Toward Globally Optimal Event Monitoring & Aggregation For Large-scale Overlay Networks

Yongning Tang, Ehab Al-Shaer, Bin Zhang
School of Computer Science, Telecommunications and Information Systems
DePaul University,
Chicago, IL, USA
{ytang,ehab,bzhang}@cs.depaul.edu

***Abstract*— Overlay networks have emerged as a powerful and flexible platform for developing new disruptive network applications. The performance and reliability of overlay applications depend on the capability of overlay networks to dynamically adapt to various factors such as link/node failures, overlay link quality, and overlay node characteristics. In order to achieve this, the overlay applications require scalable and open overlay monitoring services to monitor, aggregate globally distributed events and take appropriate control actions. In this paper, we propose the techniques and algorithms to create an optimal event monitoring and aggregation infrastructure (called *MOON*) that minimizes the monitoring latency (i.e., event retrival/detection time) and event aggregation cost (i.e., intrusiveness) considering the large-scale geographical and network distribution of overlay nodes. The proposed monitoring infrastructure, *MOON*, clusters and organizes overlay nodes efficiently such that overlay applications can globally monitor and query correlated events in an overlay network with minimum latency and monitoring cost. Our simulations and experimental studies show the evaluation of *MOON* under many various topological structures, network sizes, and event aggregation volumes.**

I. Introduction

With more deployment of overlay applications and services [1], [29], [32], overlay networks are becoming large-scale distributed virtual networks on top of the Internet [1], [28], [16]. Monitoring the status of overlay nodes, discovering overlay resources (e.g., node or link capacity), reporting critical events (e.g., suspicious network flows) are significant to providing a service-oriented architecture required by overlay applications [29], [20], [18]. Developing general overlay event monitoring and aggregation infrastructure that can be used by overlay applications to *observe* and *react* is key to providing a flexible and controllable overlay network platform. Fault path discovery and recovery, multi-path routing, performance tuning, CDN, overlay on-demand services, and global intrusion detection and protection are just few examples of overlay services that require scalable distributed event monitoring and aggregation mechanism. Overlay applications (including overlay administration applications) can use the monitoring system to define or subscribe to specific events to be reported and aggregated in order to discover a global trend and perform control actions. Aggregation functions can also be defined by the overlay applications based on their goals and needs.

However, developing an efficient event monitoring infrastructure for large-scale overlay networks will require addressing the following challenges:

- Scalability: Overlay networks might contain hundreds to thousands of participating nodes. Thus, the overlay monitoring infrastructure must be able to aggregate a large volume of distributed events in a timely manner.
- Resilience: Overlay networks operate in an open service model. Since overlay nodes may not necessarily exhibit high reliability, the nodes may join/leave overlay networks at any time. The constructed event monitoring infrastructure should be able to adapt to overlay dynamics with minimum cost.
- Efficiency: Overlay event monitoring infrastructure should aggregate and retrieve events with minimum latency and intrusiveness.

Events can be any important notification/information, such as reporting the network/node status, a change in network/node performance, or user-defined MIB objects [24]. Event aggregation functions can be as simple as filtering and averaging functions, or complex analysis functions, such as time series or threshold anomaly detection functions.

In this paper, we propose a novel approach to construct an optimal hierarchical architecture for event monitoring and aggregation over large-scale overlay networks. The presented architecture called Manageable Open Overlay Network *or MOON* allows for large-scale distributed monitoring and correlation with minimum event detection time and cost. There are three steps for constructing *MOON* infrastructure: (1) constructing the overlay virtual map for coarse-grained clustering of overlay nodes, (2) performing fine-grained clustering for constructing the monitoring hierarchy, and (3) selecting the optimal set of overlay nodes as event aggregation servers. The aggregation servers represent the monitoring access points for overlay applications to define monitoring functions and run event queries. Once the monitoring functions (events and aggregation functions) have been defined, the target events are globally aggregated and delivered through *MOON* hierarchy to the subscribing overlay applications with minimal latency and cost.

The paper is organized as follows. In Section II, we review the related work. Section III describes the virtual map and *MOON* hierarchial construction, as well as the optimal aggregation server selection. In Section IV, we describe *MOON* implementation and present our extensive simulation and experimental study for evaluating *MOON*. Section V shows our conclusion and future work.

II. Related Work

There is numerous research work in the area of scalable distributed event monitoring [4], [5], [23]. Recently, significant amount of research work was focused on overlay network monitoring tools and frameworks to enable an adaptive and dynamically configurable overlay network environment.

Overlay Event Monitoring Tools: Many tools were developed to monitor overlay networks. They are mostly designed to accomplish specific monitoring tasks such as gathering node statistics [13], top-like monitoring [14], debugging [2], [15], monitoring links and failures [11], [32]. They mainly focus on simply event collecting and reporting rather than event processing. In addition, they can not be used as an open framework for implementing general monitoring tasks or event aggregation functions.

Overlay Event Monitoring Frameworks: There are a number of research projects to develop overlay event monitoring frameworks. They commonly provide more general monitoring platforms than overlay tools by integrating more features like topology and resource discovery, node membership management, event processing, and application configuration. X-Bone [30] is a distributed system to support dynamic overlay deployment and management by using two-layer IP tunneling. Multicast is used for resource discovery, which is a main deployment limitation. Astrolabe [8] is a robust large-scale distributed information management system. It achieves scalability through a hierarchical structure and robustness through a randomized peer-to-peer protocol. However, it is not clear that how effectively and efficiently an administratively generated hierarchy can serve as a communication substrate for the large-scale networks. Astrolabe also adopts a Propagate-All and unstructured gossiping techniques to attain robustness, which requires aggressive replication of the aggregates. Ganglia [18] is a hierarchical distributed monitoring system that relies on a multicast-based listen/announcement protocol to monitor node status within clusters. Ganglia mainly focuses on scalable monitoring and reporting rather than information aggregation. Opus [7] is an overlay utility for resource monitoring and allocation. It basically uses probabilistic techniques to selectively probe various characteristics of the network. A monitoring aggregation technique was proposed by Opus but it does not consider the overlay topology, event aggregation cost and latency. The Scalable Distributed Information Management System (called SDIMS) [31] aims to develop a "distributed operating systems control plane" that uses DHT to serve as a backbone to aggregate monitoring information for a large-scale distributed service. Other overlay management systems such as MON [22] and T-man [21] use gossip-style for overlay management and create on-demand overlay structures for executing instant management commands. Although they are simple and fast techniques, it is not clear how a gossip-based approach could produce a scalable or optimal architecture.

Although many of these techniques attempt to offer an efficient event monitoring and correlation service, they are not necessarily designed to develop an optimal monitoring architecture for large-scale overlay networks. More specifically, the distribution and placement of management entities to produce minimum monitoring delay and cost in overlay networks were not sufficiently addressed in the above work. In addition, many of them offer task-specific monitoring tools rather than an open event monitoring and aggregation platform for large-scale overlay networks.

III. Overlay Event Monitoring Infrastructure

MOON is designed as a global event monitoring infrastructure for large-scale overlay networks to provide efficient, scalable and adaptive event aggregation and retrieval. Events can be generated from passive or active monitoring tools. To achieve our goals, the following problems need be tackled: (1) how to organize overlay nodes efficiently considering their geographic and network distribution; and (2) how to construct a scalable and adaptive event aggregation and retrieval infrastructure. In the following section, we describe our approach to addressing these problems.

A. Overlay Virtual Map Discovery: Geo-based Clustering

Overlay map discovery is a process to convert an IP address of an overlay node into a virtual overlay ID (VID) that represents the geographic location of a node relative to other nodes in the overlay grid. This is an important step towards discovering the basic overlay structure/topology and thus employing the appropriate event monitoring infrastructure. Second, we use this result to perform a fine-grained clustering and aggregation, and to build an optimal monitoring infrastructure. The objective of overlay map discovery is to build a new name space such that an overlay node can locate itself with respect to its peers. The overlay map discovery problem can be defined as follows: Given a set of overlay nodes $O = \{n_1, n_2, \cdots, n_N\}$ with their IP addresses $I = \{IP_1, IP_2, \cdots, IP_N\}$, the goal is to design a mapping service (P) that maps a network address (IP) into a virtual location identification (VID) in the name space S, where N is the total number of overlay nodes. Thus, we have $P : IP(n_i) \rightarrow VID(n_i)$, $(1 \leq i \leq N, VID \in S)$.

Obviously, if every overlay node probes the other nodes, we may create an accurate mapping service. However, this approach suffers a serious scalability problem because it requires $O(N^2)$ network measurement, which causes high computation and network overhead. Our technique tackles the above problem by using a passive approach to construct VID that can be used to perform the fine-grained clustering and measurement. Such VID is also useful for location sensitive overlay applications like content delivery networks and web caching services.

VID consists of two components: location code (LC) and cluster ID (CID) (CID will be discussed in Section III-B). Location code (LC) presents the geographic characteristic, and cluster ID (CID) represents network characteristic of overlay nodes. In the following, we first introduce a geographic based addressing scheme, then present coarse-grained geographic-based clustering.

1) Hierarchical geographic addressing scheme: In order to create an overlay virtual map, we propose the following hierarchical geographic addressing scheme. For a given global map as shown in Fig.1, we evenly divide it into 12 partitions

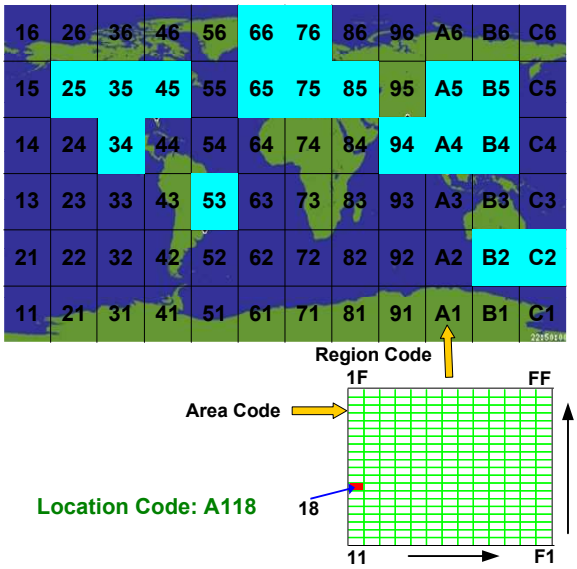


Fig. 1. **Geographic Based Clustering**

along longitude (from $-180°$ to $+180°$) labelled with prefix "1" in the most west to prefix "C" in the most east; and 6 partitions along latitude (from $-90°$ to $+90°$) labelled with postfix "1" in the most south to postfix "6" in the most north, all in hex notation. In this way, the global map is divided into $12 \times 6 = 72$ equally-sized spaces which are called *regions*. Each prefix and postfix are combined as a *region code* and each *region* occupies a $30° \times 30°$ geographic space along longitude and latitude respectively. We further divide each *region* into 15 partitions, labelled with the prefix "1" in the most west/south to the prefix "F" in the most east/north along latitude/longitude respectively. The corresponding geographic slot is called an *area*. Each prefix and postfix are combined as an *area code* and each *area* occupies a $2° \times 2°$ geographic space along longitude and latitude respectively.

Formally, each *region* is identified by the *region code* consisting of $\{XY|X \in [1,C]_{hex}; Y \in [1,6]_{hex}\}$, where X and Y represent the labels along longitude and latitude respectively. Each *area* is identified by the *area code* consisting of $\{UV|U,V \in [1,F]_{hex}\}$, where U and V represent the labels along longitude and latitude respectively. To completely identify an overlay node, we use a *Location Code* (LC) which combines *region code* XY and *area code* UV. For example, as shown in Fig.1, an overlay node with LC $A118$ refers to the node located in the region A1 and the area 18. Please note that the overlay monitoring infrastructure does not depend on these specific numbers in the proposed addressing scheme. However, finding appropriate addressing schema can effectively reduce active probing overhead. We mark this as our future work.

2) IP-to-LC conversion for coarse-grained geographic clustering: Coarse-grained geographic clustering puts the overlay nodes with the same location codes into the same cluster, which consists of the following two steps:

- Firstly, to convert the IP address to the corresponding longitude and latitude (G,T) of a given overlay node (n) by using an utility function called IP2Geo(). The utility function IP2Geo() is developed by using publicly available API (e.g. CAIDA's NetGeo service) or proprietary product (e.g. Akamais EdgeScape [1]) that takes an IP address and gives (G,T).
- Secondly, to convert (G,T) to the corresponding location code LC by using an utility function Geo2LC(). $Geo2LC()$ uses Location Code Conversion formula to convert (G,T) to the corresponding overlay node's location code $LC(n)$ as follows.

$$LC(n) = \{\lceil P \rceil_{hex} \ll 12\} \bigvee \{\lceil Q \rceil_{hex} \ll 8\} \bigvee \{\lceil \frac{G - \lfloor P \rfloor \times 30}{2} \rceil_{hex} \ll 4\} \bigvee \{\lceil \frac{T - \lfloor Q \rfloor \times 30}{2} \rceil_{hex}\} \quad (1)$$

where P and Q are calculated as $P = (G+180)/30$ and $Q = (T+90)/30$. As shown in Eq.1, the first two terms decide the *region code* and the last two terms determine the *area code*. We get *Location Code* by concatenating (ANDing) them together.

For example, in the Multimedia Networking Lab at DePaul University, we have a Planet-lab machine called *planet-lab1.mnlab.cti.depaul.edu* with IP 140.192.37.134. Using the utility function $IP2Geo()$, we get its geographic information $(-87.63, 41.88)$ (here its longitude G is $-87.63°$ and latitude T is $41.88°$). Then based on Eq.1, we get the location code for this machine as 4526.

As a result of the geographic clustering, the *Location Code* can identify the relative geographic location (in different directions) of any two nodes in different areas as shown in Fig.1. The *regions* with light color represent the ones currently having overlay Planet-lab [28] nodes. A logic *area* still spans a very large geographic space (approximately 200×100 miles), which implies that there is still potentially a large number of overlay nodes in the same *area*, and the overlay nodes in the same *area* could also be relatively far from each other. Within the same area, the geographic distance does not accurately reflect the relative location among nodes in terms of their network distances. Thus, conducting a network distance based fine-grained clustering is necessary to put the closer nodes together in order to reduce monitoring overhead. In the following, we introduce our fine-grained network similarity-based clustering technique to achieve this goal.

B. Fine-grained network similarity-based clustering

Traditional clustering techniques, like k-mean and k-center, can not be directly applied to solve such problem because (1) the number of clusters is unpredictable and depends on the node distribution; (2) the pair-wise distance between nodes is not determined unless we conduct N^2 probings among all nodes, which is not scalable (N is the total number of nodes in the same logic *area*). Here we propose a novel network similarity-based clustering technique which has the following desirable features: (1) it finds optimal cluster number; (2) the cluster size is adjustable; (3) it requires only $O(N)$ probes. We assume all overlay nodes have registered themselves with a central registration server (denoted as RS). The process of network similarity-based clustering consists of the following four steps:

1) *Landmark selection:* For a given *area* A, the initial bootstrap node n_b (e.g. the node with lowest IP or highest computing capacity) used to construct a communication hierarchy is designated by the registration server RS. Then RS randomly chooses one node from each of A's closest neighbor *areas* in each geographic direction (e.g. north, northeast) to be its landmark node. The set of selected landmark nodes is denoted as *L*. The information of L is further disseminated to all nodes in the same *area* using a hierarchical communication structure, which is constructed for each *area* as follows: (1) All nodes in the *area* A are sorted based on their IPs; (2) Every D nodes (D is a configurable parameter) form a group such that the node with the lowest IP is designated as the group leader; (3) Step (2) is recursively repeated until $\lceil N/D \rceil < 2D$, where N is the number of nodes in the *area* A. The node n_b in each *area* runs this algorithm after obtaining the node list information from RS; then it sends the result (hierarchic structure information) to the designated group leaders in the hierarchical structure, which pass this information down in the hierarchy till it reaches the leaf nodes.
2) *Limited active network probing:* all nodes in the *area* A independently conduct probing to all landmark nodes in L within the given period $T \times rand()$ seconds in order to avoid probing explosion. Here $rand() \in (0, 1]$. The probing result of each *node* is forwarded in a distance vector format to the direct leader in the hierarchy. Each leader will wait until it receives all distance vectors or timeout. If it times out, the leader will query the on-line status of these nodes.
3) *Similarity-based clustering:* The distance vector information of all nodes in *area* A is ultimately aggregated by n_b, which calculates the similarity value between every pair of nodes and forms fine-grained clusters by using the similarity-based clustering algorithm as described below. The main objective of similarity clustering is to put the similar (i.e. close) nodes into the same cluster and the dissimilar (i.e. far) nodes into different clusters. Finally, the node n_b disseminates the result to all nodes in *area* A via the same hierarchical communication mechanism.

The similarity between any pair of nodes in an *area* is calculated using an utility function called *Similarity Function* (SF). The goal of *Similarity Function* is to label closer nodes with higher similarity and distant nodes with less similarity. There are several methods to calculate the similarity. *Cosine Vector Distance*, shown in Eq.(2), is commonly used as a means of evaluating the distance between two nodes [12]. We also calculate the global similarity threshold, $S_{threshold}$, which is the average similarity value of all nodes in the same *area* as follows: $S_{threshold} = \sum_{i \in A} \sum_{j \in A, i \neq j} S_{ij}/[N(N-1)]$.

$$S_{ij} = SF(V_i, V_j) = \frac{(V_i \times V_j)}{|V_i| \times |V_j|} = \frac{\sum_{m=1}^{k} d_{im} \times d_{jm}}{\sqrt{\sum_{m=1}^{k} d_{im}^2} \times \sqrt{\sum_{m=1}^{k} d_{jm}^2}} \quad (2)$$

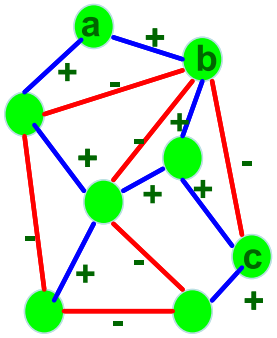


Fig. 2. **Similarity Graph**

Based on Charikar et al's research work in [10], we propose a similarity graph as shown in Fig.2 to formulate the similarity feature in an overlay network. Given similarity threshold ($S_{threshold}$), for any two overlay nodes u and v, associate an edge with either a label "+" if $SF(u, v) \geq S_{threshold}$ (such as node a and b in Fig.2); or a label "-" if $SF(u, v) < S_{threshold}$ (such as node b and c in Fig.2). Apparently, the nodes with "+" edge are regarded as similar; otherwise they are regarded as dissimilar. For a clustering assignment $C = (C_1, \cdots, C_k)$, let C(v) be the set of vertices in the same cluster as v. We call an edge e_{uv} a *positive benefit* if $e_{uv} = (+)$ and yet $u \in C(v)$; we call an edge e_{uv} a *negative benefit* if $e_{uv} = (-)$ and yet $u \notin C(v)$. The total benefit of a clustering C is denoted as $B(C)$, which is the summation of *positive benefit* ($B_p(C)$) and *negative benefit* ($B_n(C)$).

$$B(C) = B_p(C) + B_n(C) \quad (3)$$

$$B_p(C) = \sum \{c_e : e = (u, v) \in E(+), u \in C(v)\} \quad (4)$$

$$B_n(C) = \sum \{c_e : e = (u, v) \in E(-), u \notin C(v)\} \quad (5)$$

We introduce a binary variable x_{ij} and formalize similarity-based clustering as the following Integer Programming Problem. Here x_{ij} equals 1 if node i and j are assigned to the same cluster; otherwise, x_{ij} equals 0. Our optimization objective (Eq.6) is to find a valid clustering assignment of $\{x_{ij}\}$ to maximize the total benefit $B(C)$. An assignment of $\{x_{ij}\}$ is valid if $x_{ij} \in \{0, 1\}$ (Eq.7), $x_{ij} = x_{ji}$ (Eq.9) and $\{x_{ij}\}$ satisfies the triangle inequality (Eq.8). Lastly, Eq.10 is used to bound the cluster size between C_{min} and C_{max} as required by the overlay network administrator. Here lower bound C_{min} and upper bound C_{max} are used to restrict the maximal number of clusters and nodes in one single area or cluster to avoid overloading the event aggregation node of this area or cluster.

$$max\{\sum_{e \in E(-)} c_e(1 - x_{ij}) + \sum_{e \in E(+)} c_e x_{ij}\} \quad (6)$$

Subject to

$$x_{ij} \in [0, 1] \quad (7)$$

$$x_{ij} + x_{jk} \geq x_{ik} \quad (8)$$

$$x_{ij} = x_{ji} \quad (9)$$

$$\forall i, C_{min} < \sum_{i,j \in V} x_{ij} < C_{max} \quad (10)$$

Algorithm 1 Similarity Clustering With Cluster Size Constraints

1: $k \leftarrow \lceil N/C_{max} \rceil$, $\varepsilon \leftarrow 1/k$
2: Use $MaxAgree$ algorithm to cluster nodes
3: **for** each cluster c with $|c| > C_{max}$ **do**
4: $k_c \leftarrow \lceil |c|/C_{max} \rceil$
5: Use the farthest-point clustering algorithm to divide C into k_c sub-clusters
6: **end for**
7: **repeat**
8: Sort cluster with $|c| < C_{min}$ based on size in acceding order. For clusters with same size, sort based on $S(c) = \sum_{i,j \in c} S_{ij}$ in ascending order.
9: **for** each node i in the first cluster of sorting result **do**
10: assign i to cluster c' which produces the maximal average distance such that $\sum_{j \in c'} S_{ij}/|c'| \geq \sum_{j \in c} S_{ij}/|c|, \forall c$ and $|c'| + 1 < C_{max}$
11: **end for**
12: **until** no cluster with $|c| < C_{min}$ exist

Theorem 1: The Similarity Clustering problem of finding the clustering assignment to maximize the total benefit in the similarity graph is an NP-complete problem.

The Similarity Clustering problem can be mapped to the Maximal Agreement problem, which has been shown to be an NP-complete problem in [10]. Our sketch proof can be found in [3]. In the following, we propose a heuristic algorithm to solve this problem.

1) Heuristic algorithm for Similarity Clustering with cluster size constraints: The Similarity Clustering problem differs from Maximal Agreement problem because of the cluster size constraints. In algorithm 1, we use $|c|$ and $S(c)$ to represent the cluster size and the total similarity between each pair of nodes of a cluster c. We first calculate the minimum number of clusters needed and set the parameter ε (line 1). After using the *MaxAgree* algorithm to do the initial clustering, we further divide the oversized clusters (line 3-6) by adopting an modified *farthest-point* clustering algorithm [19]. Then we sort the undersized clusters based on their sizes in ascending order. For those clusters with the same size, we order them based on their $S(c)$ values. Our rational is that we want to redistribute the nodes to the clusters with smaller size first. For the clusters with the same size, we want to redistribute the nodes to the ones with less similarity first (line 8-9). Finally, we merge every node from undersized clusters with the one which may be incurred with the maximal average similarity increase (line 9-11).

C. Overlay Event Aggregation Optimization

As a result of the geographical based and network similarity-based clustering, the nodes in an overlay network are organized in a hierarchical structure that consists of (top-down): overlay *regions*, overlay *areas*, overlay *clusters* and *nodes*, as shown in Fig.3. However, in order to aggregate and correlate events across areas, there are potentially a very large number (i.e. more than 10^4) of active *areas* (Fig.1) that might include aggregation servers. In other words, in order to discover globally distributed events, the requests have to be sent to a very large number of event aggregation servers. However, this causes a very high overhead and delay. One of the main objectives in an efficient overlay event monitoring infrastructure is to design an optimal communication substrate to minimize event aggregation (e.g. event objects shipping cost) and information retrieval cost (e.g. event monitoring latency). We can define the problem formally as follows: Given a set of overlay nodes O, to select a subset of nodes from O as Event Aggregation Servers (EAS) such that event aggregation and event information retrieval cost can be minimized. In the following sections, we will show our solution to address this issue.

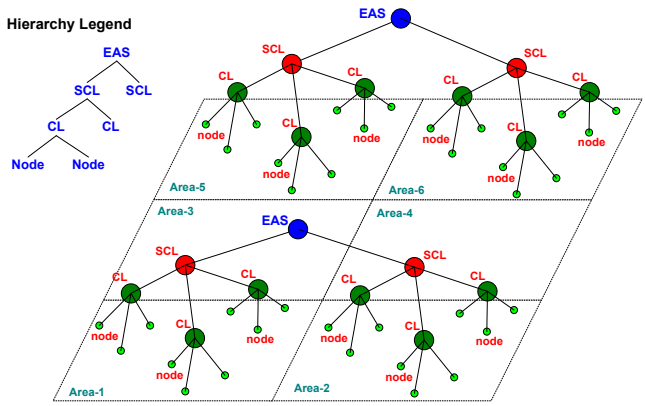


Fig. 3. **Overlay Hierarchical Event Monitoring Infrastructure**

1) Event Monitoring Architecture: Our motivation for doing geographic and network similarity-based clustering is to form a base for a hierarchical overlay event monitoring infrastructure that considers minimizing communication cost for aggregating and retrieving events. For widely distributed overlay networks, designing an efficient hierarchical infrastructure is required for achieving scalable monitoring. In the following, we identify the components of our proposed infrastructure as shown in Fig.3.

Definition 1: Cluster Leader (represented as CL) is the node which has the *maximum similarity* summation with all other nodes in the same *cluster* C.

$$CL := \{u | \sum_{u \neq v;, u,v \in C} S_{uv} = MAX_{u \in C}(\sum_{u \neq v, u,v \in C} S_{uv})\}$$

Definition 2: Cluster Leader Set (represented as $CLSet$) contains all cluster leaders in the same *Area* A.

$$CLSet := \{\{CL_i\} | \forall CL_i \in A\}$$

Definition 3: Super Cluster Leader (represented as SCL) is a cluster leader which has the *maximum similarity* summation to all CL nodes of the same *Area* A.

$$SCL := \{u | \sum_{u \neq v;\, u,v \in CLSet} S_{uv} = MAX_{u \in CLSet}(\sum_{u \neq v, u,v \in CLSet} S_{uv})\}$$

As a result of this architecture, the cluster nodes report their collected events to their cluster leaders (CL), which

consequently report these events to their super cluster leader (SCL). Thus, each SCL can aggregate event information generated in its *area*.

2) Event Aggregation Server Placement: Event aggregation is a very important service to correlate events across different *regions*, *areas* or *clusters* and discover the root cause of fault, performance and security problems. For example, a large scale of DDoS can be discovered even if it is a stealthy attack when distributed events are correlated across the network. Similarly, performance problems, such as congested overlay links or overloaded overlay nodes, can be isolated if events from different nodes sharing the same nodes/links are revealed and combined. Thus, our overlay monitoring infrastructure should also serve as a communication substrate for event aggregation and retrieval. However, the hierarchical structure constructed so far consists of a large number of *SCL* nodes on the top layer. Thus, it is desirable to construct an event aggregation layer on the top of this hierarchy from a set of optimally selected nodes called Event Aggregation Servers (*EAS*) for efficient global event correlation and retrieval.

The design objective of constructing an optimal event aggregation server layer is to minimize both event aggregation cost (AC) and event retrieval cost (RC). Theoretically, event aggregation servers can be selected from any SCL nodes. We identify the $min(N_C, |R_i|)$ number of *SCL* nodes, which are the top capable ones in this region R_i, as the candidate set to facilitate the process of *EAS* selection. Here, N_C is the number of top capable SCL nodes, and $|R_i|$ is the total number of active *areas* in *region* R.

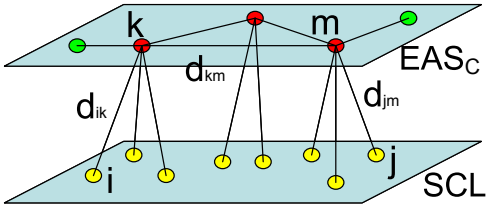


Fig. 4. **Event Aggregation Server Placement**

Therefore, we formalize Event Aggregation Server Placement Problem (*EASPP*) as following: given an undirected graph $G = (V, E)$ with two hierarchical layers, where the EAS_C layer represents *EAS* candidate nodes and the *SCL* layer represents the rest of SCL nodes. As shown in Fig.4, all SCL nodes are pre-partitioned into EAS_C and SCL layers such that $EAS_C \cap SCL = \emptyset$ and $V = EAS_C \cup SCL$ (usually $|EAS_C| \ll |SCL|$). The nodes are connected with edges $E = \{(i,k) \in V | i \in SCL, k \in EAS_C\} \cup \{(k,m) \in V | k, m \in EAS_C\}$. Each edge $(i,j) \in E$ is associated with the weight equal to d_{ij}, here d_{ij} is the distance between the *areas* represented by the corresponding nodes i and j. For each node $i \in SCL$, we associate an aggregation weight r_i, which is corresponding to the number of monitored objects that need be aggregated (e.g. CPU utilization, the number of packets dropped, queue size, etc.).

We introduce binary variable x_{ik} such that x_{ik} equals one if and only if node i ($i \in SCL$) reports to node k ($k \in EAS_C$). We also introduce binary variable y_k such that y_k equals one if node k ($k \in EAS_C$) is selected as an EAS node. We formally define *Aggregation Cost* (AC) in Eq. 11, which represents the average event transportation (i.e. reporting) cost.

$$AC = \sum_{i \in SCL} \sum_{k \in EAS_C} x_{ik} r_i d_{ik} \tag{11}$$

$$RC = \sum_{k \in EAS_C} \sum_{m \in EAS_C, k \neq m} y_k y_m w_k w_m d_{km} \tag{12}$$

Retrieval Cost (RC) is defined as Eq. (12). Here $w_k = \sum_i x_{ik}$ is the total number of SCL nodes which report to EAS server node k. Between any pair of EAS server nodes m and k, the average of event retrieval requests from m to k depends on both w_m and w_k. Intuitively, the more SCL nodes represented by node m, the more retrieval requests m may receive; the more nodes represented by k, the more chance the information on k may be queried.

In event aggregation server placement problem, we need to find an optimal assignment x_{ik} and y_k such that the total aggregation and retrieval cost can be minimized, as shown in Eq. 13. The problem can be formalized by the following Integer Programming Problem:

$$min\{ \sum_{i \in SCL} \sum_{k \in EAS_C} x_{ik} r_i d_{ik} + \sum_{k \in EAS_C} \sum_{m \in EAS_C, k \neq m} y_k y_m w_k w_m d_{km} \} \tag{13}$$

Subject to

$$x_{ik} \in \{0, 1\} \tag{14}$$
$$y_k \in \{0, 1\} \tag{15}$$
$$x_{ik} \leq y_k \tag{16}$$
$$\forall i \in SCL, \sum_{k \in EAS_C} x_{ik} = 1 \tag{17}$$

In the above formalization, the conditions 14 and 15 restrict both x_{ik} and y_k as integer variables. In the EASPP problem, SCL nodes only report to EAS server as defined by the condition 16. The condition 17 enforces every SCL node only report to a single EAS server.

Theorem 2: The problem of finding the set of event aggregation server nodes to minimize the total aggregation and retrieval cost is an NP-hard problem.

We can prove theorem 2 via a reduction from SAHLP (Single Allocation Hub Location Problem [27]). Hubs are special nodes which route the traffic in communication or transportation networks. In the hub location problem, both a set of demanding nodes and a subset of hub candidate nodes are given. There are traffic flows between any pair of nodes. The hub location problem is to locate a subset of hubs in a network and allocate non-hub nodes to hub nodes such that the summation of the costs of transporting flow between all origin destination pairs in the network is minimized. Here, Single Allocation means each node can only be allocated to one hub. The detailed proof of theorem 2 can be found in [3].

3) Heuristic Algorithm for EASPP: We developed a heuristic algorithm for EASPP based on the p-hub location approximation algorithm proposed in [9]. In our proof in [3], we show that, for each ESAPP instance, we can construct an instance of SAHLP which shares the same solution for the EASPP instance. Therefore, our approach is to firstly

TABLE I

VARIABLE AND NOTATION IN EASPP HEURISTIC ALGORITHM

EAS_c	The set of EAS server candidates
SCL	The set of SCL nodes
A_c	The set of current aggregation servers
A_p	The set of previous aggregation servers.
R	The set of remaining aggregation servers
Z_{ikmj}	The binary variable stand for whether node i sent traffic flow to node j through aggregation server k, m
$C(A)$	The total cost based on aggregation servers A
T_{ik}	How many traffic flow from node i to other node through aggregation server k.
$Cost_c$	The current total transportation cost
$Cost_p$	The previous total transportation cost based on previous aggregation set A_p

construct the corresponding SHALP problem and find the solution, then map it to EASPP. However, unlike the p-hub location problem, the EASPP does not assume any given fixed number of servers (or p-hubs). Therefore, our heuristic algorithm starts initially with the two-hub solution for the constructed SAHLP; then proceeds incrementally by adding more hubs and measuring the benefit of adding hubs until we reach the optimal set of hubs (or aggregation servers). Note that we solve the constructed SAHLP by firstly solving the Multiple-Allocation Hub Location problem. The details of the algorithm are described in Algorithm 2. After we find the solution of Multiple-Allocation problem, we can simply extract the single allocation solution to the SAHLP. Then we can directly map the SAHLP solution to ESAPP solution. The assignment of nodes to hubs represents the assignment of SCL nodes to EAS nodes. In the following, we use EAS and hubs interchangeably in our discussion of the heuristic algorithm. The variables and notation used in this algorithm are defined in Table I. The $Cost$ defined in Table I is the total transportation cost calculated based on multiple allocation hub location problem.

In Algorithm 2, we first find the optimal pair of EAS servers (A_c) which give the smallest total transportation cost in the constructed hub location problem (line 2-4). Then we add an EAS server incrementally to A_c (line 6-7) and do a greedy interchange to swap one server from A_c with one server candidate in the remaining candidate set R as long as this swapping can reduce the cost (line 8-12). After greedy interchange, we compare the cost of the current EAS server set (A_c) with the cost of the previous one (A_p). If adding more servers will reduce the cost, we will continue in step 2. Otherwise, we will stop incrementing and go to step 4 (line 14-18). In step 4, we will construct the single allocation solution based on the multiple allocation solution from step 3. In this way, each SCL node is assigned to the most frequently used EAS server, via which this SCL node sends traffic flow to the other nodes (line 21-23).

Algorithm 2 Aggregation Server Selection and Assignment

1: Step 1: /* Initialization*/
2: Construct the according hub location problem.
3: $A_c \leftarrow \emptyset$
4: Find the optimal pair of aggregation servers a_1 and $a_2 \in EAS_C$, assign $A_c \leftarrow \{a_1, a_2\}$, $R \leftarrow EAS_C \backslash A$, $Cost_c \leftarrow C(A_c)$, $Cost_p \leftarrow Cost_c$
5: Step 2: /* Server Set Finding */
6: For each $n \in R$, calculate $C(A_c \cup \{n\})$.
7: Get the n' which produce the minimum cost such that $C(A_c \cup \{n'\}) \leq C(A_c \cup \{n\})\ \forall n \in R$, assign $A_c \leftarrow A_c \cup \{n'\}$, $Cost_c \leftarrow C(A)$
8: **while** there exist a pair of nodes $n^{''} \in R$ and $a \in A_c$ such that $C(\{n^{''}\} \cup A_c \backslash \{a\}) < Cost_c$ **do**
9: $A_c \leftarrow \{n^{''}\} \cup A_c \backslash \{a\}$
10: $R \leftarrow EAS_C \backslash A_c$
11: $Cost_c \leftarrow C(A_c)$
12: **end while**
13: Step 3: /* Server Set Finalization*/
14: **if** $Cost_c < Cost_p$ and $R \neq \emptyset$ **then**
15: $Cost_p \leftarrow Cost_c$, $A_p \leftarrow A_c$
16: goto step 2
17: **else**
18: $Cost_c \leftarrow Cost_p$, $A_c \leftarrow A_p$
19: **end if**
20: Step 4: /* Nodes Assignment*/
21: find multiple allocation solution to get Z_{ikmj} based on A_c
22: For each $i \in SCL$ and $k \in A$, calculate $T_{ik} = \sum_j \sum_m Z_{ikmj}$
23: For each node i, choose k' such that $T_{ik'} \geq T_{ik}$, assign node i to aggregation server k'.

IV. IMPLEMENTATION AND EVALUATION

A. MOON Implementation

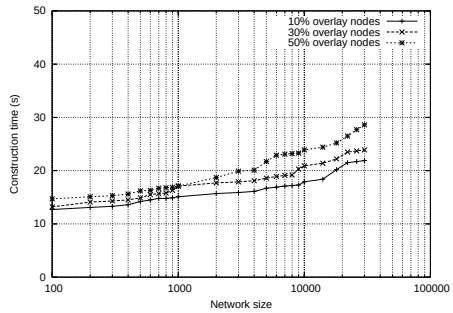
We have implemented *MOON* on Planet-lab infrastructure. We select 690 overlay nodes, whose DNS names can be resolved and the returned geographic information via IP2Geo utility function is valid. All these nodes are widely distributed as depicted in Table II. *MOON* consists of three main components: Admin Server, Event Aggregation Server and Event Agent, which conduct the following function modules as needed.

TABLE II

NODES DISTRIBUTION IN PLANET-LAB *MOON* INFRASTRUCTURE

US (511)	# of o-nodes	International (179)	# of o-nodes
.edu	345	Asia	72
.org	62	Europe	73
.net	26	Canada	21
.com	73	South America	9
.us	5	Australia	4

- Bootstrapping service module: It executes in the admin server to extract the IP address list of all overlay nodes either from a database or via unicast or multicast probing. It then uses the techniques described in Section III-A.1 to convert the IP address of an overlay node to a Location Code (LC). In our implementation, we assume that new nodes have to register their information (e.g., name, IP address, capabilities) first in the admin server before being considered part of the overlay network.
- Monitoring and Reporting Module (MRM): MRM runs on each event monitoring agent. MRM consists of three components: (1) *passive event monitoring* that uses SNMP to access public and private MIBs to observe node or link attributes/objects; (2) *active probing* that could be conducted periodically to collect more information; and (3) *event reporting* that is used by the monitoring agents to report observed events to CL or SCL for aggregation and correlation. Reporting service is triggered based on three traditional modes: periodically reporting based



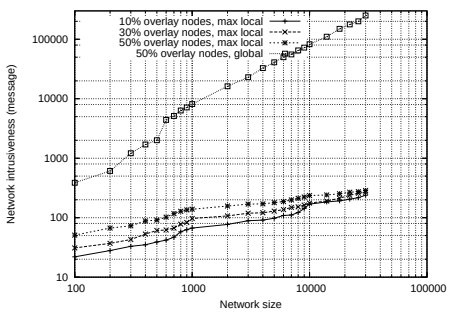


Fig. 5. **The Impact of Network Size on Scalability (a) Construction Time (b) Network Intrusiveness**

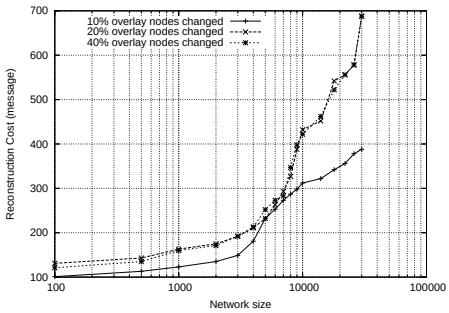


Fig. 6. The Impact of Overlay Dynamics on *MOON* Resilience

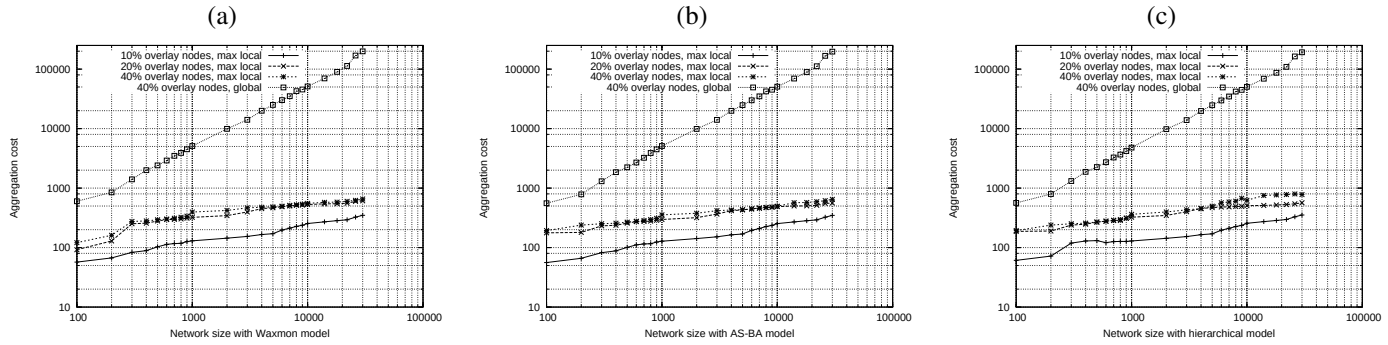


Fig. 7. **The Impact of Network Topology on Event Aggregation Cost (a) AS-level Waxman model (b) AS-level Barabasi-Albert model (c) Hierarchical model**

on schedule, threshold-based reporting, or query-based reporting.

- Event Aggregation Module (EAM) : EAM is activated on the nodes assigned as event aggregation service nodes. The event aggregation functions are user-specified and could be simple arithmetical calculations such as finding the AVG, top-N-MAX, bottom-N-MIN or custom complex processing such as Bayesian filtering, or statistic analysis. The EAM also probes hierarchically allocated nodes for periodic health reporting.
- Presentation Module: Presentation Module is a web-based interface for controlling *MOON* management infrastructure and viewing the management operations of an overlay network. Overlay network administrators can have different views based on the geographic locations, autonomous systems, area codes, etc. They can also customize and submit limited or global event queries.

B. Evaluation of MOON Infrastructure

In this section, we evaluate the construction and operation of *MOON* infrastructure. We conducted extensive simulations and experimentations on PlanetLab [28] to measure the scalability, monitoring latency, intrusiveness and resilience of *MOON* event monitoring infrastructure.

1) Simulation and Experiment Evaluation Metric: In our evaluation study, we considered the following factors that impact evaluation metrics.

- *Network size*: Both overlay and underlay network size are considered. We use a total number of network nodes that ranges from 100 to 30,000, and overlay network size that ranges from 10 to 12,000. The network size significantly impacts many performance parameters such as construction time and event aggregation latency. “
- *Network topology*: We use a synthetic topology generation tool called BRITE [25] with three types of topology models: AS-level Waxman; AS-level Barabasi-Albert model and Hierarchical models. Network topology decides node distribution, which may affect the structure of event monitoring infrastructure.
- *Event rate*: Event rate is defined as the number of events per unit of time that can be received by an event aggregation server. It is very important to use event rate to evaluate the efficiency and scalability of a distributed event monitoring system like *MOON*.
- *Number of aggregation functions*: The aggregation functions are defined by overlay applications and evaluated at different layers in the hierarchy, namely *CL, SCL,* and *EAS* levels. The number of aggregation functions that exist in *MOON* may significantly affect the performance of the event monitoring infrastructure.
- *Overlay Dynamics*: Overlay nodes may join/leave overlay infrastructure arbitrarily. This may trigger the reconstruction of the monitoring infrastructure to retain the design objectives of *MOON*.

2) The Impact of Network Size and Topology on MOON *Scalability: MOON* scalability is evaluated with respect to the following factors:

- *Construction Time* (CT): CT is defined as the elapsed time from the starting of bootstrap service until the finish of *MOON* construction. CT consists of three periods:

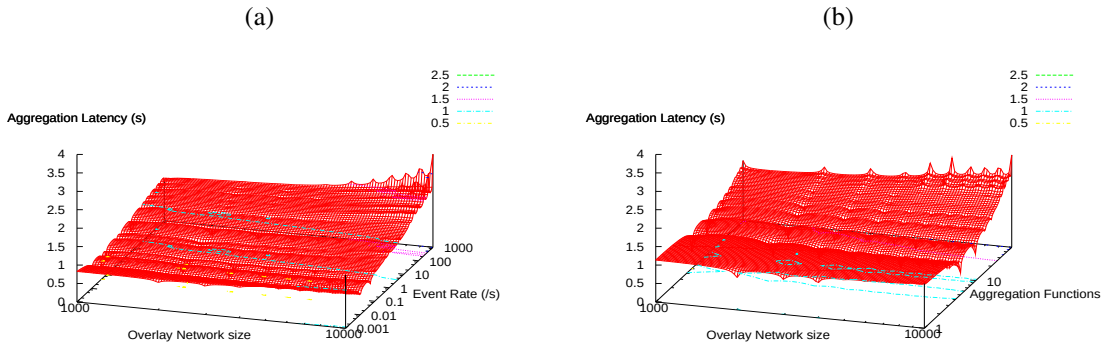


Fig. 8. **The Impact of Event Processing on Aggregation Latency (a) Event Rate change (with number of aggregation function = 1), (b) Number of aggregation functions change (with event rate = 1 event/sec)**

(1) bootstrap time, (2) clustering time, and (3) event aggregation server layer selection time.

- *Network Intrusiveness*: Network Intrusiveness is measured by the total number of messages exchanged within or across *areas* during the construction of *MOON* infrastructure.
- *Aggregation Cost*: The aggregation cost is calculated based on Eq.11, which represents the total shipping cost of the monitored objects.

We use BRITE [25] to create the following three topological types: (1) an AS-level Waxman model, (2) an AS-level Barabasi-Albert model, and (3) a hierarchical model. We simulate three different scales for each topology type: (1) a small-scale network of 100 to 1,000 nodes distributed in a relatively small area is used to simulate regional overlay networks, (2) a medium-scale network of 1,000 to 10,000 nodes that are widely distributed is used to simulate national overlay networks, and (3) a large-scale network of 10,000 to 30,000 nodes is used to simulate global overlay networks. For each kind of simulation, we selected 10% to 40% of the nodes as overlay nodes. Then we ran the simulation to generate as many random topology networks as possible and we calculated the average as the final result. For each generated topology, we used the coordinate of each node in the graph to calculate the location code (LC).

In our simulation, we defined the cluster size bounds (C_{min}, C_{max}) as $(10, 30)$, and we chose T, which is the minimal *Construction Time* as described in Section III-B as 10 seconds. Passive geographic clustering enables concurrent communication and computations to take place between agents during the *MOON* construction process. From the simulation results shown in Fig.5-a, we can clearly see that the *Construction Time* is slowly increased from 12.7 seconds to 28.6 seconds when the network size significantly increased from 1,000 to 30,000. This implies an increase rate of 0.041 $second/node$. Fig. 5-b shows that when the network size increases, the network intrusiveness (i.e. the maximum number of exchanged messages among all *areas*) slowly increases at the rate of 0.013 $message/node$. The top curve in Fig.5-b shows the total number of globally exchanged messages in all area when 40% of the nodes are overlay nodes. It is important to notice that due to the geographic and similarity-based clustering, network probing is limited within a single *area* or between nearby *areas*. The network probing used in network clustering is linearly proportional to the total number of overlay nodes in an *area*, but randomly distributed over time to avoid probing explosion.

The *Aggregation Cost* (AC) simulation results for different types of network topologies are shown in Fig.7. Here we assume each node holds 10 to 1,000 monitored objects. The AC increases with the increase of network size. However, as shown in Fig.7-a, the maximum aggregation cost in the cluster and super cluster levels increases slowly at rate of 0.014 as the network size increases. Similarly, Fig.7-b and 7-c show similar results with different network topologies.

3) The Impact of Overlay Dynamics on MOON *Resilience:* Overlay networks use an open service model, which means overlay nodes may join/leave arbitrarily. In the *MOON* infrastructure, the geographic based clustering contributes significantly in the creation of a resilient monitoring infrastructure as any local changes of the node membership in one area do not affect the other areas. Let us assume that an user requires minimum 90% accuracy in clustering, which implies to re-compute the similarity-based clustering and the related hierarchical structure if more than 10% of the overlay nodes membership (e.g. join/leave) was changed in any area. Similarly, when over 10% overlay *areas* have been re-constructed, the EAS selection will also be re-computed. Apparently, since the effect of network dynamics is limited within the areas, we focus in our simulation on the variation of global membership changes across multiple *areas* from 10% to 40% of the total number of *areas*. We assume at least 30% of nodes are overlay nodes. We conduct simulations to incrementally add/remove from 1% to 20% overlay nodes for one area up to 40% of the total number of areas. We measure the *Reconstruction Cost* (the number of messages required to reconstruct the optimal architecture) of each of these changes in overlay network. As shown in Fig.6, the reconstruction cost increases slowly (0.15%) when the network size is small or medium (i.e. the number of network $nodes < 10000$). However, for a large scale of network, the cost may increase evidently (6%). This small increase in number of messages consequently shows that the delay cost (interruption time) for updating *MOON* structure is significantly small.

4) The Impact of Event Rate and Network Size on Aggregation Latency: The event aggregation latency is the time between generating the event and reporting the event after aggregation. It is a critical parameter that affects the performance of monitoring systems. Here, the event rate is defined as the maximum number of events sent by an agent (in overlay nodes) per second. Different event types have different refreshment (i.e., update) requirements, and thereby require different refreshing frequency. For instance, the events such as the overlay node CPU or disk utilization is less time-sensitive than other critical events such as reporting anomalous traffic. In our simulation, we vary the event rate from one event every 1,000 seconds to 1,000 events per second. We also vary the number of aggregation functions from 1 to 50 running on any nodes (*CL, SCL, EAS*). As shown in Fig.8-a, when the network size increases from 1,000 to 30,000, and the event rate increases from 0.001 event/sec to 1000 event/sec (where the number of aggregation functions equals 1), the aggregation latency varies only from 0.16 sec to 2.52 sec which constitutes a very slow increase rate of 0.00236 second per increased event. Similarly, as shown in Fig. 8-b, if the number of aggregation functions increases from 1 to 50 with fixed event rate 1 event/sec, the aggregation latency increases from 0.17 sec to 2.5 sec which constitutes a slowly increasing rate of 0.047 second per aggregation function.

V. Conclusion and Future Work

Distributed event monitoring is significantly important for overlay networks in order to provide a flexible platform for disruptive network services. The scalability, flexibility and resilience are the key challenging issues in this research area. We proposed and implemented an open framework for the distributed event monitoring and aggregation in overlay networks, called *MOON*. Overlay applications can selectively subscribe to the interested events and define their event aggregation functions. *MOON* constructs the optimal monitoring architecture for aggregating and delivering events with the minimum latency and cost. *MOON* uses three techniques to accomplish this: (1) passive coarse-grained geographical-based clustering to create a virtual map of overlay nodes based on their relative locations; (2) active fine-grained similarity-based clustering to organize overlay nodes according to the network topologies, and building an efficient monitoring hierarchy; and (3) optimal allocation of overlay aggregation servers on the top-level of *MOON* hierarchy such that the event retrieval time and cost are minimized. Our evaluation study shows that *MOON* is scalable with the large-scale networks up to 30K nodes with different topology structures. We also show that the intrusiveness and maintenance overhead due to dynamic changes in the overlay networks is affordable. In our future work, we will focus on making *MOON* more adaptive to the node failures and communication performance bottlenecks, integrating necessary authentication and authorization mechanisms, and developing generic monitoring interfaces for overlay applications.

References

[1] Akamai *Technology Overview*, http://www.akamai.com/html/technology/.

[2] Jeannie Albrecht, Christopher Tuttle, Alex Snoeren, and Amin Vahdat. "PlanetLab Application Management Using Plush." *In Proceedings of ACM Operating Systems Review (SIGOPS-OSR)*, January 2006.

[3] Yongning Tang, Ehab Al-shaer and Bin Zhang. "Overlay Network Management Architecture." *CTI Technique Reports, DePaul University*, 2006

[4] Ehab Al-Shaer. "A Dynamic Group Management for Scalable Distributed Event Correlation." *In Proceedings of IEEE/IFIP Integrated Management (IM'2001)*, May 2001.

[5] Ehab Al-Shaer, Hussein Abdel-Wahab and Kurt Maly. "HiFi: A New Monitoring Architecture for Distributed Systems Management." *IEEE 19th International Conference on Distributed Computing Systems (ICDCS'99)*, Austin, Texas, May 1999.

[6] N. Bansal, A. Blum, S. Chawla. C "Correlation Clustering", *Machine Learning 56*, 2004

[7] R. Braynard, D. Kostic, A. Rodriguez, J. Chase, and A. Vahdat. "Opus: An overlay peer utility service." *In Proceedings of IEEE OpenArch*, 2002

[8] Robert Van Renesse, Kenneth P. Birman, and Werner Vogels. "Astrolabe: A Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining." *ACM Transactions on Computer Systems.*, May 2003

[9] James F. Campbell "Hub Location and the p-Hub Median Problem" *Operations Research*, 1996

[10] M. Charikar, V. Guruswami, A. Wirth. "Clustering with Qualitative Information." *in Proceedings of 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS'03)*, 2003

[11] Y. Chen, D. Bindel, H. Song, and R. H. Katz. "An Algebraic Approach to Practical and Scalable Overlay Network Monitoring." *in Proceedings of ACM SIGCOMM*, 2004

[12] Y. Chen, C. Overton and R. H. Katz. "Internet Iso-bar: A Scalable Overlay Distance Monitoring System." *in Journal of Computer Resource Management, Computer Measurement Group, Spring Edition*, 2002

[13] CoMon: *http://codeen.cs.princeton.edu/comon/.*

[14] CoTop: *http://codeen.cs.princeton.edu/cotop/.*

[15] CoTest: *http://codeen.cs.princeton.edu/cotest/.*

[16] Emulab web site: *http://www.emulab.net.*

[17] Greg Finn, Joe Touch. "Network Construction and Routing in Geographic Overlays." *ISI Technical Report ISI-TR-2002-564*, July 2002

[18] Ganglia Monitoring System: *http:///ganglia.sourceforge.net.*

[19] T. Gonzalez. "Clustering to minimize the maximum intercluster distance." *Theoretical Computer Science*, 38:293-306, 1985

[20] Ryan Huebsch, Brent Chun, Joseph M. Hellerstein, Boon Thau Loo, Petros Maniatis, Timothy Roscoe, Scott Shenker, Ion Stoica and Aydan R. Yumerefendi. "The Architecture of PIER: an Internet-Scale Query Processor." *In Proceedings of CIDR 2005*,CA, USA, January 2005.

[21] Mark Jelasity and Ozalp Babaoglu. "T-Man: Gossip-based overlay topology management." *Engineering Self-Organising Systems: Third International Workshop (ESOA 2005)*, 2005

[22] Jin Liang, Steven Ko, Indranil Gupta and Klara Nahrstedt. "MON: On-demand Overlays for Distributed System Management." *In Proceedings of 2nd USENIX Workshop on Real, Large Distributed Systems (WORLDS'05)*, 2005

[23] K.S. Lim and R. Stadler. "Real-time views of network traffic using decentralized management." *In Proceedings of IEEE/IFIP Integrated Management (IM'2005)*, May 2005.

[24] K. McClohrie and M. Rose. "Management Information Base for Network Management of TCP/IP-based Interface:" *MIB-II, RFC1213*, 1991

[25] A. Medina, I. Matta, and J. Byers. On the origin of power laws in Internet topologies, *ACM Computer Communication Review*, (Apr. 2000).

[26] Navas, J. C., Imielinski, T. "GeoCast - Geographic Addressing and Routing." *Proc. of 3rd ACM/IEEE International Conf. on Mobile Computing and Networking*, Sep. 1997

[27] Jinhyeon Sohn , Sungsoo Park "The single allocation problem in the interacting three-hub network" *Networks, Volume 35, Issue 1*, Nov. 1999

[28] PlanetLab: http://www.planet-lab.org.

[29] Mike Wawrzoniak, Larry Peterson, and Timothy Roscoe. "Sophia: An Information Plane for Networked Systems." *In Proceedings of the Second Workshop on Hot Topics in Networking (HotNets-II)*, Cambridge, MA, November 2003.

[30] "Xbone." *http://www.isi.edu/xbone*

[31] Praveen Yalagandula and Mike Dahlin. "A Scalable Distributed Information Management System." *In Proceedings of ACM SIGCOMM*, 2004.

[32] M. Zhang, C. Zhang, V. Pai, L. Peterson, and R. Wang. "Planetseer: Internet path failure monitoring and characterization in wide-area services." *In Proceeding of OSDI*, December 2004.