ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

Content Based Publisher/Subscriber Cloud Computing System using Event Matching System

Vinod N. Alone, Sanjay B. Waykar

Department of Computer Engineering, Sinhgad Institute of Technology, Lonavala, Savitribai Phule Pune University, India

ABSTRACT: today many social economic changes are impacting the real world scenario in the widespread applications like earthquake monitoring, natural disaster management, social net-working sites etc. This increasing demand of data dissemination in large-scale cloud-based applications where providing error free, fast and reliable system is required. The publish/subscribe (pub/sub) which is widely used for data dissemination lead to low matching throughput when we work on a large number of skewed subscription on cloud computing as well leads to many interrupt dissemination as a server may get fail in large numbers. Hence we propose SREM, a scalable and reliable event matching service for content-based pub/sub systems in cloud computing environment. To achieve low routing latency and reliable links among servers, we propose a distributed overlay SkipCloud to organize servers of SREM. A series of dynamics maintenance mechanisms are extensively studied.

KEYWORDS: Event Matching, Cloud Computing, Construction Overlay, SREM

I. INTRODUCTION

Today making real-time decisions is very important to help the users to carry out their task, and hence data dissemination has become very important and critical in the large scale emergency application like social networking, weather management and prediction, earthquake monitoring and many real-time natural disasters.

Recently, data dissemination in these emergency applications presents a number of fresh trends. One is the rapid growth of live content. For instance, Facebook users publish over 600,000 pieces of content and Twitter users send over 100,000 tweets on average per minute. In emergency scenarios, the sudden disasters like earthquake or bad weather may lead to the failure of a large number of users instantaneously. These characteristics require the data dissemination system to be scalable and reliable. Firstly, the system must be scalable to support a large amount of live content. The key is to offer a scalable event matching service to filter out irrelevant users.

The publish/subscribe (pub/sub) model is widely used for data dissemination because of its capacity of seamlessly expanding the system to massive size. However, most event matching services of existing pub/sub systems either lead to low matching throughput when matching a large number of skewed subscriptions or interrupt dissemination when a large number of servers fail. The cloud computing provides great opportunities for the requirements of complex computing and reliable communication. Hence we propose SREM, a scalable and reliable event matching service for content-based pub/sub systems in cloud computing environment. To keep the routing latency low and providing a robust link amongst all servers, we suggested a distributed dynamic overlay kip cloud to better organize the serve of SREM. To map the multiple subspace, we proposed hybrid space partitioning technique which ensures high matching throughput and certainly provides multiple candidate serves for each event. This will also lead for us to study a series of distributed and dynamic maintenance mechanisms.
Due to increasing demand of data dissemination in large-scale cloud-based software application where providing error-free, fast and reliable data we prefer to choose this topic for project implementation. Today many social economic

changes are impacting the real work scenario in the widespread applicationlike earthquake monitoring, natural disaster management, and social networking sites, one requires scalable and reliable data.

II. REVIEW OF LITERATURE

Distributed Multidimensional Matching algorithm which is a good algorithm to implement a pub/sub system over a Distributed Hash Table. In this we don't require any centralized schema knowledge. We map the pub and sub into regions in a different dimensional space. Then this multidimensional space is indexed with a distributed search tree, which allows matching multiple publish/subscribe attributes simultaneously. The multidimensional index gets the advantage by storing subscriptions at multiple nodes so that it can achieve a bottom-up publication matching that avoids root hotspots in the index. Also, it extends traditional publish/subscribe semantics; the matching algorithm highly supports publications with different level of range values. Here the search tree grows as the complexity of subscriptions increases. Two fault tolerance techniques, active failure detector, and period state refreshing allow the algorithms to recover from any type of crash.

Meghdoot implements a content-based publish/subscribe system over a peer to peer based Distributed Hash Table to improve scalability, based on CAN. P2P architecture offers the flexibility of incorporating additional resources at any time, thus providing performance scalability. Meghdoot doesn't impose any restrictions on type of subscriptions and allows them to be specified in terms of range predicates over all attributes in a schema. Meghdoot uses the semantics of the subscriptions and the events to store subscriptions and deliver matching events to them. Since real world datasets are not straight forward or we can say skewed, most of the existing systems fail to distribute load among peers. Meghdoot uses the characteristics of the load and uses it efficiently to distribute the load among various nodes. Subscription load leads to zone splitting, while event propagation load leads to zone replication. In Meghdoot subscriptions are replicated in an innovative and systematic way to handle fault tolerance. As there are always more publications than subscriptions it is not helpful to optimize the design of subscription state, which is done by Meghdoot. Scribe is channel/topic based publish/subscribe system. Scribe uses the different properties of Pastry like scalability, locality, fault-resilience and self-organization properties.

The Pastry works on a mechanism which builds an efficient multicast tree. Scribe creates groups of nodes and multicast them which balances the load among the participating nodes. Pastry's properties enable Scribe to expressively use locality to build an efficient multicast tree and to handle group join operations in a decentralized manner. As Scribe is channel based it has limited filtering capabilities. Hermes is a pub/sub system that is built over the Pastry DHT. Hermes uses special nodes which can be called as rendezvous nodes, which are set up through event type messages submitted prior to publishing. Herms use two types of components event brokers and event clients. Event brokers implement all the functionality of the Herms and event clients can be either publishers or subscribers. Event clients are light weight and connect to event brokers to use any type of service. Herms supports two types of content-based routing: In type-based routing subscribers receives all the events. In type and attribute based routing allows subscribers to further filter to the event type's attributes.

Terpstra is built over the Chord DHT, is a content-based pub/sub system. Instead of creating a single multicast tree from a root, it creates separate multicast trees rooted at each broker and because of this it is able to distribute node equally. This uses flooding of subscriptions to create a multicast tree. Terpstra is built on top of a dynamic peer-to-peer overlay network. Separate components in Terpstra ensure that the network self organizes itself to maintain the topology and can survive the simultaneous failure of up to half of its nodes. The main disadvantage of Terpstra is its flooding mechanism which may be drastic.

New communication paradigm - network coding, in which the intermediary relay stations are allowed to perform en-coding/decoding operations on the information they receive. Network coding is most effective for multicast communication. If the intermediary nodes are allowed to perform coding on the information they receive such as taking the xor of two packets or other operations within the appropriate finite field, then we have a coding strategy. Network coding may have the impact on the design of new networking and information dissemination protocols.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

III. SYSTEM DESIGN

SREM Design

To support large-scale users, we consider a cloud computing environment with a set of geographically distributed data centers through the Internet. Each datacenter contains a large number of servers (brokers), which are managed by a datacenter management service such as Amazon EC2 or Open Stack. All brokers in SREM as the front-end are low routing latency; these brokers are connected through a distributed overlay, called Skip Cloud. The entire content space is partitioned into disjoint subspaces, each of which is managed by a number of brokers. Subscriptions and events are dispatched to the subspaces that are overlapping with them through Skip Cloud. Subscriptions and events falling into the same subspace are matched on the same broker. After the matching process completes, events are broadcasted to the corresponding interested subscribers. The subscriptions generated by subscribers S1 and S2 are dispatched to broker B2 and B5, respectively. Upon receiving events from publishers, B2 and B5 will send matched events to S1 and S2, respectively.

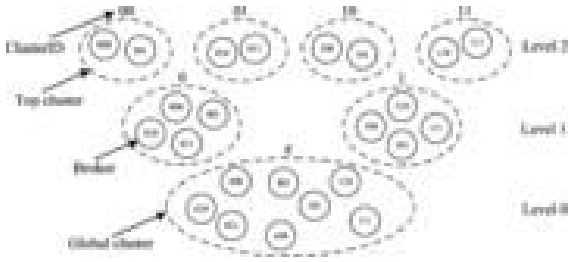

Fig 1 An example of skip cloud with eight brokers and three levels.[1]

Every broker is acknowledged by a binary string. Inefficient routing among servers. To this end, it is organized servers into Skip Cloud to reduce the routing latency in a scalable and reliable manner. Such a framework offers a number of advantages for real-time and reliable data dissemination. First, it allows the system to timely group similar subscriptions into the same broker due to the high bandwidth among brokers in the cloud computing environment, such that the local searching Time can be greatly reduced. Second, since each subspace is managed by multiple brokers, this framework is fault-tolerant even if a large number of brokers crash straightaway. Third, because the data centre management service provides scalable and elastic servers, the system can be easily expanded to Internet-scale.

SKIP CLOUD: Topology Construction

Skip Cloud systematizes all brokers into levels of clusters. The clusters at each level of Skip Cloud can be treated as a partition of the whole broker set. At the top level, brokers are systematized into numerous clusters whose topologies are complete graphs. Each cluster at this level is called top cluster. It contains a leader broker which produces a unique b-ary identifier with length of m using a hash function (e.g. MD-5). This identifier is called Cluster ID. Individually, each broker's identifier is a unique string and shares common prefix of length m with its Cluster ID. At this level, brokers in the same cluster are accountable for the same content subspaces, which provide numerous identical candidates for each event. Since brokers in the same top cluster generate frequent communication among themselves, such as updating subscriptions and dispatching events, they are organized into a complete graph to reach each other in one hop. After the top clusters have been well organized, the clusters at the rest levels can be generated level by level. Precisely, each broker decides to join which cluster at every level.

Table ISkip cloud notation

Nb	The number of brokers in skip cloud
M	The number of levels in skip cloud
Dc	The average degree in each cluster of skip cloud
Nc	The number of top cousters in skip cloud

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering
(An ISO 3297: 2007 Certified Organization)
Website: www.ijircce.com
Vol. 5, Issue 6, June 2017

IV. PROPOSED SYSTEM: INNOVATIVE STRATEGIESHPARTITION

In order to take benefit of multiple distributed brokers, SREM distributes the entire content space among the top clusters of Skip Cloud, so that each top cluster only switches a subset of the entire space and searches a small number of candidate subscriptions. SREM employs a hybrid multidimensional space partitioning technique, called HP partition, to realize scalable and reliable event matching. Generally speaking, HP partition divides the entire content space into disjoint subspaces. Subscriptions and events with overlapping subspaces are dispatched and matched on the same top cluster of Skip Cloud .To keep workload balance among servers; HP partition divides the hot spots into various cold spots in an adaptive manner

Algorithm 1. Neighbor List Maintenance

Input: views: the neighbor lists.

m: the total number of levels in Skip Cloud. Cycle: current cycle.

```
   1:    j = cycle % (m+1);
   2:    for each i in [0,m-1] do
3: Update views[i] by the peer sampling Service based on Cyclone.
   4:    for each i in [0, m-1] do
   5:    if views[i] contains empty slots then
6: fill these empty slots with other levels Items who share common prefix of length I With the
Cluster ID of views[i].
```

V. PREFIX ROUTING

Prefix routing in Skip Cloud is mainly used to efficiently route subscriptions and events to the top clusters. Note that the cluster identifiers at level i + 1 are generated by appending one b-ary to the corresponding clusters at level i. The relation of identifiers between clusters is the foundation of routing to target clusters. Briefly, when receiving a routing request to a specific cluster, a broker examines its neighbor lists of all levels and chooses the neighbor which shares the longest common prefix with the target Cluster ID as the next hop. The routing operation repeats until a broker cannot find a neighbor whose identifier is closer than itself. Algorithm 2 describes the prefix routing algorithm in Pseudo-code.
Algorithm 2. Prefix Routing

```
   1:    l =commonPrefixLength(self.ID, event.ClusterID);
   2:    if (l== m) then
   3:    process (event);
   4:    else
```

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering
(An ISO 3297: 2007 Certified Organization)
Website: www.ijircce.com
Vol. 5, Issue 6, June 2017

```
5. destB the broker whose identifier matches event.ClusterID with the longest common prefix from self.views.
6: lmax =commonPrefixLength(destB.identifier, event.ClusterID);
    7:      if (lmax ¡=l) then
8: destB the broker whose identifier is Closest to event.ClusterID from view[l].
9: if (destB and myself is in the same cluster of Level l) then
      10:  process (event);
      11:  forwardTo(destB, event);
```

USE CASE DIAGRAM

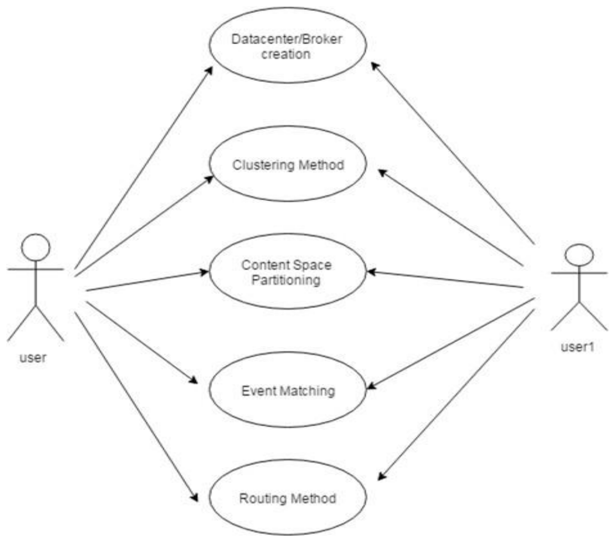

Fig 2. User case diagram

VI. SYSTEM ARCHITECTURE AND SYSTEM OVERVIEW

We will be implementing following modules in the system through the methodology.

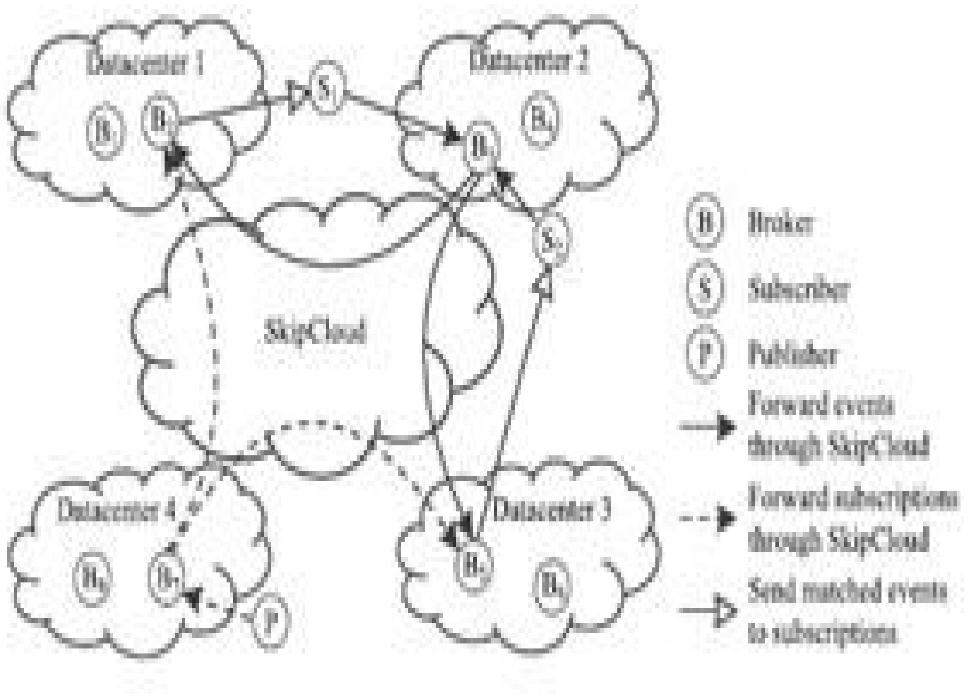
Datacenter 1
Datacenter 2
Datacenter 3
Datacenter 4
SkipCloud
B Broker
S Subscriber
P Publisher
Forward events through SkipCloud
Forward subscriptions through SkipCloud
Send matched events to subscriptions

Fig 3. System Framework [1]

are given a database of n' objects and the partitioning method constructs k' partition of data. Each partition will represent a cluster and k n. It means that it will classify the data into k groups, which satisfy the following requirements:

Each group contains at least one object

Each object must belong to exactly one group

C. Content Space Partitioning

content space is partitioned into disjoint subspaces, each of which is managed by a number of brokers. Then each top cluster only handles a subset of the entire space and searches a small number of candidate subscriptions. The whole content space into non-overlapping zones based on the number of its brokers. After that, the brokers in different cliques who are responsible for similar zones are connected by a multicast tree.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

D. Event Matching

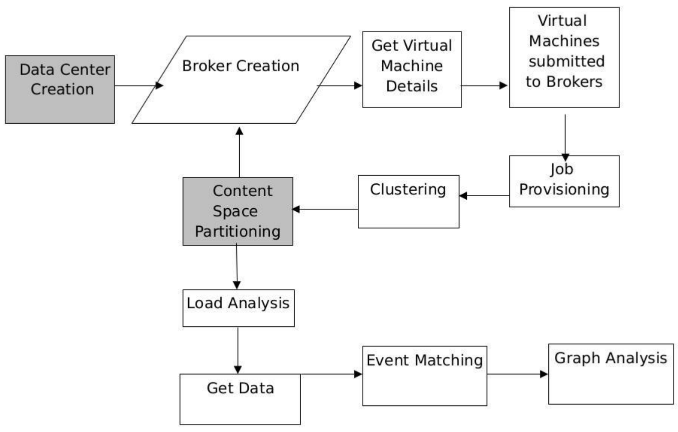

Fig 4. System Design

A. Datacenter / Broker creation

In the first module, we will focus on the Datacenter creation and Broker Creation. To support large-scale users, we consider a cloud computing environment with a set of geographically distributed data centers. Each datacenter contains a large number of servers (brokers), which are managed by a data center management service. Our approach is suitable for large and reasonably stable environments such as that of an enterprise or a data center, where reliable publication delivery is desired in spite of failures. As future work, we would like to exploit our scheme to allow for multipath load balancing and support some of P/S optimization techniques such as subscription covering. It provides an abstract and high-level interface for data producers (publishers) to publish messages and consumers (subscribers) to receive messages that match their interest.

B. Clustering Method

A cluster is a group of objects that belongs to the same class. In other words, similar objects are grouped in one cluster and dissimilar objects are grouped in another cluster. Suppose we
The data replication schemes are employed to ensure reliable event matching. For instance, it advertises subscriptions to the whole network. When receiving an event, each broker determines to forward the event to the corresponding broker according to its routing table. These approaches are inadequate to achieve scalable event matching.

E. Routing Method

The routing process usually directs forwarding on the basis of routing tables, which maintain a record of the routes to various network destinations. Thus, constructing routing tables, which are held in the router's memory, is very important for efficient routing. Most routing algorithms use only one network path at a time. Multipath routing techniques enable the use of multiple alternative paths. Prefix routing in SkipCloud is mainly used to efficiently route subscriptions and events to the top clusters. Note that the cluster identifiers at a level are generated by appending one bare to the corresponding clusters at level i. The relation of identifiers between clusters is the foundation of routing to target clusters. Briefly, when receiving a routing request to a specific cluster, a broker examines its neighbor lists of all levels and chooses the neighbor which shares the longest common prefix with the target ClusterID as the next hop. The routing operation repeats until a broker cannot find a neighbor whose identifier is closer than itself.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering
(An ISO 3297: 2007 Certified Organization)
Website: www.ijircce.com
Vol. 5, Issue 6, June 2017

IV. IMPLEMENTATION

To take advantage of the reliable links and high band-width among servers of the cloud computing environment, we Choose the Cloud Stack test bed to design and implement our prototype. To develop the prototype as modular and portable, we use ICE [13] as the fundamental communication Platform. ICE provides a communication solution that is easy to program with, and allows the developers to only focus on their application logic. Based on ICE, we add about 11,000 lines of Java code. To evaluate the performance of Skip Cloud, we implement both Skip Cloud and Chord to forward subscriptions and messages. To evaluate the performance of HPartition, The prototype supports different space partitioning policies.
Moreover, the prototype provides three different messages for-warding strategies, i.e., least subscription amount forwarding, random forwarding, and probability based forwarding

V. PARAMETERS AND METRICS

Table II Skip cloud notation

Name	Overlay	Partitioning Techniques
SREM 4	SkipCloud	HP Partition 4
SREM -1	SkipCloud	HPartition -1 (SPartition, BlueDove)
Chord-4	Chord	HPartition-4
Chord-2	Chord	HPartition-2
Chord-1	Chord HPartition-1	(SPartition, BlueDove)

We use a group of virtual machines (VMs) in the Cloud Stack test bed to evaluate the performance of SREM. Each VM is running in a exclusive physical machine, and we use 64 VMs as brokers. For each VM, it is equipped with four processor cores, 8 GB memory, and is connected to Gigabit Ethernet switches. In Skip Cloud, the number of brokers in each top cluster d is set to 2. To ensure reliable connectivity of clusters, the Average degree of brokers in each cluster Dc is 6. In HPartition, the entire subscription space consists of eight dimensions, each of which has a range from 0 to 500. The range of each dimension is cut into 10 segments by default. For each hotspot, the range of its every dimension is cut into two segments iteratively.

In most experiments, 40,000 subscriptions are generated and dispatched to their corresponding brokers. The ranges of predicates of subscriptions follow a normal distribution with standard deviation of 50, represented by sub. Besides, there are two million events generated by all brokers. For the events, the value of each pair follows a uniform distribution along the entire range of the corresponding dimension. A subspace is labeled as a hotspot if its subscription amount is over 1,000. Besides, the size of basic subscription Subset N0 in Section 4.4.2 is set to 500, and the threshold value an in Section 4.4.3 is set to 0.7.

Recall that HPartition divides all dimensions into t indi-vidual spaces Vi, each of which contains ki dimensions. We set ki to be 1, 2, and 4, respectively. These three partitioning policies are called HPartition-1, HPartition-2, and HPartition-4, respectively. Note that HPartition-1 represents the Single-dimensional partitioning which is adopted by Blue Dove. The details of implemented methods are shown in Table 1, where SREM-ki and Chord-ki represent HPartition-ki under Skip Cloud and Chord, respectively. In the following experiments, We evaluate the performance of Skip Cloud through comparing SREM-ki with Chord-ki, and evaluate the performance of HPartition through comparing HPartition-4 and HPartition-2 with the single dimensional partitioning in Blue-Dove, i.e., HPartition-1. Besides, we do not use Partition in the experiments, which is Mainly because its fine-grained partitioning technique leads to extremely high memory cost.
We evaluate the performance of SREM through a number of metrics.

1. Subscription searching size: The number of subscriptions that need to be searched on each broker for matching a message.

2. Matching rate: The number of matched events per second. Suppose the first event is matched at the moment of T1, and the last one is matched at the Moment of T2. Thus, the matching rate is Ne /T2-T1.

3. Event loss rate: The percentage of lost events in a specified time period.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have presented broker-less approach in the content-based publish-subscribe system for providing authentication and confidentiality. The approach is extremely good for a number of subscribers and publishers in the system and the number of keys maintained by them. The keys will be in cipher-text format which is labeled with credentials assigned to publishers and subscribers.

This project introduces SREM, a scalable and reliable event matching service for content-based pub/sub systems in cloud computing environment. Scalable Routing and event matching system connects the brokers through a distributed overlay Skip- Cloud, which ensures reliable connectivity among brokers through its multi-level clusters and brings a low routing latency through a prefix routing algorithm.

Through a hybrid multi-dimensional space partitioning technique, SREM reaches scalable and balanced clustering of high dimensional skewed subscriptions, and each and every event is allowed to be matched on any of its matching candidate servers. Extensive experiments with real deployment based on a Cloud-stack test bed are conducted, producing results which demonstrate that Scalable Routing and Event Matching is effective and practical, and also presents good workload balance, scalability, and reliability under various parameter settings.

VII. ACKNOWLEDGEMENT

The author Mr. Vinod Alone would like to say thank to Prof. Sanjay B. Waykar, from Department of Computer Engineering, Sinhgad Institute of Technology for his valuable inputs and support for taking this topic for survey and implementation. He has been a great motivation for me and would like to share my acknowledgment to him for presenting this paper.

REFERENCES

[1] Xingkong Ma, Student Member, IEEE, Yijie Wang, Member, IEEE, and XiaoqiangPei ”A Scalable and Reliable Matching Service for Content-Based Publish/Subscribe Systems”IEEE TRANSACTIONS ON CLOUD COMPUTING, VOL. 3, NO. 1,JANUARY-MARCH 2015
[2] A. Carzaniga, Architectures for an event notification service scalable to wide-area networks, Ph.D. dissertation, IngegneriaInformatica e Automatica, Politecnico di Milano, Milan, Italy, 1998.
[3] P. Eugster and J. Stephen, Universal cross-cloud communication, IEEE Trans. Cloud Comput., vol. 2, no. 2, pp. 103116, 2014.
[4] Y. Zhao and J. Wu, Building a reliable and high-performance content-based publish/subscribe system, J. Parallel Distrib. Comput., vol. 73, no. 4, pp. 371382, 2013.
[5] A. Gupta, O. D. Sahin, D. Agrawal, and A. El Abbadi, Megh-doot: Content-based publish/subscribe over p2p networks, in Proc. 5th ACM/IFIP/USENIX Int. Conf. Middleware, 2004, pp. 254273.
[6] W. Rao, L. Chen, P. Hui, and S. Tarkoma, Move: A large scale keyword-based content filtering and dissemination system, in Proc. IEEE 32nd Int. Conf. Distrib. Comput. Syst., 2012, pp. 445454.
[7] X. Ma, Y. Wang, Q. Qiu, W. Sun, and X. Pei, Scalable and elastic event matching for attribute-based publish/subscribe systems, Future Gener. Comput. Syst., vol. 36, pp. 102119, 2013.
[8] Muhammad Adnan Tariq, Boris Koldehofe, and Kurt Rothermel ”Secur-ing broker-less publish/subscribe systems using identity-based encryp-tion”IEEE Transactions On Parallel And Distributed Systems, Vol. 25, No. 2, February 2015
[9] M. Ion, G. Russello, and B. Crispo, Supporting Publication and Subscrip-tion Confidentiality in Pub/Sub-Networks,” Proc. Sixth Int’l ICST Conf. Security and Privacy in Comm. Networks (SecureComm), 2010
[10] L.I.W. Pesonen, D.M. Eyers, and J. Bacon, ”Encryption-Enforced Ac-cess control in Dynamic Multidomain publish/subscribe network”, Proc. ACM Int’l Conf. Distributed Event-Based Systems (DEBS), 2007
[11] P. Pietzuch, ”Hermes: A Scalable Event-Based Middleware,” Ph.D. dissertation, Univ. of Cambridge, Feb. 2004.
[12] M. Srivatsa, L. Liu, and A. Iyengar, EventGuard: A System Architec-ture for Securing Publish-Subscribe Networks, ACM Trans. Computer Systems, vol. 29, article 10,2011.
[13]roc. (2014). [Online]. Available: (http://www.zeroc.com/)