Time-Critical Event Dissemination in Geographically Distributed Clouds

Chi-Jen Wu*†, Jan-Ming Ho† and Ming-Syan Chen*†‡
*Department of Electrical Engineering, National Taiwan University, Taiwan
†Institute of Information Science, Academia Sinica, Taiwan
‡Research Center for Information Technology Innovation, Academia Sinica, Taiwan
{cjwu,hoho}@iis.sinica.edu.tw, mschen@citi.sinica.edu.tw

***Abstract*—Cloud computing has rapidly become a new infrastructure for organizations to reduce their capital cost in IT investment and to develop planetary-scale distributed applications. One of the fundamental challenges in geographically distributed clouds is to provide efficient algorithms for supporting intercloud data management and dissemination. In this paper, we present Plume, a generic distributed intercloud overlay for time-critical event dissemination services. Plume aims at improving the interoperability of interclouds in time-critical event dissemination services, such as computing policy updating, message sharing, event notifications and so forth. Plume organizes these distributed clouds into a novel quorum ring overlay to support a constant event dissemination latency. Our numerical results show that the proposed Plume greatly improves the efficiency as compared to a DHT-based overlay approach and provides better scalability than the fully-meshed approach.**

I. Introduction

In the last few years, cloud computing has rapidly become a new infrastructure for organizations to reduce their capital cost in IT investment and to develop planetary-scale distributed applications over the Internet. Cloud service providers, exemplified by Amazon Simple Storage Service (S3), Microsofts Azure Service Platform, and Google App Engine, have to scatter geographically as many datacenters as possible to support a great number of users spread around the world. Because of the demands, a growing number of vendors are introducing datacenter containers [1] to support modular datacenters that can provide predictable, repeatable components for the desired organizations. Moreover, IT organizations/enterprises, government/departments, universities may deploy their own clouds in the future [2], [3]. We can imagine that, in the future, a lot of distributed clouds, including public and private clouds, will soon be built around the globe.

One of the fundamental challenges in large-scale geographically distributed clouds is to provide efficient algorithms for supporting intercloud [4] data management and spread [5]. In other words, it is that intercloud applications/systems have to interact with a set of clouds via the Internet. Especially for time-critical services across a large geographic area, these service providers prefer to delivery the events as early as possible. Therefore, as the number of clouds continues to grow, the scale of distributed clouds may achieve a planetary-scale, with even thousand or hundred thousands of clouds. One of major challenges in these planetary-scale distributed clouds is to promptly disseminate the time-critical event to other clouds.

A typical example for such time-critical services considers disaster early warning systems [6], [7]. Assume that each country owns a disaster alert cloud for hosting the disaster early warning system and these disaster alert clouds are connected with the Internet. This disaster alert cloud is a private cloud built by the weather bureau of a country. The disaster early warning system allows end users to monitor a large disaster, such as earthquakes, hurricanes, tsunamis etc., which influences many countries. In such a system, the disaster measurement data is collected by a lot of deployed sensors, and is gathered into the disaster alert cloud of a country for representing a summary to disaster observers. However, if the monitoring disaster may cause damages to other countries, an alert notification should be rapidly disseminated to clouds across other countries to minimize the damage.

Another example is that for financial computing services or social networking services, such as SalesForce, Facebook and Twitter, may spread many geographically distributed datacenters (clouds) or rent an amount of datacenters across a large geographic area. When a user updates his/her activity with friends, the update event should quickly reflect to his/her friends who may be served on other clouds. Those business/social networking services need an efficient way to disseminate the update events among geographically distributed clouds to maintain the quality of user experiences. Thus how to share and to disseminate information among geographically distributed datacenters is critical for providing time-sensitive services in a distributed cloud computing environment.

To manage and deliver the content well, these distributed clouds have to organize themselves into a federated intercloud overlay network [8]–[11]. Currently, each large-scale cloud service provider uses its own approach to manage or deliver the content. For instance, Facebook aggregates the server log data using a tree-based overlay [12], and Amazon's Dynamo leverages a fully-meshed topology for managing data across multiple datacenters [5]. However, to meet the demand of time-critical services, this overlay requires a fundamental property, which is that scalability, constant dissemination delay and robustness in the intercloud overlay should be considered cautiously. Many existing overlay techniques can support the event dissemination in distributed systems. The fully-meshed design is the simplest way to disseminate time-critical messages to

978-1-4244-9920-5/11/$26.00 ©2011 Crown

each other nodes. This scheme takes constant transmission delay easily. However, each node in the fully-meshed design is required to maintain O(n) connections with other nodes in a distributed system of n nodes. Thus this scheme does not scale well in large-scale distributed systems.

Alternative approach is the tree-based design. Many tree-based approaches have been proposed with scalability as the main design criterion in the peer-to-peer paradigm [13]–[15]. However, most of their conception don't aim to provide a time-critical event dissemination, instead of they focus on efficient event dissemination as the communication cost incurred by delivering event among the participated nodes. Based on tree-based design, it needs O($log\ n$) hops to delivery events to all nodes, where n is the number of participated nodes. Another possible way to efficiently interconnect distributed clouds is based on Distributed Hash Tables (DHT) overlay. DHT, such as Chord [16], is inherently scalable and efficient at the Internet scale. However, the most of DHT also require at most O($log\ n$) delivery steps [14]. Gossip-based event dissemination protocols [15] also achieved scalable and fault-tolerant, however the most of them usually come with long delay in transporting messages. On the other hand, time-critical services such as those that processing financial market data or disseminating the diaster events, have tight delivery delay demands, thus these tree-based, DHT-based or Gossip-based approaches may not meet the new challenge of intercloud environments in a timely manner.

In this paper, our main focus is this problem: *how quickly updated event will be available at every clouds in a large-scale distributed clouds environment*, it is to improve the inter-operability of interclouds in computing policy updating, message sharing, event notifications and so forth. Specifically, we present Plume, a generic distributed intercloud overlay for event dissemination, that aims at forming an overlay topology for supporting time-critical event dissemination in a planetary-scale self-configuring overlay of clouds connected via the Internet. The cloud nodes in Plume are organized into a novel quorum ring overlay that is similar to the Chord ring [16]. However, the finger tables of nodes are constructed in a very different way: each cloud node independently carries out the finger table based on the concept of an overlay ring and the quorum system [17]. This eliminate the need for complex distributed overlay tree construction protocols, yet it results an efficient one-hop overlay and allows the overlay to delivery events without long event propagation delay. In addition, for scalability reasons, the size of finger table maintained by each cloud node to others is O($\sqrt{n}$). Extensive simulations are conducted to evaluate the event transmission efficient of Plume and a DHT-based design. Our numerical results show that the proposed Plume greatly improve the efficiency as compared to a DHT-based overlay approach and provide better scalability than the fully-meshed approach.

In summary, this paper makes the following contributions: 1) We present a distributed intercloud protocol, called Plume, based on a novel quorum ring to construct a low complexity cloud-level overlay for time-critical event dissemination. Also Plume is simple and easy to implement. 2) To the best of our knowledge, this paper provides the first study to address time-critical event dissemination in intercloud environments. Moreover, event dissemination provides the essential functionality for cloud computing applications that require collaboration with geographically distributed clouds. 3) We analyze the performance of Plume and different designs of distributed architectures, and also evaluate them empirically to show the performance advantages of Plume.

The rest of this paper is organized as follows. In the next section, we describe the background and related work. In Section III, we present the detailed design of the proposed Plume. Performance analysis of Plume and other schemes are presented in Section 4. In Section IV, we present performance results on Plume and a DHT-based scheme. In Sections 6 and V, we discuss the further consideration and conclude this paper.

II. Related Work

Intercloud [4] is one of the latest research fields in cloud computing research community. The one of important functionalities of intercloud is that if a cloud is out of its resources, then it could not satisfy further requests from its users. However, intercloud may bring new business niches among cloud service providers if these clouds can inter-operate each other for sharing resources. Thus the demand of creating inter-operability amongst these clouds will be occurred soon in the future Internet. Intercloud is still in its infancy. A few related results in the intercloud research area have been recently proposed by cloud computing community. Therefore, as interest in intercloud grows, so will the need for developing technologies and tools as its building blocks.

Most research on interclod communication is still in its early stage: focusing on the standardization of communication protocols, dynamic computing workload migration, etc, and there is no initial empirical results [18]. In 2010, Vint Cerf, one of pioneers of the Internet, pointed out that the need for inter-operability between different clouds will emerge in the future [8]. Cisco researchers, Bernstein *et al*. [9], discussed research issues around inter-cloud protocols and presented an architecture for intercloud networks. They also indicated that presence and messaging mechanisms of intercloud should be designed. Buyya *et al*. [10] also presented research challenges of distributed clouds, and designed a federated cloud computing environment for dynamically coordinating computing load among distributed clouds. Cross-Cloud Federation Manager (CCFM) [11] is another resources management design of federated clouds.

The ADAMANT project [19] is a pub/sub middleware that uses supervised machine learning to autonomously configure available resources among distributed clouds. It aims at designing a cloud middleware for supporting on-demand requests of cloud resources as a part of a disaster management system. Aneka-Federation [20] is a fully decentralized cloud overlay based on DHT. It is used to coordinate application scheduling and to federate resources from multiple clouds. Volley [21]

presents a data replication mechanism for reducing the inter-datacenters traffic and improve user latency, it analyzes user logs to output data migration recommendations back to the managers of datacenters. Laoutaris *et al.* [3] presented the notion of nano datacenters, which are deployed at the edge of the networks, such as set-top-boxes. They argue that the novel nano datacenters are better suited for new emerging applications, e.g., online gaming, interactive IPTV, etc. However, this conceptual design requires an efficient way to inter-operate these distributed nano datacenters.

Compared with the above efforts, we focus on the design of intercloud network protocols and intercloud overlay architecture for time-critical event dissemination as a basic building blocks of cloud computing for many-to-many messages sharing. We tackle the scalability and efficiency problems of intercloud overlay, by organizing distributed clouds into a low complexity quorum ring overlay to reduce the transmitted latency of intercloud communications.

III. Design of Plume

We propose a new intercloud overlay design, refers to Plume intercloud, in distributed clouds computing environments. The resulting overlay is then used for efficient and scalable many to many event dissemination. In this section, we begin with an introduction on the overview of Plume intercloud and its assumptions. Then, we present detail design of quorum ring overlay on which Plume is built, then present how Plume intercloud disseminate events among distributed cloud environments.

A. Overview of Plume Intercloud and its Assumptions

We consider an intercloud network with a set of geographically distributed clouds connected by the Internet, and each cloud is composed of the following components: a cloud manager, applications/services and a Plume node. We illustrate a simple overview of a Plume intercloud in Figure 1. As shown in Figure 1, these distributed clouds are organized into an intercloud overlay via Plume nodes. In each cloud, there is a physical data center management service for cloud platform vendors, such as Amazon EC2, VMware or IBM Tivoli. An application or service refers to a user level application or a running service, e.g., a virtual cloud management service [1], Facebook and Twitter hosted by multiple clouds in geographically dispersed locations.

Hence, for the scenario in Figure 1, if the user changes the cloud computing policy of the cloud manager, then the Plume node acts as a middleware service that disseminates the new policy to the rest of Plume nodes of clouds through the Plume intercloud. In other words, a Plume node preforms a connectivity gateway for a cloud utilizing control and data plane mechanisms to coordinate other distributed clouds. Moreover, a newly cloud can be deployed or join into a Plume intercloud, the new deploying cloud needs to initiate a Plume node for communication with other existing clouds.

[1]As well as CloudStatus [22] or CloudSwitch [23], it is designed as a hosted service for monitoring and managing user's applications or services.

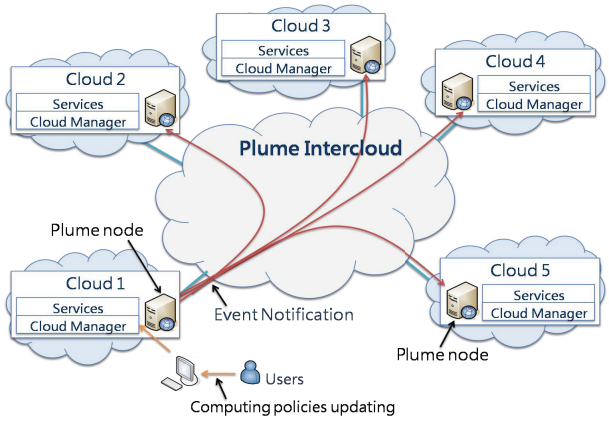

Fig. 1. An overview of a Plume intercloud network.

In this work, we assume that communications in a Plume intercloud are treated as an event dissemination. We explore the one-to-many event dissemination scenario, and it easily extends Plume to support many to many event dissemination. Formally, we assume that an intercloud network is an undirected graph $G = (V, E)$ with nodes V corresponding to Plume nodes and edges E corresponding to unidirectional overlay links. Let $u, v \in V$, an edge $e(u, v) \in E$ indicates that u and v know each other and can communicate directly through the intercloud overlay. In addition, we assume that distributed clouds are uniquely identified. In what follows, we describe our Plume intercloud in detailed descriptions.

B. Plume Intercloud

A Plume intercloud organizes the Plume nodes of these distributed clouds into an intercloud overlay based on the Quorum Ring Overlay (QRO) topology. We will describe it later. Plume intercloud ensures that any Plume node only needs at most 2 hops to reach any other Plume nodes in the intercloud network. Thus it can offer a constant dissemination latency for time-critical event delivery in large-scale distributed cloud computing environments. Specifically, a Plume intercloud as a middleware provides an interface for the cloud services or applications to share event among these distributed clouds, which consists of the basic primitive function calls: 1) $Disseminate(e)$ operation to disseminate an event e to other clouds, 2) $Join()$ and $Leave()$ operation for a Plume node to join or depart a Plume intercloud, 3) $Relay(e)$ operation that is invoked when a Plume node receives an event e and 4) a simple $Stabilize()$ process that runs periodically at each Plume node and maintains the QRO topology.

The QRO topology of Plume intercloud exploits the simple concepts of an overlay ring [16] and the grid quorum system [17] to make Plume intercloud scalable and efficient in event dissemination. As well as Chord ring, the overlay ring topology in QRO is easily imposed on it by defining the $Successor(i)$ and the $Predecessor(i)$ operations to maintain the ring edges of a Plume node i. The two operations return both successor and predecessor of the position for a Plume

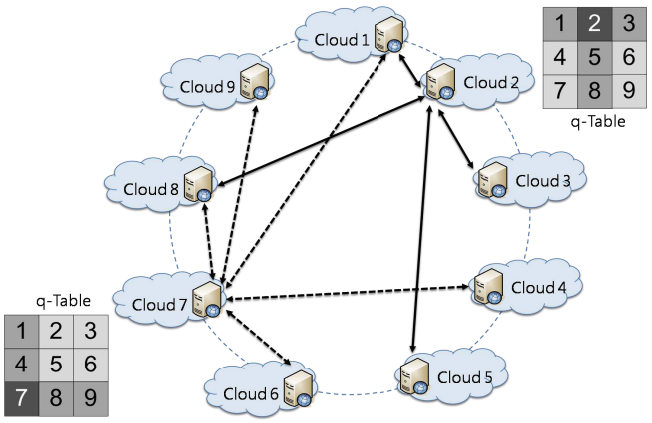

Fig. 2. An example of Quorum Ring Overlay of 9 distributed clouds.

node i maintaining the adjacency relations in the overlay ring. However, the event dissemination scheme that uses the simple ring overlay requires at last $\frac{n}{2}$ hops for delivering events to an intercloud network of size n.

To accelerate the event dissemination time, each Plume node maintains a set of Plume nodes, refers to *q-table*, to expedite the event dissemination operations. The *q-table* is constructed based on the grid quorum system. A formal definition of a grid quorum system is as follows.

DEFINITION: A grid quorum system Q is an universal set of logical points $U = \{1, 2, \ldots, n\}$ that is arranged as a $\sqrt{n} \times \sqrt{n}$ array. A subset $q_i \subseteq Q$ is defined as the set of points on the row and the column passing through logical point i in the array. Thus, it is clear that $q_i \cap q_j \neq null$, for any point i and j, $1 \leq i, j \leq n$. Based on the definition, for any logical point i, the size of q_i is $2\sqrt{n} - 1$ in a grid quorum system.

For an instance, in Figure 2, in which the grid quorum Q of size $\sqrt{9} \times \sqrt{9}$ is filled with number 1,...,9. The q_2 is {1,3,5,8}, and q_7 is {1,4,8,9}. It is clear that $q_2 \cap q_7 \neq null$.

Before a Plume node joins the Plume intercloud, it has to get an identifier in the grid quorum by manually configure or by centralized assignment. In a $\sqrt{n} \times \sqrt{n}$ grid quorum system, a Plume node with a grid identifier can easily pick one column and one row of entries and these entries will become its *q-table* in a Plume intercloud overlay. More precisely, at Plume node i, its *q-table* consists of a row_i and a col_i entries. Then this Plume node can obtain its ring edges and *q-table* by asking any existing Plume nodes in the Plume intercloud. Thus, the Plume nodes can construct the QRO topology according to their ring edges and *q-table*s respectively. Note that the size of ring edges plus *q-table* in some Plume nodes is $2\sqrt{n}$, e.g., when a Plume node with the grid point $i \equiv 0 \pmod{\sqrt{n}}$ or $i \equiv 1 \pmod{\sqrt{n}}$. As an example, suppose we have nine distributed clouds as in Figure 2. The Plume node in Cloud 2 has *q-table* {1,3,5,8} and its ring edges {1,3}. Plume node in Cloud 7 has *q-table* {1,4,8,9}. In addition, there are ring edges {6,8} in Plume node 7.

We now show that the construction of the quorum ring overlay using the grid quorum results in each Plume node can reach any Plume node at most two hops.

LEMMA 1: *In a stable Plume intercloud network of n nodes, each Plume node can reach any other Plume nodes in a two hops route.*

Proof: In a grid quorum system of $\sqrt{n} \times \sqrt{n}$ points, for each grid point i has a quorum set, q_i that contains a full column and a full row of elements in the array. By the definition of a grid quorum system, in any grid quorum system, $q_i \cap q_j \neq null$ for any grid point i and j, $1 \leq i, j \leq n$. Let k be the some grid points in $q_i \cap q_j$. Thus, for any grid point, $j \notin q_i$, the grid point i can reach j by passing one of grid points, $l \in k$. Then it is clear that any Plume node i in a quorum ring overlay can reach any other plume nodes in a two hops route. ■

Next we discuss how a Plume node maintain its *q-table*. In a Plume intercloud, every Plume node initiates a $Stabilize()$ procedure to maintain the disconnected Plume nodes in its *q-table*. The presences of Plume nodes in *q-table* can easily be aware by exchanging heartbeat messages with ring edges and Plume nodes in its *q-table* periodically. If failure Plume node $j \in$ the column of Plume node i's *q-table* then the Plume node i can replace it by any active Plume nodes in the row_j of the failure Plume node j. For example, in Figure 2, if the cloud 5 is disconnected with the Plume intercloud then cloud 2 will replace it by cloud 4 to maintain the overlay connectivity. Or if the cloud 1 is disconnected with the Plume intercloud then cloud 7 will replace it by cloud 2.

LEMMA 2: *If each Plume node in the Plume intercloud retains the correct ring edges, i.e., successor and predecessor, then the q-table of Plume nodes can be recovered from any Plume node failure after a stabilization time.*

Proof: In a stable Plume intercloud network of n nodes, the $Stabilize()$ procedure starts at a Plume node p by stabilizing its ring edges, i.e, successor and predecessor which have already the correct entries for any Plume node. After another run of stabilization, the successor of p's successor becomes correct, and so on. After at most $\sqrt{n}$ stabilization runs, Plume node p can correct all of row entries in its *q-table*. After stabilizing row entries, each Plume node has correct row entries. And the remained col entries can be retrieved by asking these stabilized row entries. ■

C. Event Dissemination Mechanism

Minimizing the event dissemination time is important to a time-critical event service, we contend that. We develop a quick event dissemination algorithm, in which time-critical events are disseminated to all Plume nodes via the *q-table* edges of the QRO topology. Based on the QRO topology, Plume intercloud can typically provides swift event dissemination in large-scale distributed clouds. For simplicity, we assume that the QRO is with a perfect grid quorum, although the algorithm can tolerant node failure. The event dissemination algorithm can be invoked by any Plume node to initiate a disseminating operation among its *q-table* edges. The event dissemination algorithm is represented as follows. In algorithm 1, we assume that a Plume node i requires to

Algorithm 1 Plume $Disseminate(e)$ Algorithm

```
1: /* Plume node s is invoked to disseminate an event e */
2: for all Plume node i ∈ s's q-table do
3:     e.ttl ← 2
4:     Send(e, i)
5: end for
```

Algorithm 2 Plume $Relay(e)$ Algorithm

```
1: /* Plume node r receives an event e from s */
2: if e.ttl -1 ≤ 0 then
3:     return
4: end if
5: if Plume node r ∈ col_s then
6:     for all Plume node i ∈ row_r do
7:         e.ttl ← 1
8:         Send(e, i)
9:     end for
10: else if Plume node r ∉ col_s then
11:     /* the event dissemination operation is done */
12:     return
13: end if
```

disseminate events to every other nodes in a Plume intercloud. And algorithm 2 presents the $Relay(e)$ operation in a Plume node r that receives an event e from s.

As previously mentioned, the event dissemination algorithm of Plume intercloud does not requires a complex dissemination algorithm, the simple event dissemination technique can exhibit much improve event dissemination time based on the low diameter property of QRO topology. For an instance, in Figure 2, the Plume node s of Cloud 2 is invoked to disseminate an event to other clouds in the Plume intercloud. Thus, these Plume nodes of Cloud 1, 3, 5 and 8 receive the event from s directly. And the Plume nodes of Cloud 5 and 8 $\in col_2$ will relay the event to Cloud 4, 6 $\in row_5$ and Cloud 7, 9 $\in row_8$ respectively.

In discussing the average dissemination delay in Plume intercloud, we simplified the transmission delay as hop counts, i.e., one signal hop costs a delay of 1.

LEMMA 3: *In a stable Plume intercloud network of n nodes, the average dissemination delay caused by the Plume intercloud is equal to* $\frac{2(n-\sqrt{n})}{n-1}$.

Proof: In a stable intercloud network of n nodes, the originator Plume node disseminates the events to its *q-table* with one hop, and these Plume nodes in its *q-table* should relay the events with another hop. Therefore, given any Plume node in the Plume intercloud, the size of *q-table* Plume nodes receive the event with a delay of 1. The size of *q-table* is $2\sqrt{n}-1$. And there is a total of $(n-2\sqrt{n})$ Plume nodes receive the event with a delay of 2. Thus, the total dissemination delay in Plume intercloud is approximately equal to $1 \times (2\sqrt{n}) + 2 \times (n-2\sqrt{n}) = 2(n-\sqrt{n})$ and the average dissemination delay in Plume intercloud is $\frac{2(n-\sqrt{n})}{n-1}$. ■

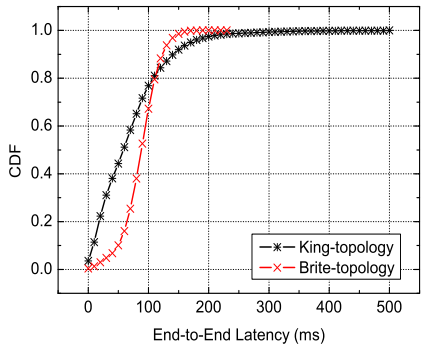

Fig. 3. End-to-end latency distribution over all pairs of King topology and Brite-topology

IV. PERFORMANCE EVALUATION

In this section, we present the details of the framework used for the experiments. Our implementation of a packet-level simulator and two intercloud overlay designs, including Plume and a DHT-based intercloud [20], was written in Java. We implement one of the most popular DHT system, Chord, as the event dissemination overlay [14]. We apply two physical topologies to simulate Internet networks. 1) King-topology: This is a real Internet topology from the King data set. The King data set delay matrix is derived from Internet measurements using techniques that described by Gummadi *et al* [24]. It consists of 2,048 DNS servers. The latencies are measured as RTTs between the DNS servers. 2) Brite-topology: This is a AS topology generated by the BRITE topology generator [25] using the Waxman model where alpha and beta are set to 0.15 and 0.2, respectively. In addition, HS (size of one side of the plane) is set to 1,000 and LS (size of one side of a high-level square) is set to 100. Totally, the Brite-topology consists of 1,000 nodes.

The simulated topology places every Plume node at a position on the King-topology or the Brite-topology, chosen uniformly at random. Every simulation results is the average 50 runs. The average delay of King-topology is 77.4 milliseconds and 96.2 milliseconds in the Brite-topology. In Figure 3, we show the CDF of the King-topology and the CDF of the Brite-topology.

First, we investigate the event dissemination latency of intercloud overlay designs. Figure 4 plots the average event dissemination latency of both two intercloud designs in the Brite-topology with the 95% confidence interval. Figure 5 plots these results in the King-topology. As shown as Figure 4 and 5, for Plume intercloud, the average event dissemination latency dose not grow with the number of Plume nodes. However, in Chord-based design, an event dissemination operation that may visit a logarithmic number hops to delivery event to other clouds [14]. The expected event dissemination latency for Plume intercloud is about 150 milliseconds in King-topology and is approximately 200 milliseconds in Brite-topology. The confidence intervals are computed over 50 independent runs.

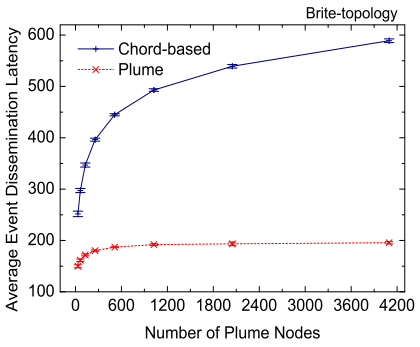

Fig. 4. Average event dissemination latency vs. number of Plume nodes in the Brite-topology

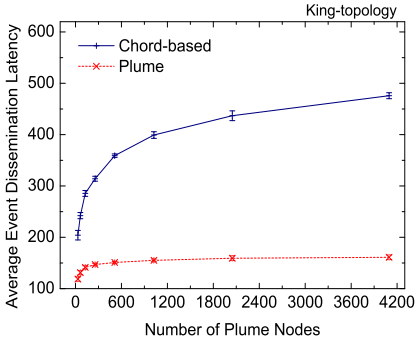

Fig. 5. Average event dissemination latency vs. number of Plume nodes in the King topology

Figure 6 plots that the relative performance of various intercloud overlay designs in the event dissemination latency for two different network topologies. We run experiments in which the number of Plume nodes is set to 1024. As shown in Figures 6, the curve for Chord-based design has a flatter tail in the King-topology. This is due to the fact that the distribution of physically link latency in king-topology is more skewed than the distribution of Brite-topology. These results show that Plume provides a suitable technique for time-critical event dissemination services.

V. Conclusions

We present Plume, an initial design of an intercloud overlay for efficient time-critical event dissemination in large-scale geographically distributed cloud computing environments. The design of Plume avoids the bottlenecks of centralized approaches or long dissemination latency of tree-based and DHT-based approaches. Overall, Plume intercloud achieves constant event dissemination latency, it can be used as a building block of intercloud for deploying efficient time-critical event dissemination services in distributed cloud environments.

References

[1] K. Church, A. Greenberg, and J. Hamilton, "On delivering embarrassingly distributed cloud services," *Proc. of ACM HotNets*, 2008.

[2] "Datacenter dynamics - gartner: private clouds to enjoy more investment through 2012 than public services," *http://www.datacenterdynamics.com*, 2009.

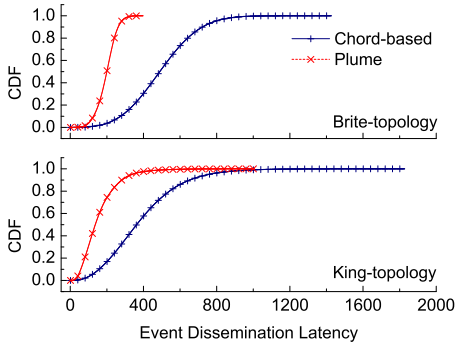

Fig. 6. The CDF of event dissemination latency in Brite and King topology

[3] N. Laoutaris, P. Rodriguez, and L. Massoulie, "Echos: Edge capacity hosting overlays of nano data centers," *ACM SIGCOMM CCR*, 2008.

[4] "Wikipedia-intercloud," *http://en.wikipedia.org/wiki/Intercloud*.

[5] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: amazon's highly available key-value store," *Proc. of ACM SOSP*, 2007.

[6] "Disaster alert network," *http://www.ubalert.com/*.

[7] "The australian early warning network, availabile at http://www.ewn.com.au/."

[8] V. Cerf, "Technology review: Integrating the clouds," *http://www.technologyreview.com/computing/24173/*, 2010.

[9] D. Bernstein, E. Ludvigson, K. Sankar, S. Diamond, and M. Morrow, "Blueprint for the intercloud-protocols and formats for cloud computing interoperability," *Proc. of IEEE International Conference on Internet and Web Applications and Services*, vol. 0, pp. 328–336, 2009.

[10] R. Buyya, R. Ranjan, and R. N. Calheiros, "Intercloud: Scaling of applications across multiple cloud computing environments," *Proc. of IEEE International Conference on Algorithms and Architectures for Parallel Processing*, vol. 0, pp. 328–336, 2010.

[11] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "How to enhance cloud architectures to enable cross-federation," *Proc. of IEEE International Conference on Cloud Computing*, vol. 0, pp. 337–345, 2010.

[12] "Facebook developers page, inc, availabile at http://developers.facebook.com/opensource/."

[13] S. Girdzijauskas, G. Chockler, Y. Vigfusson, Y. Tock, and R. Melamed, "Magnet: practical subscription clustering for internet-scale publish/subscribe," *Proc. of ACM DEBS*, pp. 172–183, 2010.

[14] W. W. Terpstra, S. Behnel, L. Fiege, A. Zeidler, and A. P. Buchmann, "A peer-to-peer approach to content-based publish/subscribe," *Proc. of ACM DEBS*, pp. 1–8, 2003.

[15] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," *Proc. of IEEE FOCS*, p. 482, 2003.

[16] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *Proc. of ACM SIGCOMM*, 2001.

[17] M. Maekawa, "A sqrt(n) algorithm for mutual exclusion in decentralized systems," *ACM Trans. Comput. Syst.*, 1985.

[18] I. Sriram and A. Khajeh-Hosseini, "Research agenda in cloud technologies," *Proc. of ACM Symposium on Cloud Computing*, 2010.

[19] J. Hoffert, D. Schmidt, and A. Gokhale, "Adapting distributed real-time and embedded publish/subscribe middleware for cloud-computing environments," *Proc. of ACM/IFIP/USENIX Middleware*, 2010.

[20] R. Ranjan and R. Buyya, "Decentralized overlay for federation of enterprise clouds," *Handbook of Research on Scalable Computing Technologies, IGI Global*, 2010.

[21] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan, "Volley: Automated data placement for geo-distributed cloud services," *Proc. of USENIX NSDI*, 2010.

[22] "Cloudstatus technologies, hyperic, inc." *http://www.cloudstatus.com/*.

[23] "Cloudswitch, inc." *http://www.cloudswitch.com/*.

[24] K. P. Gummadi, S. Saroiu, and S. D. Gribble., "King: Estimating latency between arbitrary internet end hosts," *Proc. of ACM IMW*, 2002.

[25] A. Medina, A. Lakhina, I. Matta, and J. Byers, "Brite: An approach to universal topology generation," *Proc of ACM MASCOTS*, 2001.