Hierarchical IP flow clustering*

Kamal Shadi
Georgia Institute of Technology, USA
kshadi3@gatech.edu

Preethi Natarajan
Cisco Systems, USA
prenatar@cisco.com

Constantine Dovrolis
Georgia Institute of Technology, USA
constantine@gatech.edu

ABSTRACT

The analysis of flow traces can help to understand a network's usage patterns. We present a hierarchical clustering algorithm for network flow data that can summarize terabytes of IP traffic into a parsimonious tree model. The method automatically finds an appropriate scale of aggregation so that each cluster represents a local maximum of the traffic density from a block of source addresses to a block of destination addresses. We apply this clustering method on NetFlow data from an enterprise network, find the largest traffic clusters, and analyze their stationarity across time. The existence of heavy-volume clusters that persist over long time scales can help network operators to perform usage-based accounting, capacity provisioning and traffic engineering. Also, changes in the layout of hierarchical clusters can facilitate the detection of anomalies and significant changes in the network workload.

CCS Concepts

•Networks → Network performance analysis; Network measurement; •Computing methodologies → Cluster analysis;

Keywords

Flow clustering, Hierarchical clustering, NetFlow, Unsupervised Machine Learning

1. INTRODUCTION

Enterprise networks and service providers are flooded with operational data aggregated in different scales, from link utilization counters all the way to flow records. The sheer amount of this data, as well as their velocity and dimensionality, is overwhelming for network operators. New methods are needed that can aggregate and analyze such data in real-time to produce a high-level understanding of the network's operating status, and to produce actionable recommendations when something appears to be out of the ordinary. As a step towards this high-level goal, a major task is to aggregate the incoming stream of flow records in both the spatial and temporal domains. The scale at which network data should be analyzed at is a crucial factor in both diagnostic and predictive tasks; for instance, an individual flow record cannot tell us much about what happens in the network, while a coarse aggregation of flows based on BGP prefixes can blur out important flow patterns and deviations. We present a flow aggregation scheme that identifies source-destination address blocks of maximum traffic density. The proposed clustering method is capable of handling flow records collected from multiple routers. The flow records are first aggregated agglomeratively to form a tree model that accounts for the entire traffic volume. Then, the constructed tree is traversed in an efficient manner to identify and mark nodes that correspond to maximum traffic density; those nodes are the identified clusters. These clusters form a parsimonious hierarchical model of the network flow that can facilitate many applications in network management, such as traffic engineering, capacity provisioning, usage accounting, and anomaly detection.

*This paper was originally published in the proceedings of the Big-DAMA 2017 SIGCOMM workshop

2. RELATED WORK

Our clustering approach is similar in terms of objectives to that of Estan et al. [4] but with important methodological differences. That work finds clusters with traffic volume larger than a given threshold. The threshold is set as a percentage of the total traffic; this is equivalent to specifying a desired number of clusters. Our method does not require a given number of clusters. This fundamental difference is more important when the dataset contains traffic from different routers and locations. The method of [4] and similar volume-based approaches [13, 7, 12] stretch and merge cluster boundaries just to meet the volume threshold constraint [1].

Density-based clustering algorithms such as DBSCAN [5] are more similar to our method. However, DBscan does not consider the hierarchical structure of IP address blocks. We introduce a density metric that is appropriate for the two-dimensional space of source-destination IP addresses. Finally, to account for the inherent hierarchical nature of IP addressing, we deploy agglomerative hierarchical clustering [6, 11] but relying on density maximization in every hierarchical level.

3. METHOD

We use the term *"Macroflows"* as a 2-tuple aggregation of network packets with the same IP source and destination addresses, over a given time period. Macroflows can be represented as points in a 2D space of source and destination addresses. In this 2D space, an "IP rectangle" (*IPREC*) $R = (S, D)$ represents all macroflows with source addresses in S and destination addresses in D, where S and D are two continuous ranges of IP addresses in the source and destination dimensions, respectively. Each IPREC R has a volume

V_R, which is the total traffic volume within its boundaries. The size of the IPREC R is denoted by A_R and it is the total number of IP address pairs in R. We also define the operator *merge*, denoted by $\oplus$, as the minimum bounding rectangle of two IPRECs.

Let the volume density[1] of R be:

$$density(R) = \frac{V_R}{A_R}. \quad (1)$$

We are interested in finding IPRECs with maximum local volume density. Given the inherent hierarchical nature of IP addressing, we rely on an agglomerative clustering approach that first creates a MacroFlow Tree (MFT) model, and then identifies the clusters by an efficient analysis of the MFT.

3.1 MFT construction

The root of an MFT has no parent and its IPREC covers the whole 2D IP space of the given macroflow dataset. The leaves of the MFT correspond to individual macroflows (IPRECs of size one). Each internal MFT node is an IPREC with a single pointer to a *parent* node, and two or more pointers to its *children* nodes. If n_i is the parent of n_j then the n_i IPREC contains the n_j IPREC. The volume density of a node is that of its IPREC; likewise, merging of nodes is equivalent to merging their corresponding IPRECs.

To form the MFT agglomeratively, certain nodes are merged in each iteration. To make the MFT construction more efficient, this merging operation is performed considering only neighboring nodes. The neighborhood of a node is defined by its K nearest neighbor set (KNN_SET). Finding the KNN_SET of a node requires a distance function between IPRECs; we use the Euclidean distance between the centroids of two IPRECs.

The clustering objective is to identify IPRECs of maximum volume density. Specifically, in each iteration we identify the two nodes that generate the IPREC with the maximum volume density and merge them. So, nodes n_i and n_j are merged if:

$$i, j = args \max_{p,q \in N_\emptyset} density(n_p \oplus n_q)$$
$$\text{s.t.} \quad p \in \text{KNN\_SET}(q) \quad (2)$$

where $N_\emptyset$ is the set of all intermediate and leaf nodes that have not been merged yet. Thus, each iteration of the MFT construction consists of finding the optimal pair of IPRECs, according to eq. 2, and merging them to form a new parent node. Note that the IPRECs contained in the parent node are considered its children.

Figure 1 illustrates the iterations of the MFT construction algorithm for an example with four nodes. Merging n_1 and n_3 leads to maximum volume density in the first iteration, creating the node n_{13} as the parent of nodes n_1, n_2, and n_3. In the last iteration, all nodes are merged to form the root of the MFT.

To reduce the run-time complexity of the algorithm, we avoid re-calculating the KNN_SET of nodes at each iteration. Instead, we build a weighted KNN graph initially; each edge $p \xrightarrow{W} q$ in the KNN graph means that $q \in$ KNN_SET(p) and the merge density of p and q is W. The edge weights of the KNN graph are stored in a maximum heap data structure that supports constant-time queries for the maximum

[1]We use density and volume density interchangeably in the remainder of the paper.

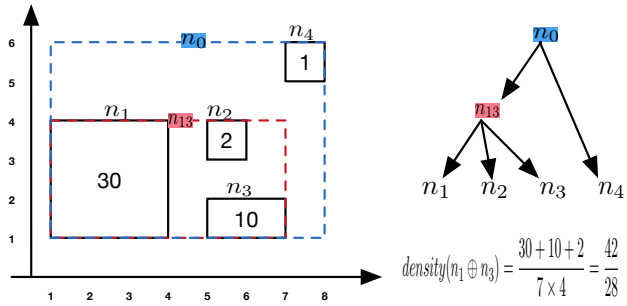


Figure 1: Left: illustration of a 2-dim IP space with four traffic-carrying regions (nodes). Each node is labeled with its traffic volume. Right: the corresponding MFT (assuming that all nodes are neighbors with each other, in terms of KNN distance). A density calculation after merging nodes 1 and 3 is also shown.

element (i.e., the optimal merge candidate of equation 2 is returned in $O(1)$ time at each iteration). Also, in each iteration two or more MFT nodes are deleted, and a new node is added. The nodes with modified neighborhoods are queried for their KNN_SET. We use the KDTree datastructre to maintain the KNN graph because it supports the insertion, deletion and KNN_SET query operations in $O(logN)$ time [2].

3.2 Cluster retrieval

The construction of the MFT hierarchy, introduced in the previous section, facilitates the detection of clusters, as described next. Consider a chain of nodes $n_i, n_{i+1}, \cdots, n_{i+k}$ that form a path in the MFT. The density change between a child and its parent on this chain is

$$\Delta density(n_j) = \frac{V_{n_j.parent} - V_{n_j}}{A_{n_j.parent} - A_{n_j}}. \quad (3)$$

If two nodes n_j and $n_j.parent$ are within the same cluster, their volume densities fluctuate around the same density, i.e., the derivative of $\Delta density(n_j)$ is expected to be close to zero. However, if the node n_j resides at the boundary of a cluster there should be a sharp density drop at n_j that will cause the derivative of the density to be significantly less than zero. We identify the nodes that experience such sharp density drops as clusters.

The MacroFlow Clustering (MFC) algorithm traverses the MFT tree from the root towards the leaves, searching for nodes at which the derivative of the density function drops significantly below zero. The problem with this procedure is that this derivative function can be highly noisy, because of the highly skewed traffic distribution. To address this issue, we rely on *regularized derivates* [3]. The method of Chartrand [3] imposes some degree of regularization in the derivative of the density function by optimizing the trade-off between the accuracy of the derivative and its smoothness.

Let $R_d(n_i)$ be the regularized derivative of $\Delta density(n_i)$. The MFC algorithm marks the node n_i as a cluster if the drop $R_d(n_i) - R_d(n_{i+1})$ is δ-times larger than the maximum of these drops in the l preceding nodes, i.e., we set the minimum length of the "plateau" in the density function

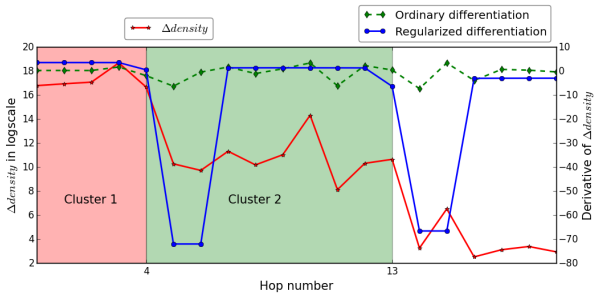


Figure 2: Volume density and its two derivatives (ordinary and regularized) for a sample path from a leaf of the MFT to the root. There are two clusters in this path, at hop numbers 4 and 13 (see large drops of density).

to l steps, and the minimum relative magnitude drop to δ ($\delta > 1$). The selection of δ balances detection sensitivity and robustness to noise, which is a common tradeoff in clustering algorithms. We empirically evaluate the accuracy of MFC as δ varies in the next section.

Figure 2 shows a sample walk of length 20 in an MFT constructed based on synthetic data. Two clusters are embedded on this walk at indices 4 and 13. Note the large drops at these cluster boundaries in the regularized derivative function (but not in the ordinary derivative function). The indices of these drops are returned as the final clusters.

A naive approach would be to traverse all paths, from the root to every leaf of the MFT, to detect clusters. This is mostly redundant however since many root-leaf paths on the MFT are expected to share several nodes (especially consecutive paths in a depth-first-search (DFS) traversal). We define the similarity of two paths, P_1 and P_2, by

$$sim(P_1, P_2) = \frac{|P_1 \cap P_2|}{min(|P_1|, |P_2|)} \quad (4)$$

which is the number of common nodes normalized by their minimum path length. Given a path P_i, we are not traversing consecutive paths $P_{i+1}, \cdots, P_{i+k}$ if $sim(P_i, P_{i+k}) > 80\%$.

4. EVALUATION WITH SYNTHETIC DATA

To evaluate the accuracy of the MFC algorithm, we created synthetic $n \times n$ traffic matrices M, with n being the number of addresses in each dimension. The element M_{ij} is the traffic volume from address i to j. It has been previously shown in [8] that the log-normal distribution is a good model for the volume distribution in a traffic matrix. To generate the traffic matrix we leverage the method of [9]: for every traffic exchange $i \rightarrow j$

$$M_{ij} = p_i p_j \quad (5)$$

where p_i and p_j are identically distributed exponential random variables with mean $\frac{1}{\lambda}$. The parameter λ controls the heterogeneity of the traffic matrix.

To embed hierarchical clusters in the traffic matrix, each element of the matrix is scaled by g^{h_k}, where k is a cluster, h_k is the depth of that cluster in the hierachy (e.g., $h_2 = 3$ in figure 4) and g is a parameter to contol the density gap between clusters and their surroundings. Furthermore, since traffic matrices are typically very sparse (i.e., many pairs of addresses do not exchange any traffic), we zeroed 90% of the smaller elements in M.

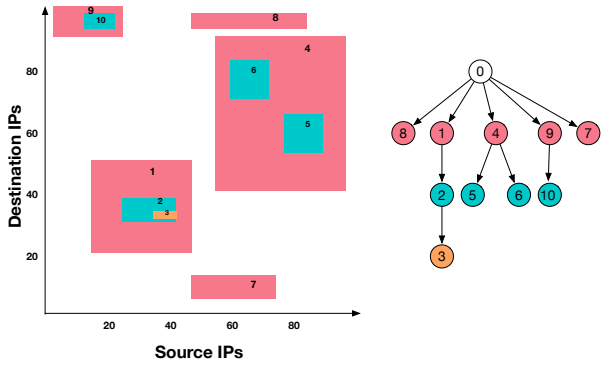


Figure 3: An example of a hierarchical traffic matrix and its corresponding cluster tree on the right.

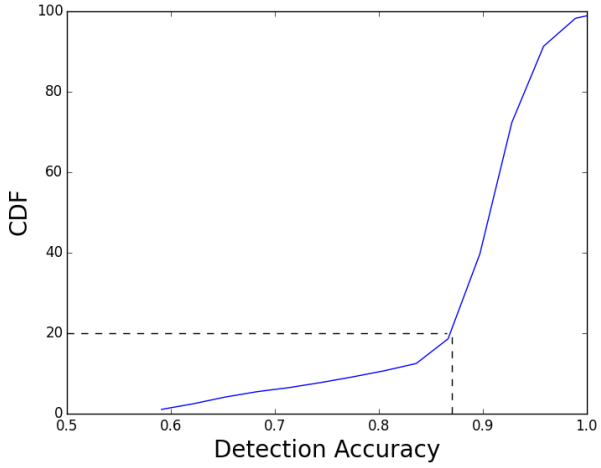


Figure 4: Detection accuracy of MFC algorithm evaluated by our synthetic data

We generated 1000 synthetic traffic matrices with $\lambda = 2$, $g = 5$ and $n = 1000$. For each matrix, we run MFC with $\delta = 5$, $l = 4$ and $K = 10$. We measure the cluster detection accuracy using the Jaccard similarity between the ground-truth and the detected clusters. For a cluster C_{g_i} in the ground-truth dataset, the detection accuracy is

$$R_{C_{g_i}} = \max_k(J(C_{g_i}, C'_k)) \quad (6)$$

where C'_k is a detected cluster k, and $J(C_{g_i}, C'_k)$ is the Jaccard similarity between those corresponding two clusters. The accuracy over the whole set of clusters is given by the following metric,

$$R = \frac{\sum_i V_{C_{g_i}} \times R_{C_{g_i}}}{\sum_i V_{C_{g_i}}} \quad (7)$$

where each cluster C_{g_i} is weighted by its volume.

The main result of this evaluation study is that the detection accuracy is larger than 92% in half of the generated traffic matrices. The accuracy is at least 87% when we consider 80% of all traffic matrices (see figure 4).

5. CASE STUDY

5.1 Collected enterprise macroflow dataset

We collected NetFlow version-9 records from one of the major backbone routers of a large international enterprise on November 16, 2016 for four hours starting at 10 a.m. local time. Approximately three terabytes of traffic was routed through router in every hour. In the remainder of the paper, we use D1, D2, D3 and D4 to refer to the pre-processed macroflow dataset for each successive hour of the collected data. Given that most macroflows are quite small in volume, we removed the smaller flow records so that we only keep those macroflows that capture 90% of the total traffic volume. The number of macroflows in each dataset is 621509, 622625, 769624, and 943041 respectively. We then separated macroflows in /8×/8 blocks; blocks with fewer than 1000 macroflows were discarded.

The four macroflow datasets were further partitioned to ingress, egress and transit traffic, as follows. Ingress traffic has a destination address within that router's enterprise site, while the source address is outside that site. The opposite is true for egress traffic. Macroflows where neither the source nor the destination address are within that router's site, is transit traffic. For the collected dataset 40% of the traffic is transit, 35% ingress and 25% egress. This distribution is approximately the same for all four traces.

5.2 Results

5.2.1 Static analysis

When we apply the MFC algorithm on the D1 dataset, we identify 760 egress, 1336 ingress and 1365 transit clusters.

The D1 clusters are of different sizes in the source or destination dimensions. The first two graphs in Figure 6 show the distribution of cluster sizes. Sizes vary from /14 to /28 prefixes in both dimensions. However, there is a strong mode at the /24 prefix length in both the source and destination dimensions. We confirmed that this enterprise uses mostly /24 address blocks (in private IP space) for different organization units. Note that the variability of subnet sizes is greater in transit traffic; ingress and egress traffic have a sharper decrease around the /24 mode. Also, the distribution of subnet sizes in the source dimension of ingress traffic is very similar to that of subnet sizes in the destination dimension of egress traffic. This is expected, given that if an organization unit generates a lot of traffic in both the ingress and egress directions, its address block will appear in both ingress and egress traffic clusters.

Figure 6c illustrates the hierarchical depth of the detected clusters in D1. The hierarchical depth of a cluster α is the number of clusters upstream of α in the macroflow tree. Figure 6c shows the distribution of hierarchical depth for clusters with no downstream cluster (leaf clusters). Note that most macroflows are in a cluster with hierarchical depth of four or smaller. Transit traffic tends to have clusters of larger hierarchical depth than both ingress and egress traffic.

The previous results were based on the default MFC algorithm parameters ($\delta = 5,\ K = 10,\ l = 4$). To test the robustness of the MFC algorithm with respect to these parameters, we varied each of these parameters (one at a time) and measured the Adjusted Rand Index (ARI) between the new clusters and the clusters that correspond to the default parameter values. The bottom row in Figure 6 shows the ARI metric for these comparisons. This analysis was perfomed based on the largest (by volume) /8×/8 block in each traffic category. MFC returns highly similar clusters (ARI larger than 90%) in a wide range around the default parameter value. The robustness is stronger for k and δ compared to l.

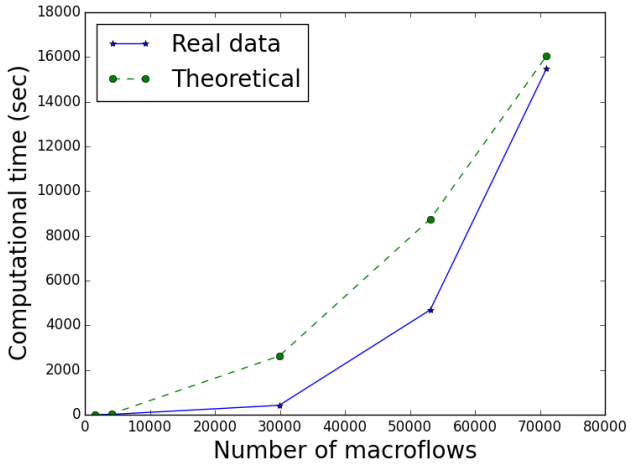


Figure 5: Run-time of MFC algorithm on five /8 × /8 blocks versus the normalized theoretical run-time.

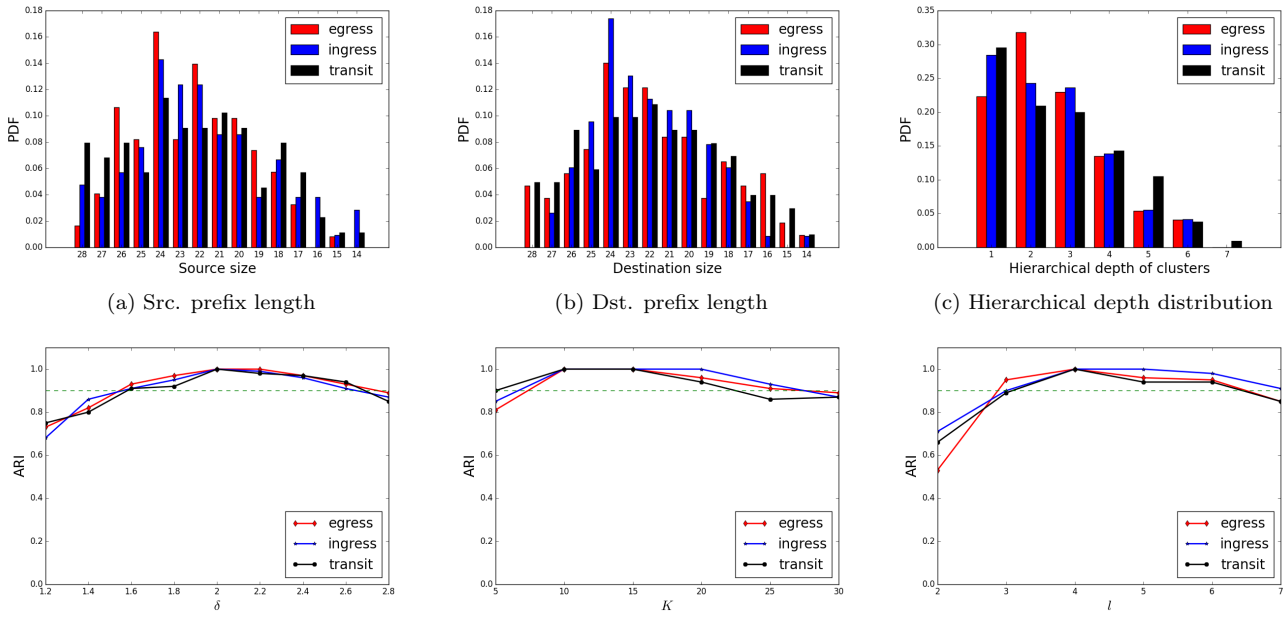
As with other hierarchical clustering algorithms, the run-time complexity of MFC is $O(N^2 logN)$, where N is the number of macroflows. Figure 5 shows the actual time it takes for MFC to run on real datasets of different sizes. We also show the theoretical run-time complexity normalized so that it has the same value with real computation time for a dataset of size $N = 1546$. This graph is based on five /8×/8 blocks of the D1 dataset. The MFC algorithm can be easily parallelized and run on data collected from multiple routers given that one can assign different large address blocks to separate MFC instances.

5.2.2 Temporal analysis

We now investigate the extent at which the identified clusters persist over time. In other words, if a cluster is detected during a time epoch, will it also be detected in subsequent time epochs? And if so, how similar will the corresponding cluster boundaries be?

We run the MFC clustering algorithm for the four 1-hour datasets and examine the temporal persistence of the top-10 clusters. For each cluster, we find its corresponding cluster in another trace by searching for the cluster with the maximum Jaccard similarity. The Jaccard similarity between two clusters is defined as the intersection area of their corresponding IPRECs divided by their union area. The average Jaccard index between all pairs of traces is shown in Table 1, averaged over all top-10 clusters. Note that the average Jaccard similarity is typically higher than 90% for successive 1-hour time epochs, at least for egress and ingress traffic. The Jaccard index of transit clusters is lower but still higher than 60%.

Finally, Figure 7 illustrates the clusters' IPRECs for different time slices. The first two rows are plotted based on the D1 and D2 datasets. The last row corresponds to a more recent dataset, collected on January 18th, 2017 - same weekday and daytime as D1. The subplots in each column of the



(a) Src. prefix length (b) Dst. prefix length (c) Hierarchical depth distribution

(d) Sensitivity analysis for δ (e) Sensitivity analysis for K (f) Sensitivity analysis for l

Figure 6: The results of running MFC on the D1 dataset. Top: the distribution of cluster sizes in the source and destination dimensions; also, the distribution of the clusters' hierarchical depth. Bottom: sensitivity analysis of MFC with respect to its three parameters.

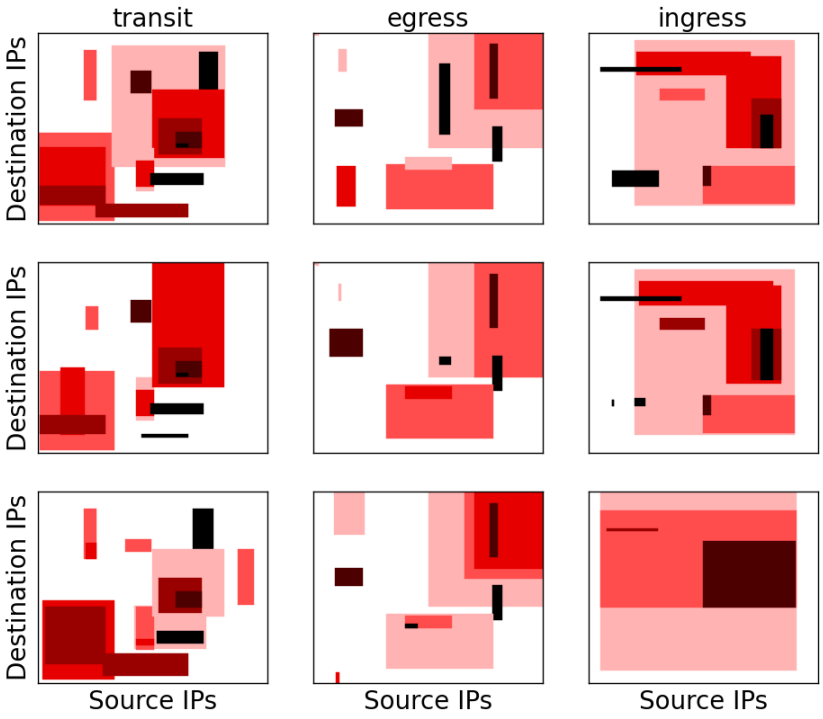


Figure 7: Cluster layouts for different traffic types (transit, egress, ingress) and time periods. Darker colors represent denser clusters. **Top:** Clusters in a /16 × /16 block of D1 (the block with the maximum number of clusters) for the three different traffic types. **Middle:** Clusters in the same block but for D2. **Bottom:** Clusters in the same block but for an one-hour dataset collected on January 18, 2017 at 10am local time (two months after the collection of the D1 trace).

	D1	D2	D3	D4
D1	100, 100, 100	94, 91, 81	83, 95, 64	83, 83, 41
D2		100, 100, 100	92, 88, 67	71, 85, 56
D3			100, 100, 100	88, 79, 65

Table 1: Average Jaccard similarity index between the top-10 clusters in our four datasets. The three figures in each entry refer to egress, ingress, and transit traffic, respectively.

figure are based on a same /16×/16 block. The block in each column was selected so that it has the maximum number of clusters in D1. Although one can see some changes across different time slices (especially as the time gap increases), there still exists significant similarity in the overall cluster layout for different time slices.

The source code of the MFC algorithm is available at [10].

6. CONCLUSIONS

We presented a method to cluster and analyze NetFlow data, without requiring a given number of clusters or a certain spatial aggregation scale (e.g., BGP prefix lengths). We applied this method in flow data from an enterprise network and observed that these clusters maintain a level of stationarity that can be leveraged for practical network operations. Instead of trying to understand what happens at the level of individual flows, network operators can focus their analysis at the level of these clusters. We believe that it should be possible to leverage historic and current NetFlow at the resolution of the identified clusters to generate actionable recommendations and applications, e.g., tracking the evolution of a cluster traffic for capacity planning and anomaly detection by identifying abrupt changes in a cluster. Developing such applications will be the next step of this research.

Acknowledgments. The authors would like to thank Dr. Mehdi Nikkhah for providing us with technical help during this project. The authors would also like to thank the ACM Big-DAMA SIGCOMM 2017 anonymous referees for their valuable comments and helpful suggestions. This work was supported by a research gift from Cisco Systems Inc and by the National Science Foundation (award number 1440585).

7. REFERENCES

[1] Agarwal, D., Barman, D., Gunopulos, D., Young, N. E., Korn, F., and Srivastava, D. Efficient and effective explanation of change in hierarchical summaries. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining* (2007), ACM, pp. 6–15.

[2] Bentley, J. L. Multidimensional binary search trees used for associative searching. *Communications of the ACM 18*, 9 (1975), 509–517.

[3] Chartrand, R. Numerical differentiation of noisy, nonsmooth data. *ISRN Applied Mathematics 2011* (2011).

[4] Estan, C., Savage, S., and Varghese, G. Automatically inferring patterns of resource consumption in network traffic. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications* (2003), ACM, pp. 137–148.

[5] Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd* (1996), vol. 96, pp. 226–231.

[6] Franti, P., Virmajoki, O., and Hautamaki, V. Fast agglomerative clustering using a k-nearest neighbor graph. *IEEE Transactions on Pattern Analysis and Machine Intelligence 28*, 11 (2006), 1875–1881.

[7] Mahmood, A. N., Leckie, C., and Udaya, P. An efficient clustering scheme to exploit hierarchical data in network traffic analysis. *Knowledge and Data Engineering, IEEE Transactions on 20*, 6 (2008), 752–767.

[8] Nucci, A., Sridharan, A., and Taft, N. The problem of synthetically generating ip traffic matrices: initial recommendations. *ACM SIGCOMM Computer Communication Review 35*, 3 (2005), 19–32.

[9] Roughan, M. Simplifying the synthesis of internet traffic matrices. *ACM SIGCOMM Computer Communication Review 35*, 5 (2005), 93–96.

[10] Shadi, K. https://github.com/kamalshadi/mfc.

[11] Vesanto, J., and Alhoniemi, E. Clustering of the self-organizing map. *IEEE Transactions on neural networks 11*, 3 (2000), 586–600.

[12] Wang, J., Miller, D. J., and Kesidis, G. Efficient mining of the multidimensional traffic cluster hierarchy for digesting, visualization, and anomaly identification. *Selected Areas in Communications, IEEE Journal on 24*, 10 (2006), 1929–1941.

[13] Zhang, Y., Singh, S., Sen, S., Duffield, N., and Lund, C. Online identification of hierarchical heavy hitters: algorithms, evaluation, and applications. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement* (2004), ACM, pp. 101–114.