Online Traffic Classification Based on Swarm Intelligence

Takumi Sue, Yuichi Ohsita, and Masayuki Murata
Graduate School of Information Science and Technology, Osaka University
Osaka, Japan
{t-sue, y-ohsita, murata}@ist.osaka-u.ac.jp

***Abstract*—In this paper, we propose a new traffic classification method which constructs hierarchical clusters using the features of uncompleted flows. By constructing the hierarchical groups, we can identify the similarity of the groups. If flows of a new application construct a new group in the lower layer, but they are classified in an existing group in the upper layer, the manager can estimate the characteristic of the new application from the characteristic of the existing group. In our method, the hierarchical groups are constructed based on the clustering method called AntTree; the each flow moves over the tree and find the nodes whose similarity to the nodes exceeds the threshold. By setting the threshold based on the number of monitored packets of the flow, we classify the flow if the features of the flow become sufficiently accurate. Otherwise we wait another packets that improve the accuracy of the features.**

Keywords*—*Traffic Classification; Hierarchical Clustering; Swarm Intelligence

I. Introduction

As the Internet has become playing an important role in our society, the number of types of services provided through the Internet increases. The requirements to the network depend on the type of the service. The video streaming application requires enough bandwidth according to the bit rate of the video. On the other hand, the interactive application such as online game requires the communication with low latency instead of the large bandwidth.

The network managers should manage their network so as to provide sufficient network performance required by each service. For example, Miyamura et al. [1] proposed a method that constructs a virtual network for each service. In this method, each virtual network is dynamically reconfigured so as to provide a performance required by the service corresponding to the network. To manage the network based on the types of the service, we need to classify the traffic based on the application.

The traditional classification of the traffic uses the port numbers [2]. However, in recent years, a large number of types of the applications, such as YouTube and network game, have become provided through HTTP [3]. All of these applications uses 80 or 443 port. As a result, the traffic classification using the port numbers is no longer applicable in the Internet.

Therefore, the traffic classification methods based on the features of the traffic have been proposed [2], [4]–[6]. The packet sizes and packet arrival intervals depend on the types of the application, and the protocol used by the application. The traffic classification methods based on the features monitors the packet size or packet arrival intervals for each flow. Then, they classify the flows based on the monitored features by using the clustering methods, in which the flows are grouped so that the flows with the similar features belongs to the same group.

The traffic classification should be performed as soon as possible after the flow arrives. Even if the network manager sets the rule to relay the flow according to the types of the application, the rule cannot work before the classification of the flow is completed. The existing method, however, cannot classify the flow before monitoring the flow is completed. One approach is to classify the flow after the predefined number of packets are monitored. By setting the required number of packets to the small value, we can classify the flow soon after the flow arrives. However, the features of the flow obtained by monitoring the small number of packets may be inaccurate, and some application may be difficult to classify based on such inaccurate information.

Another problem in the existing traffic classification is that the group constructed by the classification methods does not imply the characteristic of the group. When flows of a new application comes, the flows are classified into a new group. However, the existing classification methods do not provide the information whether the newly constructed group has the similar characteristic to the existing other groups. As a result, it is difficult to estimate the characteristic of the new application.

In this paper, we propose a new traffic classification method to solve the above problems. Our method is based on the hierarchical clustering [7]. In the hierarchical clustering, the groups of the flows are hierarchically constructed; the flows are clustered into a small number of groups in the upper layer, and the flows belonging to the same group in the upper layers are clustered into several groups in the lower layer. By constructing the hierarchical groups, we can identify the similarity of the groups. If flows of a new application construct a new group in the lower layer, but they are classified in an existing group in the upper layer, the manager can estimate the characteristic of the new application from the characteristic of the existing group.

In our method, the hierarchical groups are constructed based on the clustering method called AntTree [8]. AntTree is inspired by the behavior of ants constructing the tree. In the AntTree, an ant, which corresponds to an item required to be classified, walks on the tree constructed by the other ants. Then, if the ant find the ant whose similarity exceeds the threshold, the ant is connected to the similar ant. If the nearby ants do not have the similarity exceeding the threshold, the ant updates the threshold and goes to another place on the tree. By continuing this process, the hierarchical tree, where similar ants are connected, is constructed.

In our method, we extend AntTree to consider the accuracy of the features. The features of the flow become accurate as the number of monitored packets increases. We classify the flow if the features of the flow become sufficiently accurate. Otherwise we wait another packets that improve the accuracy of the features. To achieve this, we extend the AntTree by setting the threshold of the similarity to connect the ants considering the number of monitored packets. By doing so, if the quite similar group to the newly arrived flow exists, the flow is classified soon after the flow starts. Otherwise, our method waits another packet to improve the accuracy, and the flow is classified after the sufficient packets are monitored.

The rest of this paper is organized as follows. Section II explains the related work. Section III explains our online hierarchical traffic classification method. In Section IV , we conduct an experiment with real traffic data. The conclusion and future work are mentioned in Section V.

II. Related Work

This section explains the related work.

A. *Traffic Classification*

There are several papers proposing a method to classify the traffic using the features of the monitored traffic.

Roughan et al. proposed a method to classify the traffic through the supervised machine learning [4]. In this method, the newly obtained data is classified as the class of its nearest neighbor from the training data set. Zhang et al. improved the accuracy of the nearest neighbor approach when the number of the training data set is small by incorporating the correlation information of the flows [9]. Moore et al. also proposed a method to classify the traffic based on the supervised machine learning [5]. In this method, the traffic is classified by using the Naïve Bayes classifier. Nguyen et al. also used the Naïve Bayes to classify the traffic [10]. This method uses the features of a small number of most recent packets to obtain the features. Then, applying the Naïve Bayes, this method classify the current flows. Zhang et al. also used the Naïve Bayes classifier [11]. In this method, flow correlation information is modeled by bag-of-flow. Then, the features are extracted for the represent traffic flows. The traffic is classified by aggregating the output of the Naïve Bayes classifiers using the extracted features. Li et al. proposed a method to construct the decision tree from the training data set [12]. Then the traffic is classified based on the constructed decision tree. Jin et al. proposed a traffic classification method using multiple simple linear binary classifiers [13]. In this method, each classifier can be easily trained. Then, combining the multiple classifiers, we can accurately classify the traffic.

The supervised machine learning methods described above require the training data set. However, as we discussed in Section I, it is difficult to prepare the training data set including the suitable labels for the traffic, because the new applications, which are unknown when the system to classify the traffic starts, emerge.

The methods to classify the traffic based on the clustering have also been proposed.

Erman et al. applied the clustering method to the traffic classification [14]. They used the K-means method, DBSCAN, and AutoClass, and demonstrated that these clustering algorithms can classify the traffic accurately. Bernaille et al. proposed a method to classify the traffic online using the K-means method [6]. In this method, the clusters are constructed offline by using the training data set. Then, flows are classified online by searching the cluster corresponding to the flow. However, this approach cannot classify the traffic which corresponds to the application, which was not included in the training data set. The clustering method can be used to solve this problem. Zhang et al used the K-means method to detect the unknown flows [15].

Recently, Wang et al. extended the K-means method to improve the accuracy of the traffic classification [16]. In this method, the accuracy is improved by considering the information of the flow inferred from the packet headers as the constraint on the clustering.

However, the existing traffic classification methods based on the clustering have the following two problems. (1) These methods do not consider the case that the new applications emerges. Though the method proposed by Zhang et al [15] can detect the unknown flows, it cannot estimate the characteristic of the new flow. (2) These methods assume that the accurate features of the flow are obtained before classifying the traffic. However, the accurate features may not be able to be obtained before the flow is completed.

In this paper, we propose a clustering method which solves the above problems. Our clustering method can be applicable to the existing traffic classification method based on the clustering; our clustering method can be run by using any features of the flows.

B. *Clustering*

There are many algorithms to construct the clusters.

K-means is one of the most popular clustering algorithms. In the K-means method, the number of clusters k is given as a parameter. Then, the k clusters are constructed so as to minimize the distance from each data point to the center of the cluster the data point belongs to, which is defined by the mean of the data points within the cluster. However, the result of the K-means only indicates the cluster which each data point belongs to. Thus, we cannot understand the similarity of the data points belonging to different clusters.

The hierarchical clustering methods can solve the above problem. In the hierarchical clustering methods, the data points are clustered into a small number of groups in the upper layer, and the data points belonging to the same group in the upper layers are clustered into several groups in the lower layer. By constructing the hierarchical groups, we can identify the similarity of the groups.

The ClusTree is one of the hierarchical clustering methods, which allow to update the clusters online [7]. This method constructs the tree, including two kinds of nodes, inner node and leaf node. The leaf node indicates a fine-grained cluster, and has the pointer to the feature values of the corresponding data points. The inner node corresponds to the coarse-grained cluster, which includes the data points included in its children nodes. The inner node has the pointer to the aggregated features of the data points included in its children nodes. In the ClusTree, the clusters can be updated by (1) searching the leaf node corresponding to the new data point from the root node, and (2) updating the features on the path from the root node to the leaf node.

The ClusTree can be updated online, but does not consider the case that the features are inaccurate. Therefore, in this paper, we propose a new clustering method based on the ClusTree, considering the case that the features are inaccurate.

AntTree [8] is another hierarchical clustering method. AntTree is a method inspired by the behavior of the ants constructing a tree. In this process, each data point acts as an ant; each data point walks around the tree to find the node similar to the flow, and connects it to the found node. In the AntTree, the nodes in the constructed tree are the data points. Therefore, it is difficult to interpret the constructed tree hierarchically; we need to determine the data points corresponding to the inner nodes when constructing the fine-grained flow. However, the idea of the AntTree is useful to handle the inaccuracy of the features. In the AntTree, each data point has a threshold to determine whether the other data points are similar to it. Then, each data point walks around the tree based on the threshold. Though the original AntTree updates the threshold based on the number of nodes the data points visited, we can consider the inaccuracy of the features by setting the threshold based on the accuracy of the features. Therefore, in this paper, we introduce the method based on the AntTree to search the cluster each data point belongs to.

III. Online hierarchical traffic classification method

This section explains our online hierarchical traffic classification method.

A. Overview

In this paper, we develop the traffic classifier, which classifies the flow passing the classifier. The flow is defined by the set of packets between the same IP address pairs using the same server port. We regard the well-known ports as the server ports, and the other ports as the client ports. In this paper, the packets using the same server port is regarded as the packets belonging to the same flow even if the packets have the different client port, because some application such as Web browser uses the multiple TCP connections for the same transaction.

The traffic classifier monitors the flows. When the traffic classifier receive a packet, it identifies the flow the packet belongs to. Then, it stores the information of the packet. The traffic classifier updates the features of the flow each time a packet of the flow are monitored.

The traffic classifier classifies the flow based on its features. To classify the flow, we use the hierarchical clustering methods. In the hierarchical clustering, the groups of the flows are hierarchically constructed; the flows are clustered into a small number of groups in the upper layer, and the flows belonging to the same group in the upper layers are clustered into several groups in the lower layer. By constructing the hierarchical groups, we can identify the similarity of the groups.

Each time the features of the flow is updated, the traffic classifier runs the clustering method. That is, the clustering is performed before the flow completed by using the inaccurate features, which leads to wrong classification. Therefore, we use the clustering method considering the accuracy of the features. The accuracy of the features of the flow increases as the number of monitored packets becomes large. Thus, we classify the flow if the features of the flow become sufficiently accurate. Otherwise we wait another packets that improve the accuracy of the features.

To achieve this, we extend the AntTree. In the AntTree, an ant, which corresponds to an item required to be classified, walks on the tree constructed by the other ants. Then, if the ant find the ant whose similarity exceeds the threshold, the ant is connected to the similar ant.

If the nearby ants do not have the similarity exceeding the threshold, the ant updates the threshold and goes to another place on the tree. By continuing this process, the hierarchical tree, where similar ants are connected, is constructed.

In this paper, we set the threshold of the AntTree based on the number of received packets; the flow with a small number of monitored packets is connected to the node only if the very similar node exists. On the other hand, the flow with a large number of monitored packets is easier to be connected.

B. Data structure

In this paper, we construct the hierarchical cluster based on ClusTree [7]. Figure 1 shows the data structure of the constructed cluster.

In this data structure, the feature values of the flows are stored in a table, and updated each time a packet corresponding the flow arrives. We denote the feature value of the flow f as the vector F_f.

This cluster has two kinds of nodes, *inner node* and *leaf node*. The leaf node indicates a fine-grained cluster, and includes l to$L$ flows. Each leaf node has the pointer to the feature values of the corresponding flows.

The inner node corresponds to the coarse-grained cluster, which includes the flows included in its children nodes. That is, by constructing the tree of inner nodes, we can hierarchically construct clusters; the inner node near root node corresponds to the coarser-grained cluster, and the node near leaf corresponds to the finer grained node.

The inner node construct the tree structure by connecting it to$m$ to M children. Each inner node has the entries corresponding to its children. Each entry has the abstracted clustering features and the pointer to the corresponding child.

The abstracted clustering features corresponding to the child node c have the following values.

- The number of flows included in the abstracted features n_c
- The sum of the features S^{linear}, which is calculated by

$$S_c^{\text{linear}} = \sum_{f \in F_c} F_f$$

where F_c is the set of flows included in cluster of the node c.

In our method, there are flows that have not been classified into any clusters. We maintain such flows in a list called *unclassified flows*.

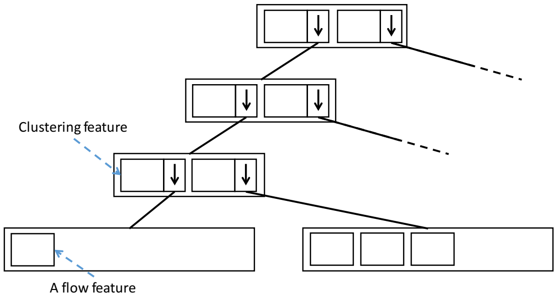


Figure 1. Data structure of the hierarchical cluster

C. Process to update the tree

When the traffic classifier receives a packet, it identifies the flow of the packet, and stores the packets. Each time a packet of the flow arrive, the features of the flow are updated. At the same time, the data structure of the tree is updated.

The process to update the tree depends on whether the flow is already classified into one of the clusters or included in the unclassified flows.

1) Update the data of the flows that is included in the unclassified flows: We denote the flow finding the cluster by f^{new}, the location of f^{new} by $p_{f^{\text{new}}}$, the node whose feature is the most similar to f^{new} among the children of $p_{f^{\text{new}}}$ by $c_{p_{f^{\text{new}}}}$. T_f^{sim} and T_f^{dissim} are the thresholds for the flow f. $Sim(f,c)$ indicates the similarity between the flow f and the node c, which is calculated by

$$Sim(f,c) = 1 - \frac{distance(f,c) - distance_{min}}{distance_{max} - distance_{min}} \qquad (1)$$

where $distance(f,c)$ is defined by

$$distance(f,c) = \left| F_f + \frac{S_n^{\text{linear}}}{n_c} \right|.$$

In this paper, the flow f^{new} moves over the tree by performing the following rule once per one arrival packet.

- If $p_{f^{\text{new}}}$ is the root node
  1) if the root node has no children, make a leaf node and insert the pointer to f^{new} to the newly added node
  2) Otherwise, go to $c_{p_{f^{\text{new}}}}$
- If $p_{f^{\text{new}}}$ is not the root node
  1) If $Sim(f^{\text{new}}, c_{p_{f^{\text{new}}}}) \geq T_{f^{\text{new}}}^{\text{sim}}$,
     a) Go to $c_{p_{f^{\text{new}}}}$
     b) If $c_{p_{f^{\text{new}}}}$ is a leaf node,
        i) Insert the pointer to f^{new} to $c_{p_{f^{\text{new}}}}$
        ii) Update the abstracted clustering features of the ancestors a of $c_{p_{f^{\text{new}}}}$ by adding F_f to S_a^{linear} and 1 to n_a
        iii) If there exists a node whose number of children exceeds the upper limit (L or M), add new a node
  2) If $Sim(f^{\text{new}}, c_{p_{f^{\text{new}}}}) < T_{f^{\text{new}}}^{\text{sim}}$
     a) $Sim(f^{\text{new}}, c_{p_{f^{\text{new}}}}) < T_{f^{\text{new}}}^{\text{dissim}}$, go back to the parent node of $c_{p_{f^{\text{new}}}}$
     b) Otherwise, stay at $c_{p_{f^{\text{new}}}}$

In the above steps, $T_{f^{\text{new}}}^{\text{sim}}$ and $T_{f^{\text{new}}}^{\text{dissim}}$ is updated by

$$T_{sim}(a_i) = T_{sim}(a_i) \times \alpha_1, \qquad (2)$$

and

$$T_{dissim}(a_i) = T_{dissim}(a_i) \times \alpha_2, \qquad (3)$$

where α_1 and α_2 are the parameters.

Addition of the new node in the above steps is done by the following steps. First, we calculate $\sum_{c \in C_a} \frac{S_n^{\text{linear}}}{n_c}$ where a is a node whose number of entries exceeds the upper limit, and C_a is a set of the children of a. Then, we select the $c^{\max}$ whose $\frac{S_n^{\text{linear}}}{n_{c^{\max}}}$ is the most different from $\sum_{c \in C_a} \frac{S_n^{\text{linear}}}{n_c}$. Finally, we remove the entry for $c^{\max}$ from a and add the node including the entry for $c^{\max}$. The parent of the newly added node is set to the parent of a. If the parent of a also has more entries than the upper limit after the above process, we perform the same process for the parent again.

IV. Experiment

In this paper, we used our traffic classifier to classify the flows from one computer in our laboratory, where 43 flows are monitored. The computer accessed Web servers, an Exchange server, and so on through the 80 or 443 port. To classify the flows, we use the features shown in Table I. In this features, we define the downstream packets as the packets from the servers whose port number is a well-known port, and the upstream packets as the packets to the servers.

In this experiment, we set the initial values of $T_{sim}(a_i)$ and $T_{dissim}(a_i)$ to 1.0. We set α_1 and α_2 to 0.7. We started the classification after 5 packets per flow were received. The features of each flow were updated until more than 100 packets of the flow were received. We stopped updating the features after 100 packets were received, because the features do not change significantly after a sufficient number of packets are monitored.

Figure 2 shows the tree constructed by our classification. Table II shows the features of the flow grouped by the cluster in the lowest layer. Tables III and IV show the abstracted clustering features divided by the number of flows included in

TABLE I
FEATURES USED IN OUR EXPERIMENT

Name	Description
$E(s^{\mathrm{down}})$	Average of the size of the downstream packets
$E(s^{\mathrm{up}})$	Average of the size of the upstream packets
$\sigma(s^{\mathrm{down}})$	Standard deviation of the size of the downstream packets
$\sigma(s^{\mathrm{up}})$	Standard deviation of the size of the upstream packets
$E(i^{\mathrm{down}})$	Average of the interval of the arrival of the downstream packets
$E(i^{\mathrm{up}})$	Average of the interval of the arrival of the upstream packets
$\sigma(i^{\mathrm{down}})$	Standard deviation of the interval of the arrival of the downstream packets
$\sigma(i^{\mathrm{down}})$	Standard deviation of the interval of the arrival of the upstream packets

the abstracted clustering features. To illustrate the result of the clustering, we plot the scatter graph of the clustering features of each flow. In Figures 3 and 4, we colored the flows based on the group constructed in the 2nd layer and the 3rd layer of the tree, respectively.

These figures show that our clustering method can group flows into clusters so that the flows with the similar feature are included in the same cluster, in any layers. That is, our method can classify the flows even before the flow is completed.

These figure also indicates that the constructed clusters are mainly based on the packet sizes, and packet arrival interval does not have a large impact on the cluster.

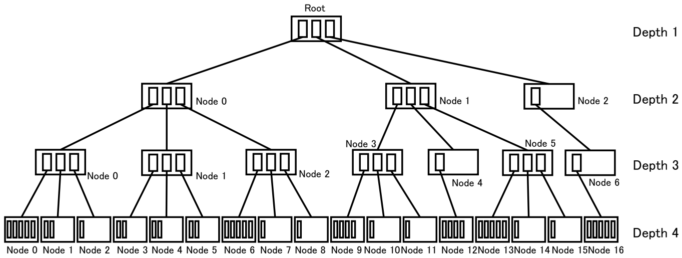


Figure 2. Tree constructed by our classification

V. Conclusion

In this paper, we proposed a new traffic classification method that construct hierarchical groups of the similar flows. Through the experiment, we demonstrated that our classification method enables grouping similar flows into the same clusters. That is, our method can classify the flows even before the flow is completed. Results also indicates that the constructed clusters are mainly based on the packet sizes, and packet arrival interval does not have a large impact on the cluster.

Our future work includes further verification of our method using larger traffic data and discussion on more appropriate features calculated from packet information.

References

[1] Takashi Miyamura, Yuichi Ohsita, Shin'ichi Arakawa, Yuki Koizumi, Akeo Masuda, Kohei Shiomoto, and Masayuki Murata, "Network virtualization server for adaptive network control," in *Proceedings of 20th ITC Specialist Seminar on Network Virtualization - Concept and Performance Aspects*, May 2009.

[2] A. Callado, C. Kamienski, G. Szabo, B. P. Gero, J. Kelner, S. Fernandes, and D. Sadok, "A survey on internet traffic identification," *Communications Surveys & Tutorials, IEEE*, vol. 11, pp. 37–52, Aug. 2009.

[3] L. Popa, A. Ghodsi, and I. Stoica, "Http as the narrow waist of the future internet," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, (New York, NY, USA), pp. 6:1–6:6, ACM, 2010.

[4] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, "Class-of-service mapping for QoS: A statistical signature-based approach to IP traffic classification," in *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, IMC '04, (New York, NY, USA), pp. 135–148, ACM, 2004.

[5] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, pp. 50–60, June 2005.

[6] L. Bernaille, R. Teixeira, and K. Salamatian, "Early application identification," in *Proceedings of the 2006 ACM CoNEXT Conference*, CoNEXT '06, (New York, NY, USA), pp. 6:1–6:12, ACM, 2006.

[7] P. Kranen, I. Assent, C. Baldauf, and T. Seidl, "The ClusTree: indexing micro-clusters for anytime stream mining," *Knowledge and Information Systems*, vol. 29, no. 2, pp. 249–272, 2011.

[8] H. Azzag, N. Monmarche, M. Slimane, and G. Venturini, "AntTree: a new model for clustering with artificial ants," *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, vol. 4, pp. 2642–2647, Dec. 2003.

[9] J. Zhang, Y. Xiang, Y. Wang, W. Zhou, Y. Xiang, and Y. Guan, "Network traffic classification using correlation information," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 24, pp. 104–117, Jan 2013.

[10] T. T. T. Nguyen, G. Armitage, P. Branch, and S. Zander, "Timely and continuous machine-learning-based classification for interactive ip traffic," *IEEE/ACM Trans. Netw.*, vol. 20, pp. 1880–1894, Dec. 2012.

[11] J. Zhang, C. Chen, Y. Xiang, W. Zhou, and Y. Xiang, "Internet traffic classification by aggregating correlated naive bayes predictions," *Information Forensics and Security, IEEE Transactions on*, vol. 8, pp. 5–15, Jan 2013.

[12] W. Li, M. Canini, A. W. Moore, and R. Bolla, "Efficient application identification and the temporal and spatial stability of classification schema," *Comput. Netw.*, vol. 53, pp. 790–809, Apr. 2009.

[13] Y. Jin, N. Duffield, J. Erman, P. Haffner, S. Sen, and Z.-L. Zhang, "A modular machine learning system for flow-level traffic classification in large networks," *ACM Trans. Knowl. Discov. Data*, vol. 6, pp. 4:1–4:34, Mar. 2012.

[14] J. Erman, M. Arlitt, and A. Mahanti, "Traffic classification using clustering algorithms," in *Proceedings of the 2006 SIGCOMM workshop on Mining network data*, pp. 281–286, 2006.

[15] J. Zhang, C. Chen, Y. Xiang, W. Zhou, and A. Vasilakos, "An effective network traffic classification method with unknown flow detection," *Network and Service Management, IEEE Transactions on*, vol. 10, pp. 133–147, June 2013.

[16] Y. Wang, Y. Xiang, J. Zhang, W. Zhou, G. Wei, and L. Yang, "Internet traffic classification using constrained clustering," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 25, pp. 2932–2943, Nov 2014.

TABLE II

FEATURES OF THE FLOWS: DEPTH 4

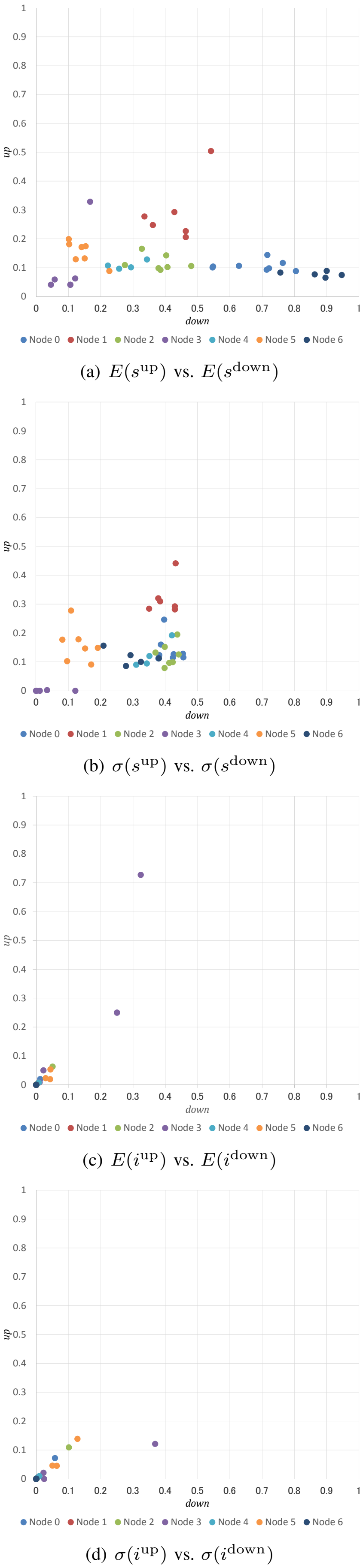
flow number	$E(s^{\mathrm{down}})$	$E(s^{\mathrm{up}})$	$\sigma(s^{\mathrm{down}})$	$\sigma(s^{\mathrm{up}})$	$E(i^{\mathrm{down}})$	$E(i^{\mathrm{up}})$	$\sigma(i^{\mathrm{down}})$	$\sigma(i^{\mathrm{up}})$
Node 0								
1	0.72084	0.0982946	0.42354	0.114737	1.01073e-005	1.51213e-005	3.37173e-005	3.97892e-005
2	0.627727	0.106697	0.426203	0.126686	5.652e-005	0.000123704	9.43008e-005	0.000106826
3	0.713851	0.0930637	0.381509	0.123607	0.000609879	0.000972389	0.00136518	0.00154726
4	0.804338	0.0884975	0.378517	0.112426	2.81074e-005	6.92139e-005	6.58568e-005	8.45793e-005
5	0.763775	0.116724	0.386436	0.159844	0.0123994	0.0198522	0.0582105	0.072384
Node 1								
6	0.546043	0.101218	0.4561	0.115731	5.26767e-005	0.000108162	8.91574e-005	8.57252e-005
7	0.547945	0.104547	0.454592	0.128433	0.000902717	0.00128282	0.00165487	0.00184167
Node 2								
8	0.715821	0.144243	0.396433	0.246516	2.88237e-005	3.63364e-005	8.34268e-005	9.56236e-005
Node 3								
9	0.427973	0.293015	0.349902	0.284289	3.13866e-005	3.08145e-005	5.17207e-005	4.05901e-005
10	0.541256	0.504292	0.431807	0.441602	0.0011992	0.000964186	0.00186101	0.00159072
Node 4								
11	0.361422	0.248002	0.384193	0.309865	0.000404118	0.000285903	0.000602977	0.000354651
12	0.335109	0.277778	0.378048	0.320403	0.000174889	0.000145212	0.000162109	0.000161237
Node 5								
13	0.463263	0.20624	0.429613	0.281791	0.00074491	0.000975223	0.00158266	0.0018436
14	0.463263	0.226884	0.429613	0.292398	0.000326853	0.000513898	0.000358468	0.000505234
Node 6								
15	0.378691	0.0986175	0.412466	0.0966242	9.41781e-005	7.83116e-005	0.000343073	0.000214059
16	0.384721	0.0927376	0.397592	0.0789838	1.71589e-005	1.75337e-005	5.07459e-005	4.32654e-005
17	0.32715	0.165906	0.398255	0.151882	0.00278449	0.00313977	0.00231127	0.00203568
18	0.406562	0.102182	0.422961	0.0993938	0.000132655	0.000113818	0.000257254	0.000232465
19	0.402588	0.143075	0.437178	0.194868	6.77042e-005	0.00115388	0.00010484	0.0017407
Node 7								
20	0.274734	0.109589	0.369642	0.132467	0.000220875	0.000434846	0.000330604	0.00039426
Node 8								
21	0.479959	0.10624	0.441488	0.125773	0.0506274	0.0633271	0.101089	0.109396
Node 9								
22	0.0570776	0.0593607	0.0114155	0	0.0222335	0.0500198	0.0248326	6.13253e-006
23	0.105784	0.0410959	0	0	0.250259	0.250248	3.545e-006	6.00934e-005
24	0.105784	0.0410959	0	0	0.250257	0.250257	7.90377e-006	0.000219129
25	0.167047	0.328767	0.121385	0	0.323694	0.727734	0.36831	0.121438
Node 10								
26	0.0456621	0.0410959	0	0	0.0105705	0.00906027	0.022708	0.0213338
Node 11								
27	0.1207	0.0628615	0.0339138	0.0021309	0.00817775	0.00833702	0.00765219	0.00837687
Node 12								
28	0.293715	0.10136	0.342828	0.0946878	0.000162252	0.000128599	0.000429047	0.000293705
29	0.3431	0.128742	0.420671	0.191927	0.000136547	0.00013721	0.000158327	8.46613e-005
30	0.256722	0.0967783	0.350721	0.120472	0.00795401	0.00795477	0.00984362	0.0097643
31	0.222	0.107827	0.309717	0.0900621	0.000672249	0.000859237	0.00212733	0.00252721
Node 13								
32	0.153349	0.174458	0.190907	0.148768	2.02533e-005	2.86792e-005	5.66575e-005	6.71647e-005
33	0.102055	0.181602	0.0811775	0.176932	9.99759e-005	7.07622e-005	0.000182625	0.000142923
34	0.140665	0.171487	0.131139	0.178177	2.1e-005	1.51033e-005	2.84602e-005	1.38837e-005
35	0.150158	0.132479	0.151779	0.146517	8.32722e-005	8.31819e-005	0.000175796	0.000215069
36	0.100761	0.199391	0.107929	0.277773	0.0292866	0.023471	0.0504462	0.0461822
Node 14								
37	0.122273	0.129427	0.0958468	0.102736	0.044121	0.0536464	0.127781	0.138994
Node 15								
38	0.226636	0.0888128	0.170655	0.0909378	0.0431425	0.0196648	0.0638903	0.0455347
Node 16								
39	0.946356	0.0748792	0.208481	0.156522	3.45042e-005	0.000112505	6.80385e-005	0.000201225
40	0.756059	0.0831219	0.380303	0.112472	0.000272307	0.000536175	0.000593526	0.000963796
41	0.895826	0.0653595	0.278181	0.0857372	7.59293e-005	0.000120305	0.000233297	0.000279578
42	0.862609	0.0769847	0.324408	0.0999886	6.9086e-006	1.35787e-005	1.75598e-005	2.69238e-005
43	0.899784	0.088946	0.292179	0.123184	3.14463e-005	7.06452e-005	5.85814e-005	6.59837e-005

TABLE III
FEATURES OF THE FLOWS: DEPTH 3

flow number	$E(s^{\mathrm{down}})$	$E(s^{\mathrm{up}})$	$\sigma(s^{\mathrm{down}})$	$\sigma(s^{\mathrm{up}})$	$E(i^{\mathrm{down}})$	$E(i^{\mathrm{up}})$	$\sigma(i^{\mathrm{down}})$	$\sigma(i^{\mathrm{up}})$
Node 0								
1-5	0.726106	0.100655	0.399241	0.12746	0.0026208	0.00420653	0.0119539	0.0148325
6-7	0.546994	0.102882	0.455346	0.122082	0.000477697	0.00069549	0.000872014	0.000963698
8	0.715821	0.144243	0.396433	0.246516	2.88237e-005	3.63364e-005	8.34268e-005	9.56236e-005
Node 1								
9-10	0.484614	0.398653	0.390854	0.362946	0.000615291	0.0004975	0.000956365	0.000815653
11-12	0.348266	0.26289	0.38112	0.315134	0.000289504	0.000215557	0.000382543	0.000257944
13-14	0.463263	0.216562	0.429613	0.287094	0.000535882	0.00074456	0.000970564	0.00117442
Node 2								
15-19	0.379942	0.120503	0.41369	0.12435	0.000619237	0.000900662	0.000613436	0.000853234
20	0.274734	0.109589	0.369642	0.132467	0.000220875	0.000434846	0.000330604	0.00039426
21	0.479959	0.10624	0.441488	0.125773	0.0506274	0.0633271	0.101089	0.109396
Node 3								
22-25	0.108923	0.11758	0.0332002	0	0.211611	0.319565	0.0982885	0.0304309
26	0.0456621	0.0410959	0	0	0.0105705	0.00906027	0.022708	0.0213338
27	0.1207	0.0628615	0.0339138	0.0021309	0.00817775	0.00833702	0.00765219	0.00837687
Node 4								
28-31	0.278884	0.108677	0.355984	0.124287	0.00223127	0.00226995	0.00313958	0.00316747
Node 5								
32-36	0.129397	0.171883	0.132586	0.185633	0.00590221	0.00473374	0.0101779	0.00932424
37	0.122273	0.129427	0.0958468	0.102736	0.044121	0.0536464	0.127781	0.138994
38	0.226636	0.0888128	0.170655	0.0909378	0.0431425	0.0196648	0.0638903	0.0455347
Node 6								
39-43	0.872127	0.0778583	0.29671	0.115581	8.42191e-005	0.000170642	0.0001942	0.000307501

TABLE IV
FEATURES OF THE FLOWS: DEPTH 2

flow number	$E(s^{\mathrm{down}})$	$E(s^{\mathrm{up}})$	$\sigma(s^{\mathrm{down}})$	$\sigma(s^{\mathrm{up}})$	$E(i^{\mathrm{down}})$	$E(i^{\mathrm{up}})$	$\sigma(i^{\mathrm{down}})$	$\sigma(i^{\mathrm{up}})$
Node 0								
1-8	0.680042	0.106661	0.412916	0.140997	0.00176102	0.0028075	0.00769962	0.00952319
9-14	0.432048	0.292702	0.400529	0.321725	0.000480226	0.000485873	0.000769824	0.000749338
15-21	0.379201	0.116907	0.411369	0.125713	0.00770635	0.00975218	0.0149268	0.0162938
Node 1								
22-27	0.100342	0.0957128	0.0277857	0.00035515	0.144199	0.215943	0.0705857	0.025239
28-31	0.278884	0.108677	0.355984	0.124287	0.00223127	0.00226995	0.00313958	0.00316747
32-38	0.142271	0.153951	0.132776	0.160263	0.0166821	0.0138543	0.0346516	0.0330214
Node 2								
39-43	0.872127	0.0778583	0.29671	0.115581	8.42191e-005	0.000170642	0.0001942	0.000307501



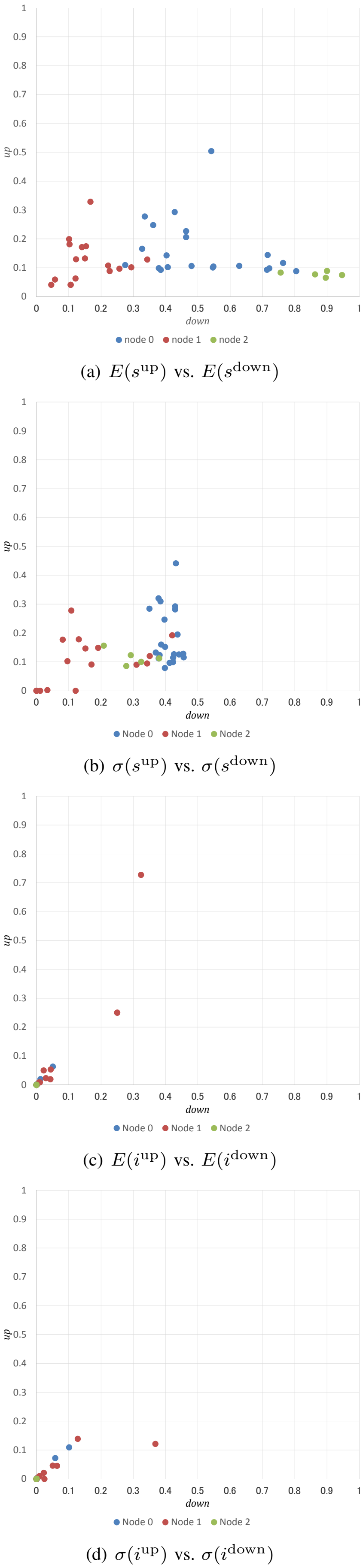
(a) $E(s^{\mathrm{up}})$ vs. $E(s^{\mathrm{down}})$

(b) $\sigma(s^{\mathrm{up}})$ vs. $\sigma(s^{\mathrm{down}})$

(c) $E(i^{\mathrm{up}})$ vs. $E(i^{\mathrm{down}})$

(d) $\sigma(i^{\mathrm{up}})$ vs. $\sigma(i^{\mathrm{down}})$

Figure 3. Features distribution: depth 3



(a) $E(s^{\mathrm{up}})$ vs. $E(s^{\mathrm{down}})$

(b) $\sigma(s^{\mathrm{up}})$ vs. $\sigma(s^{\mathrm{down}})$

(c) $E(i^{\mathrm{up}})$ vs. $E(i^{\mathrm{down}})$

(d) $\sigma(i^{\mathrm{up}})$ vs. $\sigma(i^{\mathrm{down}})$

Figure 4. Features disribution: depth 2