NetCluster: A clustering-based framework to analyze internet passive measurements data

NetCluster: a Clustering-Based Framework to Analyze Internet Passive Measurements Data

Elena Baralis[a], Andrea Bianco[b], Tania Cerquitelli[a,*], Luca Chiaraviglio[c], Marco Mellia[b]

[a] *Dipartimento di Automatica e Informatica, Politecnico di Torino, Torino, Italy*
[b] *Dipartimento di Elettronica e Telecomunicazioni, Politecnico di Torino, Torino, Italy*
[c] *Mascotte, INRIA, I3S, CNRS, Université de Nice Sophia, Sophia Antipolis, France*

Abstract

Internet measured data collected via passive measurement are analyzed to obtain localization information on nodes by clustering (i.e., grouping together) nodes that exhibit similar network path properties. Since traditional clustering algorithms fail to correctly identify clusters of homogeneous nodes, we propose the NetCluster novel framework, suited to analyze Internet measurement datasets. We show that the proposed framework correctly analyzes synthetically generated traces. Finally, we apply it to real traces collected at the access link of Politecnico di Torino campus LAN and discuss the network characteristics as seen at the vantage point.

1. Introduction and Motivations

Internet is a complex distributed system which grows and evolves in time, with new services and technologies that are rolled out at a constant pace. Today, cloud computing and Content Delivery Networks (CDNs) are the key technologies that enable efficient and high-demanding services over the Internet, with reports stating that today about 70% of web traffic is being served by the cloud and CDN infrastructures of top players like Google, Akamai,

*Corresponding author
Email addresses: elena.baralis@polito.it (Elena Baralis), bianco@tlc.polito.it (Andrea Bianco), tania.cerquitelli@polito.it (Tania Cerquitelli), luca.chiaraviglio@inria.fr (Luca Chiaraviglio), mellia@tlc.polito.it (Marco Mellia)

or Amazon [1]. In this scenario, the topological and geographical location of nodes offering some service is today becoming more and more difficult to retrieve, especially considering the above mentioned CDNs and cloud based services for which little or no information is provided by their owner. This has created a tangled world wide web that is very hard to unwind, impairing ISPs and network administrators capabilities to understand the traffic flowing on their network.

Notice that several solutions are currently being available to find the both the topological and geographical position of an IP address[1]. However, these databases exhibit well-known limits especially when asked information about CDN or cloud service [2, 3]. Worst of all, little information is known about the methodologies followed by the maintainers of these databases, so that their accuracy can hardly be evaluated.

In this paper, we propose the idea of exploiting passive measurements to derive node topological information. We leverage information that can be passively extracted by traffic that is observed at the edge of a network, e.g., at a Campus or Company access router, or at an ISP Point of Presence, where several client nodes access service in the Internet [24].

We wish to classify (by grouping together) sets of nodes that exhibit similar network path properties, such as delay, loss, throughput, as measured at a given network observation point (vantage point) by passively observing packets. Clustering algorithms are exploited to achieve this goal. We show that, by exploiting passive monitoring of traffic exchanged from/to a vantage point, it is possible to group servers to form homogeneous sets. Our aim is to retrieve information about the topological information. For example, our methodology allows to discover which IP subnets are present in the same datacenter. The geographical information of a cluster of nodes can then be eventually added later using external databases or additional information. In general, dealing with homogeneous clusters makes the mining of server characteristics easier than considering single server IP addresses.

Our methodology can thus be instrumental to complement, verify and update the information provided by other IP location databases. This is however outside the scope of our work, which instead focuses on the data mining methodology.

[1]See for example http://www.iplocation.net/, http://www.ipfingerprints.com/ or http://www.maxmind.com/

Data mining algorithms [8] allow sifting through large amounts of data and picking out relevant information. Based on a notion of object similarity, clustering methods assign objects to different groups. More precisely, clustering algorithms partition a data set into subsets (called clusters), so that objects in each subset share some common trait - often measured as proximity according to some defined distance measure. It is therefore natural to apply clustering techniques to identify sets of Internet nodes that exhibit similar network properties. However, in this paper we show that traditional clustering algorithms fail to correctly identify clusters of homogeneous nodes. Therefore, we propose "NetCluster", a novel framework to achieve this goal. Most clustering algorithms prefer a non dense measurement space, so that borders between clusters could be easily defined. However, this assumption does not hold on the typical Internet measurement dataset. Thus, new approaches are needed, as the one proposed in this paper.

Given a network trace of TCP flows generated by Internet nodes, NetCluster analyzes a small set of flow features (dimensions in a metric space). It relies on a data preprocessing phase tailored to the specific features of the considered networking application domain. More specifically, the original traffic trace is split in disjoint subsets, independently analyzed. The partition is driven by the values of dimensions for which a distance definition hardly holds (e.g., the TTL feature). Then, nodes that exhibit similar network properties are grouped by an appropriate clustering algorithm. To reach on-line processing speed, an incremental version of the selected clustering algorithm is proposed. This new algorithm, while preserving its effectiveness, is characterized by a reduced complexity, which provides enhanced execution performance. Finally, a post-processing analysis refines the cluster definition identified in the previous step. Extensive tests performed both on artificial traffic and real traffic traces prove that NetCluster outperforms several well known clustering algorithms.

The main contribution of the paper is twofold:

- the definition of a framework that analyzes passive measurements and allows the identification of clusters of Internet nodes that are network-wide (topologically) close to each other, as seen from the measurement point;

- the definition of an incremental clustering algorithm tailored to the analysis of typical Internet measurement datasets.

We provide also an the analysis of characteristics of the servers as seen at a measurement point.

The paper is organized as follows. Section 2 introduces the architecture of the proposed framework. We describe the considered measurement setup and the selection of the dataset to analyze in Section 3, whereas the proposed incremental clustering algorithm is presented in Section 4, together with a brief discussion of alternative clustering techniques. Results obtained from both real and synthetic traffic traces are then presented in Section 5. Section 6 describes related research activities, while conclusions are drawn in Section 7.

2. The NetCluster Framework

We wish to identify clusters (groups) of Internet nodes (objects) that, from a given vantage point, are network-wide similar, i.e., show similar network-related characteristics. Nodes will be grouped according to the features of the TCP connections originated at or received by the nodes. The NetCluster framework includes two blocks: (i) Data acquisition and preprocessing (see Section 3), and (ii) clustering (see Section 4).

The data acquisition and pre-processing activities are detailed in Section 3. Intuitively, network data information is acquired by passively monitoring traffic entering/leaving an edge node. The TCP flows are rebuilt and characterized by the most significant variables for node classification from a localization point of view. Finally, the generated flows are segmented in disjoint subsets subsequently analyzed in a separate fashion.

We tested several different clustering approaches to evaluate their ability in grouping objects in our application scenario. We found that most off-the-shelf algorithms suffer from significant drawbacks (see Section 4 for theorical discussion and Section 5 for experimental results). The main reasons are the difficulty in setting input parameters and the tendency to create excessively fragmented or excessively large clusters. The WaveCluster algorithm [20] has been identified as an appropriate algorithm for analyzing Internet measurement datasets. This algorithm has been modified (i) to introduce a refinement process, which allows the improvement of the clustering quality in the context of Internet measurements, and (ii) to operate incrementally, thus allowing real-time data processing. Besides proposing an effective clustering algorithm, the NetCluster framework is designed to overcome the

limitations shown by well-known clustering algorithms when applied to the node localization problem.

3. Data Acquisition and Pre-processing

A passive probe sniffs all packets flowing on the access link (vantage point) that connects the edge network to the Internet. Therefore, the passive probe can monitor all incoming and outgoing packets, i.e., packets directed to a node inside the network and generated by a node in the Internet, and vice versa. The probe runs **Tstat** [24, 25], a passive monitoring tool that permits to derive network and transport layer measurements. Tstat rebuilds each TCP connection by matching incoming and outgoing segments. Thus, a flow-level analysis can be performed, **like the one in** [25]. A TCP flow is identified by snooping the signaling flags (SYN, FIN, RST), and the status of the TCP sender is rebuilt by matching sequence numbers on data segments with the corresponding acknowledgment (ACK) numbers.

Since we are interested into properties of the path to a given server IP address, we restrict our attention to the sequence of packets received by hosts in the monitored networks, i.e., those packets that traveled toward the vantage point. Among the large set of available measurements describing flow behavior, the following were selected, because they contain the most useful information for node classification from a localization point of view:

- the minimum IP Time-To-Live (TTL) observed on packets belonging to the TCP flow, i.e., the number of hops from the remote node to the vantage point[2];
- the minimum Round-Trip-Time (RTT) observed on a TCP flow, i.e., the minimum time lag between the observation of a TCP segment and the observation of the corresponding ACK, a variable strongly related to the distance between the two nodes;
- the flow reordering probability ($P\{reord\}$), which can be useful to distinguish different paths;

[2]The initial TTL value is set by the source, typical values being 64 and 128. TTL values are converted to the range 0-64 for normalization purposes.

- the flow dropping probability ($P\{drop\}$), that can be used to separate a low-speed, unreliable or congested paths;
- the flow duplicate probability ($P\{dup\}$), that can highlight a destination served by multiple paths[3].

Thus, an object is defined by the record $\{TTL, RTT, P\{reord\}, P\{drop\}, P\{dup\}\}$ (see Section 5.1 for the experimental analysis). As a case study, only TCP flows which last more than $P = 100$ packets are considered, to obtain reliable estimates on reordering, dropping and duplicate probabilities in particular. This choice is also helpful in focusing the analysis on long-lived flows where the network path characteristics have more impact and thus can provide valuable information.

The server IP address could be another possible candidate variable. However, it does not provide a proper "distance" information. For example, two consecutive IP addresses might belong to two different ISPs operating at very different locations. Furthermore, the server IP address does not take into account the network condition. Since we look for a dynamic node localization, a key requirement is that the variables depend on link state and on node congestion.

Network traces are then pre-processed to improve cluster quality and algorithm efficiency. In the context of Internet measurements, some dimensions of the measurement space represent indices for which even a small variation corresponds to a huge distance in the measurement space. For example, two objects with different TTLs should belong to distinct clusters, because, if they were sharing the same path, they would likely have the same TTL[4]. In this case, it is appropriate to enforce an a-priori dataset partitioning to increase the effectiveness of the clustering algorithm.

Considering the measurement space under analysis, we segmented the original dataset into a number of disjoint subsets on the values of the TTL dimension. In particular, objects are assigned to different groups on the basis of their TTL value and only objects (connections) with the same TTL

[3] $P\{reord\}, P\{drop\}$ and $P\{dup\}$ are computed by observing the TCP sequence and acknowledgement numbers carried by segments of a given flow. We refer the reader to [25] for more details.

[4] This consideration is correct if the measurement window is not too large, so that the Internet routes are stationary, as reported in [26]. **Similarly, we assume anycast routes are stable.**

value are then analyzed by the clustering algorithm. Thus, the clustering algorithm will process a four dimensional measurement space characterized by $\{RTT, P\{reord\}, P\{drop\}, P\{dup\}\}$. Since the clustering algorithm will operate on a smaller dimension space (4 dimensions) and a reduced set of objects (due to segmentation), its execution will be also more efficient.

4. Clustering

Clustering algorithms group objects into sets (clusters) on the basis of the available information describing objects [8]. The aim is to obtain clusters containing similar objects, whereas objects in different clusters should be dissimilar. The larger the similarity among objects within a cluster, and the larger the difference between clusters, the higher the clustering quality. Similarity is normally measured according to a notion of distance among objects and clusters in a measurement space describing object features.

More formally, let us consider a measurement space X and a set of objects $A = \{x_1, \ldots, x_N | \ x_i \in X\}$ which have to be grouped (clustered) into K subsets. The clustering algorithm finds a partition $\mathcal{C} = \{C_1, \ldots, C_K\}$, such that $\bigcup_i C_i = A$ and $C_i \cap C_{j \neq i} = \emptyset$, with K possibly being unknown a priori. The subsets in the partition are named clusters. Clusters contain "similar" objects, whereas objects associated with different clusters should be "dissimilar", the similarity being measured via object-to-object or cluster-to-cluster distances.

Clustering algorithms rely on many different approaches. The algorithms we considered for the NetCluster framework are introduced in Section 4.1, while Section 4.2 discusses the motivations of our final choice in favor of the WaveCluster algorithm. Section 4.3 describes NetCluster, an improved version of the WaveCluster algorithm. Finally, Section 4.4 describes an extension of NetCluster specifically designed to analyze network traffic data, which operates incrementally. This novel incremental algorithm allows the analysis of measurement data captured in real-time.

4.1. Clustering Algorithms

We considered the following algorithms to be interesting candidates for network traffic analysis: (i) DBSCAN [18], (ii) EM (Expectation-Maximization) [22], and (iii) WaveCluster [20]. DBSCAN is a well-known representative of density-based approaches, whose objective is to identify portions of the measurement space characterized by a high density of objects. The Expectation-

maximization (EM) algorithm is a representative of the model-based methods, which assume that each cluster can be modeled by means of a mathematical model and group objects by determining the best fit between the mathematical model and the objects. Finally, WaveCluster is a representative of grid-based methods, which exploit a multi-dimensional grid to quantize the measurement space and define clusters by aggregating adjacent cells characterized by a high object density. These approaches are discussed in more details in the following.

Other approaches, such as hierarchical-based methods exploit an agglomerative or divisive approach to generate a hierarchical collection of clusters. The (most common) agglomerative approach [33] initially assigns each data object to a singleton cluster. The two closest clusters are then iteratively merged using a cluster proximity measure (e.g., MIN or single link, MAX or complete link, group average, Wards method) [36]. Among them both the group average and Wards measures are less susceptible to noise and outliers than both MAX and MIN measures. However, they tend to identify clusters with globular shapes, which could be a limitation for our research aim. Hierarchical-based methods are often used when the underlying application requires the creation of a taxonomy, which is not the case for our application scenario.

Other approaches, such as partitioning approaches (e.g., K-Means [23]), deal with the same issue as the EM algorithm. However, to automatically initialize the K-Means parameter (i.e., the number of clusters) the work [39] proposed the construction of a minimum spanning tree by means of Prims algorithm. The technique has been validated by analyzing astrophysics images (e.g., satellite view of Paris, images of the planet Mars) to show the effectiveness of the method in automatically initializing the K-Means algorithm. Since the authors of [37] stated that the proposed approach is effective when the data to be analyzed are approximately distributed according to a Poisson distribution, it was not considered in our scenario.

The DBSCAN algorithm. The DBSCAN algorithm [18] exploits the notion of "dense" neighborhood to define clusters. Density is defined as the number of objects which are in a particular area of the measurement space. DBSCAN explores the space by growing existing clusters as long as the number of objects in their neighborhood is above a given threshold. More specifically, DBSCAN requires two input parameters, which define a density threshold in the measurement space: A real number ϵ and an integer number

$minPts$. A high density area in the measurement space is an n-dimensional sphere with radius ϵ which contains at least $minPts$ objects. DBSCAN iteratively considers objects in the dataset and analyzes their neighborhood. If there are more than $minPts$ objects whose distance from the considered object is less than ϵ, then the object and its neighborhood originate a new cluster.

DBSCAN is effective in finding clusters with arbitrary shape and is capable of identifying outliers as low density areas in the measurement space. However, setting appropriate values for the ϵ and $minPts$ parameters is a rather difficult task [27]. Ref. [37] presented a comparative study between DBSCAN and FN-DBSCAN (fuzzy neighborhood DBSCAN) and demonstrated that the latter is more robust than the former for dataset characterizing by various shapes and densities. In particular, from the result quality FN-DBSCAN is able to identify more realistic and robust results with respect to the DBSCAN algorithm. However, to find the optimal values for the FN-DBSCAN parameters is rather a difficult task, as for DBSCAN.

The Expectation-Maximization algorithm. The Expectation-Maximization (EM) is a general iterative procedure used in statistics to find maximum likelihood estimates of unknown parameters for which a likelihood function can be built. The EM algorithm for clustering [22] exploits the EM approach to group objects in clusters. It computes the parameters of a Gaussian mixture model distribution. A mixture is a convex combination of N probability distributions where each distribution represents a cluster. At the beginning, parameters are assigned with random values. Then, the algorithm iterates recomputing them, until a convergence threshold is reached. More precisely, consider a set of vectors, where each vector is interpreted as an object picked among one of N Gaussian distributions. The EM algorithm groups together vectors originated by the same distribution. Each distribution models a cluster. For each distribution (i.e., cluster) the EM algorithm estimates (i) the mean and the standard deviation, (ii) the sampling probability, i.e. the probability that one of the N Gaussian distributions is used as a data source. The entire procedure is repeated until model parameters converge.

A relevant requirement of the EM algorithm is its need to set the number N of the Gaussian components, i.e., the number of clusters, before the algorithm is applied. This issue, which is relevant in our application domain, is further discussed in Section 4.2.

The work [38] proposed an effective approach to automatically discover

an optimal number of clusters to initialize the EM parameter. However, the number of required iterations to discover an estimating model similar to the real model (i.e., the best value for the cluster number) may significantly increase by increasing the number of data points and the number of dimensional attributes. Since the robust EM clustering algorithm [38] is expensive in terms of computation it was not considered because it may not effectively support real-time analysis of passive measurement data, which is our scenario.

The WaveCluster Algorithm. The WaveCluster algorithm (WC), analogously to other grid-based methods (e.g., Clique [19]), quantizes the measurement space by means of a a multi-dimensional grid. Each object is assigned to a single cell, which is "dense" if it contains enough objects. Objects belonging to adjacent dense cells are assigned to the same cluster, while objects not belonging to any clusters are labeled as noise. More specifically, the WaveCluster algorithm [20] is characterized by three main steps: (i) Quantization, (ii) wavelet transform, and (iii) cluster definition. In step (i), the measurement space is quantized into a finite set of cells. The (multidimensional) cell size is an input parameter. For each cell, its density, defined as the number of objects belonging to the cell, is computed. This phase reduces the number of objects to analyze, because clustering will operate on cells and not on objects. The next step is the application of the discrete Wavelet transform to smooth the density value of each cell according to the density values of adjacent cells. Thus, regions containing high density cells are emphasized, whereas regions containing low density cells are smoothed out. In step (iii) clusters are defined on the transformed space. A cluster is the maximal set of connected cells with a non negligible density. A threshold parameter needs to be defined to identify high density cells.

The Wavecluster algorithm exploits the wavelet transform to deal effectively with outliers and noise. However, it requires the definition of a density threshold, whose value may be hard to set appropriately. Furthermore, similarly to other grid-based approaches, it may scale poorly in the number of dimensions, because the number of cells is exponential in the number of dimensions.

4.2. Selection of the Clustering Algorithm

The following main issues guided our choice of the most suitable clustering algorithm: (i) The amount of knowledge needed to correctly set the input

parameters, and (ii) the ability to deal with noisy data, i.e. the sensitivity to outliers, missing, or erroneous data. Noise reduction is a fundamental requirement for the analysis of any measurement data affected by noise, such as Internet measurements.

All clustering algorithms require the definition of several input parameters. A proper setting of these parameters is often crucial to obtain high-quality clusters. However, appropriately setting input parameters is very difficult, because the best choice usually depends on the considered dataset. Parameters may be defined by checking clustering quality in scenarios in which the "best" clustering configuration is known (i.e., in synthetically generated datasets). However, there is no guarantee that the same parameter values would provide a high-quality clusterization when applied to datasets characterized by a different data distribution. For this reason, clustering algorithms whose parameter values can be somehow related to known data characteristics are preferable.

A specific input parameter required by some clustering algorithms (e.g., the EM algorithm) to operate correctly is the number of clusters. Since we would like to infer Internet node location information and properties by analyzing passive network measurements on data generated by relatively long TCP connections, it is definitely difficult to "guess" the correct number of clusters in our application scenario. Obviously, several clustering sessions with different values of the final number of clusters may be run. However, this procedure is quite time consuming. Furthermore, the selection of the best clustering result is not straightforward.

Among the clustering algorithms described in Sec.4.1, the grid-based WaveCluster algorithm showed remarkable properties in the context of network traffic analysis: Scalability, ability to remove noise, multi-resolution analysis capability, and straightforward parameter setting. Scalability in the number of objects is provided by the grid-based approach. Noise reduction is achieved via the Wavelet transform. The cell size may be defined separately for each variable in the measurement space. Hence, multi-resolution analysis may be obtained by appropriately managing the grid granularity. Furthermore, most WaveCluster parameters can be easily related to the network characteristics of our analysis scenario, as shown later, thus simplifying the critical task of parameter setting. The other algorithms require instead the definition of parameters which do not straightforwardly correspond to connection-related characteristics. Finally, WaveCluster does not require the a-priori knowledge of the number of clusters.

When analyzing Internet measurement data, the classical algorithms identify either few, very large clusters, or a large number of highly fragmented clusters.[5] This is due to the nature of the Internet measurements, in which the intrinsic variability of the measurements tends to spread-out objects. Furthermore, the WaveCluster algorithm creates large clusters, being the cell adjacency the criterion adopted to form clusters. The NetCluster algorithm enhances the WaveCluster algorithm by means of a reshaping process to refine cluster definitions. This is a key point that makes the NetCluster algorithm more robust and precise than classical clustering techniques on networking data. We also propose a novel incremental version of this algorithm, named *Incremental NetCluster*. This incremental version may operate in an on-line fashion and it will be shown to provide a clustering partition very similar to the one provided by the non-incremental version, while drastically reducing the processing time, thus achieving real-time processing capabilities.

The following section details the proposed NetCluster algorithm, while Section 4.4 describes the incremental version of NetCluster. Then, our algorithm choice is validated in Sec. 5, where the quality of the clustering results provided by the considered algorithms is compared.

4.3. The NetCluster algorithm

The NetCluster algorithm (denoted as NC in the following) integrates a refinement process into the WaveCluster algorithm [20]. **We assume that the reader has a basic knowledge of WaveCluster. For more details, the reader is referred to [20].** In particular, the following steps are performed: (i) quantization, (ii) wavelet transform (identified by the Mexican Hat wavelet transform), (iii) cluster definition, (iv) cluster resize. The first three steps are common to both approaches[6], while the cluster resize phase characterizes NC only. Since cell adjacency is the criterion adopted to form clusters in WaveCluster, the WC algorithm tends to intrinsically build large clusters. To counter-balance this behavior, we enforce a maximum "range" for feature values in each identified cluster. For example, clusters including objects with very different RTT values (e.g., ranging in $[0, 500]$ms) would not be allowed.

To reach this goal, we first define, for each dimension in the metric measurement space, a maximum extension (i.e., a range) allowed when grouping

[5] These aspects will be better highlighted in the Performance Results Section.

[6] For more details on the first three steps see the WaveCluster algorithm [20].

cells in the same cluster. This process defines a multi-dimensional ellipse bounding the measurement space. Then, for each obtained cluster, the cell with the maximum frequency is selected[7]. This cell becomes the center of a multi-dimensional ellipse. All cells of the current cluster which are included in the multi-dimensional ellipse are assigned to a newly created cluster. The process is then iteratively repeated for all the cells of the original cluster. At end of this process, the original big cluster is split into several new smaller clusters. This refinement process is applied to all the clusters defined by the WC algorithm.

At the end of the resize phase, all the objects belonging to the same cluster are characterized by a distance smaller that twice the ellipse "radius". Bounding the size of each cluster requires to add new parameters to the process, denoted in the following as Net_p (see Section 5 for Net_p values exploited in our implemntation). These parameters can be derived by common sense networking-based domain-knowledge. For example, considering the RTT, a natural choice would be to limit the range of values in a given cluster to about $\pm$5ms when considering Wide Area Networks. Similarly, reordering or loss probability are expected to range in a $\pm 1\%$ range or less.

4.4. The Incremental NetCluster algorithm

Both the WC and NC algorithms operate on the full trace, i.e., all the objects in the trace (typically a very large number of objects) are processed together after being collected. However, in a network measurement scenario, the objects are not available all together at the same time. In particular, batches of objects are measured and collected in a stream-like environment. Hence, we propose an incremental version of the NC algorithm, denoted in the following as NCI, which incrementally reclusters objects after a new data batch is received. More specifically, a small batch of objects, denoted as B_t, is collected every Δk seconds. The new clustering is computed as a function of the new objects in B_t and the clusters generated from the previous batches.

The WC algorithm [20] is not very efficient because the transform step and the cluster discovery are performed on all the cells of the quantized space. Furthermore, both the current and past objects need to be stored and processed at each run, thus severely reducing the applicability of the

[7]The chosen cell order may affect the result. However, we experimentally verified that by modifying the cell order we did not obtain better results.

approach to the on-line analysis. Conversely, when dealing with batches of objects, only a small portion of the quantized feature space is modified, both in terms of cell density and cluster shape. This effect is due to the fact that each batch contains a limited number of objects compared to the full trace.

The main ideas that inspired NCI are (a) recomputing the wavelet transform only over the subset of cells modified by the current batch B_t and (b) exploiting the wavelet transform and the information in the current batch B_t to incrementally define the new clusters. In general, the t-th execution of the NCI algorithm on batch B_t entails the following steps: i) Quantization, to count the cell density for the new batch B_t in the quantized space, ii) cell status update, to update the wavelet transform only on cells whose density has been changed by the new data objects in B_t, iii) cluster discovery, to update the set of relevant clusters, and iv) cluster resize and (cluster) label assignment, to refine clusters by identifying the most connected groups of data objects.

Algorithm 1 The NCI algorithm

Require: Data batch B_t, minimum frequency threshold ρ_F, network parameters Net_p
Ensure: set of clusters
1: Quantize feature space for the data batch, then assign data objects to the cells
2: Apply Wavelet transform only for cells whose density has changed
3: Find the adjacent cells in the new space
4: Assign island labels to the cells by exploiting ρ_F
5: Assign cluster labels to the islands
6: Resize clusters by exploiting network parameters
7:
8: **return** set of clusters

Algorithm 1 reports a pseudo code of NCI. At the t-th execution, the NCI algorithm analyzes the new data collected at time t, i.e., B_t. Besides the data, the configuration parameters ρ_F and Net_p are required. ρ_F is the minimum frequency threshold for the wavelet transform, while the network parameters Net_p allow resizing clusters at the end of the analysis to split large clusters and identify strongly connected groups.

The first step of the algorithm is the computation, on the current batch B_t, of the cell density in the quantized space. The wavelet transform is then computed only for the cells for which the density has been changed by the objects of the current batch. This approach reduces the run time needed to analyze a data batch.

To perform the next step, the algorithm exploits the concept of island, defined as the maximum set of adjacent/connected cells characterized by a

wavelet transform value larger than a minimum frequency threshold ρ_F. For each cell, its neigborhood, identified by the Mexican Hat wavelet transform, is explored to identify relevant cells. The algorithm assigns an island label to each cell. **For an arbitrary cell three cases are given. (i) If the cell is adjacent to an island, it is added to it. (ii) If its adjacent cells do not belong to an island, a new island label is assigned to the cell. (iii) Two islands can be merged in one because they are connected by the cell. In this case, a new label is assigned to the newly generated island.An example of the procedure is reported in Fig.1. The figure on the top reports the new island formation, labelled as '3'. The figure on the bottom reports the merging of the island, i.e., island 3 and island 2 are merged in the new island 4**. Different values for ρ_F may slightly change the selection of relevant cells. A high value of ρ_F limits the number of relevant cells and increases the number of cells discarded as noise. Low values of ρ_F may identify rather large groups of conncected cells.

The next step defines the new cluster set. For each island the corresponding density is computed as the sum of its cell densities. Only islands for which the sum of their cell densities is larger than zero become clusters. More specifically, due to the smoothing effect of the wavelet transform filter, some cells in the neighborhood of a high density cell may be characterized by a density equal to zero. Hence, islands may emerge from cells with density equal to zero. When the island total density is equal to zero, it should not become a cluster.

Finally, the clusters are resized by exploiting the network parameters $Net_p.$, i.e., big clusters are split into several small and more connected clusters to avoid excessively large clusters (see Section 4.3).

5. Performance Results

To test the effectiveness of the proposed framework, we ran a large set of experiments on both synthetic and real traces. The experimental validation addressed the following issues: (i) Clustering algorithms comparison (Section 5.2), (ii) algorithm sensitivity and robustness (Section 5.3), (iii) incremental NetCluster performance (Section 5.4), and (iv) characterization of the Internet traffic (Section 5.5).

We analyzed three classical clustering algorithms (i.e., DBSCAN [18], EM [22], and WaveCluster, denoted as WC [20]), NetCluster, denoted as

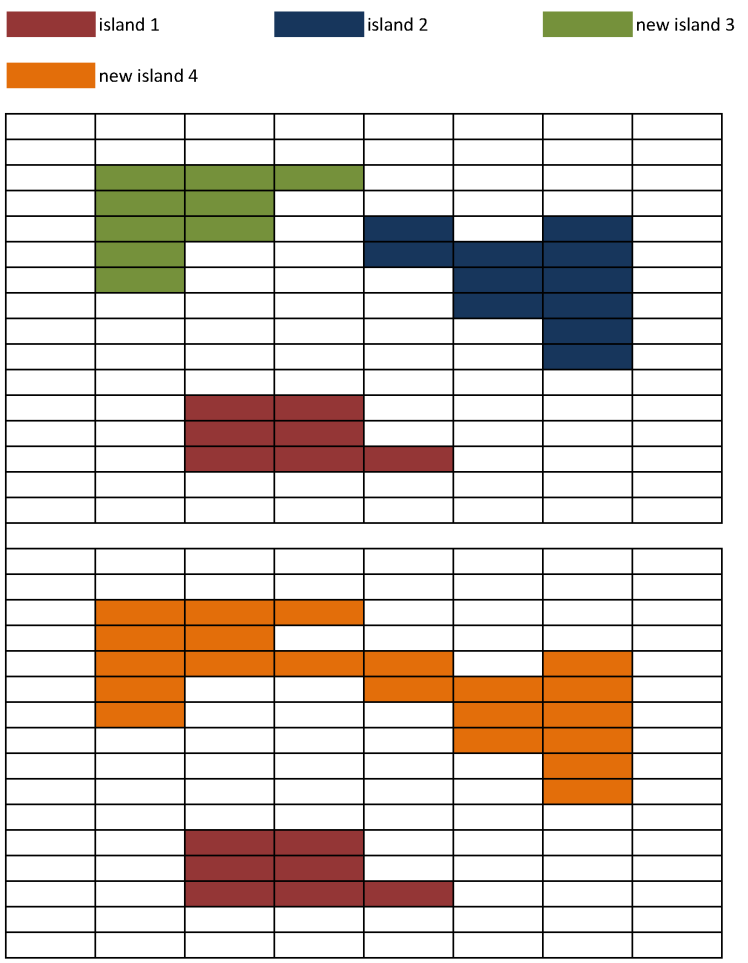


Figure 1: Island formation (top) and island merge (bottom) for an exemplary two dimensional dataset.

Table 1: Default Parameters for the Clustering Algorithms

Algorithm	Parameters
DBSCAN	$minPts = 4$ $\epsilon = 0.005$
EM	$N = 59$
WC	$P\{reord\} = P\{dup\} = 0.01$ $RTT = 5ms$
NC	$P\{reord\} = P\{dup\} = 0.01$ $RTT = 5ms$
	$RTT_{MAX} = 10ms$ $P\{reord\}_{MAX} = P\{dup\}_{MAX} = 0.09$
NCI	as for NC, with $\rho_F = 1.5$

NC, and Incremental NetCluster, denoted as NCI. Unless otherwise specified, we used the set of parameters reported in Tab. 1 to run the algorithms. In particular, the RTT_{MAX}, $P\{reord\}_{MAX}$, and $P\{dup\}_{MAX}$ represent the Net_p parameter set. Experiments have been performed on an AMD Athlon 64 X2 Dual-Core PC with 2 Gbyte main memory running WinXp.

5.1. Datasets and feature selection

We first ran a set of experiments in which several connections to known servers were artificially generated. We considered a set of N HTTP mirrors located in different known geographical positions. For each HTTP server, we downloaded the same file L times. Therefore, a clustering algorithm should identify N clusters, each one including L objects. This scenario, named "known servers" in the remainder of the paper, is a simple case study that permits to know a priori the "correct" clustering. This idealized environment permits to detect if the clustering algorithms are able to correctly identify correlated groups of servers. Since the "optimal" solution is known, this trace allowed us to evaluate the quality of the clusters obtained by different algorithms.

To characterize each TCP flow, we exploited Tstat. Clustering algorithms are then run. Identified clusters are compared to the expected set of clusters (i.e., "optimal" solution). We run several experiments, considering first the set of $N = 59$ UBUNTU mirrors to download the distribution of "wget", and then the set of $N = 25$ sourceforge mirrors to download the distribution of "visualwget". Moreover, L is set to 100. The experiments were repeated both during the day and the night, to observe the impact of different network conditions.

For real traces, we presented the results for a 24-hours trace collected on Politecnico di Torino Campus LAN access link. We collected Internet traffic exchanged between our Campus LAN and the external network (i.e.,

Table 2: **The error percentage and the number of clusters by considering different sets of features**

Set of features	Error	Cluster number
$\{TTL, RTT, P\{reord\}, P\{drop\}, P\{dup\}\}$	12%	54
$\{TTL, P\{reord\}, P\{drop\}, P\{dup\}\}$	60%	32
$\{TTL, RTT\}$	40%	54
$\{RTT, P\{reord\}, P\{drop\}, P\{dup\}\}$	81%	12

Internet). This scenario, named "real trace" in the remainder of the paper, represents a fairly large dataset that covers a large set of Internet servers. TCP flows are again characterized by exploiting Tstat. 24 subsets of objects were considered, one set for each hour. Since in this scenario no a-priori information is available on the "optimal" clustering, we exploited different measures (e.g., cluster homogeneity [33], adjusted rand index [29]) to prove the effectiveness of the NetCluster framework.

In all scenarios, measurement data are pre-processed by partitioning them on the TTL value (see Section 3). Thus, TCP flows characterized by different TTL are analyzed in different clustering sessions.

Feature selection. Among the large set of available measurements describing flow behavior, we selected $\{TTL, RTT, P\{reord\}, P\{drop\}, P\{dup\}\}$ as the most useful information for node classification from a localization point of view. Our selection was based on theoretical aspects (see Section 3) and experimental analysis. In particular, we have studied the order of importance of the features by evaluating the performance of the NetCluster algorithm on the known servers dataset. The error percentage (computed as [33]) and the number of clusters obtained at the end of the clustering have been analyzed by considering different subsets of the considered features. **Tab. 2 reports the main results.** The most important parameters are the RTT and the TTL. In particular, when the TTL is not considered, the error is more than 80% and the number of clusters (i.e., 12) is very low with respect to the real value (i.e., 59). By removing the RTT from the analysis, the error is more than 60% and the number of clusters (i.e., 32). By only considering the $\{TTL, RTT\}$, the error is close to 40%. To minimize the error and identify a number of clusters close to the real value, all features need to be considered in the analysis.

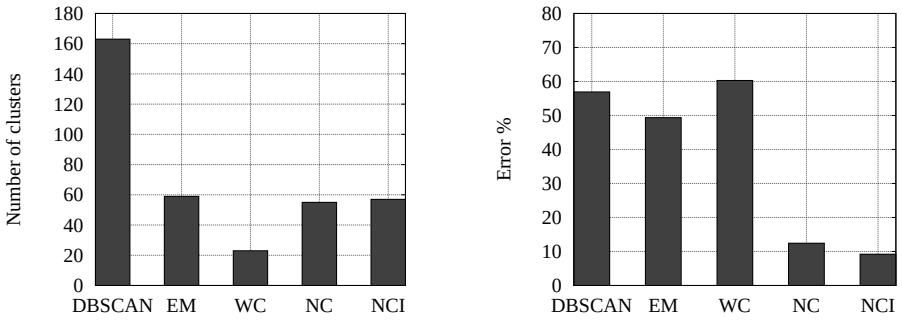


Figure 2: Number of identified clusters (left) and error percentage considering different clustering algorithms (right). The ground truth is 59 clusters

5.2. Clustering algorithm comparison

Clustering algorithms are compared to show their ability in correctly identifying clusters of nodes which exhibit similar network path properties. We addressed the following issues: (i) Characterization of clustering results and (ii) agreement measures between two clustering results.

Characterization of clustering results. The left plot of Fig. 2 reports the number of clusters identified on the known servers trace by the considered clustering algorithms (i.e., DBSCAN, EM, WaveCluster (WC), NetCluster (NC), and Incremental NetCluster (NCI)). Since the known server trace contains 59 UBUNTU mirrors, 59 clusters should be identified. EM requires the number of clusters as an input parameter. Hence, it (obviously) identifies the correct number of clusters. DBSCAN identifies a large number of small clusters, while WC tends to identify few, large clusters. Both NC and NCI, after the resize phase, identify a number of clusters very close to the expected one (55 and 57 respectively).

The right plot of Fig. 2 reports the error percentage in the clustering composition yielded by the different algorithms. For each cluster $C = 1, \ldots, K$, let $N_{OK}(C)$ be the number of flows belonging to the dominant server, i.e., the server to which refers the majority of flows in the cluster. Let $N_{KO}(C)$ be the number of flows in C referring to other servers and $|D|$ the total number of flows in the dataset. The error percentage [33] is then evaluated as

$$\eta = \frac{\sum_C N_{KO}(C)}{|D|} \times 100$$

The reported results show that NetCluster performs significantly better than the other algorithms, showing an error percentage of about 10%, whereas the other algorithms range around 50%. Note that EM, even if considering exactly N clusters, shows a large error percentage, because it does not explicitly address noise.

The good performance of the NetCluster algorithm is largely due to its (final) resize phase. Fig. 3 shows a graphical representation of part of the measurement space after the first two steps of NetCluster, i.e., quantization and wavelet transform, performed on the real trace. The resulting cell density versus $\{RTT, P\{reord\}\}$ is plotted for a given TTL value ($TTL = 52$). At this processing step, clusters referring to a given server already emerge. However, the group of cells with small RTT values forms a single set of connected cells, i.e., a single huge cluster. The NetCluster resize phase allows splitting this cluster into several smaller clusters, better partitioning the original dataset.

We further explored the composition of the clusters obtained after NetCluster resize phase. Fig. 4 plots the cluster homogeneity [33], defined as $100 \times$ the ratio between the number of objects from the prevailing server versus the total number of objects in the cluster. The higher the cluster homogeneity is, the higher the clustering quality. Clusters are sorted in decreasing values of homogeneity. 39 clusters contain only flows from a single server, while only 3 clusters are characterized by homogeneity smaller than 50%. Investigating further, the most heterogeneous clusters group together flows coming from mirrors very close to each other (e.g. Bern and Lausanne (CH)).

Since the NetCluster algorithm is an improved version of the WaveCluster algorithm, in the following we will consider the NetCluster algorithm as representative of both approaches.

Agreement measures between two clustering results. To compare clustering results obtained with different clustering algorithms, agreement measures (e.g., rand index [29], adjusted rand index [30]) between algorithm pairs are exploited. The Rand Index [29] computes the number of pairwise agreements between two partitions of a set. Hence, it may be exploited to provide a measure of similarity between the cluster sets obtained by two different clustering techniques.

Let O be a set of n objects, and X and Y two different partitions of set O to be compared. The Rand Index, denoted as R_{index}, is computed as follows:

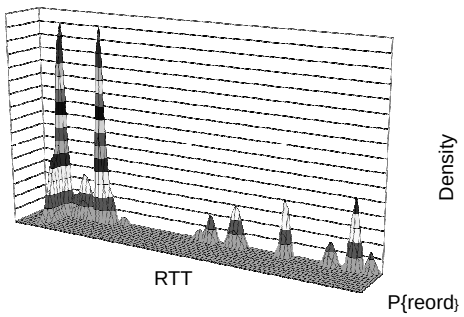


Figure 3: Graphical representation of cell density in the two-dimensional measurement space RTT (round-trip time), P(reord) (reordering probability).

$$R_{index} = \frac{a+d}{\binom{n}{2}} \tag{1}$$

where a denotes the number of pairs of elements in O which are in the same cluster both in X and Y, and d denotes the number of pairs of elements in O which do not belong to the same cluster neither in X nor in Y. Thus, the term $(a+d)$ is the number of pair wise agreements of X and Y, while $\binom{n}{2}$ is the number of different pairs of elements which can be extracted from O. The R_{index} ranges from 0 to 1, where 0 indicates that the two partitions do not agree for any data pair, and 1 indicates that the two partitions are equivalent.

Since the Rand index does not yield zero as expected value of two random partitions, the adjusted Rand index has been proposed in [30] as follows:

$$ARI = \frac{R_{index} - expected\_value}{maximum R_{index} - expected\_value} \tag{2}$$

where the expected value is computed as the weighted average of all values that the R_{index} can take and the maximum R_{index} is the maximum value that R_{index} can assume (i.e., 1). The adjusted Rand index is bound by 1 and is 0 when the index equals its expected value. Since the Rand index is always larger than the adjusted Rand index, the sensitivity of the adjusted Rand index is higher than that of the Rand index [30].

We first compared the clustering algorithms on the known servers trace. For this dataset, the correct clusters are known. Tab. 3 reports the values of the adjusted Rand index when separately comparing the result of each

Table 3: Known servers trace: Adjusted Rand Index

	Adjusted Rand Index
DBSCAN	0.25
EM	0.48
NC	0.87
NCI	0.86

Table 4: Known servers trace: Adjusted Rand index for the pairwise algorithm comparison

	NC	DBSCAN	EM	NCI
NC	1	0.29	0.21	0.85
DBSCAN	–	1	0.22	0.31
EM	–	–	1	0.24
NCI	–	–	–	1

clustering algorithm with the known solution. Both NC and NCI provide good quality clusters, which are very close to the correct partition.

We also computed the adjusted Rand index given by the pairwise comparison of the cluster sets provided by the DBSCAN, EM, NC, and NCI approaches. Tab.4 and Tab.5 report the obtained values for the known servers trace and for the real trace respectively. Consider the known servers. The index value is rather low for most algorithm pairs. Hence, most algorithms yield fairly different clusters. Not surprisingly, the (NC, NCI) index value is the highest, suggesting that the two algorithms provide rather similar data partitions. These results highlight that, even when two algorithms yield a similar number of clusters, the cluster composition may be quite heterogeneous. Similar considerations hold for the real trace.

5.3. Algorithm sensitivity and robustness

We analyzed the robustness of the clustering quality to parameter settings. For each considered algorithm, a set of experiments have been run

Table 5: Real trace: Adjusted Rand index for the pairwise algorithm comparison

	NC	DBSCAN	EM	NCI
NC	1	0.31	0.2	0.77
DBSCAN	–	1	0.2	0.32
EM	–	–	1	0.2
NCI	–	–	–	1

to find the optimal input parameter settings, using, when available, tools to optimize algorithm performance (e.g., OPTICS [27]).

The evaluation of the robustness of the clustering quality to parameter settings requires the validation of the discovered clustering structures. Performing the latter activity in the real world environment is the most difficult task because no a-priori information is known to evaluate the goodness of the resulting clusters. Thus, we only report results considering the dataset of known servers, i.e., the UBUNTU dataset collected during the day. This idealized environment permits to detect if the clustering algorithms are able to correctly identify correlated groups of servers.

The EM algorithm requires as input parameter the number of clusters, which is in general very difficult to define, given the wide range in which it may vary. Hence, the analysis of algorithm sensitivity and robustness has been addressed for the DBSCAN algorithm and for the NetCluster framework only. The characteristics of the NCI algorithm are analyzed in Section 5.4.

DBSCAN algorithm. Tab. 6 shows the percentage of correctly clustered flows when varying the input parameters ϵ (i.e., the radius of the neighbourhood) and $minPts$ (i.e., the minimum number of points within ϵ radius) of DBSCAN. As $minPts$ increases, the percentage of correctly clustered flows decreases, because more objects are labeled as noise[8]. The larger the radius ϵ, the larger the obtained clusters. Thus, also in this case, the number of correctly clustered flows decreases. Small values of ϵ may yield fragmentation and, thus, large error rates. Bold values reports the optimal parameters in our experimental setting (38.27%) and the parameters selected by the OPTICS algorithm [27] (17.64%). OPTICS has been proposed to simplify DBSCAN parameter setting. However, it selects a parameter configuration far from the optimal one, because its computation is based on the number of clusters and not on the error, which is unknown a priori. Hence, devising an appropriate parameter configuration for DBSCAN in a real operational environment may be a complex task.

Table 7 reports the corresponding number of clusters N_C for each parameter configuration. N_C is inversely proportional to both ϵ and $minPts$ (as expected). The parameter configuration provided by OPTICS yields 64 clusters, a number which is quite close to the real number of servers (59).

[8]**In our case the noise is the instance of data that the algorithm cannot cluster due to their lack of neighbors**

Table 6: DBSCAN: Percentage of correctly clustered flows

MinPts	Eps							
	0.0005	0.001	0.003	0.004	0.005	0.007	0.01	0.05
2	26.95%	31.78%	37.23%	38.25%	26.33%	24.75%	16.62%	5.99%
4	26.08%	30.96%	36.98%	**38.27%**	26.11%	24.57%	**17.74**%	5.95%
6	24.75%	29.33%	36.14%	37.60%	25.14%	24.15%	16.92%	5.95%
8	23.31%	28.19%	35.25%	36.46%	25.41%	24.57%	17.59%	5.95%
10	22.62%	26.58%	34.06%	35.77%	24.75%	24.30%	18.26%	6.19%

Table 7: DBSCAN: Number of clusters

MinPts	Eps							
	0.0005	0.001	0.003	0.004	0.005	0.007	0.01	0.05
2	434	385	368	355	286	231	112	32
4	171	193	210	**208**	163	136	**64**	27
6	93	125	149	151	124	99	52	27
8	57	82	109	111	93	78	53	27
10	46	67	82	88	79	68	47	29

208 cluster are instead found when the set of parameters that minimizes the error is selected. The cluster quality provided by DBSCAN is strongly dependent on the values chosen for the parameters. Hence, its robustness is rather limited. Furthermore, the $minPts$ and ϵ parameter values are not related to networking characteristics, thus preventing the exploitation of domain-knowledge when selecting the appropriate parameter configuration.

NetCluster framework. NetCluster parameters are the cell extensions in the different dimensions. In the following, we analyze RTT and P_{reord}, which are the most relevant. Indeed, P_{loss} and P_{dup} are typically negligible in our scenario given the very good connectivity of our campus LAN to the Internet. In more details, P_{dup} has been found to be not zero only in few cases, where possibly the path from the server was crossing some faulty links/interfaces. Fig. 5(a) reports the variation of the number of created clusters versus the RTT and P_{reord} cell extension parameters. The algorithm is rather sensitive to the variation of P_{reord}. A large number of clusters is obtained with very low values of P_{reord} (e.g., less than 0.5%), while the variation of RTT does not substantially affect the number of clusters.

Fig. 5(b) shows the total error by varying the RTT and P_{reord} cell extension parameters. The variation of RTT affects the total error. The error can be reduced by setting RTT=[2,5] ms as input parameter, which intuitively

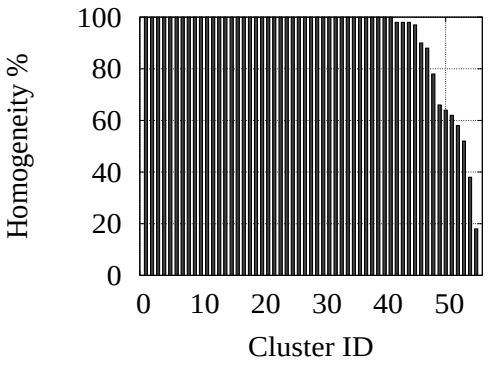
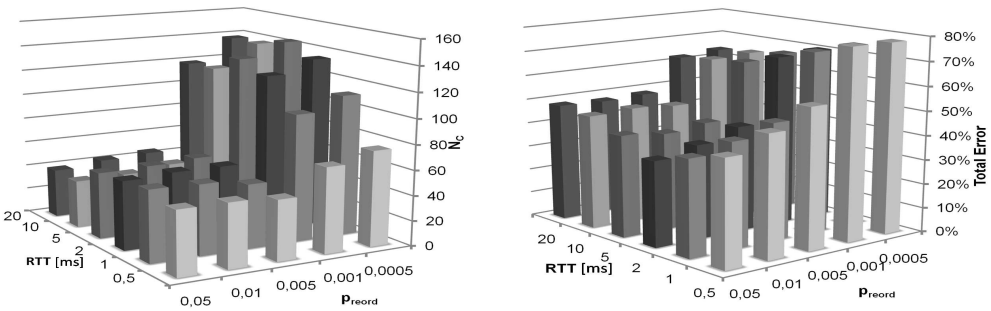


Figure 4: Clusters homogeneity considering NC.



(a) Number of clusters

(b) Total error in percentage

Figure 5: NetCluster framework on known server trace

Table 8: **Error percentage and number of clusters by varying ρ_F**

ρ_F	1	1.25	1.5	1.75	2
Number of clusters	76	63	55	54	54
Error percentage	12%	13%	12%	16%	17%

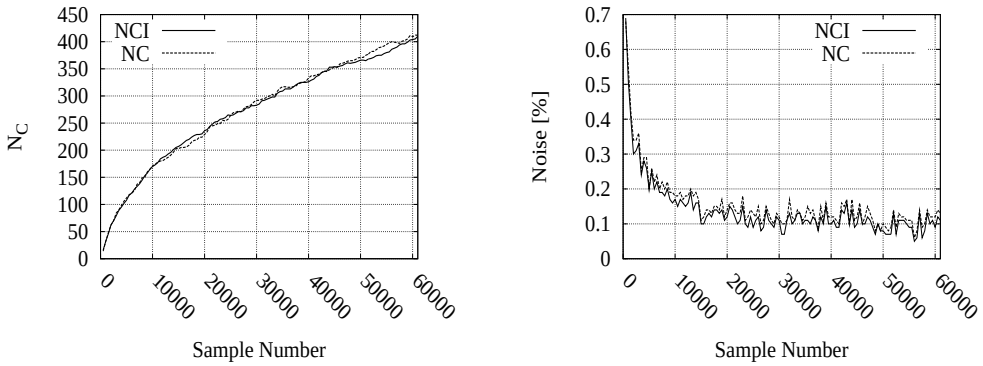
suggests that samples that differ less than 5 ms in RTT can be grouped in the same cell. Setting the parameters to "network-meaningful" values (e.g., $RTT_{MAX} = 10$ms or $P\{reord\} = 0.01$ as previously discussed) is relatively simple and provides a quite robust clustering quality.

To give more insight, we analyzed the error percentage and the number of clusters by varying ρ_F. As expected, the sensitivity to ρ_F is quite limited. For low values of ρ_F more clusters are generated, since a higher number of cells are considered. On the contrary, when ρ_F increases, more cells are considered, and the error is slightly increased. At last, we choose $\rho_F = 1.5$ so to ignore those cells with a too small number of point in them.

5.4. Incremental NetCluster performance

To analyze the performance of NCI, we simulated on-line incoming blocks of traffic flows. We considered traffic flows of the real trace collected from 11:00 a.m. to 4:00 p.m. The traffic trace was partitioned in batches containing 500 flows. Let N be the number of batches and i the index of the current batch (e.g., $B(i) = 500$ means that batch i contains 500 flows). To compare the clustering results obtained by means of the NC and NCI algorithms, for each batch i, the NC algorithm is run on all the first $k = 0, 1 \ldots i$ batches, while NCI is only run on the single batch $B(i)$. The performed experiments addressed the following issues: (i) Characterization of clustering results, in terms of cluster number, noise percentage, and adjusted rand index, and (ii) time performance of the two approaches.

Characterization of clustering results. The results shown in Fig.6 compare NCI and NC performance. For each sample in the set of batches B, Fig.6(a) plots the number of identified clusters, while Fig.6(b) plots the noise percentage of NCI and NC, i.e., the percentage of objects erroneously classified as noise. NC and NCI show a similar increasing trend in the cluster number, by identifying 10 clusters when they analyze the first batch $i = 0$, and more than 400 clusters for the last batch. Both NCI and NC show a fastly decreasing trend in the noise percentage, keeping the noise bounded



(a) Cluster number analysis (b) Noise Percentage

Figure 6: NCI versus NC on Internet traffic trace

Table 9: Adjusted Rand Index for Different Block Sizes considering the Real Clustering

Algorithms	Block Size		
	10	100	1000
NCI w.r.t. NC	0.84	0.91	0.81

below 20% after 17,000 samples.

Tab. 9 reports the adjusted rand index values given by comparing the cluster sets obtained by the NC and NCI algorithms when varying the block size. The real trace is discussed as representative dataset. The block size does not consistently affect the values of the adjusted rand index. This confirms that NCI and NC produce similar clusters.

Finally, Fig.6 reports the variation of the adjusted rand index between NCI and NC for the real trace as a function of the sample number, for different block sizes. Also in this case, the values of the adjusted rand index are very similar considering different blocks. In particular, the adjusted rand index values rapidly converge to 0.94.

Time performance. Fig. 8 shows the CPU time required to run the NC and NCI algorithms for different block size. A logarithmic scale is used to report the CPU time. The dataset including all TCP flows collected during the entire day (i.e., including night) is considering. The dataset contains 100,000 objects. NC requires roughly 2s to process each block, almost independently of the block size. On the contrary, NCI exhibits an exponential decrease in the required CPU time when reducing the block size. As shown in the previous paragraph, the quality of the final clustering structure is only

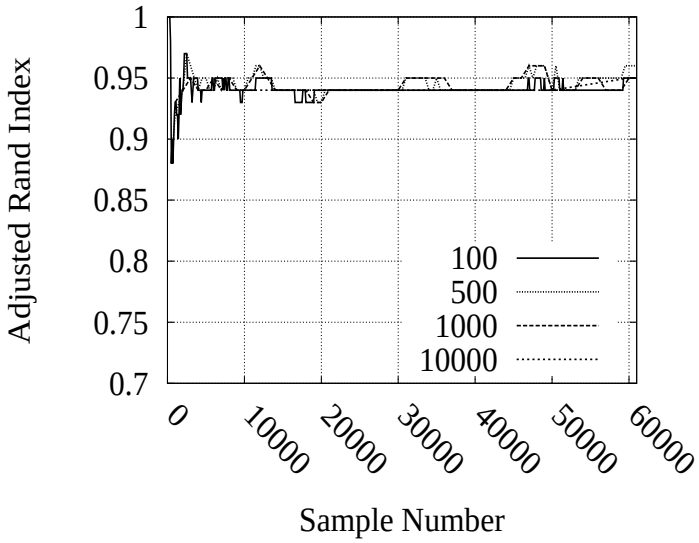


Figure 7: Adjusted Rand Index on Internet traffic trace for different block sizes.

marginally affected by the block size. Thus, it is possible to select the block size that better fits application domain characteristics or computational time constraints without reducing the clustering quality.

5.5. Characterizing Internet Traffic

In this section we discuss the results of the analysis performed on the real trace to characterize Internet traffic by means of the NetCluster framework. Recall that the real trace has been obtained from the Campus access link trace. The trace contains accesses to a large number of nodes, geographically distributed over the entire Internet. The 24-hours trace is split in 24 subsets, each lasting one hour. Since in this scenario no control information (i.e., no label) on the flows characteristics is available, we analyzed cluster homogeneity to evaluate the quality of the clustering results. We only discuss the results provided by the NetCluster framework, which has been shown in the previous sections to be the most reliable in finding homogeneous clusters. The other considered clustering algorithms were also tested, but the provided cluster homogeneity was lower than the one provided by NetCluster.

Fig. 9 depicts the number of flows with more than 100 packets (solid line) and the number of identified clusters (solid bars) for each hour. As expected, the number of clusters follows a day-night trend, as does the number of TCP

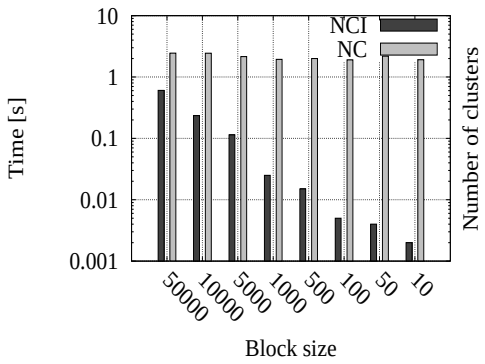


Figure 8: NetCluster execution time: NCI and NC versus block size.

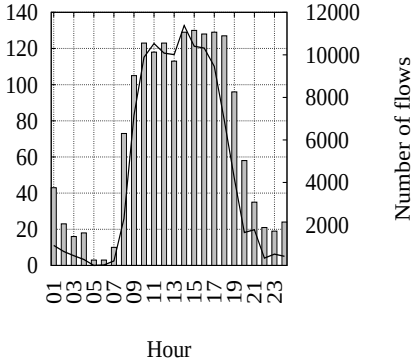


Figure 9: Cluster Numbers and TCP flows (solid line) in the dataset.

flows, due to the higher load offered during the day. In particular, during peak hours from 9:00 a.m. to 6:00 p.m., the number of clusters is almost stationary, varying between 110 and 130.

To show the quality of the identified clusters, Tab. 10 shows the IP addresses, the number of flows and the DNS server name[9] of objects belonging to the largest cluster (i.e., time slot 14:00). By looking at the IP addresses, it can be observed that 97% of the contacted servers are Google servers (belonging to different subnets), while only 11 servers are not registered by Google. However, all servers are located in Amsterdam, the Netherlands[10].

[9]The server name has been retrieved by performing DNS reverse lookup of the server IP addresses. The name is thus generic.

[10]The whois.net website was used to collect the information.

Table 10: Composition of a cluster

IP Address	**Flow %**	Number of Flows	Server Name
66.249.93.X	59%	217	ug-in-fX.google.com
66.249.91.X	22%	83	ik-in-fX.google.com
64.233.183.X	16%	58	nf-in-fX.google.com
82.94.210.200	2%	7	-
194.109.217.140	0.5%	2	emo.blender.org
62.50.24.217	0.5%	2	amst2.eu.psigh.com

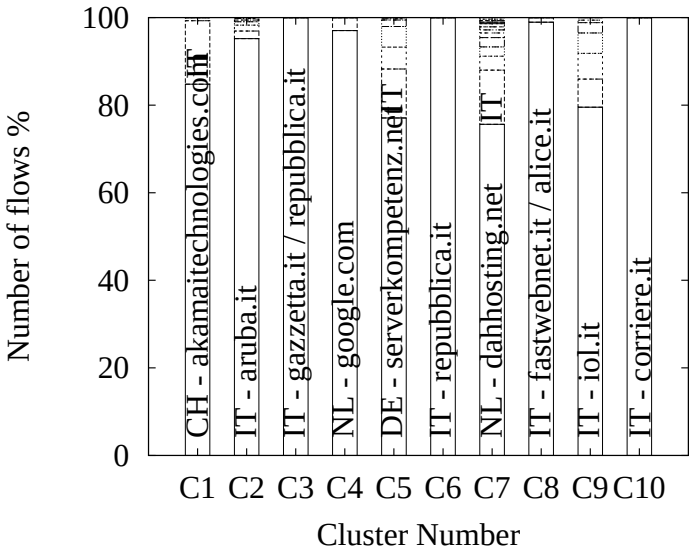


Figure 10: Graphical representation of cluster structure considering the 10 largest clusters. The dataset considers flows active from 12:00 to 13:00.

Fig. 10 reports the breakdown of the largest ten clusters, showing the countries to which the IP addresses belong to, and the largest ISP name, as provided by the WHOIS service. Cluster homogeneity is quite astonishing. For example C1, C3, C6, C8, and C10 contain objects which belong to a single ISP in a single Country, or even of a single service (e.g., C10). Other clusters, e.g., C5 and C7, group servers that are located in the same country mainly, with about 20% of flows that belong to other countries, which could possibly be verified further by using some other geolocation services. Overall, NetCluster is very effective in correctly grouping servers together, and could be seen as a promising technique to validate and improve the topological information discovery of servers.

6. Related work

In the past, several research groups have been attracted by Internet Tomography. Starting from the seminal work of Yardi [4], in which the authors study how to derive the traffic matrix from link load measurements, several authors studied how to infer internal network properties from data measurements. Most works focus on path and topology characteristics. For example,

in [10] and [11] the authors describe methods to infer the packet loss rates in internal networks routers from the traffic required among edge nodes.

Ref. [34] proposed to understand the Internets structure by only analyzing two passive measurements (i.e., the source IP address as unique identifiers of hosts and the TTL information). To generate IP sources that are topologically close to each other the version of the EM algorithm proposed in [35] has been used. Our work extends the one in [34] by considering a large set of passive measurements. We also proposed an incremental clustering approach to deal with the analysis of passive measurements captured in real-time.

Bottleneck link identification [12] is another research area of network tomography. This approach can be based on either sending active probes, or passively inferring informations exploiting TCP flow observation. Passive monitoring [13] is usually preferred, since it allows obtaining information without injecting additional traffic, thus having no impact on network status.

More recently, a new problem attracted the attention of the research community: How to define a network-wide positioning system to locate nodes. Node position knowledge is then exploited to improve network performance. For example, considering P2P systems, the knowledge of other peers position could be exploited when building the overlay topology, so that neighboring nodes are logically connected together to avoid exchanging data over long (and possibly congested) paths. Currently, some commercial application like [15] exploits geolocalization informations during the peer selection, though this feature is usually limited to the same Autonomous System. While several proposals have been defined to derive node position in the Internet (see [16] and [17] to cite only the seminal paper and the most recent one), to the best of our knowledge all previous works require the explicit cooperation of end systems and, possibly, specialized nodes (called landmarks) to achieve the goal. Thus, signaling and active probes are often adopted. Conversely, we pursue an approach purely based on passive measurements of data traffic.

A parallel effort has been devoted to designing and developing incremental clustering algorithms which are able to process new data as they are added to the collection. In particular, incremental clustering algorithms (e.g., [6], [5], [7]) are able to update the clustering structure after insertion and/or deletion of new objects. Among the previous approaches, the aim of the work presented in [7] is to identify clusters of objects characterized only by categorical (i.e., not numerical) features. Thus, this approach cannot be exploited to analyze traffic network data.

The COBWEB algorithm [6] is able to change the clustering structure

concurrently with the collection of new data. It clusters a dataset in the form of a classification tree, where the root node represents the whole dataset, the leaves represent single objects and internal nodes represent clusters. Each cluster is characterized by a probabilistic description. The tree is built incrementally with a single read of the dataset. When new data is available, it is temporally added to each cluster to compute a metric called *category utility*. This measure evaluates the similarity of the data belonging to the same cluster and the dissimilarity of data belonging to other clusters. New data is expected to improve the overall category utility. To achieve this goal four actions can take place: (i) New data is added to an existing class, (ii) a new cluster is created, (iii) an existing cluster is split, and (iv) two existing clusters are merged.

The incremental version of DBSCAN [5] is able to deal with both insertion and deletion of new objects, while the Incremental NetCluster algorithm only manages new object insertions. However, in this analysis domain, object deletion is not a relevant issue. On the other side, the Incremental NetCluster algorithm presented in this paper provides better performance than DBSCAN, in terms of both accuracy and efficiency.

The idea of exploiting clustering algorithms to derive node location information by analyzing Internet measured data collected via passive measurements was first introduced in [9]. This paper significantly improves over [9] by introducing Incremental NetCluster, a novel and scalable approach suitable for on-line analyses unfeasible with the approach in [9]. Furthermore, a significantly more extensive set of experiments has been performed to thoroughly explore the effectiveness of the NetCluster framework.

7. Conclusions

NetCluster is a framework that explois Internet passive measurements to derive node location information. Extensive tests performed both on artificial and Internet traffic traces prove that NetCluster performs better than several well known clustering algorithms. The Incremental NetCluster algorithm is very efficient, thus it is potentially able to deal with the analysis of measured data captured in real-time. The experimental results show that the NetCluster framework effectively allows obtaining network-wide information on the Internet structure, as seen at a vantage point.

Although we focus on segments generated by relatively long TCP connections characterized by at least 100 segments, the proposed approach can be

easily extended to provide appealing information to devise novel and more intelligent applications. For example, considering Content Delivery Networks, nodes could directly contact the closest server without leveraging on pure load-balancing techniques or centralized control schemes. Similarly, considering P2P applications, the knowledge of other peers location could be exploited to improve the structure of the overlay topology. This will be beneficial to the network as well, because it will enforce traffic flow locality properties.

Furthermore, up to now, NetCluster performs an a-priori dataset partition based on the TTL to increase the effectiveness of the clustering approach. Since load balancing strategies may affect TTL, we plan to address this issue by pushing ad-hoc technique in NetCluster.

8. Acknowledgement

The research leading to these results has received funding from the European Union under the FP7 Grant Agreement n. 318627 (Integrated Project "mPlane").

References

[1] V.Gehlen, A.Finamore, M.Mellia, M.M.Munaf, "Uncovering the big players of the web", Traffic Monitoring and Analysis - 4th International Workshop, TMA 2012, Vienna, March 2012.

[2] I.Poese, S.Uhlig, M.A.Kaafar, B.Donnet, B.Gueye, "IP geolocation databases: unreliable?", ACM SIGCOMM Computer Communication Review, vol.41, no.2, pp. 53, 56, April 2011.

[3] Y.Shavitt, N.Zilberman, "A geolocation databases study", Selected Areas in Communications, IEEE Journal on, vol.29, no.10, pp.2044,2056, December 2011.

[4] Y.Yardi, "Network tomography: estimating source-destination traffic intensities from link data", *Journal American Statistics Association*, v.91, pp.365-377, 1996.

[5] M. Ester M., H.P. Kriegel, J. Sander, H.P. Wimmer, and X. Xu. "Incremental clustering for mining in a data warehousing environment". *The*

24rd International Conference on Very Large Data Bases, pp. 323-333, 1998.

[6] D.H. Fisher. "Incremental knowledge acquisition via incremental conceptual clustering". *Machine Learning*, Springer, v. 2, n. 2, pp. 139-172, 1987.

[7] A. Pons-Porrata, G. Daz, M. Cortes, L. Ramrez. "An incremental clustering algorithm based on compact sets with radius alpha". *CIARP*, pp. 518-527, 2005.

[8] J.Han, M.Kamber, "Data mining: concepts and techniques", *Morgan Kaufmann*, San Francisco, 2006.

[9] E.Baralis, A.Bianco, T.Cerquitelli, L.Chiaraviglio, M.Mellia, "NetCluster: a Clustering-Based Framework for Internet Tomography", *IEEE ICC 2009*, Dresden, Germany, June 2009.

[10] M.Coates, R.Nowak, Y.Tsang, "Passive network tomography using EM algorithms", *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Salt Lake City, Utah, USA, May 2001.

[11] R.Caceres, N.G.Duffield, J.Horowitz, D.F.Towsley, "Multicast-based inference of network-internal loss characteristics", *IEEE Transactions on Information Theory*, v.45, n.7, pp.2462-2480, Nov. 1999.

[12] E.Brosh, G.Lubetzky-Sharon, Y.Shavitt, "Spatial-temporal analysis of passive TCP measurements", *IEEE INFOCOM 2005*, Miami, Florida, USA, March 2005.

[13] Y.Tsang, M.Yildiz, P.Barford, R.Nowak "Network radar: tomography from round trip time measurements", *ACM IMC 2005*, Taormina, Italy, October 2004.

[14] V.Aggarwal, A.Feldmann, C.Scheideler, "Can ISPs and P2P systems cooperate for improved performance?", *ACM SIGCOMM Computer Communications Review*, v.37, n.3, pp.29-40, July 2007.

[15] http://www.joost.com

[16] S.Ratnasamy, M.Handley, R.Karp, S.Shenker, "Topologically-aware overlay construction and server selection", *IEEE INFOCOM 2002*, New York, NY, USA, June 2002.

[17] C.Barakat, W.Dabbous, M.A. Kaafar, L.Mathy, K.Salamatian, T.Turletti, "Securing internet coordinate embedding systems", *ACM SIGCOMM 2007*, Kyoto, Japan, August 2007.

[18] M.Ester, H.P.Kriegel, J.Sander, X.Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise", *2nd Int. Conf. on Knowledge Discovery and Data Mining*, Portland, Oregon, USA, August 1996.

[19] R.Agrawal, J.Gehrke, D.Gunopulos, P.Raghavan, "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications", *ACM SIGMOD*, Seattle, Washington, USA, June 1998.

[20] S.Chatterjee, G.Sheikholeslami, A.Zhang, "WaveCluster: A multi-resolution clustering approach for very large spatial databases", *VLDB*, New York, NY, USA, August 1998.

[21] M.Ester, H.P.Kriegel, J.Sander, X.Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise", *Second International Conference on Knowledge Discovery and Data Mining*, 1996, pp.226-231.

[22] G.McLachlan, T.Krishnan, "The EM algorithm and extensions", *John Wiley and Sons*, Wiley series in probability and statistics, 1997.

[23] B.H.Juang, L.R.Rabiner , "The segmental K-Means algorithm for estimating parameters of hidden Markov models", *IEEE Trans. Acoust., Speech, Signal Processing*, v.9, n.38, pp.1639-1641, 1990.

[24] A.Finamore, M.Mellia, M.Meo, M.Munafò, D.Rossi, "Experiences of internet traffic monitoring with tstat", *IEEE Network, March/April 2011*, Vol. 25, No. 3, pp. 8–14, ISSN: 0890-8044, March/April 2011

[25] M.Mellia, M.Meo, L.Muscariello, D.Rossi, "Passive analysis of TCP anomalies", *Computer Networks*, Vol. 52, No. 14, pp. 2663–2676, ISSN: 1389-1286, 2008

[26] J.Rexford, J.Wang, Z.Xiao, Y.Zhang, "BGP routing stability of popular destinations," *ACM SIGCOMM Internet Measurement Workshop*, Marseille, France, November 2002.

[27] M.Ankerst, M.Breuning, H.Kriegel, J.Sander, "OPTICS: ordering points to identify the clustering structure," *SIGMOD Rec.*, v. 28, pp 49-60, June 1999.

[28] A.P.Dempster, N.M.Laird, D.B.Rubin, *Journal of the Royal Statistical Society. Series B (Methodological)*, v.39, n.1, pp.1-38, 1977.

[29] W.M.Rand, "Objective criteria for the evaluating of clustering methods," *J. Am. Stat. Assoc.*, v.66, pp.846-850, 1971.

[30] L.Hubert, P.Arabie, "Comparing partitions ," *Journal of Classification*, v.2, pp.193-218, 1985.

[31] F.Pukelsheim "The three sigma rule ," The American Statistician, v.48, n.2, pp.88-91 1994.

[32] S.Guha, R.Rastogi, K.Shim, "Cure: an efficient clustering algorithm for large databases", *ACM SIGMOD*, Seattle, WA, USA, June 1998.

[33] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to data mining*, second ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2006.

[34] B. Eriksson, P. Barford, R. Nowak. *Network discovery from passive measurements.* SIGCOMM 2008: 291-302

[35] A. K. Jain M. Figueiredo. *Unsupervised learning of finite mixture models.* IEEE Transaction on Pattern Analysis and Machine Intelligence, 24:381-396, 2002

[36] T. Hastie, R. Tibshirani, J. Friedman *The elements of statistical learning: data mining, inference, and prediction.* Second Edition. Springer. February 2009

[37] E. N. Nasibov and G. Ulutagay. *Robustness of density-based clustering methods with various neighborhood relations.* Fuzzy Sets Syst. 160, 24 (December 2009), 3601-3615.

[38] M.-.S Yang, C.-Y. Lai, and C.-Y. Lin. *A robust EM clustering algorithm for Gaussian mixture models.* Pattern Recogn. 45, 11 (November 2012), 3950-3961.

[39] L. Galluccio, O. Michel, P. Comon, E. Slezak, A. O. Hero (2009). *Initialization free graph based clustering,* I3S Laboratory Internal Report, ISRN I3S/RR-2009-08-FR, MAY 2009