IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON MOBILE SERVICE COMPUTING WITH INTERNET OF THINGS

Received March 24, 2019, accepted April 5, 2019, date of publication April 16, 2019, date of current version June 20, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2911573

Mining Frequent Items Over the Distributed Hierarchical Continuous Weighted Data Streams in Internet of Things

SHUZHUANG ZHANG[1], (Member, IEEE), YU ZHANG[2], (Member, IEEE), LIHUA YIN[3], TINGTING YUAN[4], (Member, IEEE), ZHIGANG WU[1], AND HAO LUO[1]

[1]Institute of Network Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China
[2]College of Computer and Control Engineering, Nankai University, Tianjin 300071, China
[3]Cyberspace Institute of Advanced Technology (CIAT), Guangzhou University, Guangzhou 510006, China
[4]Inria Diana Sophia Antipolis-Mediterranee, 06000 Nice, France

Corresponding author: Lihua Yin (yinlh@gzhu.edu.cn)

This work was supported in part by the National Key Research and Development Program under Grant 2016YFB0801205.

ABSTRACT Recently, with the increasing supply of band width and the diversification of applications in the Internet of Things (IoT), it has been a challenging problem to identify frequent items (also called heavy hitters) in high-speed and dynamically changing data streams. As well as, these data streams are from multiple sources in a distributed environment. To solve it, we propose the distributed-tracking schemes for continuously mining frequent items in the multi-level, non-regular tree-based communication structure. Our method employs a combination of local tracking and delays updating at each node to produce highly communication-efficient and space-efficient solutions. To reduce the communication cost, it only sends the frequency increments to violate a pre-defined threshold through a hierarchy of intermediate nodes, which is interposed between the monitoring nodes and the root node. With the information gathered, the root node continuously reports the set of frequent items. Two optimization approaches are proposed to minimize the worst-case total communication and minimize the worst-case maximum load on any link under any input streams. We perform extensive simulations with real traffic traces to evaluate the performances of the two optimization approaches.

INDEX TERMS Frequent items, distributed weighted data streams, multi-level communication structure, continuous tracking.

I. INTRODUCTION

Recently, with the increasing supply of band width and diverse applications in traditional Internet and Internet of Things [1], it is of great significance to identify frequent items (also called heavy hitters) in high-speed and variable data streams on designing high quality data integration, connection, communication mechanism and a number of applications, such as discovering denial-of-service (DoS) attacks, warning heavy network users, monitoring traffic trends, balancing traffic load, and discovering virus signatures [2]–[7].

Most of the existing work focuses on detecting frequent items at a single node [8]–[16]. However, in many applications, data sets are physically distributed over a large number of nodes [17]. For example, in IP-network monitoring, an ISP (Internet Service Provider) may have packet traces collected at hundreds (or even thousands) of ingress and egress routers, and the amount of data collected at each router can be in the order of several terabytes. In order to detect a DDoS attack, we need to find the destination IP addresses that occur frequently in IP traffic aggregated over the union of all packet traces at all ingress and egress points. A similar scenario emerges within the sensor networks. For example, many sensors are deployed in the field to collect environ-mental information and the frequently occurring sensor measurements are usually the most interesting sort of data [18]. In this case, an item can be a value that a sensor read at a particular point in time, the same item may appear multiple times at one or more sensor nodes. In summary, the underlying infrastructure comprises several remote nodes that (each with its own local data source) can exchange information through a communication network.

The associate editor coordinating the review of this manuscript and approving it for publication was Tie Qiu.

Furthermore, to identify frequent items in a distributed environment also needs to consider the communication cost. For example, in IP network monitoring, given the gigantic and evolving nature of these physically distributed data sets, it is usually infeasible to transport all the data to a single location for completing processing due to the prohibitively high communication cost. In wireless sensor networks, the main factor affecting battery life is the communication cost; therefore, in order to improve the availability useful life of the network, we need to reduce the communication cost. Communication cost is one of the most important measure of complexity during the entire tracking period. Besides, the monitoring is continuous, which need real-time track of measurements or events. In this situation, a continuous query is placed instead of a one-shot query which only requires one-shot responses. This yield yet further challenge.

This paper studies how to design communication-efficient algorithms for continuously tracking the global frequent items over the distributed weighted data streams. The remainder of the paper was organized as follows: in section II we defined the problem and discussed the related work, in section III we proposed the method of hierarchical continuous distributed frequent item mining, in section IV we analyzed its complexity, in section V we proposed two optimization approaches. We presented the experimental results in section VI, and concluded the paper in section VII.

II. PRELIMINARIES

We first define the multi-level, non-regular tree-based communication structure by use of an examples, and formalize it.

A. PROBLEM DEFINITION

Here, suppose that a distributed monitoring environment is a tree-based hierarchical structure. It consists of $m \geq 1$ monitoring nodes, one central root node and several intermediate nodes. Each monitoring node is a leaf which observes and summarizes a single local data stream. Partial knowledge of these summaries will be then relayed through a hierarchy of intermediate nodes that is interposed between the monitoring nodes and the root node. With the knowledge gathered, the root node continuously reports the set of frequent items over the union of all m distributed data streams. An example of the tree-based communication structure is shown in Figure1. For the sake of clarity, we number the nodes by levels from top to bottom and from left to right. In this figure, there are 7 local data streams (i.e., $S_1, S_2, S_3, S_4, S_5, S_6, S_7$) processed by monitoring nodes 1, 4, 5, 6, 7, 8 and 9 respectively, root node islabelledby0,andintermediate nodes arelabelledby2 and 3 respectively.

Denote the distributed data streams as $S_1, S_2, \ldots, S_m$. S_i consists of a sequence of tuples $(v, c_{i,v,t})$, ordered by timestamp t, where i is the ID of data stream($1 \leq i \leq m$), $v \in U$ ($\{1 \ldots k\}$), k is a positive integer) is the item name, $c_{i,v,t} > 0$ is the weight of the item. Denote the union of data stream $S_1, S_2, \ldots, S_m$ as S, reordered by the timestamp. The goal of the root node is to continuously supply the frequent items in the global data stream S which satisfies the following properties:

(1) Output each item v which satisfies $f_v \geq \varphi N$
(2) Output no item v which satisfies $f_v < (\varphi - \epsilon) N$.

Where the true frequency of item v in S is denoted as f_v (i.e., the total weights of tuples throughout S with the same item name v), the total frequency in S is denoted as N (i.e., the total weights of all tuples throughout S), the user-specified error bound is denoted as ϵ, the user-specified support threshold is denoted as φ.

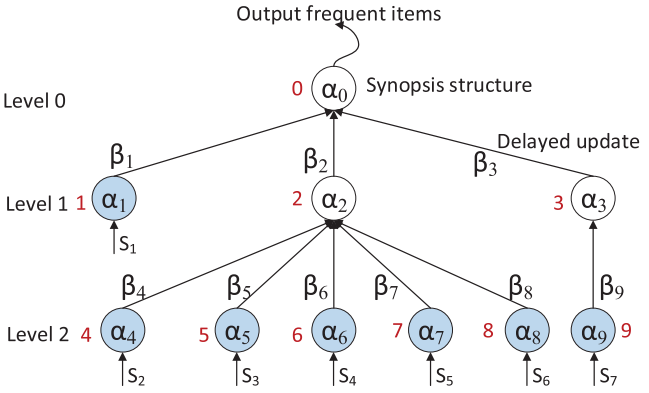


FIGURE 1. An example of the tree-based communication structure.

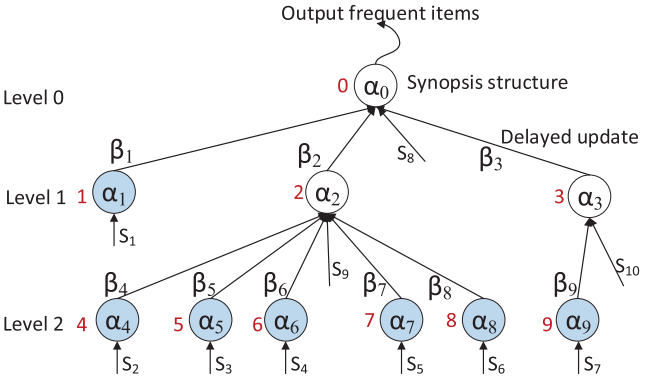


FIGURE 2. An extension of the tree-based communication structure.

In practice, the intermediate nodes and the root node might need to directly process the items from the local data streams as well. For example, in Figure 2, besides the messages from the child nodes, the root node and the intermediate nodes 2 and 3 need to process the items from the local data streams S_8, S_9, S_{10} respectively. For clarity, we call them extended structure (where the intermediate nodes and the root node might process the local data streams) and the normal structure (where only the monitoring nodes process the local data streams) respectively in the following section.

B. RELATED WORK

Most of the existing work focuses on mining frequent items in data streams at a single node [8]–[16]. Only a few works investigated this problem in a distributed scenario.

Tong *et al.* [17] proposed a local threshold-based deterministic algorithm and a sketch-based sampling approximate algorithm to track distributed probabilistic frequent items. Manjhi *et al.* [19] gave a method to find frequent items in the union of multiple distributed data streams. However, this method only can be applied to the case of the one-shot query and its communication structure is restricted to balanced, regular trees. Subsequently, Manjhi *et al.* [20] extended the above work to the non-regular trees. Cormode *et al.* [21] introduced an approach to continuously track approximate quintiles summaries of collections of physically distributed streams with communication cost $O\left(\frac{k}{\epsilon^2}log\,(n)\right)$, and this method can be extended to continuously track the frequent items in a distributed environment with the same communication cost. As well, the communication structure was extended from the single-level mode to the multi-level model. The major drawback is its high communication cost, which means the total error tolerance must be large enough to prevent small deviations from triggering summary refreshes. Babcock and Olston [22] designed some heuristics for monitoring the top-k values over remote data streams. This approach can be adapted to continuously track the frequent items over remote data streams [23], [24]. There are two main disadvantages: first, this method remains heuristic in nature which means it lacks of a theoretical analysis; second, only the single-level communication structure is considered. Yi and Zhang [25] proposed a deterministic algorithm for continuously tracking frequent items over remote data streams with worst-case communication cost $\mathrm{O}\left(\frac{k}{\epsilon}log\,(n)\right)$, however the authors only consider the single-level communication structure. Huang *et al.* [26], [27] designed two randomized algorithms for finding frequent items over distributed streams, one [23] only supports the one-shot query, and the other [24] supports the continuous query with communication cost $\mathrm{O}\left(\frac{\sqrt{k}}{\epsilon}log\,(n)\right)$. However, both of them only consider the single-level communication structure. Several research [28]–[30] have considered the problem of finding items with frequency above a fixed threshold, not some fraction φ of the total frequency as in this paper. Zhao *et al.* [28] and Zhao *et al.* [29] proposed respectively two methods to find icebergs (items whose frequency of occurrence is above a certain threshold) over remote data sets. A major drawback is that they only consider the one-shot query, which is not a continuous monitoring solution. Keralapura *et al.* [30] considered the "threshold counts" problem where returning the aggregate frequency counts of the items that are continuously monitored by distributed nodes whenever the actual counts exceed a given threshold value. However, the authors only consider the single-level communication structure.

In summary, none of the previous works have proposed communication-efficient algorithms for continuously tracking the global frequent items over the distributed weighted data streams in the multi-level, non-regular tree-based communication structure.

III. HIERARCHICAL CONTINUOUS DISTRIBUTED FREQUENT ITEM MINING

Our goal in this paper is to make the root node which is in the tree based communication structure effectively track the frequent items of the update streams observed at the monitoring nodes. To achieve this goal, the root node needs to maintain each item's frequency as well as the total frequency. One simple way is propagating all the updates from the monitoring nodes to the root node. In this way, it is apparently very wasteful. Another simple method is periodically sending the approximate frequent items and total frequency maintained in the monitoring nodes to the root node. However, periodically sending cannot provide continuous frequent items. This is because that the accuracy of returned frequent items can be guaranteed only at the time of sending. Moreover, the communication cost is higher.

In this paper, we improve the optimized weighted frequent algorithm (*OWFrequent* [31]) to continuously track the global frequent items over the distributed weighted data streams. Our method employs a combination of local tracking and delayed updating at each node to produce highly communication and space-efficient solutions. To reduce the communication cost, only the frequency increments of the items violating a pre-defined threshold are relayed through a hierarchy of intermediate nodes interposed between the monitoring nodes and the root node.

A. ESTIMATE EACH ITEM'S FREQUENCY IN THEROOTNODE

The monitoring nodes will not send the exact frequency counts of the items to the root node, but send the frequency increments of the items. Therefore, a new factor of frequency increment is added in *OWFrequent* to track the increment on the frequency count since last update to the parent node. In order to reduce the communication cost, the frequency increment will not be sent to the parent node until its value violates a pre-defined threshold (i.e., delayed updating). To be precise, all the nodes in tree structure work as follows:

A-1) Each monitoring node (numbered s) uses *OWFrequent* to maintain an α_s-synopsis structure (a min-heap with $\left\lceil\frac{1}{\alpha_s}\right\rceil$ entries, $1 < \alpha_s < \epsilon$ represents the error tolerance of node s) for the items in the local data stream $S_{i_s}(1 \leq i_s \leq m)$. When a new item $\left(v, c_{i_s,v}\right)$ arrives, if v is in the α_s-synopsis structure, then increase $\Delta_v = \Delta_v + c_{i_s,v}$, where Δ_v represents the frequency increment of item v and its initial value is 0. Otherwise, v enters the α_s-synopsis structure (i.e., when replacing the minimum item in *OWFrequent*), and set $\Delta_v = c_{i_s,v}$. When $\Delta_v \geq \beta_s N_{i_s}$, the monitoring node sends a message (v, Δ_v) to its parent node, and resets $\Delta_v = 0$, where β_s represents the delayed update coefficient of node s ($0 < \beta_s < \epsilon$), N_{i_s} represents the total frequency of all items in S_{i_s}.

A-2) Each intermediate node (numbered s') receives all messages from its child nodes, and uses *OWFrequent* to

maintain an $\alpha_{s'}$-synopsis structure(a min-heap with $\left\lceil \frac{1}{\alpha_{s'}} \right\rceil$ entries, where $1 < \alpha_{s'} < \epsilon$ represents the error tolerance of node s'). When a new message (v, δ_v) arrives, if v is in the $\alpha_{s'}$-synopsis structure, then increase $\Delta_v' = \Delta_v' + \delta_v$, where Δ_v' represents the frequency increment of item v and its initial value is 0. Otherwise (v is not in the $\alpha_{s'}$-synopsis structure), v enters the $\alpha_{s'}$-synopsis structure (i.e., when replacing the minimum item in *OWFrequent*), and set $\Delta_v' = \delta_v$. When $\Delta_v' \geq \beta_{s'} N_{s'}'$, the intermediate node sends a message $\left(v, \Delta_v'\right)$ to its parent node, and resets $\Delta_v' = 0$, where $0 < \beta_{s'} < \epsilon$ represents the delayed update coefficient of node s', N_s'' represents the total frequency of all messages received by node s'.

A-3) the root node receives all messages from its child nodes and uses *OWFrequent* to maintain an α_0-synopsis structure (a min-heap with $\left\lceil \frac{1}{\alpha_0} \right\rceil$ entries, $1 < \alpha_0 < \epsilon$ represents the error tolerance of root node).

B. ESTIMATETHETOTAL FREQUENCY IN THEROOTNODE

Two counters are added in each node dedicated to record the total frequency as well as its frequency increment which will not be sent to the parent node until its value violates a pre-defined threshold (i.e., delayed updating). To be precise, all the nodes in the tree structure work as follows:

B-1) Each monitoring node (numbered s) adds two counters N_s'' and Δ_s'', where N_s'' records the total frequency of all items in the local data stream S_{i_s} $(1 \leq i_s \leq m)$ processed by node s (labeled $N_s'' = N_{i_s}$), Δ_s'' records the frequency increment of N_s''. When a new item $\left(v, c_{i_s,v}\right)$ arrives, increment $N_s'' = N_s'' + c_{i_s,v}$ and $\Delta_s'' = \Delta_s'' + c_{i_s,v}$. When $\Delta_s'' \geq \beta_s N_s''$, the monitoring node sends a 0-message $\left(0, \Delta_s''\right)$ to its parent node, and resets $\Delta_s'' = 0$. Where β_s represents the delayed update coefficient of node s.

B-2) Each intermediate node (numbered s') adds two counters $N_{s'}''$ and $\Delta_{s'}''$, where $N_{s'}''$ records the total frequency of all 0-messages received by node s', $\Delta_{s'}''$ records the frequency increment of $N_{s'}''$. When a new 0-message $(0, \delta)$ arrives, increment $N_{s'}'' = N_{s'}'' + \delta$ and $\Delta_{s'}'' = \Delta_{s'}'' + \delta$. When $\Delta_{s'}'' \geq \beta_{s'} N_{s'}''$, the intermediate node sends a 0-message $0, \Delta_s''$ to its parent node, and resets $\Delta_s'' = 0$, where $\beta_{s'}$ represents the delayed update coefficient of node s'.

B-3) The root node adds one counter N_0'' which records the total frequency of all 0-messages received. Note that the estimated total frequency in the root node is N_0''.

C. IDENTIFY THE DISTRIBUTED FREQUENY ITEMS IN THE ROOT NODE

When it comes a frequent item query, the root node scans through each entry in the min-heap maintained by *OWFrequent* and reports item v if $\hat{f}_v \geq \left\lceil w N_0'' \right\rceil$, where $\hat{f}_v$ represents the estimated frequency of item v maintained by OWFrequent, $0 < w < \varphi$ represents the output threshold and N_0'' represents the estimated total frequency.

IV. COMPLEXITY ANALYSES

A. SPACE AND TIME COMPLEXITY ANALYSIS

Theorem 1: For each node s, the required memory space is $O\left(\frac{1}{\alpha_s}\right)$, the required time to process one item is $O\left(\log\left(\frac{1}{\alpha_S}\right)\right)$, where α_S represents the error tolerance of node s $\alpha_s > 0$.

Proof: Since node s needs to maintain an α_s-synopsis structure (i.e. a min-heap with $\left\lceil \frac{1}{\alpha_s} \right\rceil$ entries in *OWFrequent* [32]) for all the items and messages are received, the memory space required by node s is $O\left(\frac{1}{\alpha_S}\right)$ Note that it only required 2 more counters in node s to estimate the total frequency N(see B-1,B-2,B-3 in sectionIII). Since the min-heap in node s has $\left\lceil \frac{1}{\alpha_s} \right\rceil$ entries, it requires $O\left(\log\left(\frac{1}{\alpha_s}\right)\right)$ time to process one item.

Corollary 1: The errors tolerance of each node is bounded by its memory size. That is to say, if the memory size of node s is μ(measured the number of entries in the min-heap), then $\alpha_s \geq \frac{1}{\mu}$, where α_s represents the error tolerance of node s.

Proof: Refer to Theorem 1.

In practice, the memory size of each node is usually fixed, according to Corollary 1, then the minimum error tolerance of each node is fixed as well.

B. COMMUNICATION COMLEXITY ANALYSIS

Theorem 2: No matter in the normal or extended structure, the total number of messages sent by node s(except the root node) to its parent node is no more than $2\frac{\log\left(\tilde{N}_s\right)}{\beta_s}$, where β_s represents the delayed update coefficient of node s; $\tilde{N}_s$ represents the total frequency of all items in T_s (T_s is the sub-tree rooted at node numbered s).

Proof: In the first scenario where node s is in the normal structure (i.e., only the monitoring nodes process the local data streams). There are two cases for the inspected node s.

Case 1: node s is a monitoring node which processed the local data stream S_{i_s}, then according to A-1 and B-1 in section III, node s will not send a message to its parent node until N_{i_s} and N_s'' are increased by a factor of $1+\beta_s$, node is bounded by $\frac{\log(N_{i_s})}{\log(1+\beta_s)} + \frac{\log\left(N_s''\right)}{\log(1+\beta_s)}$, where N_s'' represents the total frequency of all items in S_{i_s} and $N_s'' = N_{i_s}$. Obviously, in this case $N_{i_s} = \tilde{N}_s$, as there only one node(i.e. the monitoring node s) in T_s. Since $\frac{1}{\log(1+\beta_s)} \approx \frac{1}{\beta_s}$ (for $0 < \beta_s < 1$), the number of messages sent by monitoring node s to its parent node is about $2\frac{\log\left(\tilde{N}_s\right)}{\beta_s}$.

Case 2: node s is an intermediate node, then according to A-2 and B-2 in section III, node s will not send a message to its parent node until N_s' and N_s'' are increased by a factor of $1+\beta_s$ the number of messages sent by this node is bounded by $\frac{log\left(N_s'\right)}{log(1+\beta_s)} + \frac{log\left(N_s''\right)}{log(1+\beta_s)} \approx \frac{log\left(N_s' + logN_s''\right)}{\beta_s}$, where N_s' represents

the total frequency of all messages(except the 0-messages) received by node s, $N_s^{''}$ represents the total frequency of all 0-messages received by node s. Since each node only send the frequency increments to its parent node, we have $N_s^{'} \leq \tilde{N}_s$ and $N_s^{''} \leq \tilde{N}_s$, thus, the number of messages sent by node s to its parent node is no more than $2\frac{\log\left(\tilde{N}_s\right)}{\beta_s}$.

In the second scenario where node s is in the extended structure (i.e., the intermediate nodes and the root node might directly process the local data streams). There are two cases for the inspected node s as well. Similar to that of the first scenario, it is easy to prove that in the extended structure, whether the monitoring node or the intermediate node, the number of messages sent by node s to its parent node is no more than $2\frac{\log\left(\tilde{N}_s\right)}{\beta_s}$.

In conclusion, no matter in the normal or extended structure, the total number of messages sent by node s to its parent node is no more than $2\frac{\log\left(\tilde{N}_s\right)}{\beta_s}$. Note that the root node will not send any messages since it has no parent node.

Corollary 2: No matter in the normal or extended structure, the total number of messages sent by all the nodes is no more than $2\log(N)\sum_{s=1}^{z-1}\frac{1}{\beta_s}$, where z represents the total number of nodes, β_s represents the delayed update coefficient of node s, N is the total frequency of all items in the union of all local data stream.

Proof: Because the root node (numbered 0) will not send any messages, according to Theorem 2, the total number of messages sent by all nodes is no more than $2\sum_{s=1}^{z-1}\frac{\log\left(\tilde{N}_s\right)}{\beta_s}$. Since for each $1 \leq s \leq z-1$, we have $\tilde{N}_s \leq N$ (no matter in the normal or extended structure). Consequently, $2\sum_{s=1}^{z-1}\frac{log\left(\tilde{N}_s\right)}{\beta_s} \leq 2log(N)\sum_{s=1}^{z-1}\frac{1}{\beta_s}$. To sum up, no matter in the normal or extended structure, the total number of messages sent by all nodes is no more than $2\log(\mathrm{N})\sum_{\mathrm{s}=1}^{\mathrm{z}-1}\frac{1}{\beta_s}$.

V. APPROPRIATE SETTINS OF PARAMETERS

A. OPTIMIZATION APPROACHES

In this subsection, we first consider how to select the delayed update coefficient to use at each node in order to achieve one of the two objectives: (1) minimizing the worst-case total communication cost under any input streams, (2) minimizing the worst-case load on any link under any input streams.

We propose algorithm 1 to determine the optimized delayed update coefficients which ensures a relatively small amount of total communication. According to Corollary2, the total number of messages sent by all nodes is no more than $2\log(N)\sum_{s=1}^{z-1}\frac{1}{\beta_s}$. Hence, we can use algorithm1 to minimize $\sum_{s=1}^{z-1}\frac{1}{\beta_s}$, then we have a relatively small amount of total communication in practice.

The algorithm 2 is proposed to determine the optimized delayed update coefficients which ensure the minimum worst-case maximum load on any link.

Algorithm 1 Total Communication Optimization

Require: the user-specified error bound ϵ

1: let $s_1, s_2, \ldots, s_m$ represent the monitoring nodes;

2: calculate $\alpha_{\mathrm{s_i}}^{'}$ $(1 \leq i \leq m)$ and $\alpha_{max}^{'}$, where $\alpha_{s_i}^{'}$ represents the sum of each node's error tolerance in the path from monitoring node s to the root node, $\alpha_{max}^{'}$ represents the maximum of all $\alpha_{s_i}^{'}$

3:set $\beta_{s_i}^{''} = Max\left\{\frac{\epsilon-\alpha_{max}^{'}}{2}, \frac{\epsilon}{2}-\alpha_{s_i}^{'}\right\}(1 \leq i \leq m)$;

4:let $\beta_{s_i}^{'}$ $(1 \leq i \leq m)$ represents the sum of each node's delayed update coefficient in the path from $\mathrm{s_i}$ to the root node;

5:Caculate $\beta_1, \beta_2, \ldots, \beta_{z-1}$ to minimize $\sum_{s=1}^{z-1}\frac{1}{\beta_s}$ subject to the constraints: $\beta_{s_1}^{'} \leq \beta_{s_1}^{''}, \beta_{s_2}^{'} \leq \beta_{s_2}^{''}, \ldots, \beta_{s_1}^{'} \leq \beta_{s_1}^{''}$

Algorithm 2 Maximum Link Load Optimization

Require: the user-specified error bound $\in$

1: let $s_1, s_2, \ldots, s_m$ represent the monitoring nodes in descending order according to their levels(if equals, sort the node from left to right);

2: calculate $\alpha_{s_i}^{'}$ $(1 \leq i \leq m)$ and $\alpha_{max}^{'}$;

3: set $\beta_{s_i}^{''} = Max\left\{\frac{\epsilon-\alpha_{max}^{'}}{2}, \frac{\epsilon}{2}-\alpha_{s_i}^{'}\right\}(1 \leq i \leq m)$;

4: **for** each monitoring node S_i from S_{i1} to S_m **do**

5: let k_{s_i} represents the number of nodes in the path from $\mathrm{S_i}$ to the root node of which delayed update coefficients have not yet been determined;

6: let $\beta_{s_i}^{'''}$ represents a sum of each determined node's delayed update coefficient in the path from S_i to the root node.

7: set each undetermined node's delayed update coefficient $\beta_{\mathrm{s_i}}^{'}$ in the path from $\mathrm{S_i}$ to the root node according to $\frac{\beta_{s_i}^{''}-\beta_{s_i}^{'''}}{k_{s_i}}$.

8: **end for**

B. THE INITIALIZATION PHASE

In this subsection, we demonstrate the requirements for an initialization phase in our proposed method.

Theorem 3: No matter in the normal or extended structure, the maximum message sending rate from node s (except the root node) is $\frac{2}{\beta_s t}$ (messages/s), where β_s is the delayed update coefficient of node s and t represents the time in seconds.

Proof: Let $M_s(t) = 2\frac{\log\left(\tilde{N_s}t\right)}{\beta_s}$. For clarification, in simplest case that $\tilde{N_s}t = p_s t$, where $\mathrm{p_s} > 0$ is a constant. Therefore the maximum message sending rate from node s is $M_s^{'}(t) = \frac{2}{\beta_s t}$ (messages/s) (derivative of $M_s(t)$).

Corollary 3: No matter in the normal or extended structure, the total message sending rate from all nodes is no more than $\frac{z}{t}\sum_{s=1}^{z-1}\frac{1}{\beta_s}$ (messages/s), where z represents the total number of nodes, β_s represents the delayed update coefficient of node s and t represents the time in seconds.

Proof: Refer to Theorem 3.

From Theorem 3, the message sending rate from each node decreases rapidly with the change of time. However, at the very beginning the rate might be too high for the connected link to transmit (i.e., the message sending rate might exceed the bandwidth limit of the connected link at the very beginning). That is to say, Theorem 3 demonstrates the requirement for an initialization phase, i.e., each node may have to wait some initialization time before the connected link is able to transmit the generated messages. In practice, at the beginning, each node will handle the incoming items (messages) as usual; however, it will send any messages to its parent node after the initialization time. Strictly speaking, in order to guarantee the correctness of theory, each node needs to send all the items whose frequency increments exceeding the pre-defined threshold to its parent node right after the initialization time. However, in practice, when a query is submitted after a long time (e.g., one hour), the returned global frequent items will be hardly affected by the initialization time due to its short duration (e.g., just several seconds in practice). Meanwhile, the total communication overhead will be significantly reduced by preventing the nodes from sending any messages in the initialization phase.

VI. EXPERIMENTAL EVALUTION

In our experiments, we tracked the distributed TCP and UDP elephant flows (i.e., frequent items) defined by the famous five-tuple (i.e., source IP address, destination IP address, source port, destination port and protocol) in the extended communication structure depicted in Figure2.

We used three real traffic traces as input for our distributed monitoring system. The CERNET trace is a TCP packet header trace collected at an OC-48 link of CERNET (China Education and Research Network) on May 31, 2007 in both directions. The CAIDA48 trace [32] is an anonymized packet header trace measured at an OC-48 west coast peering link on August 14, 2002 and we used the first hour traffic of one direction (Direction-1). The CAIDA192 trace [33] is an anonymized packet header trace measured at an OC-192 link from the equinix-sanjose Internet data collection monitor on July 17, 2008 and we used the first 18-minute traffic of one direction (Direction-A). The characteristics of these three traffic traces are given in table 1. Notice that we just list the approximate numbers of packets and flows (including both TCP and UDP).

TABLE 1. Characteristics of traffic traces.

Traces	Packets(mil.)	Flows (mil.)	Duration
CERNET	1,940	85	9 hours
CAIDA48	420	27	1 hour
CAIDA192	400	27	18 mins

In Figure2, the packets of each trace are distributed to all the nodes, i.e., nodes 0,1,2,3,4,5,6,7,8,9. In this experiment, all the packets to all nodes at first are uniformly distributed in Figure2. And then in order to evaluate the performance under other probability distributions, we used random distributions to update different nodes. The random distributions are created by generating random probabilities associated with each node in Figure2. These probabilities are used to send the packets from the input trace to these nodes. Note that a different random distribution is generated for every simulation run. We repeat the simulation 500 times to ensure that we capture a variety of different random distributions.

In this experiment, the error bound ϵ ranged from 0.006 to 0.01 (i.e., 0.006, 0.008, 0.01), the support threshold $\varphi = \epsilon$. For simplicity, we set $\alpha 1 = \alpha 2 = \alpha 3 = \alpha 4 = \alpha 5 = \alpha 6 = \alpha 7 = \alpha 8 = \alpha 9 = 0.1\epsilon$. Both of the optimization approaches (i.e., Algorithm 1 and 2) are explored in the experiments. The following two metrics are employed to evaluate the communication complexity:

- Total communication cost, measured the total number of messages sent by all nodes.
- Maximum link load, measured the maximum number of messages sent by any node.

A. UNIFORMLY DISTRIBUTED PACKETS

Here we explore the effect of using a uniform distribution to update different nodes. Figures3, 4 and 5 show total communication cost accumulate over time with error bounds 0.006, 0.008 and 0.01.

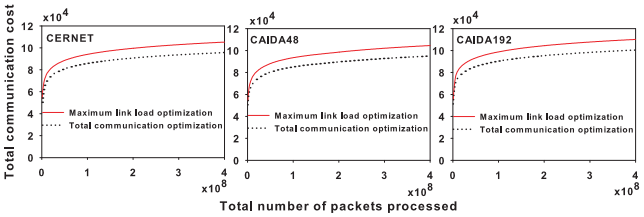


FIGURE 3. Total communication cost over time with a uniform distribution ($\epsilon = 0.006$).

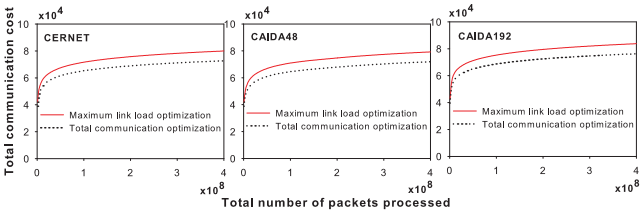


FIGURE 4. Total communication cost over time with a uniform distribution ($\epsilon = 0.008$).

Just as expected, in each figure the total communication optimization method requires less communications than the maximum link load optimization method. And then, the total communication cost decreases rapidly with the error bound (seeFigure3 ($\epsilon = 0.006$)), This figure presents the highest total communication overhead, whereas Figure5 ($\epsilon = 0.01$) presents the lowest total communication overhead. This is because that the total communication cost has a strong reverse relation with the delayed update coefficients according to

Corollary 2. Generally speaking, the greater the error bound is, the bigger the delayed update coefficients will be (see Algorithms 1 and 2). Therefore, the total communication cost has a strong reverse relation with the error bound. At last, there is a sudden spike in the total communication cost followed by a gradual increase in each figure. This is because that the total message sending rate from all nodes has a strong reverse relation with the time according to Corollary 3.

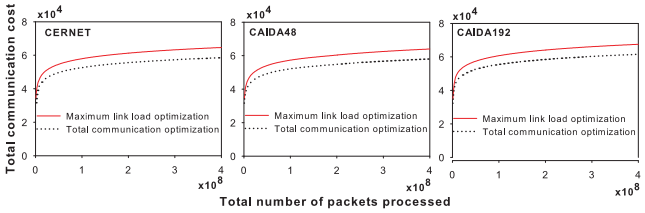


FIGURE 5. Total communication cost over time with a uniform distribution ($\epsilon = 0.01$).

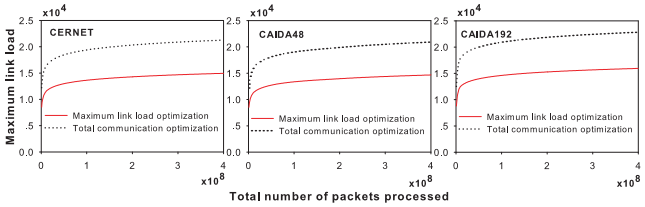


FIGURE 6. Maximum link load over time with a uniform distribution ($\epsilon = 0.006$).

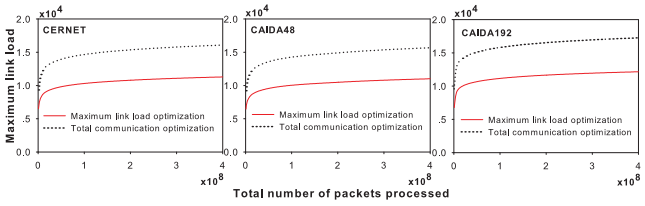


FIGURE 7. Maximum link load over time with a uniform distribution ($\epsilon = 0.008$).

Figures 6, 7 and 8 show how maximum link loads accumulate over time with error bounds 0.006, 0.008 and 0.01. Just as expected, in each figure the maximum link load of the maximum link load optimization method is much less than that of the total communication optimization method. The maximum link load decreases rapidly with the error bound, i.e., Figure6 ($\epsilon = 0.006$) presents the biggest maximum link load, whereas Figure8 ($\epsilon = 0.01$) presents the smallest maximum link load. This is because that the maximum link load has a strong reverse relation with the delayed update threshold according to Theorem 2. And in general, a greater error bound will result in bigger delayed update thresholds. Therefore, the maximum link load has a strong reverse relation with the error bound. At last, just as the total communication cost, a sudden spike in maximum link load is exhibited followed by a gradual increase in each figure. This is because that each node's message sending rate has a strong reverse relation with the time according to Theorem 3.

In all the above 6 figures, both the total communication cost and the maximum link load increase rapidly at first, however keep almost steady as time goes on. These results demonstrate the requirements for an initialization phase. In fact, extending the initialization phase to cover the first 107 packets in our experiments will drastically reduce the total communication cost and the maximum link load

B. RANDOMLY DISTRIBUTED PACKETS

Previous results were based on selecting which node to update uniformly. Here we explore the effect of using random distributions to update different nodes.

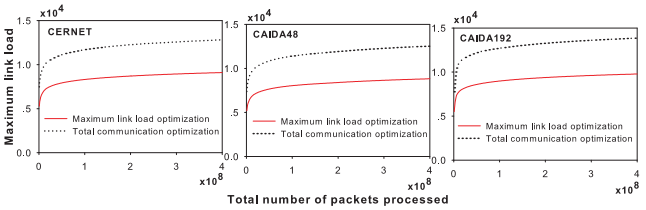


FIGURE 8. Maximum link load over time with a uniform distribution ($\epsilon = 0.01$).

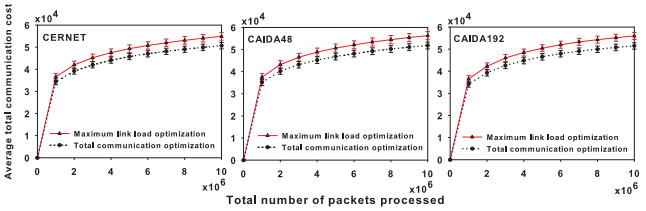


FIGURE 9. Average total communication cost over time with random distributions ($\epsilon = 0.006$).

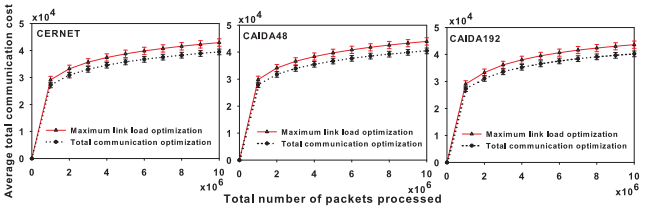


FIGURE 10. Average total communication cost over time with random distributions ($\epsilon = 0.008$).

Figures 9, 10 and 11 show the average total communication cost over 500 repetitions with random incoming packet distributions ($\epsilon = 0.006, 0.008, 0.01$). The error bars in the figures represent the standard deviations of the total communication cost generated by the 500 simulation runs.

Figures 12, 13 and 14 show the average maximum link loads over 500 repetitions with random incoming packet distributions ($\epsilon = 0.006, 0.008, 0.01$). The error bars in the figures represent the standard deviations of the maximum

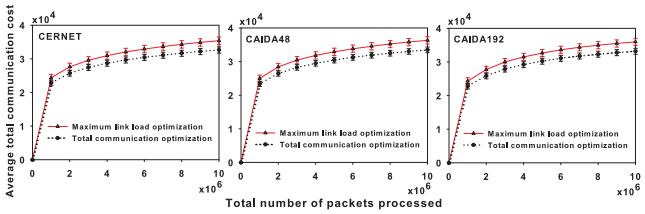


FIGURE 11. Average total communication cost over time with random distributions ($\epsilon = 0.01$).

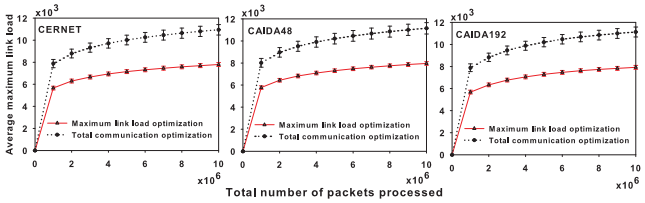


FIGURE 12. Average maximum link load over time with random distributions ($\epsilon = 0.006$).

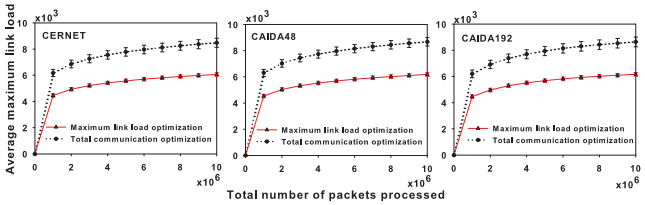


FIGURE 13. Average maximum link load over time with random distributions ($\epsilon = 0.008$).

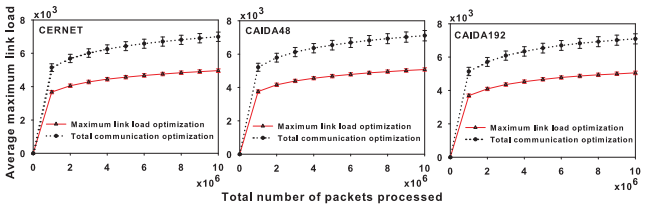


FIGURE 14. Average maximum link load over time with random distributions ($\epsilon = 0.01$).

link loads generated by the 500 simulation runs. From all the 6 figures we can see that the effect of using random distributions is relatively small.

VII. CONCLUSION

We have given a method to efficiently identify global frequent items in distributed data streams in a continuous fashion, which is a fundamental problem in many network and sensor monitoring fields.

In this paper, we consider normal and extended communication structures to investigate continuously tracking the distributed frequent items in order to continuously report the frequent items in the root node.

we propose several algorithms running respectively on the monitoring nodes, intermediate nodes and the root node to maintain each item's estimated frequency as well as the estimated total frequency. The proposed algorithms make use of the delayed updating concept to significantly reduce the communication overhead. In addition, we analyze the theoretical complexity and propose two parameter optimization approaches about a low total communication cost and a small maximum link load, respectively. Last, Experiments with real traffic traces confirm the theoretical analyses and demonstrate the performances of the two optimization approaches. We believe that the communication overhead can be further reduced with randomized algorithms. We will work on this direction in the future.

REFERENCES

[1] T. Qiu, X. Wang, C. Chen, M. Atiquzzaman, and L. Liu, "TMED: A spider-Web-like transmission mechanism for emergency data in vehicular ad hoc networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 9, pp. 8682–8694, Sep. 2018.

[2] Z. Tian, W. Shi, Y. Wang, C. Zhu, X. Du, S. Su, Y. Sun, and N. Guizani, "Real time lateral movement detection based on evidence reasoning network for edge computing environment," *IEEE Trans. Ind. Informat.*, to be published. doi: 10.1109/TII.2019.2907754.

[3] Z. Tian, S. Su, W. Shi, X. Du, M. Guizani, and X. Yu, "A data-driven method for future Internet route decision modeling," *Future Gener. Comput. Syst.*, vol. 95, pp. 212–220, Jun. 2019.

[4] Q. Tan, G. Yue, J. Shi, X. Wang, B. Fang, and Z. Tian, "Toward a comprehensive insight into the eclipse attacks of tor hidden services," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1584–1593, Apr. 2019. doi: 10.1109/JIOT.2018.2846624.

[5] M. Li, Y. Sun, Y. Jiang, and Z. Tian, "Answering the min-cost quality-aware query on multi-sources in sensor-cloud systems," *Sensors*, vol. 18, no. 12, p. 4486, 2018.

[6] Z. Tian, M. Li, M. Qiu, Y. Sun, and S. Su, "Block-DEF: A secure digital evidence framework using blockchain," *Inf. Sci.*, vol. 491, pp. 151–165, Jul. 2019.

[7] T. Qiu, H. Wang, K. Li, H. Ning, A. K. Sangaiah, and B. Chen, "SIGMM: A novel machine learning algorithm for spammer identification in industrial mobile cloud computing," *IEEE Trans. Ind. Informat.*, vol. 15, no. 4, pp. 2349–2359, Apr. 2019. doi: 10.1109/TII.2018.2799907.

[8] B. Lahiri and S. Tirthapura, "Identifying frequent items in a network using gossip," *J. Parallel Distrib. Comput.*, vol. 70, no. 12, pp. 1241–1253, 2010.

[9] K. Inoue, T.-T. Hoan, and C.-K. Pham, "Frequent items counter based on binary decoders," *IEICE Electron. Express*, vol. 15, no. 20, 2018, Art. no. 20180808.

[10] S. Galea, A. W. Moore, and G. Antichi, "Revealing hidden hierarchical heavy hitters in network traffic," in *Proc. ACM SIGCOMM Conf. Posters Demos*, 2018, pp. 81–83.

[11] B. He and and J. Pei, "Research on mining global maximal frequent itemsets for health big data," in *Proc. IEEE 3rd Inf. Technol. Mechatronics Eng. Conf. (ITOEC)*, Oct. 2017, pp. 1143–1146.

[12] M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams," *Theor. Comput. Sci.*, vol. 312, no. 1, pp. 3–15, 2004.

[13] P. Bose, E. Kranakis, P. Morin, and Y. Tang, "Bounds for frequency estimation of packet streams," in *Proc. SIROCCO*, 2003, pp. 33–42.

[14] R. M. Karp, S. Shenker, and C. H. Papadimitriou, "A simple algorithm for finding frequent elements in streams and bags," *ACM Trans. Database Syst.*, vol. 28, no. 1, pp. 51–55, Mar. 2003.

[15] E. D. Demaine, A. López-Ortiz, and J. I. Munro, "Frequency estimation of Internet packet streams with limited space," in *Proc. Eur. Symp. Algorithms*, in Lecture Notes in Computer Science, Sep. 2002, pp. 348–360.

[16] J. Misra and D. Gries, "Finding repeated elements," *Sci. Comput. Program.*, vol. 2, no. 2, pp. 143–152, 1982.

[17] Y. Tong, X. Zhang, and L. Chen, "Tracking frequent items over distributed probabilistic data," *World Wide Web*, vol. 19, no. 4, pp. 579–604, Jul. 2016.

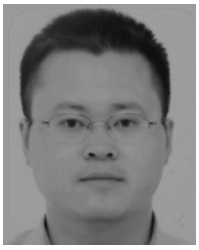
[18] T. Qiu, R. Qiao, and D. O. Wu, "EABS: An event-aware backpressure scheduling scheme for emergency Internet of Things," *IEEE Trans. Mobile Comput.*, vol. 17, no. 1, pp. 72–84, Jan. 2018.

[19] A. Manjhi, V. Shkapenyuk, K. Dhamdhere, and C. Olston, "Finding (recently) frequent items in distributed data streams," in *Proc. 21st IEEE Comput. Soc. Int. Conf. Data Eng. (ICDE)*, Washington, DC, USA, vol. 1, no. 21, Apr. 2005, pp. 767–778.

[20] A. Manjhi, S. Nath, and P. B. Gibbons, "Tributaries and deltas: Efficient and robust aggregation in sensor network streams," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2005, pp. 287–298.
[21] G. Cormode, M. Garofalakis, S. Muthukrishnan, and R. Rastogi, "Holistic aggregates in a networked world: Distributed tracking of approximate quantiles," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, New York, NY, USA, 2005, pp. 25–36. doi: 10.1145/1066157.1066161.
[22] B. Babcock and C. Olston, "Distributed top-k monitoring," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, New York, NY, USA, 2003, pp. 28–39.
[23] R. Fuller and M. Kantardzic, "FIDS: Monitoring frequent items over distributed data streams," in *Proc. 5th Int. Conf. Mach. Learn. Data Mining Pattern Recognit. (MLDM)*. Berlin, Germany: Springer-Verlag, 2007, pp. 464–478. doi: 10.1007/978-3-540-73499-4.
[24] R. Fuller and M. Kantardzic, "Distributed monitoring of frequent items," *Trans. Mach. Learn. Data Mining*, vol. 1, no. 2, pp. 67–82, Oct. 2008.
[25] K. Yi and Q. Zhang, "Optimal tracking of distributed heavy hitters and quantiles," in *Proc. 28th ACM SIGMOD-SIGACTSIGART Symp. Princ. Database Syst.*, New York, NY, USA, 2009, pp. 167–174.
[26] Z. Huang, K. Yi, Y. Liu, and G. Chen, "Optimal sampling algorithms for frequency estimation in distributed data," in *Proc. IEEE INFOCOM*, Shanghai, China, Apr. 2011, pp. 1997–2005.
[27] Z. Huang, K. Yi, and Q. Zhang, "Randomized algorithms for tracking distributed count, frequencies, and ranks," in *Proc. 31st Symp. Princ. Database Syst. (PODS)*, New York, NY, USA, 2012, pp. 295–306.
[28] Q. G. Zhao, M. Ogihara, H. Wang, and J. Xu, "Finding global icebergs over distributed data sets," in *Proc. 25th ACM SIGMOD-SIGACT-SIGART Symp. Princ. Database Syst. (PODS)*, New York, NY, USA, 2006, pp. 298–307.
[29] H. Zhao, A. Lall, M. Ogihara, and J. Xu, "Global iceberg detection over distributed data streams," in *Proc. IEEE 26th Int. Conf. Data Eng. (ICDE)*, Long Beach, CA, USA, Mar. 2010, pp. 557–568.
[30] R. Keralapura, G. Cormode, and J. Ramamirtham, "Communication-efficient distributed monitoring of thresholded counts," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, New York, NY, USA, 2006, pp. 289–300.
[31] Y. Zhang, Y. Sun, J. Zhang, J. Xu, and Y. Wu, "An efficient framework for parallel and continuous frequent item monitoring," *Concurrency Comput., Pract. Exper.*, vol. 26, no. 18, pp. 2856–2879, Dec. 2014.
[32] *CAIDA OC48 Peering Point Traces Dataset*. Accessed: May 21, 2018. [Online]. Available: http://www.caida.org/data/passive/passive_oc48_dataset.xml
[33] *The CAIDA UCSD Anonymized Internet Traces 2008*. Accessed: May 21, 2018. [Online]. Available: http://www.caida.org/data/passive/passive_2008_dataset.xml

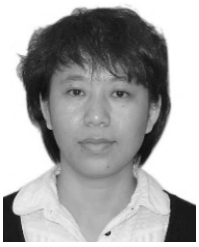
SHUZHUANG ZHANG (M'82) received the B.S. and M.S. degrees in computer science and technology from Yanshan University, in 2007, and the Ph.D. degree in information security from the Harbin Institute of Technology, Harbin, China, in 2011.

He is currently a Lecturer with the Institute of Network Technology, Beijing University of Posts and Telecommunications. His primary research interests include network security and information content security.



YU ZHANG (M'81) received the B.S., M.S., and Ph.D. degrees in information security from the Harbin Institute of Technology, Harbin, China, in 2010.

He is currently an Assistant Professor with the College of Computer and Control Engineering, Nankai University, Tianjin, China. His research interests mainly include data mining and network security.



LIHUA YIN was born in 1973. She received the Ph.D. degree from Harbin Institute of Technology, Harbin, China. She is currently a Professor and a Ph.D. Supervisor with Guangzhou University. Her current research interests include computer networks and network security. She is a member of the China Computer Federation.

TINGTING YUAN (M'81) received the Ph.D. degree from the Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2018.

She currently holds a postdoctoral position with the Diana Team, Inria, France. Her current research interests include artificial intelligence, software-defined networking, vehicle networks, and so on.

ZHIGANG WU was born in 1972. He received the B.S., M.S., and Ph.D. degrees in computer system architecture from the Harbin Institute of Technology, Harbin, China, in 2000.

He is currently an Associate Professor with the Institute of Network Technology, Beijing University of Posts and Telecommunications. His research interest includes network security.

HAO LUO was born in 1979. He received the B.S., M.S., and Ph.D. degrees in computer system architecture from the Harbin Institute of Technology, Harbin, China, in 2006.

He is currently an Associate Professor with the Institute of Network Technology, Beijing University of Posts and Telecommunications. His research interest includes network security.