CLUSTER ANALYSIS OF TRAFFIC FLOWS ON A CAMPUS NETWORK

Asim Karim[1], Irfan Ahmad[2], Syed Imran Jami[3], and Mansoor Sarwar[2]

[1]Dept. of Computer Science
Lahore University of Management Sciences
Opp. Sector U, DHA, Lahore 54792
Pakistan
akarim@lums.edu.pk

[2]School of Science and Technology
University of Management and Technology
52-L, Gulberg III, Lahore 54600
Pakistan
{irfank, sarwar}@umt.edu.pk

[3]Dept. of Computer Science
NUCES (FAST)
Shah Latif Town, Karachi
Pakistan
imranjami2000@yahoo.com

ABSTRACT
Large quantities of network traffic flow data are generated on university campus networks. These data contain information on the sources and destinations of individual flows encoded as IP addresses. The cluster analysis of such data can reveal useful knowledge for web cache designing, user profiling, and network resource management. However, popular clustering algorithms such as k-means and DBSCAN are not directly applicable to datasets containing IP addresses. Moreover, such generic algorithms can yield results that are difficult to interpret.

This paper presents the cluster analysis of network traffic flows using a hybrid clustering algorithm. The algorithm integrates the longest prefix matching concept of TCP/IP traffic routing and the nearest neighbor algorithm. The similarity between IP addresses is determined by the longest prefix match. Similar IP addresses are then grouped together by an adapted version of the nearest neighbor algorithm. The algorithm provides automatic clustering that does not require input parameters such as the desired number of clusters and similarity threshold value. Furthermore, the algorithm yields 'natural' clusters consistent with the characteristics and usage of IP addresses. The test results are verified using nslookup. About 90% of the clusters were correctly identified by the algorithm.

KEY WORDS
Data Mining, Clustering, Network Traffic Analysis, IP Addresses

1. Introduction

Clustering is a key technique for analyzing large amounts of data. It partitions the data into groups such that the data objects within a group are more similar to each other than to the data objects in other groups. Clustering algorithms require the definition of a similarity measure for comparing data objects. Furthermore, they typically require simplifying heuristics on search to speed up the process. Popular clustering algorithms, such as the k-means and DBSCAN algorithms [1, 2], use generic similarity measures and search heuristics. As such, they cannot take advantage of domain-specific knowledge to improve the quality and efficiency of clustering.

The analysis of IP addresses contained within network traffic flows can yield useful traffic regularities that can help in network engineering. Clustering is a useful technique for segmenting network traffic flows on the basis of IP addresses. However, the popular clustering algorithms are not applicable directly to IP addresses based datasets. Moreover, the results of such clusterings tend to be less meaningful from a network engineering perspective because the unique characteristics of IP addresses are not considered in the clustering.

This paper presents the cluster analysis of traffic flows measured on a campus network based on their source and destination IP addresses using a new hybrid clustering algorithm. The algorithm integrates the longest prefix matching concept with the nearest neighbor clustering algorithm to provide automatic and meaningful clustering of IP addresses based datasets. The algorithm determines the similarity between two IP addresses by their longest prefix match value. Subsequently, an adaptation of the nearest neighbor algorithm is used to group similar addresses. The results are verified using traceroute and nslookup utilities.

The rest of the paper is arranged as follows: Section 2 presents an overview of the measurement setup. Section 3 describes some current algorithms for clustering IP address data. The hybrid algorithm for clustering IP addresses based datasets is described in section 4. The results of applying the algorithm are presented in section 5, and the concluding remarks are given in Section 6.

2. Traffic Flow Measurement Setup

Network traffic measurement has many applications and is typically done by monitoring network traffic flows [3]. The network traffic flows dataset used in this work is collected from Lahore University of Management Sciences (LUMS) which operates a campus-wide local area network (LAN) with approximately 1800 users.

Figure 1 shows the layout of the campus network. The computers in the student labs, administrative and

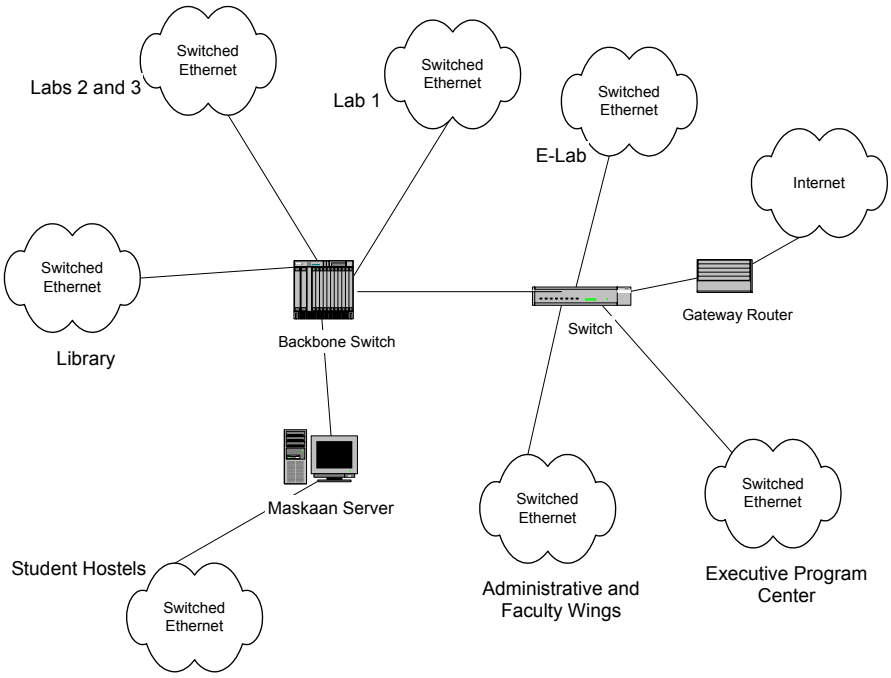

Figure 9: Layout of Campus Network

Figure 1 Layout of Campus Network

faculty offices, and an executive program center are interconnected via switched Ethernet LAN. Since students are the majority users of the network, we focus on measuring and analyzing traffic flows generated by students only. Students access the network through computers in Lab 1, Lab 2, Lab 3, E-Lab and Student Hostels (Figure 1). Using the measurement setup, we have collected network flows data generated by students from Lab 2, Lab 3 and Student Hostels for a period of one month.

The measurement setup consists of NeTraMet [4] which is the open source implementation of the measurement architecture recommended by the Realtime Traffic Flow Measurement (RTFM) working group [5] of the Internet Engineering Task Force (IETF). NeTraMet has a configurable packet-matching engine, which can be programmed according to measurement specific needs. We collected information on all flows where each flow is defined by {source IP, destination IP, source port, destination port, bytes sent, bytes received, start time, duration}.

3. Clustering and IP Addresses

Clustering is a fundamental problem in various fields for which numerous algorithms have been proposed over the years. These algorithms typically employ heuristics to guide the search to an approximate solution that is otherwise NP-hard to solve exactly. There has been renewed interest in clustering algorithms in recent years with the goal of discovering useful regularities in large and complex data [1, 2]. In this section popular clustering algorithms and their applicability to IP addresses based datasets is reviewed.

3.1 k-Means and k-Medoids Algorithms

A widely used clustering algorithm is the k-means algorithm. It identifies a cluster by the mean of the objects in it [1, 2]. The similarity between objects is determined by the Euclidean distance between them. Consequently, the k-means algorithm is directly applicable to numeric data only. The k-means algorithms can be applied to IP addresses by considering the decimal form of IP addresses and taking each octet as a dimension in a 4-dimensional space. This approach, however, will yield clusters that are inconsistent with the IP address' class based classification. For example, given three IP addresses IP1 = 128.195.4.115, IP2 = 213.195.4.115, and IP3 = 128.195.0.1, the k-means algorithm will group addresses IP1 and IP2 together, even though they belong to different network classes, instead of IP1 and IP3.

The k-medoids algorithm [1] has a similar logic to that of the k-means algorithm except that it can be applied to non-numeric data as well. It identifies a cluster by the median object (medoid) of the data objects in it. The similarity between objects is determined by metrics such as simple matching coefficient and Jaccard coefficient [2]. The k-medoids algorithm suffers from the same limitation as the k-means algorithm with regards to IP addresses. Furthermore, both k-means and k-medoids algorithms require the final number of clusters as an input.

Input:
D = {d_1, d_2, d_3, … , d_n} // Dataset
A // Adjacency matrix showing similarity between objects
t // threshold on minimum similarity
Output:
K // Set of clusters
Nearest Neighbor Algorithm:
$K_1 = \{d_1\}$;
$K = \{K_1\}$;
k = 1;
for i = 2 to n do
find the d_m in some cluster K_m in K such that similarity(d_i, d_m) is the greatest
if similarity(d_i, d_m) $\geq$ t then
$K_m = K_m \cup d_i$
else
k = k + 1
$K_k = \{d_i\}$
end if
end for

Figure 2 Nearest Neighbor Algorithm

3.2 Nearest Neighbor Algorithm

The nearest neighbor algorithm [1] is a clustering technique that does not require computation of the mean or median. It iteratively merges objects into the existing clusters that are closest to them. Its original implementation requires a threshold value for the minimum similarity measure for a cluster. If an object does not pass the threshold value (i.e. it is less similar) then it is included in a new cluster. Oftentimes the maximum number of neighboring objects to be considered is taken as a constraint in addition to the threshold on the minimum similarity measure.

The nearest neighbor algorithm is given in Figure 2. Figure 3 illustrates a typical iteration in the algorithm. The circles in the figure represent data objects in the dataset. The solid circle is an initial cluster (K1). Based on the given minimum similarity (maximum Euclidean distance) threshold, five clusters are created from nearest neighbor object merges.

The nearest neighbor algorithm is popularly used for clustering [6, 7, 8]. In [6] an adaptive online learning system is presented in which the most adequate materials for different students is retrieved by identifying clusters of these students. More interestingly, the nearest neighbor algorithm is often modified to be more appropriate for a given problem domain [7,8]

The original nearest neighbor algorithm, however, is not directly applicable to IP addresses based datasets. The original algorithm proposes a numeric similarity measure such as Euclidean distance. As described earlier, the Euclidean distance is not a good basis for grouping IP addresses. Nonetheless, a variation of the algorithm can be implemented on IP addresses based datasets, as described in section 4.

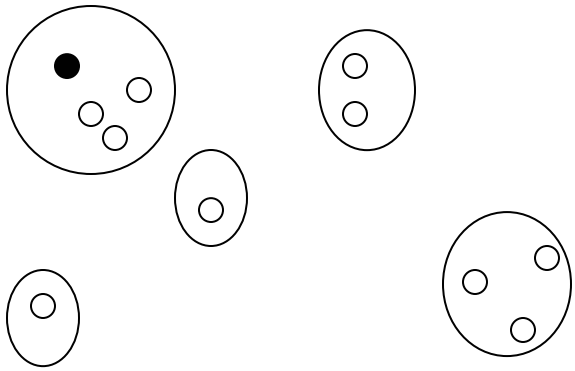

Figure 3 Illustrating Nearest Neighbor Algorithm

3.3 Other Algorithms

In recent years, new clustering algorithms have been developed for finding useful knowledge in large and complex datasets. A density-based algorithm in this category is DBSCAN [1,2]. DBSCAN does not require the number of clusters to be known a priori or the composition of the initial clusters. However, it is based on the density of the data objects. That is, data objects are viewed as points in n-dimensional space and density-connected points are grouped into clusters. This algorithm can only be applied to numerical datasets that can be plotted in n-dimensional space. With IP addresses, this is not possible since IP addresses are not defined like the ordinary numbers.

Evolutionary approaches like genetics algorithms can also be used for clustering [9]. Such algorithms perform a global search in the candidate solution space, and they tend to cope better with attribute interaction. It has been shown that evolutionary algorithms for clustering outperform the k-means algorithm and tend to increase as the number of clusters is increased [10, 11]. But it has been well proven that evolutionary algorithms are one or two orders of magnitude slower than conventional techniques [9], so it is unwise to use this approach for IP addresses based datasets.

3.4 Longest Prefix Matching

Longest prefix matching is employed for efficient routing table lookups during TCP/IP communications [12]. Figure 4 illustrates the concept of longest prefix matching. The figure shows two 16-bit strings. Starting from the left, the two strings match bit-by-bit until bit 10. Thus, the longest prefix match is 10.

The concept of longest prefix matching has been used to cluster IP addresses in the networks community, as described by several researchers [3, 13 14, 15]. However, these works do not present a general algorithm

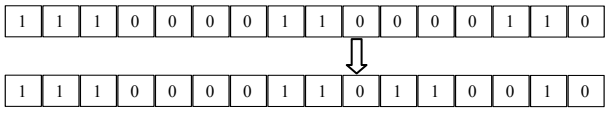

Figure 4 Longest Prefix Matching

for clustering datasets having IP address attributes. Rather, they describe the use of longest prefix matching to guide the grouping of IP addresses for specific tasks such network routing and server replications. As such, their approaches are interactive and not general and automatic.

4. Hybrid Algorithm for Clustering IP Addresses

We present a hybrid algorithm for clustering IP addresses based datasets that integrates judiciously the use of predefined IP address classes, the longest prefix matching concept for TCP/IP network traffic routing, and an adaptation of the nearest neighbor algorithm. The overall algorithm is a two level hierarchical technique that clusters large sets of IP address objects automatically without requiring inputs such as number of clusters desired or similarity thresholds. The final clusters obtained are based on unique characteristics of IP addresses, and hence, provide information that can be used for network engineering.

The algorithm for clustering IP addresses is shown in Figure 5 and described subsequently. First, clusters are created on the basis of IP address classes. This represents the level 1 clustering yielding four clusters, one for each class (class A, B, C, and D and E). Level 1 clustering may be skipped; however, its use will improve the computational performance of the algorithm and yield better final clusters.

Level 2 clustering involves the IP addresses within each class. First, the longest prefix match among the IP addresses is calculated and stored in an adjacency matrix. Then, each IP address is considered in turn and its cluster is created with all IP addresses with which it has the longest prefix match. This is the nearest neighbor approach where a cluster is created based on the nearest (or most similar) neighbors. However, unlike in the original nearest neighbor algorithm (Figure 2), a new cluster is created for every IP address with the IP addresses with which it has the longest prefix match. As such, IP addresses may be relocated from one cluster to another whenever their longest prefix match is greater with another IP address. In this way, clusters are modified iteratively as each IP address is considered based on the longest prefix match, a natural measure of similarity for IP addresses.

Notice that the hybrid algorithm does not require the input of a threshold value for the similarity, as required in the original nearest neighbor algorithm (Figure 2). This

Input:
D = {IP_1, IP_2, IP_3, ….. , IP_n) // IP addresses

Output:
K // Set of clusters

Algorithm:
Level 1: Cluster on the basis of IP address classes (optional)

Cluster 1 contains class A addresses (D_1={IP_1,…,IP_{n1})
Cluster 2 contains class B addresses (D_2={IP_1,… IP_{n2})
Cluster 3 contains class C addresses (D_3={IP_1,…,IP_{n3})
Cluster 4 contains outliers having class D and E addresses (D_4={IP_1,…,IP_{n4})

Level 2: Cluster within each class
do the following for each level 1 cluster (notation assumes level 1 is skipped)

Create Adjacency Matrix showing Longest Prefix Matching (LPM) among IP Addresses as A
K_1 = {IP_1}
K= {K_1}
c = 1 // cluster number
for i = 1 to n do
  for all IP_j (j = i+1 to n) with maximum LPM(IP_i, IPj)
    $K_c = K_c \cup IP_i$
    if IP_j is in some other cluster K_m
      if LPM ($K_c < K_m$)
        $K_c = K_c \cap IP_i$ // Remove IP_i from cluster K_c
        $K_m = K_m \cup IP_i$ // Add IP_i in cluster K_m
      else
        $K_m = K_m \cap IP_i$ // Remove IP_i from cluster K_m
        $K_c = K_c \cup IP_i$ // Add IP_i in cluster K_c
      end if
    end if
  end for
  c = c + 1 // for next cluster
end for
end do

Figure 5 Algorithm for Clustering IP Addresses

makes the algorithm automatic. More importantly, a threshold value has no meaning in the context of longest prefix match and IP addresses; two IP addresses may have a low longest prefix match but they may still belong to the same IP address class, and hence, cluster. Also, note that the computational complexity of the algorithm is O(n2/2) which is better than O(n2) for the original nearest neighbor algorithm.

To illustrate the algorithm, consider the 10 different destination IP addresses taken from traffic flows on the campus network shown in Table 1. These addresses belong to class B (i.e. level 1 clustering has already been done). The longest prefix match among these IP addresses

Table 1 Sample IP Addresses from a Campus Network

No.	Decimal	32-bit binary
1.	'128.104.176.132'	'10000000011010001011000010000100'
2.	'128.105.7.11'	'10000000011010010000011100001011'
3.	'128.121.232.233'	'10000000011110011110100011101001'
4.	'128.121.243.165'	'10000000011110011111001110100101'
5.	'128.121.26.137'	'10000000011110010001101010001001'
6.	'128.143.22.23'	'10000000100011110001011000010111'
7.	'128.148.19.67'	'10000000100101000001001101000011'
8.	'128.148.32.110'	'10000000100101000010000001101110'
9.	'128.171.157.175'	'10000000101010111001110110101111'
10.	'128.174.245.159'	'10000000101011101111010110011111'

is given in the adjacency matrix shown in Figure 6. The adjacency matrix is symmetric. Thus, the lower triangular portion is set to zero. Moreover, the diagonal values of the matrix are set to zero (the longest prefix match of an IP address with itself) since this value is not required. Starting from the first row of the adjacency matrix, a cluster is created containing that addresses and all addresses with which it has the longest prefix match. The final clusters obtained are shown in shown in Figure 7.

The ten IP addresses are grouped into 6 clusters. The correctness of the clusters is verified by using the traceroute (traceroute) and domain name lookup (nslookup) utilities. Cluster 1 IP addresses belong to University of Wisconsin, cluster 4's belongs to University of Virginia, cluster 5's belong to Brown University, and cluster 6's belong to University of Hawaii and University of Illinois. IP addresses of cluster 2 and 3 are unresolved by nslookup and tracert. Among the clusters obtained, only one cluster (cluster 6) has incompatible addresses.

5. Cluster Analysis of Traffic Flows Based on Destination IP Addresses

The algorithm is applied to a 7 day subset of the traffic flows dataset collected on our campus network. This dataset is clustered based on the destination IP address of each flow. Duplicate traffic flows with the same destination IP addresses are removed since they do not represent useful information for the analysis done here. The pruned dataset contained 10,525 traffic flows with distinct destination IP addresses.

Adjacency matrix

0	15	11	11	11	8	8	8	8	8
0	0	11	11	11	8	8	8	8	8
0	0	0	19	16	8	8	8	8	8
0	0	0	0	16	8	8	8	8	8
0	0	0	0	0	8	8	8	8	8
0	0	0	0	0	0	11	11	10	10
0	0	0	0	0	0	0	18	10	10
0	0	0	0	0	0	0	0	10	10
0	0	0	0	0	0	0	0	0	13
0	0	0	0	0	0	0	0	0	0

Figure 6 Adjacency Matrix

Cluster 1 '128.104.176.132' ; '128.105.7.11'
Cluster 2 '128.121.232.233' ;'128.121.243.165'
Cluster 3 '128.121.26.137'
Cluster 4 '128.143.22.23'
Cluster 5 '128.148.19.67'; '128.148.32.110'
Cluster 6 '128.171.157.175';'128.174.245.159'

Figure 7 Final Clusters

The results of clustering the dataset are given in Table 2. Level 1 clustering yields 4 clusters corresponding to the IP address classes A, B, C, and D and E. The IP addresses in classes D and E are considered as outliers (for web cache design, user profiling, etc) because they are not available to users. This class of addresses is not clustered further. Level 2 clustering yields 2610 clusters, which corresponds to 24.8% of the size of the original dataset.

The results are verified by using traceroute and domain name lookup (nslookup) utilities [16]. This procedure identified 1556 clusters (about 60%). Among these 1556 clusters, 1400 were found to be correct clusters representing natural groups of IP addresses. Thus, 90% of the clusters were correct. The problem of not being able to resolve IP addresses and to verify clusters is also faced by other researchers [14]. In [14] only 50% of the addresses were resolved using nslookup. Nevertheless, our algorithm exhibits a large correct clustering percentage considering the diverse use of IP addresses in today's network communication.

The robustness of the algorithm is evaluated by changing the order in which the dataset is read. The results given above are for in-order consideration of the objects in the dataset. Similar datasets are then placed in reverse order and in random order to test the dependence of algorithm on the order in which the objects are considered. In our simulation, it is found that the algorithm is unaffected by different orderings and yields practically the same results to that obtained from in-order consideration. This observation shows that the algorithm can identify significant traffic flow clusters under any sequence ordering.

Table 2 Cluster Analysis of Traffic Flows Based on Destination IP Addresses

Level 1 Clustering
No. of clusters = 4

No. of Flows in Cluster/Class			
Class A	Class B	Class C	Others
1302	799	2261	6163

Level 2 Clustering
No. of clusters = 2610
No. of clusters identified = 1556
No. of valid clusters = 1400

No. of Clusters in Each Class		
Class A	Class B	Class C
780	480	1350

No. of clusters identified = 1556
No. of valid clusters = 1400

6. Conclusion

This paper presents the cluster analysis of network traffic flows based on the destination IP addresses of each flow. The clustering is performed using a hybrid clustering algorithm that integrates judiciously the longest prefix matching concept from the networking domain with the nearest neighbor clustering algorithm to provide an effective automatic way of grouping IP addresses based datasets. The longest prefix match is adopted as the similarity measure for the clustering. An adaptation of the nearest neighbor algorithm is used to cluster similar IP address objects. The algorithm is automatic and does not require the input of desired number of clusters or similarity threshold value. The algorithm is applied to a traffic flows dataset measured on a campus network. The obtained clusters are verified using traceroute and nslookup utilities. About 90% of the clusters are correctly formed by the algorithm. This means that the clusters represent flows to destinations that belong to the same domain.

This work will help network administrators and designers with useful knowledge for designing web caches, user/traffic profiling, and network resource management. We are currently working on extending our algorithm to incorporate other attributes such as source and destination port numbers, starting and ending times, and number of bytes sent and received. The clustering of such datasets can provide more interesting patterns for network engineering.

Acknowledgment

This work was supported by a research assistant grant from Lahore University of Management Sciences.

References

[1] M. H. Dunham, *Data mining: introductory and advanced topics* (Pearson Education, 2003).
[2] J. Han and M. Kamber, *Data mining – concepts and techniques* (Morgan Kaufmann, 2001).
[3] C. Estan and G. Varghese, New directions in traffic measurement and accounting, *Proc. ACM SIGCOMM Internet Measurement Workshop*, 2001,
[4] N. Brownlee, *NeTraMet*, http://www2.auckland.ac.nz/net/Accounting/ntm.Release.note.html, 2003.
[5] IETF, *IETF Realtime Flow Measurement WG*, http://www2.auckland.ac.nz/net/Internet/rtfm/, 1999.
[6] B. Shih, W. Lee, The application of nearest neighbor algorithm on creating an adaptive online learning system, *31st ASEE/IEEE Frontiers in Education Conference*, Reno, NV, 2001.
[7] H. Samet, K-nearest neighbor finding using the MaxNearestDist estimator, Dept. of Computer Science, Univ. of Maryland, http://www.cs.umd.edu/~hjs/slides/iciap2003.pdf, 2003.
[8] D. Tunkelang, Making the nearest neighbor meaningful, *Proc. SIAM Int. Conf. on Data Mining*, 2002. .
[9] A. A. Freitas, *Data mining and knowledge discovery with evolutionary algorithms* (Springer, 2002).
[10] L. Meng, Q. H. Wu and Z. Z. Yong. A faster clustering algorithm: real world applications of evolutionary computing, *Proc. Evoworkshops*, 2000.
[11] K. Krishma and M. N. Murty, Genetic k–means algorithm, *IEEE Transactions on Systems, Man and Cybernatics - Part B: Cybernetics, 29*(3), 1999, 433–439.
[12] M. Waldvogel, Fast longest prefix matching: algorithms analysis, and applications, Swiss Federal Institute of Technology, Zurich, http://marcel.wanda.ch/Publications/waldvogel00fast.pdf, 2000.
[13] C. Estan, S. Savage, G. Varghese, Automatically inferring patterns of resource consumption in network traffic, *Proc. ACM SIGCOMM '03*, Karlsruhe, Germany,, 2003.
[14] B. Krishnamurthy, J. Wang, On network-aware clustering of web clients, *Proc. ACM SIGCOMM '00*, Stockholm, Sweden, 2000.
[15] R. Vaarandi, A data clustering algorithm for mining patterns from event logs, *IEEE Workshop on IP Operations and Management*, 2003.
[16] ARIN, *NSlooup Utility*, http://ws.arin.net/, 2005.