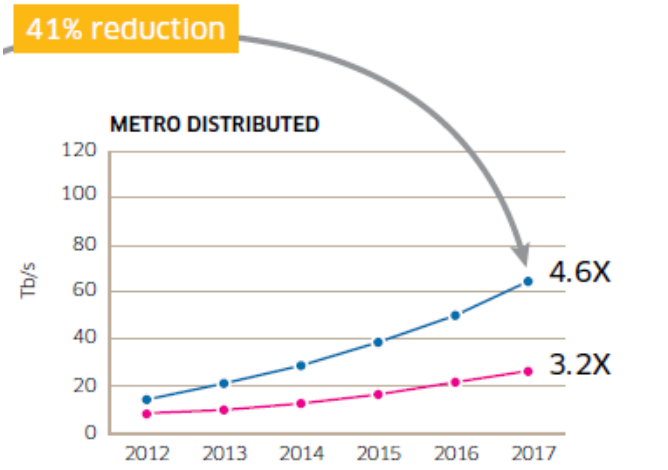


Fig10: METRO DISTRIBUTED

This master thesis was written by two students. The systematic literature review was done twice by each student separately (for reducing bias); the results of the research were compared and summed in the end. Through whole phases (both researching and writing) assigned mentor observed and helped. In the first phase of doing a systematic literature review, planning, we used the search terms 'cloud architecture' OR 'cloud architecting'. While screening the articles we realized that the term 'cloud design' is also used very often in the same manner as the previous terms. Therefore, the term of 'cloud design' was also added to the main search terms and the additional reference scanning was done. Although doing the systematic literature review twice, there might be some missed articles. Explanation for this could be that researches used different terms from 'architecting' and/or 'designing' cloud applications in their studies. Our selected list of studies consists of studies found by different search libraries (defined in sub-chapter 3.1), summed with

studies found by using snowball effect.

VII. CONCLUSION

In this paper, a new technique has been proposed to get optimized Web Service Composition. To achieve the Quality of Service, a CORS mechanism is used to make the cross domain call. Using HTTP Redirect, the fast data retrieval access can be obtained which causes high throughput and reliability. Also, to increase the efficiency of QoS, Channel API is used for multiuser chat where the web services call made on every chat also, it update the information quickly. For security reasons, Open Authentication protocol is used. Thus our whole process satisfies the non-functional properties of QoS. Computational cost and time complexity is reduced by our algorithm called approximation algorithm. In a large web service repositories the above process are applicable to obtain the optimal QoS. Moreover, the overall global QoS constraints are highly obtained.

REFERENCES

[1] Alcatel Lucent white paper, "Bell Labs Metro Network Traffic Growth: An Architecture Impact Study," 2013.
[2] NFV white paper, "Network Functions Virtualization," (http://portal.etsi.org/NFV/NFV_White_Paper.pdf), 2012.
[3] Ericsson white paper, "The Telecom Cloud Opportunity: How Telecom Operators Can Leverage Their Unique Advantages in the Emerging Cloud Market," 2012.
[4] IBM Institute for Business Value, "The Natural Fit of Cloud with Telecommunications," 2012.
[5] L. Velasco et al., "Elastic Operations in Federated Datacenters for Performance and Cost Optimization," Elsevier Computer Commun., vol. 50, 2014, pp. 142–51.
[6] L. Contreras et al., "Towards Cloud-Ready Transport Networks," IEEE Commun. Mag., vol. 50, 2012, pp. 48–55.
[7] L. Velasco et al., "Cross-Stratum Orchestration and Flex grid Optical Networks for Datacenter Federations," IEEE Network, vol. 27, 2013, pp. 23–30.
[8] M. Jinno et al., "Multiflow Optical Transponder for Efficient Multilayer Optical Networking," IEEE Commun. Mag., vol. 50, 2012, pp. 56–65.
[9] D. King and A. Farrel, "A PCE-Based Architecture for Application-Based Network Operations," RFC 7491, March 2015.
[10] ONF Solution Brief, "Open Flow-enable Transport SDN," May 2014.
[11] L. Velasco et al., "In-Operation Network Planning," IEEE Commun. Mag., vol. 52, 2014, pp. 52–60.
[12] ETSI GS NFV 001, "Network Functions Virtualization (NFV): Use Cases," V1.1.1, Oct. 2013.
[13] L. Contreras et al., "Towards Cloud-Ready Transport Networks," IEEE Commun. Mag., vol. 50, 2012, pp. 48–55.
[14] L. Velasco et al., "Cross-Stratum Orchestration and Flex grid Optical Networks for Datacenter Federations," IEEE Network, vol. 27, 2013, pp. 23–30.
[15] M. Jinno et al., "Multiflow Optical Transponder for Efficient Multilayer Optical Networking," IEEE Commun. Mag., vol. 50, 2012, pp. 56–65.
[16] D. King and A. Farrel, "A PCE-Based Architecture for Application-Based Network Operations," RFC 7491,
March 2015. [10] ONF Solution Brief, "Open Flow-enable Transport SDN," May 2014.
[17] L. Velasco et al., "In-Operation Network Planning," IEEE Commun. Mag., vol. 52, 2014, pp. 52–60.
[18] ETSI GS NFV 001, "Network Functions Virtualization (NFV): Use Cases," V1.1.1, Oct. 2013.
[13] Ll. Gifre et al., "Experimental Assessment of ABNOdriven Multicast Connectivity in Flex grid Networks," IEEE J. Lightw. Technol. (JLT), vol. 33, pp. 1-8, 2015.
[19] M. Mishra et al., "Dynamic Resource Management using Virtual Machine Migrations," IEEE Commun. Mag., vol. 50, 2012, pp. 34–40.
[20] J. Barrera, M. Ruiz, and L. Velasco, "Orchestrating Virtual Machine Migrations in Telecom Clouds," Proc. OFC, 2015.

BIOGRAPHIES

1. Mrs. Kavitha S.K., M.C.A., M.Phil Research Scholar, Department of Computer Science, D.K.M. College for Women (Autonomous), Vellore, TamilNadu, India.

2. Ms. Siva Sankari A., HOD, Department of Computer Science, D.K.M. College for Women (Autonomous), Vellore, TamilNadu, India.

3. Mrs. Sangeetha Lakshmi G., Asst. Prof Department of Computer Science, D.K.M. College for Women (Autonomous), Vellore, TamilNadu, India.