applied sciences

Article

Application Communities Detection in Network

Shuzhuang Zhang *, Yingjun Qiu, Hao Luo and Zhigang Wu

Institute of Network Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China; qiuyingjun@bupt.edu.cn (Y.Q.); luohao@bupt.edu.cn (H.L.); wuzhigang@bupt.edu.cn (Z.W.)
* Correspondence: zhangshuzhuang@bupt.edu.cn

Received: 1 November 2018; Accepted: 14 December 2018; Published: 22 December 2018

Abstract: The continuous growth of Internet traffic and its applications causes more difficulties for analyzing Internet communications. It has become an increasingly challenging task to discover latent community structure and find abnormal behavior patterns in network communication. In this paper, we propose a new type of network community—the application community—which can help understand large network structure and find anomaly network behavior. To detect such a community, a method is proposed whose first step is aggregating the nodes according to their topological relationships of the communication. It then clusters different application nodes according to the communication behavior modes in the same topological partition. Empirical results show that this method can accurately detect communities of different applications without any prior knowledge. In addition, it can identify the communities more accurately than other methods. Thus, this research greatly benefits the administration of IoT and cyber security.

Keywords: application community; communication behavior mode; topological relationship; cluster

1. Introduction

As internet hosts and applications continue to grow, it becomes increasingly important to understand large network structure and traffic patterns between the interactions of nodes for efficient network management [1–4] and security [5–7]. However, with the increasing supply of bandwidth and diversification of applications of the Internet, detailed analysis of all traffic streams in large-scale networks has become challenging for network measurement and administration. In recent years, researchers have been focusing on the behavior modes of network nodes from the perspective of network communities.

Compared with monitoring the hosts one by one, analyzing the overall characteristics and behavior modes of network communities is equivalent to "compressing" the original data [8]. Meaning, we only need to study a small number of hosts and their traffic flows to determine the overall characteristics and behavior modes of the communities. Therefore, we can understand the overall status of the network more quickly and effectively. Studying network communities can be used for unknown network traffic detection [9,10], network traffic analysis [11,12], botnet detection [13,14], and application identification [15]. Such research can also provide an important basis for network administrators to perform resource configuration and virus detection, which is significant for maintaining network security.

Typically, there are two types of methods for defining communities. One is to define communities based on the similarity of hosts' communication behaviors. For example, in a complex network, a community is defined as a set of nodes [16,17] that are closely related to each other and sparsely connected to the outside based on connectivity between nodes. While the other is to define communities based on the closeness of the communication relationships. For example, for computer networks, a community is generally considered to be composed of a group of hosts with common goals [18]. That is, a host cluster that accesses the same websites or uses the same services is called a community.

Therefore, community detection methods can be divided into two categories. The first one is based on topological partitioning [11,19–21] and the second one is based on host behavior clustering [22–24].

However, the communities detected by these two methods can only show that the member hosts in the communities have similarities in a certain aspect. With the increasing variety of network applications, these two communities cannot fully reflect the current state of the hosts or the use of network resources. This is because the clustering method is easily affected by encryption and obfuscation technologies. When a node is involved in multiple applications at the same time, it is difficult to apply topological partitioning for accurate community division. For example, a content delivery network node caches the contents of multiple websites. All hosts accessing this node are generally regarded as a cluster, but cannot be distinguished by the different websites accessed.

In this paper we propose a new type a community and its corresponding detection method. Compared with the above traditional communities, it has two important characters:

1. It uses the <IP, Port> two-tuple as a communication node to describe communication relations. Thus it can identify the roles (server nodes or client nodes) of nodes more accurately, which benefits the measurement and analysis of the entire application traffic mode of interaction.
2. It combines communication topological relationship and traffic behavior clustering to obtain more accurate community identification results. For example, we will find IP nodes that carry multiple services simultaneously and distinguish between normal users and malicious users who are accessing a service port on the same host.

The rest of this paper is organized as follows. In Section 2 we discuss the related work on community detection. In Section 3 we define the application community and introduce the method of community detection. In Section 4 we show and evaluate the results of community detection. In Section 5 we analyze and discuss typical communities in network to illustrate the practical value of this method and conclude this paper.

2. Related Work

Network communities that distinguish different behavior modes are important for network security administration and therefore of significant concern by researchers. According to different goals, community detection methods are divided into two categories: host-based behavior clustering and topology-based partitioning. These two methods are introduced as follows.

In network communication, the traffic streams generated by the same network application show similar behavioral modes. With these modes, the communication behavior of the hosts can be quantitatively represented, achieving host classification. However, because the number of applications and hosts in the network cannot be determined in advance, unsupervised and semi supervised clustering methods are often used to distinguish hosts with different behavior modes. Terzi et al. [25] extracted features such as the ports, target Internet protocol addresses (IPs), and loads to describe the communication behavior of the source IPs, and then clustered the source IPs to identify the botnet. Wei et al. [24] identified active hosts from network traffic data, selected relevant features from the message headers of these hosts, and built a host tree diagram using hierarchical clustering algorithms to find host clusters with similar behaviors. Jakalan et al. [19] designed an algorithm to find important IP nodes, extracted 15 communication mode features, used the dbscan clustering algorithm to obtain the host cluster, and analyzed the clustering results by comparing the feature values of the hosts between different clusters. Shadi et al. [26] aggregated the IPs in the terabyte traffic data into a tree structure according to the amount of data transmitted and the associated address blocks to find the IP block with the largest traffic in an enterprise network. Dewaele et al. [22] proposed nine features to describe the host communication mode. He then used the unsynchronized clustering method of the minimum spanning tree to distinguish different types of hosts.

Since the communication behavior exhibited by the hosts is similar to the social behavior in the social network. The community detection method in the social network is also used in the computer network. Iliofotou et al. [23] proposed a method for classifying IPs based on tags that only needs the connection between IPs and the usage of a few IP hosts to classify all IPs. Xu et al. [11,21] proposed a bipartite-based method to study hosts with similar behaviors by analyzing the communication relationships between hosts. Their method first divides the entire network into two independent sets according to the network segments in which the hosts are located. The connection between the two sets represents the communication relationship between the hosts. This method then uses a single-mode projection to count the visits of hosts on the same network segment to hosts outside the network segment, and constructs a similarity matrix. Finally, the host clustering method is used to divide the hosts on the same network segment into several clusters based on behavior modes. Similarly, Jakalan et al. [19,20] constructed a bipartite graph using the Net Flow data obtained from a boundary route. In order to describe the similarities of IPs in the administrative domain, they constructed a bipartite graph and applied a single-mode projection to construct a similar matrix to obtain the final community partitioning result.

Both of the above-mentioned methods consider the hosts (or IPs) as a whole, and partition a host (or IP) into a unique community. In fact, one client can use multiple applications at the same time, and one server can also provide multiple services at the same time, which allows one host (or IP) to associate with multiple communities at the same time. To improve the analysis of traffic flow interactions, in this paper we propose a method to define and detect network application community. This method considers the hosts providing different services and participating indifferent applications, so that the communication relationship can be more clearly expressed. After segmenting the communication relationship diagram, we cluster the hosts according to data transmission modes and finally identify communities that are associated with different applications.

3. Application Community

3.1. Application Community Definition

Traditionally, a community is defined as a set of nodes that are closely related to each other and sparsely connected to the outside based on connectivity between nodes. While in this paper, we need to consider both communication relationships and traffic behavior characteristics to distinguish multiple services carried by the same IP nodes and the normal and malicious connections that are accessing a service port on the same host. The nodes in each application community satisfy the following conditions.

1. There is a group of service nodes and other member nodes have communication relations to these service nodes;
2. When the member nodes communicate with service nodes, the characteristics of data transmission modes are similar.

When the set of nodes that meet these two conditions, we call it an application community (here in after referred to "the community"). In Figure 1, Servers A and B provide different services. Clients 1–7 and Servers A and B communicate based on the connection. The line type is used to indicate different data transmission behaviors. According to the above description of communities, the nodes in the graph can be divided into three communities, namely {A, 1, 2}, {A, 3, 4, 5}, and {B, 4, 5, 6, 7}. Among these, Servers A and B are leader nodes determining the community type, and Clients 1–7 are member nodes that communicate with the leader nodes to obtain a certain service. Although {1, 2, 3, 4, 5} communicates with the A node, {A, 1, 2} and {A, 3, 4, 5} do not belong to the same community because {1, 2} and {3, 4, 5} differ in data transmission behavior. The difference in data transmission behavior may because that (1) Server A provides multiple services at the same time or (2) Client {1, 2} or {3, 4, 5} access the server in an unconventional manner. Using the method proposed in this paper, one can identify different communities when (1) the same node participates in different applications, and (2) the same service node provides normal services or is attacked.

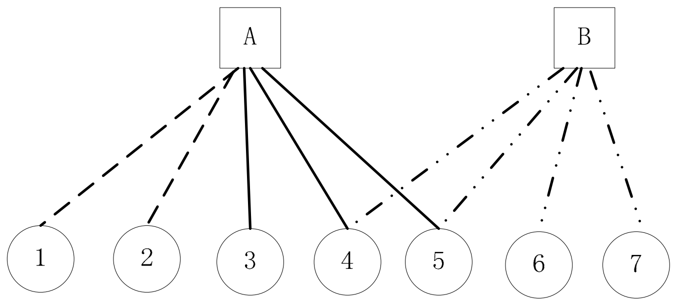

Figure 1. Diagram of application community.

3.2. Application Community Detection Method

According to the definition of an application community, the community detection method proposed in this paper is divided into two steps. The first step is to divide the network relationship diagram into multiple partitions according to the nodes' communication relationships. A partition consists of a leader node (usually a server), multiple member nodes (usually clients), and a connection between them (traffic). One or more services or applications may exist in a partition. Therefore, according to the statistical characteristics of the traffic flow, the nodes in the same partition are further divided to find a set of hosts whose connectivity and traffic characteristics are similar, which is the final community identification result.

3.2.1. Network Topological Partitioning Based on Communication Relationship

A network application community is usually composed of a service node and multiple client nodes. If you can find the server in the network first, then the server and its neighboring hosts will naturally form a partition, which is helpful for identifying the community.

Diagrams are used here to represent network communication relationships overtime. A large number of previous studies [8,9,11,18–21] use IP as a node in the communication. If data are transmitted between two IPs, the nodes represented by the two IPs are connected by one side. There are two limitations to this representation. First, it is impossible to determine from the topology whether an IP is a server or a client. For example, in Figure 2, node 1 may indicate that (1) one server receives many client visits or that (2) one client uses multiple applications to communicate with multiple servers at the same time. Second, it is impossible to identify the situation (through topology) when a service node provides multiple services. Since there is the possibility that an IP provides multiple services, even if node 1 in Figure 2 is known as a server node, it is not known exactly which services are provided and which clients correspond to each service.

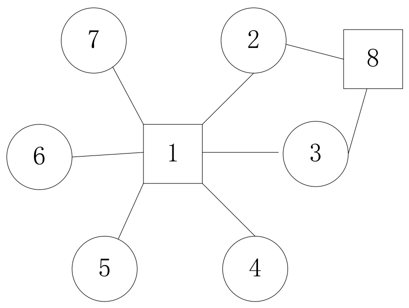

Figure 2. Communication relationships.

For most network application services, they open a fixed port to respond to requests from the client. The use of the port is very helpful for accurately determining the type and quantity of services provided by a server. Therefore, in this paper, the <IP, Port> two-tuple is used as a node in the network topology diagram. The communication relationship between the nodes is represented by a connection. There are two advantages of doing this:

1. The server node and client node can be accurately distinguished. When a client communicates with a server, it usually uses multiple ports and the server usually only uses a fixed port. This difference in ports allows the server nodes to have a large degree of power, so that the identity of a node can be clearly identified. Even if an IP accesses multiple servers at the same time, the phenomenon that the server nodes have a large degree of power will not change.
2. It can be identified that the same IP bears multiple services. The same IP will open different ports for different services, so the <IP, Port> two-tuple as a node can split a server IP into multiple topology nodes according to different ports. Each node corresponds to a service provided by the server IP. A client node's access to various services can also be easily identified.

Server nodes and client nodes are form a common structure—the C/S structure. There is usually one service node and multiple client nodes. In this structure client node only communicates with the service node and does not communicate directly with other client nodes. Therefore, there is a significant difference between the server node and client node in the number of adjacent nodes. Which means, for a certain network service or application, a server node has many adjacent nodes and a client node has few adjacent nodes. We define the adjacency matrix A as follows.

$$A_{ij} = \begin{cases} 1 & i \text{ and } j \text{ has a connection, } i \neq j \\ 0 & \text{Other} \end{cases} \tag{1}$$

For node i, the definition of a set is

$$P_i = \{p | A_{ip} = 1\} \tag{2}$$

$$Q_i = \{q | A_{pq} = 1, p \in P_i, q \notin P_i, q \neq i\} \tag{3}$$

where $|P_i|$ and $|Q_i|$ represent the number of elements in sets P_i and Q_i, respectively. In Figure 1, for all nodes $i = 1, 2, 3, 4, 5, 6, 7, 8$, their P_i and Q_i values are shown in Table 1.

Table 1. P_i and Q_i for each i.

i	P_i	Q_i
1	{2, 3, 4, 5, 6, 7}	{8}
2	{1, 8}	{3, 4, 5, 6, 7}
3	{1, 8}	{2, 4, 5, 6, 7}
4	{1}	{2, 3, 5, 6, 7}
5	{1}	{2, 3, 4, 6, 7}
6	{1}	{2, 3, 4, 5, 7}
7	{1}	{2, 3, 4, 5, 6}
8	{2, 3}	{1}

It can be seen from Table 1 that the core node in the C/S structure, $|P_i| > |Q_i|$, and the edge nodes, $|P_i| < |Q_i|$. Therefore, for any node, its $|P_i|$ and $|Q_i|$ values can reflect its type or identity in the network. In order to verify this idea, here we use the NetFlow log collected on an enterprise boundary route, with <IP, Port> as the node, to calculate the $|P_i|$ and $|Q_i|$ values of each node. According to the well-known service port (20, 21, 22, 53, 80, 123, 443, 1080, 8080), we label the client and service nodes. The results are shown in Table 2, indicating that 94% of the server nodes satisfy $|P_i| > |Q_i|$ and 98% of the client nodes satisfy $|P_i| \leq |Q_i|$.

Table 2. $|P|$ and $|Q|$ values for different types of nodes.

	$\mathbf{|P| > |Q|}$	$\mathbf{|P| \le |Q|}$	**Total Number of Nodes**
Server nodes	5006	318	5324
Client nodes	2122	154,789	156,911

Based on the above idea, the communication entities in a network (nodes represented by <IP, Port>) can be divided into different subgraphs. The specific process is shown in Algorithm 1, where the variable c is the subgraph number to be assigned. The vector C records the subgraph number, which each node belongs to. It skips if a node has been assigned a subgraph number. If the value of $|P_i| + |Q_i|$ is very small (not larger than 2), the partitions formed by node i and its neighbors are small enough to constitute an influential subgraph.

Algorithm 1 Network Relationship Division

```
Input: list of nodes
Output: list of partition numbers to which the node belongs
Initialize all element values in C to $-1$, $c = 0$
For $i$ in nodes:
    If C[$i$] ! = $-1$:
        continue
    End If
    calculate $P_i$ and $Q_i$
    If $|P_i| + |Q_i| > 2$ and $|P_i| > |Q_i|$:
        C[$i$] = $c$
        For each $j$ in $P_i$:
            C[$j$] = $c$
        End For
        $c$ += 1
    End If
End For
return C
```

3.2.2. Node Clustering Based on Traffic Behavior

In most cases, the result of node partitioning obtained in Section 3.2.1 can identify an application community in the network. However, there may be malicious entities sending malicious messages, such as distributed denial-of-service (DDos) attacks, worms, and Trojans, based on the inherent communication relationship. Therefore, in this paper we further differentiate the communication traffic flow according to the statistical characteristics of the transmission data before obtaining the final community identification result. The members of a community obtained through the above two step shave a very high similarity in both communication relationship and communication mode.

After the division process in Section 3.1, entities with close communication relationships are divided into the same partition. In most cases, a pair of IPs will only communicate with one type of service in a short period of time. Therefore, traffic flows between entities in the same partition can be aggregated according to IP pairs to extract data transmission behavior characteristics. There are two advantages to doing this:

1. Increasing the number of sample scan avoid the influence of outlier traffic flows on the results of community partitioning, which can reflect the data transmission behavior between a pair of IPs in more detail.
2. After clustering, the number of instances used for clustering is greatly reduced, reducing the time for clustering calculations.

The clustering features selected in this paper are identified by the five-tuple (<source IP, source port, destination IP, destination port, proto>), and the statistical information of each traffic flow is recorded, such as the number of packets and the number of bytes in each direction. During the clustering process, all traffic flows between each pair of IPs are combined and a feature vector is extracted to describe the data transmission behavior between the pair of IPs. Each vector includes these several elements:

1. Number of nonrepeating ports used by the source IP.
2. Number of nonrepeating ports used by the destination IP.
3. Average value of the protocol number.
4. Minimum number of up flow packets.
5. Median number of up flow packets.
6. Maximum number of up flow packets.
7. Minimum number of down flow packets.
8. Median number of down flow packets.
9. Maximum number of down flow packets.
10. Minimum average packet length for up flow.
11. Average median packet length for up flow.
12. Maximum of average up flow packet length.
13. Minimum of average down flow packet length.
14. Median of average down flow packet length.
15. Maximum of average down flow packet length.

Features 1 and 2 reflect the usage of a pair of communication IP ports. Feature 3 is used to reflect the usage of a pair of communication IP transmission layer protocols. In this paper, we only study traffic flows with protocol numbers 6 (TCP) and 17 (UDP). Features 4–9 reflect the distribution of the number of IP transmission packets, while Features 10–15 reflect the distribution of the length of a pair of IP transmission packets. By extracting the above features, the data transmission behavior between pairs of nodes with in a partition can be quantitatively described.

In order to unify the weights of each dimension, we normalize the actual (IP pair) feature values in the same partition before clustering. In this paper, we use maximum and minimum scaling to scale the feature to [0, 1]. If there is an N pair of IP pairs (i.e., instances of clustering) in a partition, the scaling formula is

$$f_{ik} = \frac{F_{ik} - \min(F_k)}{\max(F_k) - \min(F_k)} \tag{4}$$

where i represents the instance number to be clustered and F_{ik} the i-th eigenvalue of the k instance. $k = 1, 2 \ldots, 15$, and F_k represents a sequence of all the eigenvalues of the k-th dimension of all instances to be clustered. Values of $\max()$ and $\min()$ represent the maximum and small values of F_k, respectively. Variable f_{ik} is the normalized eigenvalue.

Finally, in this paper we use the dbscan-clustering algorithm [27] to process all session traffic in each subgraph. This algorithm is a density-based clustering algorithm that does not require the number of clusters to be specified. Without any prior knowledge, the number of types of traffic flows in a partition and the number of flows in each category are uncertain. Therefore, it is suitable to use the dbscan algorithm. When the dbscan algorithm runs clustering, there are three important parameters to specify:

1. eps

This parameter is the maximum distance that is allowed to constitute a cluster between instances. In this paper, we set the parameter value to 1.0. If the distance between two instances is less than the value, they are considered to be within the same cluster. If an instance is more than the distance of eps from all other instances, it is considered an outlier and to be labeled as -1. To simplify the end result, we consider that all outliers belong to the same community.

2. min_samples

The parameter min_samples denotes the minimum number of instances in a cluster. In this paper, the parameter value is set to 2. This means, there are at least two instances in a cluster. This parameter, together with eps, determines how densely the instances are in the same cluster. For different data sets, the values of these two parameters should also be adjusted accordingly.

3. metric

This parameter is the way the distance between instances is calculated. In this paper, we use the Manhattan distance to indicate the similarity between two instances. For instances i and j, the calculation formula of the Manhattan distance between them is

$$\text{Manhattan distance}(i,j) = \sum_{k=1}^{15} \left| f_{ik} - f_{jk} \right| \tag{5}$$

The dbscan algorithm takes the normalized eigenvector matrix as input. Each row of the matrix represents a traffic flow between a pair of IPs and each column represents a feature dimension. The algorithm outputs an array of labels, each element of the array corresponding to each row of the matrix. The tag is used to indicate the class of the traffic flow represented by the corresponding vector. All IPs involved in the same traffic flow are the results of the final network community identification.

4. Experimental Results

4.1. Datasets

To obtain the experimental dataset, we collected two weeks of traffic logs from the boundary route [28] of an enterprise network that contains 150 active users. The dataset contained approximately 30 million traffic flows, including 209 intranet nodes and 54,147 outer network nodes. The logs were stored in a format similar to Netflow and exported as readable text.

4.2. Experimental Results

4.2.1. Modularity

We evaluate the results of topological partitioning based on communication relationships by calculating the modularity [1]. The modularity is widely used because it is simple to calculate and can accurately reflect the quality of a global partition. The modularity can be calculated by the following formula.

$$Q = \frac{1}{2m} \sum_{i \neq j} \left(\mathrm{A_{ij}} - \frac{k_i k_j}{2m} \right) \delta\left(C_i, C_j\right) \tag{6}$$

where i and j represent any two nodes in the graph; m is the number of sides in the entire network; A_{ij} is the value of the row and C_i, C_j column in the adjacency matrix (1 or 0); and k_i, k_j represent the modularity of nodes i and j, i.e., the number of adjacent nodes. The letter i represents the numbering of the subgraphs j and $\delta\left(C_i, C_j\right)$. The definition is

$$\delta\left(C_i, C_j\right) = \begin{cases} 1 & C_i = C_j \\ 0 & C_i \neq C_j \end{cases} \tag{7}$$

Figure 3 shows the results of every hour of data per day, including 214,735 traffic flows and 16,145 IPs. The community identification was performed every 5 min. The modularity of the obtained division result is above 0.8. Therefore, the proposed method is simple and effective when comparing the sizes of $|P_{\mathrm{i}}|$ and $|Q_i|$ to determine the identity of node i and divide the network accordingly.

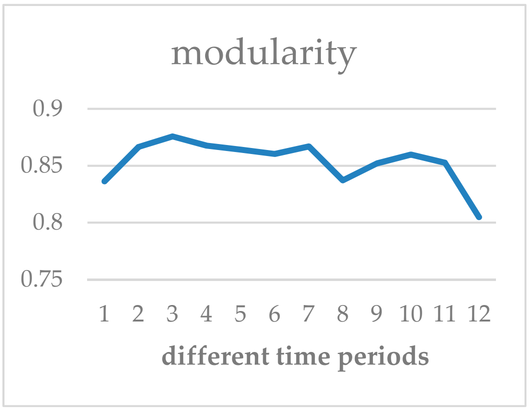

Figure 3. Modularity in different time periods.

4.2.2. Community Division Results

Figure 4 shows the number of subgraph partitions and the number of communities in each period obtained by topological partitioning and clustering. The number of communities is larger than the number of submap partitions. This is because some submap partitions are divided into different application communities due to different data transmission behaviors, such as malicious users and normal users in DDos attacks. Although the communication relationship is the same, it is divided into different communities due to different communication behaviors.

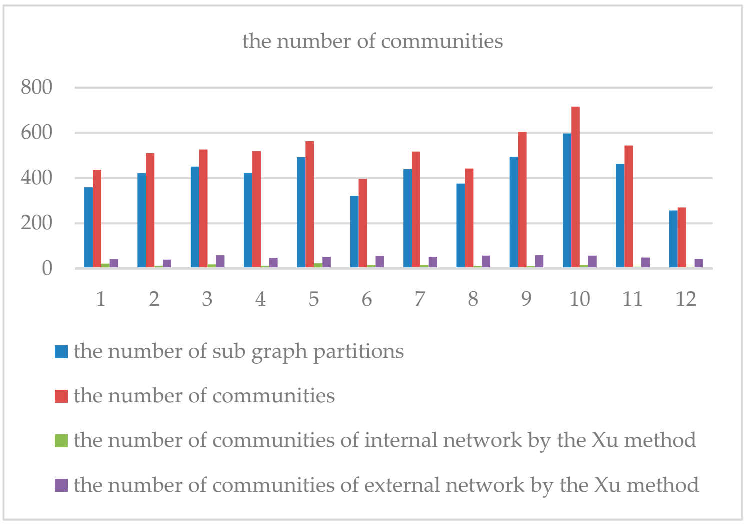

Figure 4. Number of partitions and communities in different time periods.

To illustrate the effectiveness of our method, we use the method proposed in Xu et al. [11] to identify the same data. Since the Xu method uses a single-mode projection for the bipartite graph in community identification, it can only identify communities from one side of the network at a time. We use the community detection algorithm in Xu et al. [11] for the internal and external IPs in the enterprise network; the results are shown in Figure 4. The results show that for the same dataset the number of communities identified by our method is much larger than that obtained by Xu's method. Through further analysis of the results, we found that Xu's method does not distinguish between different data transmission characteristics (such as the DDos attack situation) in the same communication relationship. Moreover, the communities identified by Xu's method do not have a service node (leader node). Thus, it is difficult to visualize the characteristics of the communities. Our method takes into account the communication relationship and data transmission characteristics when identifying the communities and retains the leader nodes in the communities. These help to more accurately identify and research the communities. This is the advantage of our method compared with existing research methods.

5. Analysis of Discussion

We used the Cytoscape method [29] to visualize some typical communities in the community detection results. The nodes represent the <IP, Port>, and connections represent the communication relations between nodes. Nodes are represented in different shapes according to the community which they belong to. The morphology of the community is shown in Figure 5, and the following phenomena can be observed by analyzing the community.

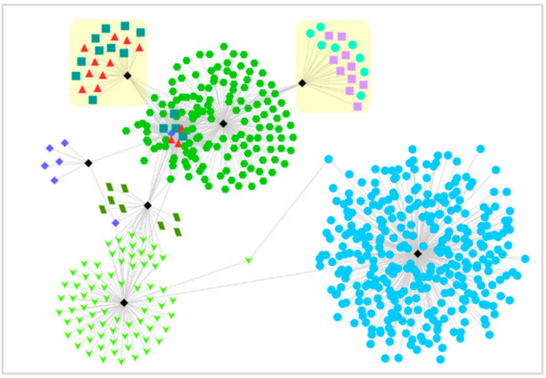

Figure 5. Community discovery results.

5.1. Normal and Abnormal Access on the Same Service Port

A service may appear as a leader node in two communities. By checking the original traffic logs, we can find that the clients in the A community frequently use a large number of ports to send the same size packets to the server in a short period of time. The clients in the B community, however, use fewer ports uploading and downloading packets, and the length is random. We separately selected one of the two communities to check the original packet, and found that the clients in the A community attack the server and the clients in the B community browse the webpage normally.

The experimental results show that the method can effectively distinguish different behavior modes on the same service port and can analyze the situation of the entire application community based on the observation of some individual traffic flows. Compared with the observation of each host, the community analysis presented in this paper can greatly reduce the necessary number of observation objects and the number of observations and can be used as a preprocessing step for traffic classification and traffic identification. For example, in this case, there were originally 10 hosts and 10 observations. After community identification, it was only necessary to take one from each of the two communities for observation.

5.2. Single-IP Bearing Multiservice

There are two communities and servers related to the same IP, but there are significant differences in the data transmission behaviors. For example, in the A community, the number of down link packets of the traffic flow is generally small (less than 3). The down link packet length is also below 80 bytes. However, in the B community (numbered 8), the number of down link packets is relatively large (greater than 100), and the average packet length of the down link is also 100 bytes or more. After analyzing the raw data, we found that the reason for this phenomenon is that the server uses two ports (53 and 443) and provides two services (DNS and https server).

5.3. P2P Communication Community

Through community identification, we find that a node IP communicates with 1773 external IPs and the IP is appeared as a leader node in 20 communities. In several communities, the IP communicates with a large number of public IPs. After verifying the traffic flows, we find that the IP is using P2P tools to download, including both eDonkey and Thunder network applications (and protocols). There are control messages with small transmission lengths and mass data transmission between nodes. Using previous community identification methods, it can only be regarded as a large community; however, our method can divide the multiprotocol simultaneous communications into different communities.

6. Conclusions

In this paper, we propose the definition of application community, which consider both communication relationships and traffic behavior characteristics. We also propose a corresponding detection method for application communities. In our application community the <IP, Port> two-tuple is used as a communication entity when establishing the communication diagram. The clustering method is used to distinguish different behavior mode. Our experiments demonstrate practical benefits of application community in profiling traffic patterns discovering emerging Internet applications and detecting anomalous traffic behaviors through synthetic traffic. It helps network administrators understand the distribution of traffic streams and complete resource deployment or virus attack prevention. The future work will include improving the algorithm for further reduction in calculations complexity of the detection method and tracking the evolution of communities in networks.

Author Contributions: Conceptualization, S.Z. and H.L.; Data Curation, S.Z.; Formal Analysis, S.Z.; Methodology, S.Z. and Y.Q.; Project Administration, S.Z.; Resources, H.L. and Z.W.; Software, Y.Q.; Supervision, Z.W.; Visualization, Y.Q.; Writing—Original Draft, S.Z. and Y.Q.; Writing—Review & Editing, Z.W.

Funding: This research received no external funding.

Acknowledgments: This study was supported by the National Key Research and Development Program (2016YFB0801200).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Guan, Z.; Li, J.; Wu, L.; Zhang, Y.; Wu, J.; Du, X. Achieving Efficient and Secure Data Acquisition for Cloud-supported Internet of Things in Smart Grid. *IEEE Internet Things J.* **2017**, *4*, 1934–1944. [CrossRef]
2. Du, X.; Guizani, M.; Xiao, Y.; Chen, H.H. Transactions papers A Routing-Driven Elliptic Curve Cryptography based Key Management Scheme for Heterogeneous Sensor Networks. *IEEE Trans. Wirel. Commun.* **2009**, *8*, 1223–1229. [CrossRef]
3. Xiao, Y.; Du, X.; Zhang, J.; Guizani, S. Internet Protocol Television (IPTV): the Killer Application for the Next Generation Internet. *IEEE Commun. Mag.* **2007**, *45*, 126–134. [CrossRef]
4. Du, X.; Xiao, Y.; Guizani, M.; Chen, H.H. An Effective Key Management Scheme for Heterogeneous Sensor Networks. *Ad Hoc Netw.* **2007**, *5*, 24–34. [CrossRef]
5. Baker, T.; Mackay, M.; Shaheed, A.; Aldawsari, B. Security-Oriented Cloud Platform for SOA-Based SCADA. In Proceedings of the 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Shenzhen, China, 4–7 May 2015.
6. Karam, Y.; Baker, T.; Talebbendiab, A. Security support for intention driven elastic cloud computing. In Proceedings of the Uksim/Amss European Symposium on Computer Modeling & Simulation, Manchester, UK, 20–22 November 2013.
7. Braem, B.; Barz, C.; Barz, C.; Rogge, H.; Freitag, F.; Navarro, L.; Bonicioli, J.; Papathanasiou, S.; Escrich, P.; Viñas, R.B.; et al. A Case for Research with and on Community Networks. *ACM SIGCOMM Comput. Commun. Rev.* **2013**, *43*, 68–73. [CrossRef]

8. Jin, Y.; Duffield, N.; Haffner, P.; Sen, S.; Zhang, Z.-L. Can't see forest through the trees? Understanding mixed network traffic graphs from application class distribution. In Proceedings of the 9th Workshop on Mining and Learning with Graphs (MLG2011), San Diego, CA, USA, 20–21 August 2011.
9. Cai, J.; Liu, W.X. A new Method of detecting network traffic anomalies. In *Applied Mechanics and Materials*; Trans Tech Publications Ltd.: Dürnten, Switzerland, 2013.
10. Chen, Z.; Hendrix, W.; Samatova, N.F. Community-based anomaly detection in evolutionary networks. *J. Intell. Inf. Syst.* **2012**, *39*, 59–85. [CrossRef]
11. Xu, K.; Wang, F.; Gu, L. Behavior Analysis of Internet Traffic via Bipartite Graphs and One-Mode Projections. *IEEE/ACM Trans. Netw.* **2014**, *22*, 931–942. [CrossRef]
12. Ni, J.; Weng, W.; Chen, J.; Lei, K. Internet Traffic Analysis Using Community Detection and Apache Spark. In Proceedings of the 2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (Cyber C), Nanjing, China, 12–14 October 2017.
13. Wang, J.; Paschalidis, I.C. Botnet detection using social graph analysis. In Proceedings of the 2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton), Monticello, IL, USA, 30 September–3 October 2014.
14. Wang, J.; Paschalidis, I.C. Botnet detection based on anomaly and community detection. *IEEE Trans. Control Netw. Syst.* **2017**, *4*, 392–404. [CrossRef]
15. Jakalan, A.; Gong, J.; Weiwei, Z.; Su, Q. Clustering and profiling ip hosts based on traffic behavior. *J. Netw.* **2015**, *10*, 99. [CrossRef]
16. Newman, M.E. Modularity and community structure in networks. *Proc. Natl. Acad. Sci. USA* **2006**, *103*, 8577–8582. [CrossRef] [PubMed]
17. Guo, B.; Yu, Z.; Zhou, X.; Zhang, D. Opportunistic iot: Exploring the social side of the internet of things. In Proceedings of the 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD), Wuhan, China, 23–25 May 2012; pp. 925–929.
18. Aiello, W.; Kalmanek, C.; McDaniel, P.; Sen, S.; Spatscheck, O.; Van der Merwe, J. Analysis of communities of interest in data networks. In *International Workshop on Passive and Active Network Measurement*; Springer: Berlin, Germany, 2005.
19. Jakalan, A.; Gong, J.; Su, Q.; Hu, H. Community Detection in large-scale IP networks by Observing Traffic at Network Boundary. In Proceedings of the World Congress on Engineering and Computer Science, San Francisco, CA, USA, 19–21 October 2015.
20. Jakalan, A.; Gong, J.; Su, Q.; Hu, X.; Abdelgder, A.M.S. Social relationship discovery of IP addresses in the managed IP networks by observing traffic at network boundary. *Comput. Netw.* **2016**, *100*, 12–27. [CrossRef]
21. Xu, K.; Wang, F.; Gu, L. Network-aware behavior clustering of Internet end hosts. In Proceedings of the INFOCOM, Shanghai, China, 10–15 April 2011.
22. Dewaele, G.; Himura, Y.; Borgnat, P.; Fukuda, K.; Abry, P.; Michel, O.; Fontugne, R.; Cho, K.; Esaki, H. Unsupervised host behavior classification from connection patterns. *Int. J. Netw. Manag.* **2010**, *20*, 317–337. [CrossRef]
23. Iliofotou, M.; Gallagher, B.; Eliassi-Rad, T.; Xie, G.; Faloutsos, M. Profiling-by-Association: A resilient traffic profiling solution for the internet backbone. In Proceedings of the 6th International Conference, Philadelphia, PA, USA, 30 November–3 December 2010.
24. Wei, S.; Mirkovic, J.; Kissel, E. Profiling and Clustering Internet Hosts. *DMIN* **2006**, *6*, 269–275.
25. Terzi, D.S.; Terzi, R.; Sagiroglu, S. Big data analytics for network anomaly detection from netflow data. In Proceedings of the 2017 International Conference on Computer Science and Engineering (UBMK), Antalya, Turkey, 5–7 October 2017.
26. Shadi, K.; Natarajan, P.; Dovrolis, C. Hierarchical IP flow clustering. In Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks, Los Angeles, CA, USA, 21 August 2017.
27. Kumar, K.M.; Reddy, A.R.M. A fast DBSCAN clustering algorithm by accelerating neighbor searching using Groups method. *Pattern Recognit.* **2016**, *58*, 39–48. [CrossRef]
28. Baker, T.; García-Campos, J.M.; Reina, D.G.; Toral, S.; Tawfik, H.; Al-Jumeily, D.; Hussain, A. *GreeAODV: An Energy Efficient Routing Protocol for Vehicular Ad Hoc Networks*; Springer: Cham, Switzerland, 2018; pp. 670–681.

29. Shannon, P.; Markiel, A.; Ozier, O.; Baliga, N.S.; Wang, J.T.; Ramage, D.; Amin, N.; Schwikowski, B.; Ideker, T. Cytoscape: A software environment for integrated models of biomolecular interaction networks. *Genome Res.* **2003**, *13*, 2498–2504. [CrossRef] [PubMed]

© 2018 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).