Policy-Aware Algorithms for Proxy Placement in the Internet

Krishnanand M.Kamath, Harpal Singh Bassali, Rajendraprasad B.Hosamani
and Lixin Gao *
University of Massachusetts
Amherst, MA 01003
harpal@cs.umass.edu, {kkamath,rhosaman,lgao}@ecs.umass.edu

ABSTRACT

Internet has grown explosively for the past few years and has matured into an important commercial infrastructure. The explosive growth of traffic has contributed to degradation of user perceived response times in today's Internet. Caching at the proxy server have emerged as an effective way of reducing the overall latency. The effectiveness of a proxy server is primarily determined by its *locality*. This locality is affected by factors such as the Internet topology and routing policies. In this paper, we present heuristic algorithms for placing proxies in the Internet by considering both Internet topology and routing policies. In particular, we make use of the logical topology inferred from Autonomous System (AS) relationships[17] to derive the path between a proxy and a client. We present heuristic algorithms for placing proxies and evaluate these algorithms for the Internet logical topology over three years. To the best of our knowledge, this is the first work on placing proxy servers in the Internet that considers logical topology.

1. INTRODUCTION

Over the past few years, there has been an explosive growth in the use of the World Wide Web. Commercialization and the development of standard tools for creating and navigating web documents have been the main reasons for this growth. The quick adoption of the Internet has also led to the degradation in the performance of the Internet, as the web clients often encounter unacceptably long response time. Server and Network congestion at peak time along with long propagation delays are believed to be the root cause of the problem. Caching improves network and system performance for browsing by saving network bandwidth, reducing the server load and achieving load balancing.

The principle behind caching is that it allows the required documents to be kept close to the clients; thus reducing the response time of web access and also decreasing network congestion. The most popular locations to place a cache are the web browser and proxy server.[1] Web browser caching has been implemented by most existing web browsers.[2] This helps in preventing the client from generating traffic for the same data again.

Caching at the proxy server have emerged as an effective way of reducing the overall latency. A proxy server can be either a replicating server or a web proxy, which is an intermediate server that acts as a caching agent between clients and server. If properly placed, a proxy server can significantly reduce web access delay and network congestion. In addition, it can reduce the server load, which may be critical during peak-time. For example, Content Distribution Networks (CDN's) have emerged as a powerful solution to improve the response times to web pages and to reduce the traffic in the Internet to some extent. CDN's consist of a set of distributed proxy servers replicating the content for better performance and availability than the centralized servers.

The effectiveness of a proxy server is primarily determined by its *locality*. This locality is affected by factors such as the Internet topology and routing policies. In this paper, we present heuristic algorithms for placing proxies in the Internet by considering both Internet topology and routing policies. In particular, we make use of the logical topology inferred from Autonomous System (AS) relationships[17] to derive the path between a proxy and a client. We present heuristic algorithms for placing proxies and evaluate these algorithms for the Internet logical topology over three years. To the best of our knowledge, this is the first work on placing proxy servers in the Internet that considers logical topology.

*The author was supported in part by NSF grant NCR-9977555, and NSF CAREER Award grant ANI-9875513. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

1.1. Previous Work

Existing work on Web performance has primarily focused on techniques for efficiently redirecting user requests to appropriate servers to reduce request latency and balance load. Research has been done from Web workload characterization[4–6] to develop techniques to enhance Web performance. Two techniques, caching and replication, have been proposed for enhancing Web performance. Many aspects of caching have been studied extensively with an exception of placement. The first work on the placement problem known to us is by Li[7] *et al.* They assume the underlying network topologies to be trees, and model the placement problem as a dynamic programming problem. The assumption of a tree structure for the underlying topology imposes restrictions on the applicability of these results to practical scenario. As shown in,[13–16] the Internet topology is far from being a neatly structured tree.

Jamin et al. examined the placement problem for the Internet in Instrumentation in.[8] The paper carries out the evaluation of two kinds of methods. The first one being the graph theoretic methods and the second class consists of heuristics for instrumenting the Internet. They also consider the case when full knowledge of the underlying topology is not available. They have shown that an Internet distance map service based on their placement techniques can offer useful hints for server selection by clients.

The paper by Qiu et. al,[10] reports interesting results to the proxy placement problem. In this paper the authors have presented comparisons between the *Tree-based Algorithm* presented in,[7] a *Greedy Algorithm*, *Random Algorithm* and a *Hot Spot* algorithm. The basic idea of the Greedy Algorithm is to place proxies one at a time in the best possible place. Subsequent placements are made such that the placements yield lowest cost in conjunction with the previous placements. Random Algorithm places proxies randomly and Hot Spot Algorithm places proxies in the top N heavily loaded ASes. The Greedy algorithm is proposed to perform best in comparison to the rest by means of experiments run on various *simulated* network topologies and a few Internet topologies. While[10] makes the generated network topologies as rich as possible by using a wide range of parameters, it also states that these network models and parameters may not be true representatives of the Internet topology. Access logs collected at the MSNBC server site are used to consider only the top (based on the generated traffic load) few clusters to reduce the high computational cost of the tree-based and Greedy algorithms. Furthermore, routing policies are not modeled when evaluating these algorithms. In our paper, in addition to considering routing policies and listing the performance of our heuristic algorithms on a range of Internet AS topologies (from March 98 to March 01), we compare its performance against the Greedy algorithm as proposed in.[10]

1.2. Organization

The rest of the paper is organized as follows. In section 2, we present a brief account on today's Internet topology and the implications of BGP policy issues on Inter-AS routing followed by the problem description. In section 3 we present an algorithm for the computation of the cost function given the Internet AS topology and the proxy placement. Section 4 lists the heuristic algorithms for a suboptimal proxy placement scheme followed by the experimental results in Section 5, obtained by running the heuristic algorithms on Internet AS topologies of the past three years. Section 6 concludes with pointers to future work.

2. BACKGROUND AND PROBLEM DESCRIPTION

2.1. Background

Today's Internet is a complex maze consisting of thousands of administrative domains also referred to as *Autonomous Systems (ASes).* Routing within an AS is determined by the intra-domain routing protocols used within the domain e.g., OSPF,IS-IS,RIP etc. A pair of ASes interconnect via dedicated links and/or public network access points, and routing between ASes is determined by the inter-domain routing protocol - Border Gateway Protocol (BGP). BGP allows each AS to choose its own policy, which determines the nature of its connectivity with its neighboring ASes. These policies impose an additional constraint that connectivity does not necessarily mean reachability. Hence it is important to study the effects of BGP routing policies on the reachability.

In this section, we summarize the results of Gao et al[17] on the implications of BGP policies on Internet routing. The commercial agreements between pairs of administrative domains can be classified into customer-provider, peering and mutual-transit agreements.[18,20] Since a customer pays its provider for connectivity to the rest of the Internet, a provider transits traffic for its customers. However, a customer does not transit traffic between two of its providers. Pair of peers agree to exchange their respective customers traffic between them free of charge. A mutual-transit agreement allows a pair of administrative domains to provide connectivity to the rest of the Internet for each other. This mutual transit relationship is typically between two administrative domains such as small ISPs or universities who are located close to each other and who cannot afford additional Internet services for better connectivity. Due to these observations ASes can have one of the four kinds of relationships. *Customer-Provider or Provider-Customer* relationship, *Peer-Peer* where administrative domains of similar size exchange traffic or *Sibling-Sibling* relationship if they belong to the same administrative domain or to the administrative domains that have a mutual-transit agreement as shown in Fig.1.

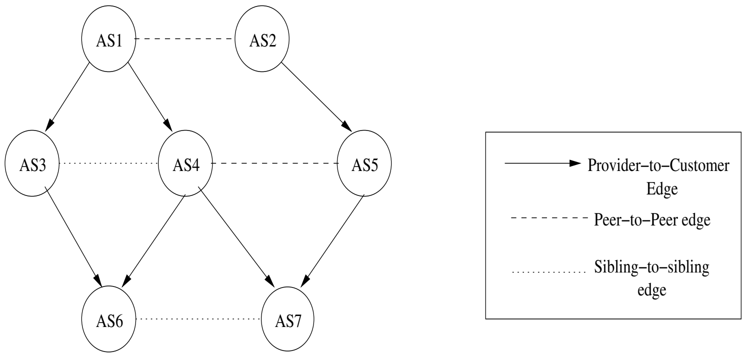

Figure 1. Relationships between ASes

The basic rule governing the export policies is that an AS does not provide transit services between two of its providers and peers. This means that there are a number of patterns of link traversal that can compose a path between any source and destination AS pair, which in turn means that the AS paths in a BGP routing table should have certain patterns.

We define two terms which will be used in the rest of the section. An AS path of a BGP routing table has sequence consisting of one or both of the below mentioned paths.

- **Uphill Path:** A sequence of edges that are either customer-provider or sibling-to-sibling edges. Formally, a path $(u_1, u_2 \ldots\ldots u_n)$ is an uphill path if and only if (u_i, u_{i+1}) is either a customer-to-provider or a sibling-to-sibling edge for all $i < n$.
- **Downhill Path:** A sequence of edges that are either provider-to customer or sibling-to-sibling edges. Formally, a path $(u_1, u_2 \ldots\ldots u_n)$ is a downhill path if and only if (u_i, u_{i+1}) is either a provider-to-customer or a sibling-to-sibling edge for all $i < n$.

With the above definitions one can rule out the patterns that are not possible in a BGP routing table. These patterns indicate some misconfiguration in the BGP settings. According to above mentioned rules governing export policies the following patterns are not possible

1. A Downhill path followed by an Uphill path
2. An uphill path followed by a peer-to-peer edge, which is followed by another peer-to-peer edge.
3. A downhill path followed by a peer-to-peer edge, which is followed by another peer-to-peer edge

Based on these observations,[17] proposes a set of heuristics to infer the underlying AS topology, which forms the basis for our algorithms. Our algorithms have been tested on the topology data obtained by running the above mentioned heuristics on the BGP routing tables obtained from the Internet Routing Registries (IRRs).

2.2. Problem Description

We begin the details of our work by presenting a detailed description of the problem that is being solved in this paper. The problem consists of a topology $\mathcal{N}$ which consists of N autonomous systems (AS). These ASes are inter-connected to each other by one or more of the relationships mentioned in the previous section. The problem we are trying to solve is to place M proxies optimally. The optimality is determined by the definition of the Figure of Merit (FOM) function. Throughout this paper, we assume that all the proxies are identical with respect to their content.

Let $\mathcal{P}$ be the set of all possible unique proxy placements.

$$\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \cdots, \mathcal{P}_K\},$$

where $K = {}^{N}C_M$ denotes the total possible ways in which M proxies can be placed on N ASes and $\mathcal{P}_i$ is the i^{th} element in the set $\mathcal{P}$.

Let $c(v, \mathcal{P}_i)$ be defined as the minimum cost for an autonomous system $v \in \mathcal{N}$ to access a proxy in AS $\mathcal{P}_i$. The optimality criterion for $\mathcal{P}_i^{th}$ set of proxy placement *figure of merit*, FM_i, is defined as

$$FM_i = \frac{\sum_{v \in \mathcal{N}} c(v, \mathcal{P}_i)}{N}$$

In simple words this denotes the average cost a node will incur in order to access a proxy, where the proxy placement is given by $\mathcal{P}_i$.

In addition we define the following notations.

- **Optimal figure of merit:** The *figure of merit* achieved by using the optimal proxy placement algorithm and denoted by *Optimal* $FM(\mathcal{N}, M)$. In other words, it is the minimum possible *figure of merit* that can be achieved given a topology, $\mathcal{N}$, and M proxies.

$$Optimal\ FM(\mathcal{N}, M) = min_i(FM_i)$$

- **Upper bound on figure of merit:** The best possible figure of merit that is achievable given the size of the topology and the number of proxies to be placed and denoted by *Upper Bound* $FM(\mathcal{N}, M)$. For a topology $\mathcal{N}$ consisting of N nodes with M proxies to placed and assuming the link cost to be 1 (in other words, assuming hopcount as the metric)

$$Upper\ Bound\ FM(\mathcal{N}, M) = \frac{(N - M)}{N}$$

and

$$Optimal\ FM(\mathcal{N}, M) \ \leq\ Upper\ Bound\ FM(\mathcal{N}, M)$$

An optimal proxy placement algorithm would choose a proxy placement $\mathcal{P}_i, i = 1, 2, \cdots, K$, that gives the optimal figure of merit, *Optimal* $FM(\mathcal{N}, M)$.

The obvious difficulty in the brute force method is the exponential complexity of $O(N^M)$ required since one needs to try all the ${}^{N}C_M$ combinations. Given the size of the present Internet, it is computationally infeasible to calculate such a term even for small values of M.

3. COMPUTATION OF SHORTEST PATH

In the Internet topology, physical connectivity between domains(ASes) does not necessarily imply reachability. In the routing literature, we do come across algorithms[12] to compute the shortest path to any other node in the graph. However, these algorithms assume that physical connectivity implies reachability. In other words, if a node A has a neighbor B who in turn has another neighbor C than it is implied that A can reach C through B. This is not true in case of an Internet topology, which uses a policy-based routing protocol like BGP.

In order to incorporate the above-mentioned restriction, we adopt a modified Dijkstra's(MD) shortest path algorithm. In MD shortest path algorithm, we maintain additional states, which account for the policy induced restrictions while computing paths between any two given nodes. Before we present the algorithm we shall define a few notations that simplify the description of the MD algorithm.

- **Source Path** A sequence of edges that are sibling-to-sibling edges.

The additional state that we maintain in the MD algorithm can take one of the following values.

- u: A path between any two given nodes is marked u, if the source has to take an *uphill path* to reach the destination. Reachability to any nodes which can be reached through *uphill path* can be advertised only to the customers and siblings of the source node.
- d: A path between any two given nodes is marked d, if the source node has to take a *downhill path* to reach the destination. Such a route can be advertised to all the neighbors. Providers and Siblings of the source store this route with a state d, since they have to take a *downhill path* to reach the source. The customers store this route as u, since they have to take an *uphill path* to reach the source.
- s : A path between any two given nodes is marked s, if the source node has to take a *source path* to get to the destination. Such routes can be advertised to all the neighbors of the source node. Customers of the source node will store this route with a state u while providers will store this route as d. It is fairly intuitive to see why this is done so. The siblings of the source node store this route as s, since the source can be reached from any of the receiving nodes by taking the current *source path* path.

We will briefly explain the working of the MD shortest path algorithm using the network topology shown in Figure 2. For sake of simplicity, we shall assume all link costs as 1 resulting in shortest path corresponding to the path with the smallest hopcount. The AS A containing the proxy shall have a cost(hop-count) of 0 to the proxy. The customer and peer of the AS A containing the proxy namely B and C will each have a cost of 1 to get to the proxy and a state u since they need to traverse a *uphill path* to get to the source. Also, since they have received a route to the proxy through a *uphill path* that is with a state u, they will forward this information(route) only to its customer and siblings namely D, E and H with the same state namely u. Now, the provider of the A namely I will have a cost of 1 to the proxy through its *downhill path* i.e. with a state d. Hence, the node I will forward this route to the proxy to all its neighbors. However, while the siblings and providers of I namely J and K get the route information through their *downhill paths* i.e. with a state d, the peers and customers namely L get the same route information through their *uphill paths* i.e. with a state information u. Finally, the sibling of the source A namely O shall behave exactly similar to the node I in all aspects except while forwarding the routing information to its sibling namely R. The sibling R of AS O continues to be on a source link and hence will have a cost of 2 to the proxy(source) with a state s. This in a nutshell, explains the working of the above algorithm in presence of the *selective export rule* imposed by the policy based inter-domain routing of BGP .

The following figure presents the pseudo code for the MD shortest path algorithm. Note that MD computes the minimum distance of any AS in a given topology. Requests from an AS can leave the AS either by taking an *uphillpath*,*downhillpath* or a *sourcepath*. Keeping this in mind the following are the additional notations used in the algorithm.

- $c(i, j)$: link cost from AS i to AS j.
- $D_l(v)$: AS v's cost to access the given proxy p by taking a route with state l , where $l \in$ {u,d,s}
- $\mathcal{N}_l$: Set of AS numbers who have a route received with a state $l \in \{u, d, s\}$ to the proxy

In Phase I of the Modified Dijkstra's Algorithm, the various variables are initialized. AS A is the node, which is chosen as the proxy. Therefore, the distance of A from the proxy is initialized to 0. For the rest of the nodes the value is initialized to ∞. The neighbours of A have their distance entries initialized to the cost of reaching A. In steps 8 and 9, we choose a node w with a minimum distance. In steps 10, 11 and 12 we check if the distance of w chosen above is ∞ and exit. In steps 13,14 and 15, we check all the customers v of AS w, which are not already in N_u, whether they have a new shortest distance to the proxy in AS A through w. Notice that we use $D_p(w)$ here for the cost upto w since there is no restriction on a customer path. In steps 16,17 and 18 we repeat the above mentioned operations for siblings. However notice that taking the sibling path does not affect the "state" of the route. In steps 19,20 and 21 we do the same for providers except for the restriction that we cannot take a provider path if we are in a u distance route. What this means is that if we have traversed a customer link or a peer link, then we can no

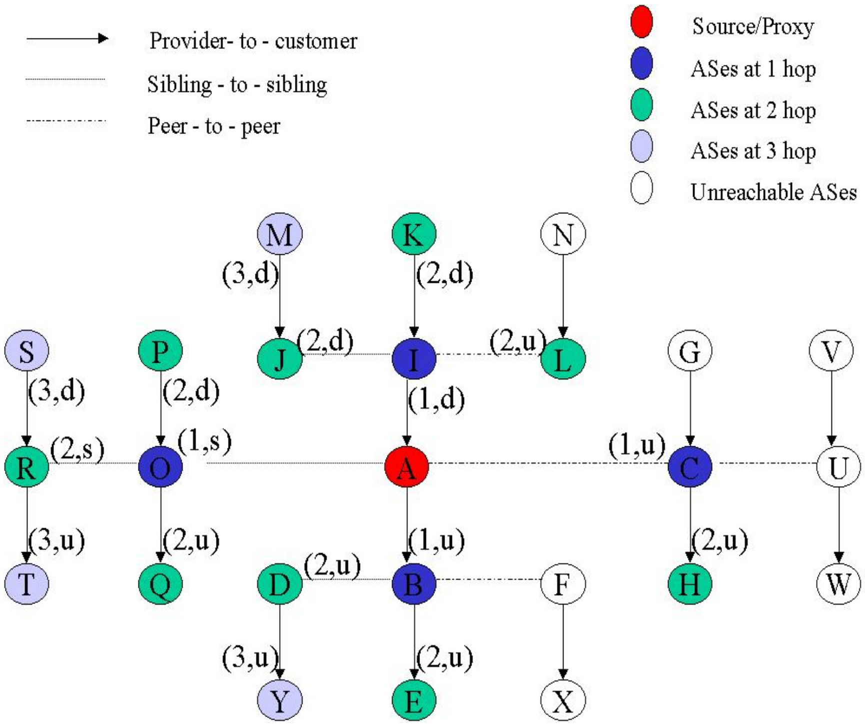

Figure 2. MD Example

longer traverse a provider link. However if we are in s or d distance states then we have only traversed siblings or provider links, therefore we can take another provider link and change the state to d to take note of the fact that we have traversed a provider link. In steps 22,23 and 24 we do a similar thing for peers except that on taking a peer link we move to the most restrictive u state. What this means is that we can no longer take another peer link or provider link. We continue this algorithm till all nodes have been added to N.

In this section, we have formulated a method for computing the *figure of merit*, given a proxy placement, for any link cost (traffic load, hopcount etc.). However, the only link cost metric we have is in terms of hopcount between any two ASes, made available by the *Route View* server,[9] and do not have any other link cost information on any of the links. In the next section, we shall present the basis and the implementation of the heuristic algorithms we shall use in formulating a sub-optimal proxy placement scheme. Due to the generalized MD algorithm that computes the figure of merit for any defined link cost, our heuristic algorithms are applicable for all link cost metrics. However, due to the lack of any other link cost metric, hereafter, we shall use hopcount as the link cost metric.

4. HEURISTIC ALGORITHMS

4.1. Basis

In this section we shall explain the basis of our heuristic by means of a small graph of 20 ASes as shown in Figure 4. The topology consists of all the relationships prevalent in today's Internet namely provider-to-customer, peer-to-peer and sibling-to-sibling relationship. The topology also exhibits the power law to *some* extent. The cost criterion used for all the following implementations is hop count. However, one should note that the Modified Dijsktra algorithm for computing the *figure of merit* (and hence, our heuristic algorithms for proxy placement) could be used for any other cost criteria as well.

To show the intuition behind our heuristic algorithm the topology shown in Figure 4 will be used. We obtained the optimal solution for proxy placements by using the brute force method and *figure of merit* computation described earlier. For $M = 1$ i.e. one proxy case we find that $AS2$ gives the *optimal figure of merit*. The intuition one obtains from the above result is that since $AS2$ was at the topmost level and had the highest number of neighbors, it was best suited for proxy placement in the sense that its average distance to the rest of the nodes(ASes) was least. The above result is the basis for the *highest-degree-first* proxy placement algorithm.

Modified Dijkstra Algorithm:
Input: Annotated AS graph $\mathcal{N}$,Proxy in AS A
Output: Minimum distance from every AS to AS A
Definition: $\mathcal{S}$ =$\{u, d, s\}$, $\mathcal{N}$ = Set of all ASes in AS graph

Phase 1: Initialization
$\mathcal{N}_s = \mathcal{N}_u = \mathcal{N}_d = \{A\}$
$D_u(A) = D_d(A) = D_s(A) = 0$
For each AS $v \in \mathcal{N}$ - $\{A\}$ AND $l \in \mathcal{S}$
    $D_l(v) = \infty$
    if v is a customer/peer of A then $D_u(v) = c(v, A)$
    if v is a provider of A then $D_d(v) = c(v, A)$
    if v is a sibling of A then $D_s(v) = c(v, A)$

Loop
Find w such that $\forall\ p \in \mathcal{S}$ AND $w \notin \mathcal{N}_p$
    $D_p(w)$ is minimum

For all $p \in \mathcal{S}$
    if $(D_p(w) == \infty)$
        exit

For all customers v of w such that $v \notin \mathcal{N}_u$
    $D_u(v) = min(D_u(v), D_p(w) + c(v, w))$, where $p \in \mathcal{S}$
    Add v to $\mathcal{N}_u$

For all siblings v of w such that $v \notin \mathcal{N}_p$, where $p \in \mathcal{S}$
    $D_p(v) = min(D_p(v), D_p(w) + c(v, w))$
    Add v to $\mathcal{N}_p$

For all providers v of w such that $v \notin \mathcal{N}_d$
    $D_d(v) = min(D_d(v), D_p(w) + c(v, w))$, where $p \in \{s, d\}$
    Add v to $\mathcal{N}_d$

For all peers v of w such that $v \notin \mathcal{N}_u$
    $D_u(v) = min(D_u(v), D_p(w) + c(v, w))$, where $p \in \{s, d\}$
    Add v to $\mathcal{N}_u$

Until all nodes in $\mathcal{N}_p$ where $p \in \mathcal{S}$

Figure 3. *Modified Dijkstra Algorithm*

In the next step, we found the ASes which would give us the *optimal figure of merit* while placing two proxies i.e. $M = 2$. These ASes turned out to be $AS11$ and $AS9$. The intuition obtained from the selection of $AS11$ continues to be the fact that $AS11$ having the largest number of neighbors is best suited for a proxy placement. However, even though $AS2$ has the highest number of neighbors along with $AS11$, $AS9$ turns out to be best suited in improving the *figure of merit*. The second intuition obtained from the above result is that the distance between two nodes containing the proxy also affects the *figure of merit*. In other words, by having proxies in two nodes which are neighbors or close to each other say $AS2$ and $AS11$ we are increasing the hop count to numerous other nodes in the other subtree namely ASes 13 to 20. The above result is the basis for the *farther-first* proxy placement algorithm.

To conclude, we find that while ASes with greater number of neighbors should be given priority while placing the proxies, the ASes containing the proxies should not be close to one another. We shall quantify the effects of both

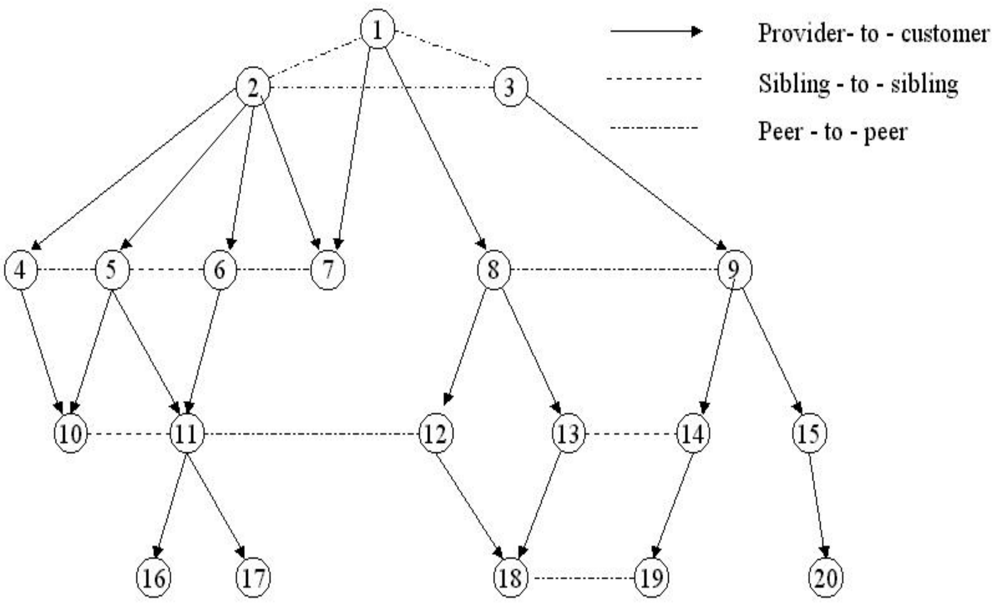

Figure 4. Heuristic Basis

heuristics in the subsequent section.

4.2. Algorithms

In the previous section, we observed that while deciding to place a proxy in a certain AS, we should take into account the connectivity of that AS. If the AS has a high number of neighbours, then it should be given more preference for being selected as a proxy. Another factor that needs to be taken into account is that the proxies that get chosen should be as far apart from each other as possible. This will help in preventing some nodes being very close to multiple proxies while others being distant from all proxies. The three proxy placement algorithms we propose are formally defined below.

1. **Highest-degree-first Algorithm**
   The Highest-degree-first algorithm attempts to place proxies in the ASes that have the highest degree i.e. the highest number of neighbors when hopcount is the *cost* function. Given N ASes and M proxies to be placed($M < N$), the ASes are sorted in the decreasing order of their number of neighbors and top M ASes are chosen for proxy placement.

2. **Farther-first Algorithm**
   The Farther-first algorithm attempts to place proxies in the ASes that have the highest degree as well as that are far apart. Given N ASes and M proxies to be placed($M < N$), the ASes are sorted in the decreasing order of their number of neighbors and top M ASes are chosen, starting from the topmost AS in the sorted list, so that no two ASes are neighbors i.e. at a hopcount 1 from each other.

3. **Optimal Algorithm**
   The Optimal algorithm is a hybrid of the above two algorithms. Given N ASes and M proxies to be placed, the ASes are sorted in decreasing order of their number of neighbors as in the above two algorithms. From the sorted AS list, $n(1 \leq n \leq M)$, proxies are chosen based on highest-degree-first algorithm and the remaining $M - n$ proxies are selected based on the farther-first algorithm. It is noted that the special cases $n = M$ and $n = 1$ are the highest-degree-first algorithm and farther-first algorithm respectively and the optimal algorithm basically computes the value of n that gives the best performance.

The above three heuristic algorithms are explained by means of the *Basic Algorithm* below which takes the value of n and the list of ASes(sortedASList) sorted based on their degrees (number of neighbors) as inputs and returns the list of proxies(proxyList). The highest-degree-first algorithm is equivalent to calling *Basic Algorithm* with input $n = M$ and the farther-first algorithms equivalent to calling *Basic Algorithm* with input $n = 1$. The optimal algorithm is an $O(M)$ search over all possible values of n i.e. call the *Basic Algorithm* M times with n varying from 1 to M and selects the proxy list which give the least figure of merit.

```
Basic Algorithm:
Input: n (Number of Highest Degree ASes) && sortedASList
Output: proxyList

Initialization: Initialize the variables
 1.  numHighDegreeAS = 1
 2.  sortedASListIndex = 1
 3.  proxyListIndex = 0
 4.  discardListIndex = 1

Phase 1: Use the sortedASList to populate proxyList
 5.  Do
 6.      If( numHighDegreeAS ≤ n ||
 7.          !isAdjacent(sortedASList[sortedASListIndex],proxyList) )
 8.              addToProxyList(sortedASList[sortedASListIndex])
 9.              numHighDegreeAS++
 10.             proxyListIndex++
 11.     Else
 12.             addToDiscardList(sortedASList[sortedASListIndex])
 13.     sortedASListIndex++
 14. Until ( sortedASListIndex > N || proxyListIndex==M )

Phase 2: Use the discardList to populate proxyList
 15. While ( proxyListIndex < M )
 16.     addToProxyList(discardList[discardListIndex])
 17.     proxyListIndex++
 18.     discardListIndex++
 19. End
```

Figure 5. *Basic Heuristic Algorithm*

Figure 5 presents the *Basic Algorithm*. We define three lists namely sortedASList, proxyList and discardList. The sortedASList is the input to the *Basic Algorithm* and is the list of ASes in a decreasing order of their number of neighbors. The proxyList is the output of the *Basic Algorithm* and is the list of the ASes selected for proxy placement. The discardList comprises of the ASes that are discarded or left out of the proxyList because they are not one of the top n high degree ASes and have one of the already selected proxies as its neighbor i.e. adjacent to one of the proxies in the proxyList. Line 1 in the initialization phase initializes the variable numHighDegreeAS which represents the number of highest degree ASes that have been selected from the sortedASList. Line 2 to 5 initializes the indexes of the above three lists. In Phase 1 of the algorithm, the sortedASList is used to populate the proxyList. Starting from the first AS in the sortedASList we run the steps from line 6 to line 13 of Phase 1 until M proxies have been found or all the N ASes in the sorted ASList have been accounted for. In lines 6 and 7 it is checked whether the AS is one of the n highest degree ASes or if it is *not* adjacent to any of the ASes in the proxyList (already selected proxies). If yes, then the new AS is added to the proxyList else it is added to the discardList. If Phase 1 of the algorithm does not populate the proxyList with the required M proxies, the phase 2 of the algorithm selects the additional proxies from the top of the discardList (which contains the discarded ASes in the decreasing order of their number of neighbors).

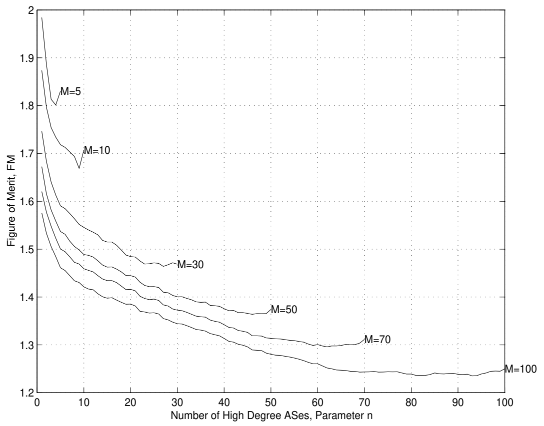

Figure 6. Figure of Merit (FM) Vs n for number of proxies (M) varying from 5 to 100

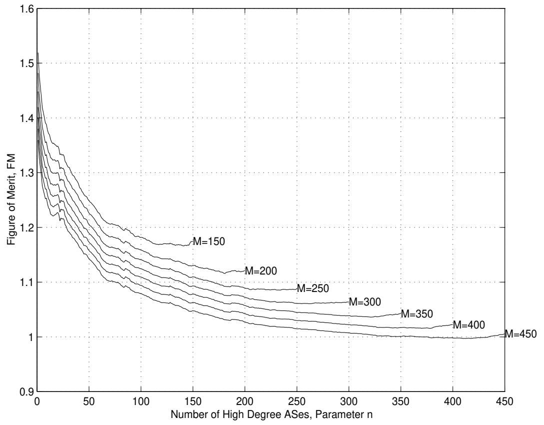

Figure 7. Figure of Merit (FM) Vs n for number of proxies (M) varying from 150 to 450

In Figures 6, 7 and 8, we plot the variation of the figure of merit(FM) as n is varied from 1 through M. The different plots are for different number of proxies(M) and give us an intuition on the weight of the highest-degree-first and farther-first algorithm in the hybrid optimal algorithm. We observe that as we increase the value of n from 1 the figure of merit begins to improve. In this portion of the graph, picking up nodes with higher number of neighbours is better than trying to place proxies far apart. This leads to more and more nodes at the top level getting proxies and their high connectivity playing a role in improving the figure of merit. In fact we observe that this improvement in the figure of merit continues till almost $n = M$. Thus we find that placing proxies in nodes with higher connectivity is better than placing proxies far apart. This can be explained by the power law[19] in the Internet where newer nodes tend to connect to nodes that are closer to the backbone of the Internet and thus in time nodes with higher connectivity have an even higher connectivity. Thus we find that in the Internet, there are nodes that have high connectivity(generally, the nodes in tier I) and although they are at a single hop from each other, they must have proxies placed in them, which would reduce the hopcount to numerous other ASes connected to them improving the overall performance.

Towards the end of the graph when n reaches M, we notice that covering the breadth of the graph begins to play a role. This leads to a worsening of the Figure of Merit(FM). Thus we find that there is an optimal value of n after which we should switch from the highest-degree-first heuristic algorithm to the farther-first heuristic algorithm. This explains the inflexion point in all these curves. Another interesting observation that was made in these plots is that for large values of M ($M > 800$) the plot becomes a flat line in the middle region. On closer observation we found that the upper bound on figure of merit was achieved i.e. each node either had a proxy in itself or one of its immediate neighbours had a proxy. It may be noted that for such large values of M, our algorithm attained the optimal solution of a brute force approach. For the cases of smaller values of M, our algorithm might be obtaining

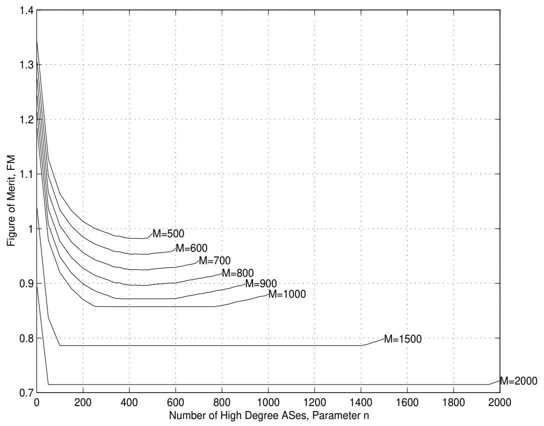

Figure 8. Figure of Merit (FM) Vs n for number of proxies (M) varying from 500 to 2000

the optimal answer but we have no means of verification.

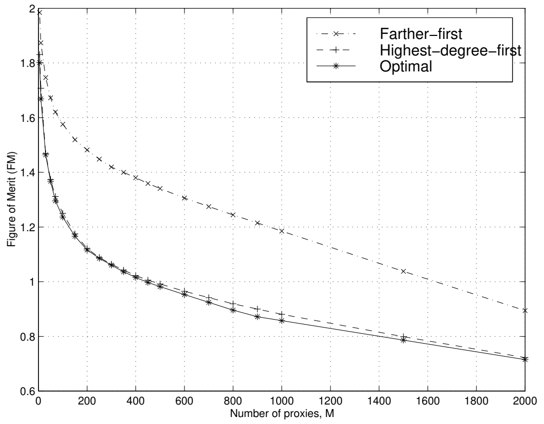

Figure 9. Comparison of FM achieved for the three heuristic algorithms for various M

In Figure 9, we compare the above three heuristic algorithms. We find that the highest-degree-first algorithm always outperforms farther-first algorithm. Also, as expected the optimal heuristic algorithm which switches from highest-degree-first heuristic to farther-first heuristic at the optimal point is seen to have the best Figure of Merit(FM) value for all values of M.

5. EXPERIMENTAL RESULTS

In the previous section, the performance of the proposed heuristics was shown over a given Internet topology. In this section, we present experimental results of using the optimal heuristic algorithms for proxy placement in the Internet. The publicly available routing table from the Route Viewer router in Oregon[9] captures the state of the Internet by establishing BGP peering sessions with 22 ISPs at 24 locations across the world. The BGP routing table in Route Viewer router is used to generate the AS topology using the algorithms by Gao.[17] Internet AS topologies at intervals of 6 months, from March 1998 through March 2001, are used in the experimental results.

Figure 10 illustrates the exponential growth of Internet over time. It is seen that there is more than three fold increase in the number of ASes over the past three years. Hence, the performance of the proposed optimal heuristic algorithm has been computed for a range of Internet topologies from March 98 (3430 ASes) through March 2001 (10461 ASes) to evaluate its performance over time and under practical constraints.

Figure 11 presents the variation in the figure of merit over different Internet topologies for a constant number of proxies placed. It was seen in Figure 10 that the number of ASes increases and the Internet topology changes over

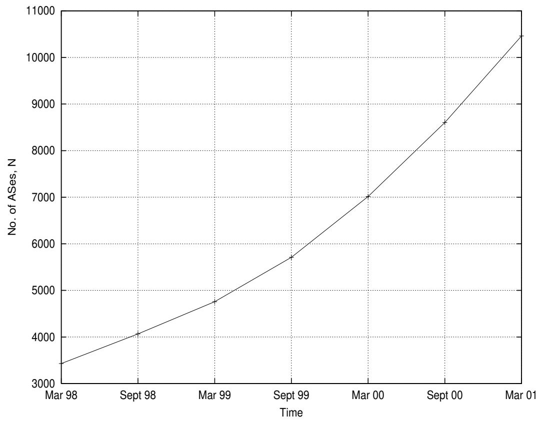

Figure 10. Growth of Internet over time.

time. To show the effect of the number of ASes, we scale A axis with the number of ASes in all the figures that follow. Two proxy placement strategies are considered namely *normal* proxy placement strategy and *constrained* proxy placement strategy and are defined below.

- *Normal* proxy placement strategy - Given an AS topology, the *normal* proxy placement strategy finds a new set of M ASes, by running the optimal heuristic algorithm. Every point in Figure 11 that represents the *normal* proxy placement strategy is obtained by running the optimal heuristic algorithm for the given AS topology.

Ideally, one would select a new set of proxies for every change in the Internet topology. However, practical considerations may inhibit a *normal* proxy placement strategy necessitating a *constrained* proxy placement strategy.

- *Constrained* proxy placement strategy - The *constrained* proxy placement strategy persists with the same set of proxies even as the Internet topology changes over time. In Figure 11, the optimal heuristic algorithm is used to select a set of M ASes for proxy placement for the AS topology of March 98 (3430 ASes) and the same set of proxies are used for the subsequent AS topologies.

The figure of merit worsens due to the growth of Internet for both the above proxy placement strategies. It is observed that the *constrained* algorithm has a poorer performance than the *normal* proxy placement algorithm due to the additional restrictions imposed.

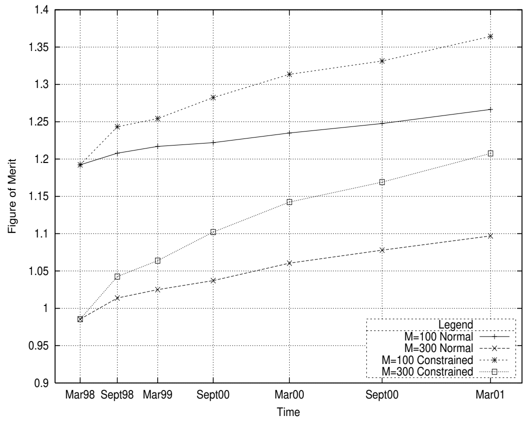

Figure 11. Performance Degradation with growth of Internet

It is noted that using a constant number of proxies over a rapidly evolving Internet leads to a slow but certain degradation in performance. Hence, a fair comparison would involve keeping the ratio of the number of proxies to number of ASes $\frac{M}{N}$ a constant. Figure 12 represents the variation in the figure of merit over different Internet topologies for a constant $\frac{M}{N} = 0.03$. The *normal* proxy placement algorithm is the same as described above wherein a new set of proxies is chosen for every Internet topology. However, the restrictions imposed for the *constrained* proxy placement strategy would hold in the following case too in the sense that using a new set of proxies may not be feasible and calls for a *incremental* proxy placement strategy defined as

- *Incremental* proxy placement strategy - The *incremental* proxy placement strategy builds upon previously obtained solutions by not changing the previous proxies and using the optimal heuristic algorithm solely to add the new proxies.

For instance, while 100 proxies need to be used for the Internet topology of March 98 ($N = 3430$) an additional 19 proxies are needed for the September 98 Internet topology($N = 4066$). While the *normal* proxy placement strategy would select the best possible proxy placement strategy for the September 98 topology by selecting a set of 119 ASes based on the optimal heuristics, the *incremental* heuristic would use the set of 100 proxies from the March 98 proxy placement solution and selecting the additional set of 19 proxies by using the optimal heuristic. As before we see that the restrictions imposed on the *incremental* proxy placement strategy leads to a degradation in the performance. However, the degradation in performance stays constant implying that using the optimal heuristic over a *incremental* proxy placement strategy is feasible for a rapidly evolving Internet. It is also observed in Figure 12 that maintaining a constant $\frac{M}{N}$ leads to an improvement in the figure of merit which indicates that maintaining a constant $\frac{M}{N}$ over an evolving Internet topology is an overkill when the aim is to maintain a constant figure of merit.

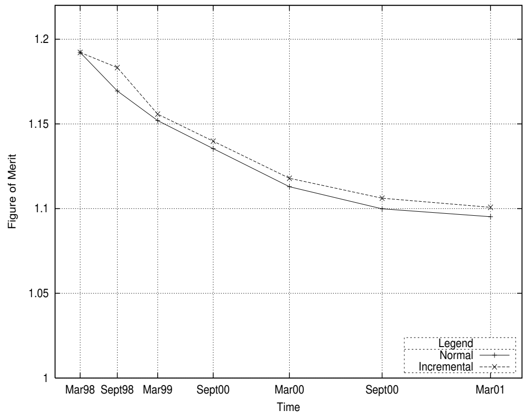

Figure 12. Performance Comparison of Normal and Incremental Proxy Placement

Figure 13 represents the variation of the number of proxies required over different Internet topologies for achieving a constant figure of merit of 1.192. It is seen that while the number of ASes increases from 3430 to 10461 over March 98 through March 01, the number of proxies required increases from 100 to 153 for the normal proxy placement strategy and to 168 for incremental proxy placement strategy. It is noted that the rate of increase of the required number of proxies for a given figure of merit is much smaller with respect to the growth rate of the Internet. This indicates the growth of Internet obeys the power law wherein the new ASes try to connect directly to the ASes with large number of neighbors. Such a growth model augurs well with the first heuristic(*Highest-degree-first*) which uses the M nodes with the highest traffic (number of neighbors) for proxy placement.

As mentioned earlier[10] is the only work to our knowledge, which attempts to solve the problem of proxy placement. Hence we have compared the performance of our algorithms with the best results for true Internet topology as mentioned in[10] i.e. by using the greedy algorithm for proxy placement. The basic idea of greedy algorithm is to choose one proxy placement at a time. In the first iteration, each of the N potential ASes are evaluated individually to determine its suitability for placing a proxy. The AS that yields the lowest cost (figure of merit) is chosen for placing the first proxy. In the second iteration, we search for a second AS which, in conjunction with the AS already picked yields the lowest cost. In general, in computing the cost, we assume that clients direct their accesses to the AS that can be reached with the lowest cost. We iterate until M ASes are chosen for proxy placement.

The Greedy Algorithm has a significant complexity and to reduce the computational cost, Greedy Algorithm implementation in[10] ranks the ASes according to the traffic load using the data collected from the MSNBC server

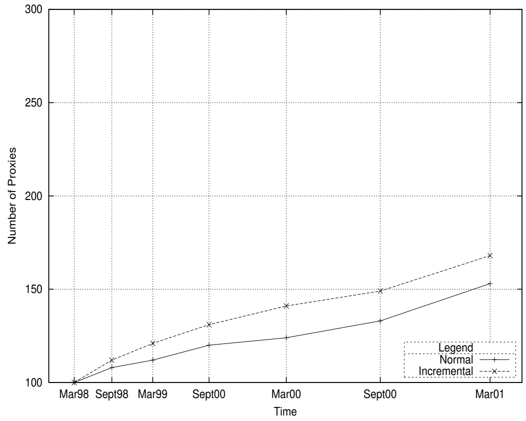

Figure 13. Proxies Needed for Constant Performance over time

logs and considers only the top, in terms of requests generated, few hundreds or few thousands of clusters for proxy placement. Due to the lack of any such data, we have ranked the ASes according to the number of neighbours in our implementation of greedy algorithm. Also,[10] does not take into account BGP policy issues and AS relationships mentioned in Section 2 while computing the cost to access a proxy as considered by our Modified Dijsktra algorithm. Hence, for a fair comparison we implement the greedy algorithm while considering the logical relationships between AS i.e. we use the MD algorithm to compute the cost to access the nearest proxy given the proxy placement instead of using physical connectivity between ASes to get the minimum cost. Figure 14 presents the variation in the figure of merit using optimal and greedy algorithm, over different Internet topologies for a constant number of proxies ($M = 100$) placed.

In experiments conducted by[10] on simulated and Internet topology, Greedy algorithm gave the best performance. We observe from figure 14 that both the optimal heuristic algorithms and greedy algorithms are comparable in performance. For earlier Internet AS topologies, the optimal heuristic algorithm performs far better than the greedy algorithm. However for more recent Internet AS topologies, while the performances of the two algorithms are comparable, the greedy algorithm does give a slightly better figure of merit than the optimal algorithm. Hence, we conclude that one cannot hope for *one single heuristic algorithm* to give the best results over a rapidly changing Internet topology. Also, one should consider the performance of incremental and constrained proxy placement strategies while evaluating the performance of any proxy placement algorithm since ideally one would prefer a proxy placement algorithm that would incur the least amount of perturbation to the system over a period of time. We have shown that the optimal heuristic algorithm does precisely that in providing proxy placements that give close to best performance and is well suited for both constrained and incremental proxy placement strategies.

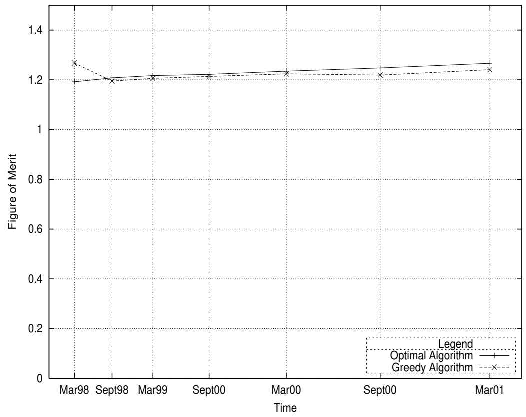

Figure 14. Comparison of Optimal Algorithm Vs Greedy Algorithm

6. CONCLUSIONS AND FUTURE WORK

Interdomain routing policies are constrained by the commercial contractual relationships. As a result of this routing at the AS level is dictated by policies set at the BGP routers of these domains. In this paper we present heuristic algorithms, which consider these restrictions during proxy placement and at the same time attempts to exploit the properties of today's Internet topology. We have performed experimental study involving data from the actual Internet as well as simulations and our results confirm with our theories. This is, to the best of our knowledge the first attempts at tackling the proxy placement problem by considering the hierarchical relationships at the Inter-AS level. We have seen that our algorithms provide the ideal solution possible at the cost of about 800-900 proxies. However we feel that there is a lot of scope for further improvement to the work outlined in this paper. In this work we have assumed that the content at all proxies is equivalent which might not be true in the real-world case. The algorithm can be extended to incorporate this feature. Also improvements can be done to this algorithm to explore the possibility of approaching the upper bound on the figure of merit with lesser number of proxies.

REFERENCES

1. N Yeager and R. McGrath "Web ServerTechnology". Morgan Kaufmann, 1996.
2. M. Baentsch, L. Baum, G. Molters, S. Rothkugel and P. Sturm "World Wide Web Caching: The Application-Level View of the Internet". *Proc. IEEE Communications Magazine,* Vol. 35, No. 6, June 1997.
3. P. Scheuermann, J. Shim and R. Vingralek "A Case for Delay-Conscious Caching of Web Documents". *Computer Networks and ISDN Systems,* Vol 29,No. 8-13, September 1997.
4. M.F. Arlitt, and C.L. Williamson. "Internet Web Servers: Workload characterization and Performance Implications" *IEEE Transactions on Networking* Vol.5,No.5,Oct 1997.
5. L. Breslau,P. Cao,L. Fan,G. Phillips, and S. Shenker. "Web Caching and Zipf-like Distributions: Evidence and Implications"
6. V.N. Padmanabhan, and L. Qiu "The Content and Access Dynamics of a Busy Web Site: Findings and Implications" In *Proc. of SIGCOMM'2000*, August-September 2000
7. B. Li, M.J. Golin,G.F. Ialiano, and X. Deng "On the potimal placement of Web Proxies in the Internet" In *Proc. of INFOCOM'99*, March 1999.
8. S. Jamin,C. Jin,Y. Jin,D. Raz,Y. Shavit,L. Zhang "On the Placement of Web Proxies in the Internet Instrumentation" In *Proc. of INFOCOMM 2000*, March 2000.
9. http://www.antc.uoregon.edu/route-views/
10. L. Qiu, V.N. Padmanabhan and G.M. Voelker "On Placement of Web Server Replicas" In *Proc. of INFOCOMM 2001*, April 2001
11. MSNBC *http://www.msnbc.com*
12. T. Corman,C. Leiserson and R. Rivest "Introduction to Algorithms". The MIT Press, Cambridge, Massachusetts, 1990.
13. R. Govindan and A. Reddy "An analysis of internet interdomain topology and route stability". *Proc. IEEE INFOCOM,* Kobe,Japan,April7-11 1997.
14. B.M. Waxman. "Routing of multipoint connections", *IEEE Journal of selected areas in communication*, pp 1617-1622, 1988.
15. M. Doar "A better model for generating test networks" *Proc. Global Internet, IEEE,*, Nov 1996.
16. E.W.Zegura,K.L.Calvert, and M.J.Donahoo Doar "A Quantitative comparison of graph-based models for internetworks" *IEEE/ACM Trans. on Networking* 5(6):770-783, Dec 1997.
17. L. Gao "On Inferring Automonous System Relationships in the Internet". **IEEE Global Internet**, Nov 2000.
18. G. Houston "Interconnection, peering and settlements-Part I". **Internet Protocol Journal**, March 1999.
19. Christos Faloutsos Michalis Faloutsos, Petros Faloutsos "On power-law relationships of the internet topology,". *Proc. of ACM SIGCOMM*, Aug 1999.
20. "Interconnectoin,peering and settlements-Part-II" **Internet Protocol Journal,** June 1999.