International Journal of Computer Applications (0975 8887)
Volume 81 - No. 7, November 2013

Data Clustering for Anomaly Detection in Content-Centric Networks

Amin Karami
Computer Architecture Department (DAC)
Universitat Politécnica de Catalunya (UPC), Barcelona, Spain

ABSTRACT

Content-Centric Networks (CCNs) have recently emerged as an innovative trend to overcome many inherent security problems in the IP-based (host-based) networks by securing the content itself rather than the channel through which it travels. In this network architecture new kinds of attacks -ranging from DoS to privacy attacks- will appear. Therefore, it is becoming necessary to design a flexible and powerful mechanism to be able to detect them in an intelligent manner the first time they are employed. In this paper, a novel anomaly detection system has been proposed to detect known and previously unknown types of attacks using an efficient unsupervised learning engine that utilizes clustering with the optimal number of clusters, high detection rate, and low false positive rate in the same time over the CCN traffics flows. This paper compares the performance of five different clustering algorithms in the proposed anomaly detection system including K-means and Farthest First as Partitioning clustering, Cobweb as Hierarchical clustering, DBSCAN as Density-based clustering and Self Organizing Map (SOM) as Model-based clustering. Results show that DBSCAN method is the most efficient one for this purpose since it outperforms the other ones in terms of high detection rate and low false positive rate in the same time.

General Terms:

Keywords:

Content-centric networking, Anomaly Detection, Clustering Analysis

1. INTRODUCTION

Recently, Content-Centric Networking (CCN also referred to as Data-Centric Net-working, Named-Data Networking, or Information-Centric Networking) as the future Internet has emerged to overcome the inherent limitations of the current Internet such as, useful trust model, content security, and protection of privacy by shifting the security mechanisms from the network nodes to content itself [2, 14, 31]. The CCN not only eliminates many security problems in the current Internet by securing the data embedded in the content to avoid of transmission of fake contents but improves also the performance in terms of response time, bandwidth usage, and accessibility [3, 7, 12]. Unlike the host-based approaches in which security, integrity and trust should be enabled in the channel, the CCN secures content itself and puts integrity and trust as the content properties. On the other hand, with this new paradigm new kinds of attacks and security challenges -from Denial of Service (DoS) to privacy attacks- will arise that an efficient and effective security mechanism should be proposed to secure content, protect privacy, and defense against unknown and new forms of attacks [41].
This new Internet architecture should be resilient to existing DoS attacks, or at least limit their effectiveness, anticipate new and undetected (unknown) attacks that take advantage of its idiosyncrasies, and incorporate basic defenses in its design [13]. Thus, one of the most important challenges of CCN design is develop a method or protocol so that detect these new forms of attacks (intrusions) and any deviation. Designing an anomaly detection system is a major approach in attempt to solve the attack (intrusion) detection problem [32]. In this sense, unsupervised learning for anomaly detection can be used to cluster traffic flows without prior knowledge to discover a pattern of traffic behaviors and find features inherent and their deviations to the problem [5, 36]. To the best of the author's knowledge, this is the first time a proposal on unsupervised learning for anomaly detection is made for Content-Centric Networks. The main contribution of this paper is to design the anomaly detection system for detection of unknown and new types of attacks and their variants using an efficient clustering with the optimal number of cluster, high detection rate and low false positive rate over the Content-Centric Network traffic flows.
The rest of the paper is as follows. Section 2 presents structure of CCN and potential DoS attacks. Clustering algorithms are described in Section 3. In Section 4, internal clustering validation is proposed. Experimental setup and discussion are presented in Sections 5 and 6. Finally, the conclusion draws in Section 7.

2. DOS ATTACKS IN CCN

All communications in CCNs are performed using two distinct types of packets: *Interest* and *Data (Content)*. The main idea in the CCN is that, an Interest request for a content object is routed towards the location of the origin content where it has been published. Any router or middle node on the way checks its cache for matching copies of the requested content. If a cached copy of any piece of Interest request is found, it is returned to the requester along the path the request came from. On the way back, all the middle nodes store a copy of content in their caches to answer to probable same Interest requests from subsequent consumers (users) [37, 40]. Each CCN router maintains three major data structures:

(1) Pending Interest Table (PIT): hold all not yet satisfied Interests that sends upstream towards potential data sources. Each PIT entry holds one or multiple incoming and outgoing physical interfaces with related Interest packets.

(2) Forwarding Interest Base (FIB): forward Interests to one or multiple physical network interfaces based on the forwarding strategies.

(3) Content Store (CS): temporarily buffers data packets for data retrieval efficiency.

In this paper, current potential DoS attacks in CCNs are considered. There are new attack opportunities in the forms of DoS attacks to make either content unavailable or deny service to users [13, 18, 41]:

(1) To make content unreachable: a source can be disrupted by sending large numbers of new and distinct Interests (Interest Flooding Attacks (IFA)) or an attacker can decline the common cache efficiency by overloading the cache when a cache receive a legal traffic. When attackers get high access control in a router, they can make disruption in routing by do not forwarding requests or enforce misbehaving in Pending Interest Table (PIT) routers in order to prevent content retrieval.

(2) To serve fake responses: an attacker can make routers believe a valid content is invalid and reply a "not valid" response, deliberately. A content can also be spoofed by injecting fake responses that are not signed or are signed with a wrong key, hoping that the user accepts the response in source. An old content (which may be unsecured) signed with the right key can be also replaced with the original one, or an attacker may get high access to the source's signing key to sign content with the correct key [39].

3. CLUSTERING ALGORITHMS

Data clustering is a data exploration technique that attempts to find groups of data based on similar characteristics to separate data objects into meaningful and reasonable groups [29]. In order for the measurement of similarities between data objects, distance metric plays an important role. While many distance measures exist, Euclidean distance is one of the most commonly used and easy metric [38]. A large number of clustering algorithms exist. The choice of the appropriate clustering algorithm depends on the type of data set and the particular purpose [5]. In this paper, the performance of the five frequently used and powerful clustering algorithms are implemented and compared in our anomaly detection system namely, K-means and Farthest First as Partitioning clustering, Cobweb as Hierarchical clustering, DBSCAN (Density-Based Spatial Clustering of Applications with Noise) as Density-based clustering and Self Organizing Map (SOM) as Model-based clustering. Partitioning approach produces sphere-like clusters where it constructs a single level (flat) partition of a data set with N data objects into K clusters. The objects in a cluster are more similar (minimum distance) to each other than to objects in other clusters. Partitioning methods usually start with a random partition and refines partitions iteratively to constructs different groups by maximizing the homogeneity within the cluster by minimizing the square error [9]. Hierarchical approach forms a classification tree or a sequential presentation scheme over the objects into meaningful categories [34]. Density-based approach finds clusters as dense regions that are separated by regions of lower density. The density region continues growing the given cluster as long as the density in the neighborhood exceeds some threshold. This type of clustering is designed to discover clusters of arbitrary shapes which are not necessarily convex [23]. Model-based approach is based on the probability model from the data that attempts to optimize the fit between the data set and some mathematical models.

3.1 K-means clustering

K-means clustering algorithm groups objects by a predefined parameter, K (the number of clusters) which objects are classified with similar values or patterns into K distinct clusters. The five steps of the K-means clustering algorithm is as follows [26]:

(1) Define the number of clusters K,

(2) Place randomly the initial group (K cluster) of centroids,

(3) Assign each object to the group with closest centroid. The distance between each cluster centroid to each object is measured by distance metrics (e.g., Euclidean distance). Then, objects are assigned to a specific cluster with a minimum distance,

(4) Recalculate the positions of the centroids in each group,

(5) Repeat step 2 until the centroids do not change any more.

K-means is computational ease and relatively memory efficient by $O(T \cdot K \cdot N)$, where N is the number of objects, K number of the clusters, and T number of the iterations. In contrast, it often terminates at a local optimum. It needs to specify K in advance, unable to handle noisy data and sensitive to outliers, does not work well with overlapping clusters, and no suitable for discovering clusters that are not hyper-ellipsoids or hyper-spheres [22].

3.2 Farthest First Clustering

The Farthest First algorithm is an implementation of the Farthest First Traversal (FFT) by Hochbaum and Shmoys (1985), which is fast, approximate, and a variant of K-means. It places each cluster center in turn at the point furthermost from the existing cluster center. This point must lie within the data area. This greatly speeds up the clustering in most of the cases since less reassignment [1]. The time complexity of FFT is $O(N \cdot K)$, where N is the number of data objects and K is the number of clusters.

3.3 Cobweb Clustering

The Cobweb [11] is one of the most commonly used algorithms in conceptual clustering which maximizes category utility to build a probabilistic hierarchical tree. The detail of Cobweb algorithm is described in [21], but the main function is defined as follows:

(1) Initialize the tree by reading in an object to form the root node,

(2) Read in the next object, and start from the root node,

(3) If the node is a leaf, go to step 4, else the current node is an internal node. Then, choose the one of maximizing category utility (locally) and repeat:
   (a) Insert the object into each of the children of the current node, and choose the one with highest category utility observed,
   (b) Create a new leaf for the object and add the leaf to the children of the current node,
   (c) Merge the two children found in the first step that have the highest and the second highest values of category utility by turning the two children of the current node into two children of the merged node. Then, insert the object into the merged node,

(d) Split the child found in Insert step with the best category utility by raising the level of its children up one level, so that the children of this node become the direct children of the parent of this node. This node is eliminated. Then the object is inserted as in step Insert,

(4) When a leaf node is reached, create a new leaf node to hold the objects, then both the old leaf node and the new leaf node are added to a new nodes as its two children, finally, the new node with two children replaces the original place of the old leaf node to become an internal node of the tree,

(5) Check the stopping condition, if it is satisfied, terminate; else, go to step 2 to begin a new iteration.

3.4 DBSCAN Clustering

The DBSCAN algorithm is power and commonly used density-based clustering algorithm with the time complexity of $O(N \cdot logN)$. The DBSCAN algorithm is particularly suited to deal with large databases, with noise, and is able to identify the clusters with different sizes and shapes [10]. It is mostly insensitive to the ordering of the points in the database. In contrast, it does not respond well to data sets with varying densities. Clusters are identified by looking at the density of points. The regions with a high density of points illustrate the existence of clusters, in contrast regions with a low density of points present clusters of noise or outliers. This algorithm needs two input parameters: Eps (the radius of the neighborhood area of point) and $MinPts$ (minimum number of points that must exist in the Eps-neighborhood). The pseudocode of DBSCAN algorithm is as follows [25]

(1) Select an arbitrary point P from data set,

(2) Retrieve all points density reachable from P by Eps and $MinPts$,

(3) If P is a core point then a cluster is formed,

(4) If P is a border point then no points are density reachable from P and DBSCAN visit the next point of the data set,

(5) Continue the process until all of the points have been processed.

3.5 Self-Organizing Map (SOM)

The SOM algorithm or Kohonen Map [16] is capable for the large data sets because the computational complexity scales linearly with the number of data. It does not need high memory, and can be implemented both in a neural, online learning manner as well as parallelized [19]. It belongs to the category of competitive learning networks, in which the output neurons compete with each other to be activated, with the result that only one is activated at any one time. This activated neuron is called a Winner-Takes-All (WTA) neuron or simply the winning neuron. Then, the lateral inhibition connections are implemented between the neurons. The result is that the neurons are forced to organize themselves. SOM involves four major stages for n-dimensional input space and m output neurons:

(1) Initialize weights vectors w_i for neuron i, $i = 1, ..., m$, randomly.

(2) An input vector x is randomly chosen to compute the distance (using a distance metric such as Euclidean distance) between x and neurons i.

(3) Determine winner (best matching) neuron k: $||W_k - X|| = min_k||W_i - X||$

(4) Update all weights vectors of all neurons i in the neighborhood of neuron k: $W_i(t+1) = W_i(t) + \eta_{k,i}(t)h_{c,i}(t)[X - W_i(t)]$, where $\eta_{k,i}(t)$ is adaption coefficient, $h_{c,i}(t)$ is neighborhood kernel centered on the winner unit:

$$h_{c,i}(t) = exp(-\frac{\|(r_b - r_i)^2\|}{2\sigma^2(t)}) \quad (1)$$

Where r_b, r_i are positions of neurons b and i on the SOM grid. Both $\eta(t)$ and $\sigma(t)$ decrease monotonically with time [35].
In the other words, the weights of every node are updated at each cycle by adding: current learning rate ($\eta_{k,i}(t)$) * degree of neighborhood with respect to winner ($h_{c,i}(t)$) * difference between current weights and input vector to the current weights.

(5) Iterate the step 2-4 until the sufficiently accurate map is acquired.

4. INTERNAL CLUSTERING VALIDITY MEASURES

Many clustering algorithms require the number of clusters by the user before running the algorithm. There is no completely satisfactory and ideal method for defining the number of the clusters for any type of clustering analysis [35]. Obviously, the quality of clustering is significantly dependent on the estimation of the correct K. Data clustering with too many partitions complicates the results for analyzing and interpreting, while too few partitions lead to the loss of information and misdirect the final decision [30]. One of the most widely used techniques for estimation of the optimal K and clustering validation are relative criteria where the algorithms run repetitively using different parameters and input values until the validity of clustering results are compared in two categories: internal and external [17, 23]. The internal validation measures are employed to choose both the optimal cluster number without any additional information and the best clustering algorithm. The internal validation measures are also the only option for clustering validation when there is no external information available, such as labeled data set [24]. *Compactness* and *Separation* are two major criteria that make similar data objects within the same clusters and place other objects in distinct clusters [33]. The compactness (intra-cluster cohesion) measures how near the data points in a cluster are based on variance. The lower variance indicates the better compactness. The separation (inter-cluster separation) measures how a distinct (well-separated) cluster is far from other clusters. In spite of variety of internal evaluation methods, in most application, expert judgments are still the key. Decision making based on numerical representations might not be individually appropriate [15]. In this work, three widely used internal validation measures have been applied:

(1) Davies-Bouldin (DB) [6] calculates the similarities between each cluster C and other clusters, and the highest value is assigned to C as its cluster similarity. Then the DB measures the average of similarity between each cluster. As the clusters should be compacted and separated, the lower DB means better clustering result.

(2) Calinski-Harabasz (CH) index [4] evaluates the cluster validity based on the average between clusters and within clusters. The maximum value of CH means better cluster configuration.

(3) Dunn index [8] is based on the minimum pairwise distance between objects in the different clusters as the inter-cluster separation and the maximum diameter among all clusters as the intra-cluster compactness. The larger value of Dunn means better cluster configuration.

Table 1. Internal clustering validity measurement

Validity index	Notation	Formula	Optimal value
Davies-Bouldin index	DB	$\frac{1}{NC}\sum_i max_{j,j\neq i}\frac{[\frac{1}{n_i}\sum_{x\epsilon C_i} d(x,c_i)+\frac{1}{n_j}\sum_{x\epsilon C_j} d(x,c_j)]}{d(c_i,c_j)}$	Min
Calinski-Harabasz index	CH	$[\sum_i n_i d^2(c_i,c)/(NC-1)]/[\sum_i\sum_{x\epsilon C_i} d^2(x,c_i)/(n-NC)]$	Max
Dunn index	Dunn	$min_i\{min_j(\frac{min_{x\epsilon C_i, y\epsilon C_j} d(x,y)}{max_k\{max_{x,y\epsilon C_k}\}d(x,y)})\}$	Max

n: number of objects in data set; c: centers of data set; NC: number of clusters; C_i: i-th cluster; n_i: number of objects in C_i; c_i: center of C_i.

5. EXPERIMENTAL SETUP

We have implemented CCN flows on a testbed configuration running the CCNx software of PARC (`www.ccnx.org`). A full duplex link interconnects the nine Linux (Ubuntu) machines from the three different countries (Spain, Finland and Brazil) including two servers (place of the origin content) and seven clients. Figure 1 shows this construction. We performed the following experiments with the main tools in CCNx: ccnsendchunks, ccncatchunks2, ccnputfile, ccngetfile, ccndsmoketest, HttpProxy, and ccnchat. We launched the ccndc routing daemon on the all hosts to configure the CCNx FIB (Forwarding Interest Table) and started this routing utility in the background by ccndstart daemon. For generation of the normal traffic, we uploaded some objects (small and large files) into the CCN repository with ccnsendchunks C API on the server side. It is a program used to chop contents in small data units called chunks and inject them from stdin to ccn. In the other side, two clients send Interest packets under the given CCNx URI (the CCNx name) with ccncatchunks2 to get desired contents and write them to stdout. We also load some files with ccnputfile (It publishes a local file as content with the CCNx name) in the CCNx repository, and retrieves the published content with ccngetfile (It retrieves content published under the CCNx name and writes it to the local file). We used HttpProxy application to run a HTTP proxy that converts HTTP Gets to CCN Interests. The ccnchat utility also allows one(s) to join an existing chat channel (or create a new one if that particular channel does not exist). For generation of abnormal traffics, we ran ccndsmoketest daemon for smoke-test of ccnd. This utility used to send the large number of Interests (Interest flooding attacks) toward servers (we called 'Smoke attack'). We also launched ccndsmoketest again from some clients to simulate a distributed Interest flooding attack. Then, we made abnormal traffics to saturate channels by sending very small contents (decreasing buffer size) from owner of origin content to requesters (we called 'AbnormalSource' behavior). Moreover, we do not forwarding content requests deliberately to requesters with killing (shutting down) ccnd cleanly (we called 'AbnormalUnreachableContent' behavior). Finally, we ran wireshark plugin for capturing CCNx packets.

5.1 Feature Construction

We employed the simple features (attributes) that are extracted from the header's area of the network packets. These intrinsic features are available in many networks, for example, the duration (length of the connection), source host, destination host, source interface, and destination interface [20]. We also used three features in each 2 seconds time interval: (1) Total number of packets sent from and to the given interface in the considered time interval, (2) Total number of bytes sent from and to the given interface in the considered time interval, and (3) Number of different source-destination pairs matching the given hostname-interface that being observed in the considered time interval. The number of packets and bytes allows to detect anomalies in traffic volume, and the third features allows to detect the network and the interface scans as well as the distributed attacks, which both result in an increased number of source-destination pairs [27]. Since generated CCN flows are very large (about 100,000 records), we sampled the data using an appropriate scheme with 30% of total traffics until clustering algorithm could be run in a timely manner. The evaluated data set (CCN flows) includes 78% normal traffic and 22% abnormal traffic.

5.2 Performance Metrics

To evaluate clustering performance and accuracy in our system we were Interested in some criteria. Firstly, we applied purity. The purity criterion ([0 1]) determines the frequency of the most common category into each cluster:

$$Purity = \frac{1}{n}\sum_{q=1}^{k}\max_{1\leq j\leq l} n_q^j \tag{2}$$

Where, n is the total number of samples; l is the number of the categories, n_q^j is the number of samples in cluster q that belongs to the original class $j(1 \leq j \leq l)$. A larger purity is desired for a good clustering.

we also used two typical indicators to quantify performance: *Detection Rate (DR)* and *False Positive Rate (FPR)*. The *Detection Rate* is the number of intrusion instances detected by the system and the *False Positive Rate* is the number of the normal instances that were incorrectly classified as intrusion [28, 36]. We calculated these two indicators over the labeled data through equations 3 and 4, respectively:

$$DR = \frac{TruePositive}{TruePositive + FalsePositive} \tag{3}$$

Where, *True Positive* is the number of the attack instances when successfully detected, and *False Positive* is the number of normal is detected incorrectly as attack.

$$FPR = 1 - \frac{TrueNegative}{TrueNegative + FalsePositive} \tag{4}$$

Where, *True Negative* is the number of the normal instances are detected as a normal (there is no attack).

Applied data set (CCN flows) consists of the different scale and distribution of attributes that these differences make it difficult to measure the similarities. To reduce the varied measurement units and scale, the data set was transformed into normal form so that all attributes can have equal impact on the computations. Normalization is the most commonly used method with an average of zero and standard deviation of one.

The proposed anomaly detection system in CCN consists of five phases that is depicted in Figure 2. Firstly, pre-processing of the data set including feature selection and normalization is required. Secondly, we apply five clustering algorithms on normalized data set. We run clustering algorithms in several times with different

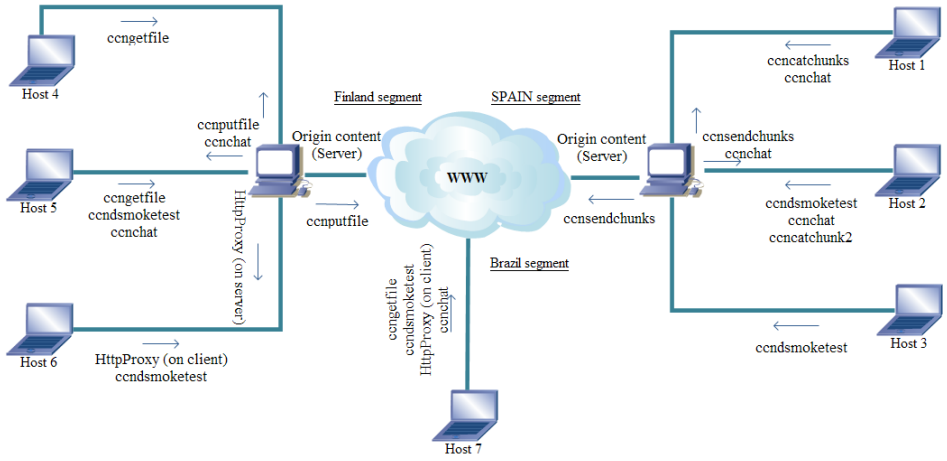

Fig. 1. Content-Centric Network Scenario

input parameters. For each clustering result, we measure the goodness of clustering with internal validation measurements and keep the three best results. Thirdly, we apply three typical criteria to check the performance of each clustering. Finally, the best clustering algorithm based on the performance measurement is selected.

6. EXPERIMENTAL RESULTS AND DISCUSSION

We conducted our experiments on Intel Pentium 2.13 GHz CPU, 3 GB RAM running Windows 7 Ultimate. Clustering techniques, clustering validity and performance measurement methods implemented on WEKA and MATLAB software.

6.1 DBSCAN Results

DBSCAN requires two input parameters: Eps and MinPts. We experimented with several values for Eps (between 0.01 and 4) and for MinPts (between 1 and 300). Then, we selected the best three results obtained from validity indices. As shown in Table 2, DB and CH present very high purity (purity=0.9936) with the maximum detection rate (DR=100%) and the low false positive rate (FPR=5.62%), with K=438, Eps=0.01 and MinPts=1. Second optimal clustering result is K =109 from DB index with purity=0.9916, Eps=0.01 and MinPts=20. According to the results, high purity with low DR indicates that the high portion of the abnormal traffics have distributed into the large normal clusters. Also the high purity with the high FPR indicates that the most normal traffics place in several small attack clusters.

Table 2. Results of DBSCAN algorithm implementation

DBSCAN result						
Criteria	Value	Eps, Pts	K	Purity	DR (%)	FPR (%)
DB	0.0442	0.01, 1	438	0.9936	100	5.62
	0.0697	0.01, 20	109	0.9916	89.96	8.85
	0.0718	0.02, 120	8	0.9801	32	16
CH	119810	0.01, 1	438	0.9936	100	5.62
	46200	0.05, 2	139	0.989	42.5	3.6
	42600	0.06, 2	106	0.989	42.5	2.2
Dunn	1.367	0.25, 1	21	0.981	77.22	18.27
	1.2661	0.3, 5	16	0.981	77.27	14.44
	0.7521	0.3, 240	8	0.98	72	17.7

6.2 K-means results

The K-means algorithm was considered with its input value (number of clusters), ranging between 30 and 400. The best three values of validity indices are shown in Table 3. The best combination of DR and FPR was obtained with K=240, which achieved a DR of 94.21% and a FPR of 6.74% from Calinski-Harabasz (CH).

Table 3. Results of K-means algorithm implementation

K-means result					
Criteria	Value	K	Purity	DR (%)	FPR (%)
DB	0.2736	165	0.989	61.58	8.19
	0.2796	250	0.9891	73.24	14.19
	0.2841	155	0.989	55.79	3.3
CH	68925	230	0.989	68.57	9.52
	61900	220	0.989	75.44	10.26
	61867	240	0.989	94.21	6.74
Dunn	1.296	65	0.989	52.5	6.5
	1.009	70	0.989	54.79	8.04
	0.882	135	0.989	65.58	8.96

6.3 Farthest First Results

For the farthest first algorithm we tried with the K between 30 and 300 (Table 4). In spite of very high purity value, there is no satisfaction DR and FPR. Insignificant results show that the farthest first algorithm could not correctly place the normal and abnormal traffics into the appropriate clusters.

6.4 Cobweb Results

The cobweb algorithm ran with the range of Acuity (minimum standard deviation for numeric attributes) between 0.01 and 2 with 30 experiments. According to Table 5, DB and CH indexes propose the good validity measurements, but their results are not ideal due to the low DR and the high FPR. The inconsistent results confirm that the cobweb algorithm realize too many normal data as noise or outlier in small clusters.

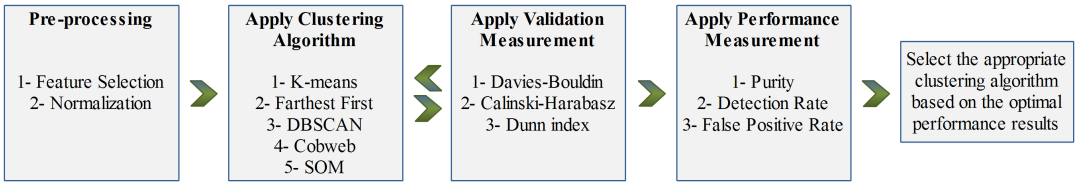

Fig. 2. Anomaly Detection System Design in CCN

Table 4. Results of Farthest First algorithm implementation

Farthest First result					
Criteria	Value	K	Purity	DR (%)	FPR (%)
DB	0.1366	260	0.993	71.03	48.29
	0.1519	130	0.993	71.03	45.12
	0.1601	110	0.993	70.08	42.12
CH	78945	250	0.97	93.17	18.22
	72901	210	0.96	95.44	20.16
	62451	240	0.959	91.91	19.74
Dunn	1.0527	50	0.981	56.68	26.8
	0.7726	40	0.981	56.68	23.37
	0.0911	60	0.981	66.67	30.52

Table 5. Results of Cobweb algorithm implementation

Cobweb result					
Criteria	Value	K	Purity	DR (%)	FPR (%)
DB	0.125	176	0.9895	64.58	31.9
	0.1583	36	0.9801	72.97	27.38
	0.1752	44	0.9801	72.97	41.01
CH	42790	276	0.9936	81.08	32.5
	35810	176	0.9895	64.58	31.9
	17320	94	0.981	74.85	32
Dunn	1.1105	64	0.978	67.46	26.12
	1.0034	58	0.978	65.41	21.89
	0.0964	60	0.982	63.18	27.51

6.5 SOM Results

Table 6 displays the three optimal results of clustering by 28 combination from 6-by-6 to 20-by-20 (height and width) neuron SOM. The learning rate also was assigned to 0.1, 0.5, 1, respectively for each combination. The obtained results are not satisfactory for anomaly detection with the low DR and the high FPR.

Table 6. Results of SOM algorithm implementation

SOM result						
Criteria	Value	neuron	K	Purity	DR (%)	FPR (%)
DB	0.0806	11-by-12	132	0.989	67.57	12.27
	0.0817	14-by-15	210	0.989	52.5	17.09
	0.0936	19-by-20	380	0.9924	47.03	34.28
CH	26013	10-by-11	110	0.989	44.79	13.11
	24602	10-by-10	100	0.9882	40.54	21.74
	20287	9-by-10	90	0.9882	40.55	11.83
Dunn	1.311	10-by-11	135	0.982	57.5	8.53
	1.101	14-by-15	182	0.982	57.79	11.04
	0.914	17-by-18	110	0.967	53.58	7.42

6.6 Anomaly Detection Performance Measurement

We continue our experiment by applying optimal algorithm results with the high detection rate (more than 85%) and the low false positive rate (under 10%) on the aforementioned data set for the anomaly detection classified into three types of intrusion: 'Smoke attack', 'AbnormalSource' behavior, and 'AbnormalUnreachableContent' behavior. The DR and FPR results are shown in Figures 3 and 4, respectively. This experiment shows that the DBSCAN (Eps=0.01, MinPts=1, K=438) is able to detect all three attacks without any false positive rate. Unfortunately, this algorithm with K=109 failed to accurately detect 'AbnormalUnreachableContent' (DR=0, FPR=4.25%) and 'AbnormalSource' (DR=74.58%, FPR=6.98%) behaviors. The K-means (K=240) detects considerably 'Smoke' attack, 'AbnormalSource' behavior (DR=100%, FPR=0), and 'AbnormalUnreachableContent' behavior with DR= 74.58% and FPR=6.98%.

Within the experimental results, we confronted with some ineligible results, either high purity with low DR and/or low FPR (e.g., Farthest First, Cobweb, SOM), or with the acceptable results by too many numbers of cluster (DBSCAN with K=438). This could because of the inherent structure of CCN which is summarized to two types of packet: Interest and Data (Content). The further research is to improve this preliminary results.

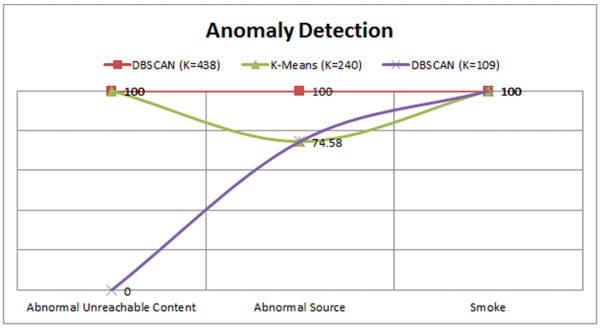

Fig. 3. Anomaly detection using DBSCAN (K=438 and 109) and K-means (K=240) clustering algorithms

7. CONCLUSION AND FUTURE WORK

In this paper, an anomaly detection system has been proposed to detect intrusions in forms of DoS attacks in Content-Centric Networks (CCNs) based on unsupervised learning algorithms. This anomaly detection system is able to detect intrusions by an appropriate performance measurement. The promising results has been obtained using DBSCN (K=438) with high purity, high detection

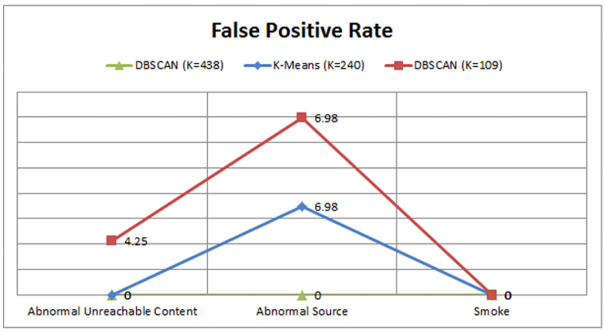

Fig. 4. False positive rate using DBSCAN (K=438 and 109) and K-means (K=240) clustering algorithms

rate, and low false positive rate. The K-means (K=240) also performs very well in detecting 'Smoke attack' and 'AbnormalUnreachableContent' behavior but it did not perform well in detecting 'AbnormalSource' behavior. We are currently working on a novel protocol to mitigate the most security challenges in CCNs, which this paper is our first experimental results and a base in order to continue other sections of our protocol. The next step in our research is to work towards several improvements of the presented approach by designing a system for effectively identifying the other types of intrusions and anomalies by optimal performance.

8. ACKNOWLEDGMENT

I would like to thank Mr. Mohammad Hovaidi Ardestani from Aalto University in Finland for his valuable support in modeling and generating CCN traffics.

9. REFERENCES

[1] S. B. Aher and L.M.R.J. Lobo. A comparative study for selecting the best unsupervised learning algorithm in e-learning system. *International Journal of Computer Applications*, 41(3), 2012.

[2] B. Ahlgren, Ch. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A Survey of Information-Centric Networking (Draft). In *Information-Centric Networking*, number 10492 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2011. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.

[3] M. H. Ardestani, A. Karami, P. Sarolahti, and J. Ott. Congestion control in content-centric networking using neural network. In *Talk and Presentation in CCNxCon 2013, 5-6th September*. Parc (Xerox Co.), California, USA, 2013.

[4] T. Caliński and J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1):1–27, 1974.

[5] G. Corral, E. Armengol, A. Fornells, and E. Golobardes. Explanations of unsupervised learning clustering applied to data security analysis. *Neurocomput.*, 72(13-15):2754–2762, 2009.

[6] D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, 1979.

[7] A. Detti, N. Blefari-Melazzi, S. Salsano, and M. Pomposini. Conet: a content centric inter-networking architecture. In *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, ICN '11, pages 50–55, 2011.

[8] J.C. Dunn. Well separated clusters and optimal fuzzy partitions. *Cybernetics*, 4:95–104, 1974.

[9] J. Erman, M. Arlitt, and A. Mahanti. Traffic classification using clustering algorithms. In *Proceedings of the 2006 SIGCOMM workshop on Mining network data*, MineNet '06, pages 281–286, New York, NY, USA, 2006. ACM.

[10] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD'96*, pages 226–231, 1996.

[11] Douglas H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Mach. Learn.*, 2(2):139–172, 1987.

[12] G. Fortino, C. Mastroianni, M. Pathan, and A. Vakali. Next generation content networks: trends and challenges. In *Proceedings of the 4th edition of the UPGRADE-CN workshop on Use of P2P, GRID and agents for the development of content networks*, UPGRADE-CN '09, 2009.

[13] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang. Dos and ddos in named-data networking. *CoRR*, abs/1208.0952, 2012.

[14] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, CoNEXT '09, pages 1–12, 2009.

[15] A. Karami, R. Johansson, and M. Riveiro. Utilization and comparison of multi attribute decision making techniques to rank bayesian network options, 2011. Master thesis in University of Skövde, Sweden.

[16] T. Kohonen. *Self-Organizing Maps*. Springer, Berlin, Heidelberg, 1995.

[17] F. Kovacs, C. Legany, and A. Babos. Cluster validity measurement techniques. Technical report, Department of Automation and Applied Informatics, Budapest University of Technology and Economics, Budapest, Hungary, 2002.

[18] T. Lauinger. Security & scalability of content-centric networking, September 2010.

[19] R. D. Lawrence, G. S. Almasi, and H. E. Rushmeier. A scalable parallel algorithm for self-organizing maps with applications to sparse data mining problems. *Data Mining and Knowledge Discovery*, 3:171–195, 1999.

[20] W. Lee and Salvatore J. Stolfo. A framework for constructing features and models for intrusion detection systems. *ACM Trans. Inf. Syst. Secur.*, 3(4):227–261, 2000.

[21] M. Li, G. Holmes, and B. Pfahringer. Clustering large datasets using cobweb and k-means in tandem. In *Proceedings of the 17th Australian joint conference on Advances in Artificial Intelligence*, AI'04, pages 368–379, Berlin, Heidelberg, 2004. Springer-Verlag.

[22] B. Liu. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*. Data-Centric Systems and Applications. Springer, 2007.

[23] O. Maimon and L. Rokach. *Data mining and knowledge discovery handbook, Chapter 15 CLUSTERING METHODS*, pages 321–352. The Kluwer International Series in Engineering and Computer Science. Springer, 2005.

[24] U. Maulik and S. Bandyopadhyay. Performance evaluation of some clustering algorithms and validity indices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1650–1654, 2002.

[25] K. Mumtaz and K. Duraiswamy. An analysis on density based clustering of multi dimensional spatial data. *Computer Science and Engineering*, 1(1):8–12, 2010.

[26] A. P. Muniyandi, R. Rajeswari, and R. Rajaram. Network anomaly detection by cascading k-means clustering and c4.5 decision tree algorithm. *Procedia Engineering*, 30:174–182, 2012. International Conference on Communication Technology and System Design.

[27] G. Münz, S. Li, and G. Carle. Traffic anomaly detection using k-means clustering. In *Proc. of performance, reliability and dependability evaluation of communication networks and distributed systems, 4 GI / ITG Workshop MMBnet*. Hamburg, Germany, 2007.

[28] J. F. Nieves and Y. Jiao. Data clustering for anomaly detection in network intrusion detection. Technical report, 2009.

[29] D. T. Pham, S. S. Dimov, and C. D. Nguyen. Selection of k in k-means clustering. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 219(1):103–119, 2005.

[30] D. Wunsch R. Xu. Survey of clustering algorithms. *IEEE Transactions on In Neural Networks*, 16(3):645–678, 2005.

[31] D. V. Rooy and J. Bus. Trust and privacy in the future internet-a research perspective. *Identity in the Information Society*, 3(2):397–404, 2010.

[32] F. Seredynski and P. Bouvry. Anomaly detection in tcp/ip networks using immune systems paradigm. *Comput. Commun.*, 30(4):740–749, 2007.

[33] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[34] K. Thompson and P. Langley. *Concept formation in structured domains. In Concept formation: Knowledge and experience in unsupervised learning*. Fisher, D.H., Pazzani, M.J., & Langley, P. (Eds.) San Francisco. CA: Morgan Kaufmann, 1991.

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

[36] Q. Wang and V. Megalooikonomou. A performance evaluation framework for association mining in spatial data. *Intelligent Information Systems*, 35(3):465–494, 2010.

[37] I. Widjaja. Towards a flexible resource management system for content centric networking. In *In Proc. of IEEE ICC'12 Next Generation Network Symposium*, 2012.

[38] I.H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition*. The Morgan Kaufmann Series in Data Management Systems. Elsevier Science, 2005.

[39] W. Wong and P. Nikander. Secure naming in information-centric networks. In *Proceedings of the Re-Architecting the Internet Workshop*, ReARCH '10, pages 1–12, 2010.

[40] M. Xie, I. Widjaja, and H. Wang. Enhancing cache robustness for content-centric networking. In *INFOCOM'12*, pages 2426–2434, 2012.

[41] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, G. Tsudik B. Zhang, K. Claffy, D. Krioukov, D. Massey C. Papadopoulos adn T. Abdelzaher, L. Wang P. Crowley, and E. Yeh. Named data networking (ndn) project. In *In Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, number PARC TR-2010-3, pages 68–73, 2010.