

Chapter 5

Placing Servers in Overlay Networks

In this chapter, we study another subproblem of the overlay network design problem: the placement of the MSNs with constraints on the client to server paths, reflecting the quality of service within the regional access networks. We envision that this imposition of service quality constraints on server to client paths is essential for the newer network services to achieve better service quality in order to attract and retain customers. While the server-to-server paths can be explicitly provisioned in order to ensure available bandwidth and routed to satisfy a delay constraint, this approach is generally not cost-effective on the more numerous client access paths. Consequently, the quality of the service is determined largely by network locations of the deployed servers. However, operating and maintaining these distributed servers represents a major cost for service providers and limits the number of servers that can be deployed.

To examine the trade-off between service quality and network cost, we ask the following question: *Given multiple networks and their estimated service parameters, how many servers are needed and where should an overlay service provider locate them to ensure a desired service quality to all its clients.* The measure of quality of service can vary from application to application: it can be delay for real-time applications, or bandwidth for content distribution applications, or a combination of both. The connection from a client to its designated MSN can stay within an ISP domain or may cross multiple domains. With the emergence of co-location services in major metropolitan areas, we expect that fewer clients

would need to be routed through multiple ISPs to reach their designated MSNs. Within an ISP network, the service provider can estimate these service quality parameters for a given client to a potential server location based on the client's network access technology and the capacities of the internal routing paths. Across the ISP domains, such estimation is also possible if the peering path between networks is explicitly indicated or if both networks guarantee a service level agreement from which we can infer the service parameters. Therefore, we assume that we can decide in advance whether or not placing an MSN at a specific location can provide a given client with the desired level of service quality. In the rest of this chapter, we use regional routers from different ISP networks to represent the aggregation of clients and use the network distance between a regional router and an MSN as the service parameter, however, our methods can apply to any generic metrics.

To answer the above question, we transform the placement problem to the set cover problem [14] and solve it using both linear programming (LP) relaxation and greedy heuristics. An instance of the set cover problem is that given a base set of elements and a family of sets that are subsets of this base set, find the minimum number of sets such that their union includes all elements in the base set. The server placement maps to the set cover problem as follows: an element corresponds to the network location of an edge router, which represents the aggregation of regional clients in an ISP network; The base element set contains all the network locations of edge routers; A set represents a potential server placement at one of the network locations; Each set includes all the network locations that are within the service range from the server location represented by the same set. By solving the set cover problem, we find the minimum number of servers and their locations, that will cover all clients within the service range. We will only consider the uncapacitated version of the set cover problem, where the servers do not have capacity limits and can serve as many clients as possible. We think this uncapacitated version is adequate since it is typically cheaper to buy more bandwidth at one location than to install a separate server. The set cover problem is NP-Hard [42] and has approximation ratio of $O(\log n)$ [23, 65]. We introduce a rounding technique to solve the integer-programming formulation of the set cover problem based on linear programming (LP) relaxation methods. The super-optimality of the LP problem

provides a lower bound to the IP formulation of the set cover problem. Using simulation, we show that this rounding technique approaches the lower bound very closely; in fact, it reaches the lower bound for a number of network configurations. Meanwhile, the greedy heuristic also provides good performance in all instances with significantly less computation complexity.

One contribution of our approach is that we have investigated several variations of the placement strategy and their associated costs. For example, what is the cost if each client should be served by a backup MSN as well as a primary MSN? What is the cost if backup MSNs are allowed to be placed at twice the range of the primary? What is the cost saving if service range can be relaxed? Or if we can compromise the service quality of some non-premier clients? Answers to these questions offers guidance to service providers on the economy of their planned services. These different placement strategies can be mapped to different node inclusion criteria when constructing a set, and we can then solve the resulting set cover instances with the same algorithms.

Another important aspect of our study is the network modeling used in our simulation. Existing network modeling tools, such as GT-ITM [88] and Tiers [20], can generate hierarchical network graphs with probabilistic network interconnections, however, they do not explicitly model the geographical locations of the network elements. In our model, we consider the potential of co-located servers which can access multiple networks from the same geographical location; this mirrors the behavior of co-location service providers in the current Internet. Therefore, when two network nodes of different networks are within a geographical vicinity, a server placed at this location can service clients, who are within the service range, in both networks. We show that these co-locations can greatly reduce the number of required servers, since they avoid long indirect paths through the network peering points by providing shortcuts from one network to another.

The rest of the chapter is organized as follows: we introduce the formal problem definitions and the algorithms in Section 5.1; we then describe the network models used in our simulations in Section 5.2; Section 5.3 presents simulation results; Section 5.4 discusses some of the related work; and in Section 5.5, we summarize our results.

5.1 Formal definitions and the Algorithms

The transformation of the placement problem to the set cover problem assumes that we are given a set of networks and their interconnections, as well as a specific routing policy that allows us to compute an end-to-end path for every pair of nodes in the networks. Typically, shortest path routing algorithm is used in intra-domain routing and for inter-domain traffic, packet flows are routed towards the nearest peering point between two networks (also called the “hot-potato” routing). With this routing policy, we can compute a routing table for each node n_i and the cost of each routing path $c(n_i, n_j)$, which is the sum of the hop distances along the path. For each node n_i , we compute a set S which includes all the nodes reachable from n_i within the routing cost of C . This is to say that if a server is placed at the location of node n_i , then all the nodes within this set can be serviced by this server. If n_i has co-location nodes, then the set S also includes all nodes reachable from each of these co-location nodes within the cost of C . By varying the criteria of including a node in a set, for example varying the cost C changes the service range of a server, we can explore different placement policies while using the same algorithms to find a solution for the set cover problem.

Let S_1, S_2, \dots, S_m be all the sets computed. The LP formulation of the set cover problem is:

$$\text{Objective: minimize } \sum_{j=1}^{j=m} x_j \quad (5.1)$$

$$\text{Subject to: } \sum_{j=1}^{j=m} a_{ij} x_j \geq 1 \quad \text{for } i = 1 \dots n \quad (5.2)$$

$$x_j \in \{0, 1\}$$

where x_j is the selection variable of S_j , a_{ij} is 1 if $n_i \in S_j$ and 0 otherwise.

A variation of the problem is to allow one primary and one backup server to cover each node. A backup server can cover twice the distance of the primary server. Let

T_1, T_2, \dots, T_m be all the backup sets, and $b_{ij} = 1$ if $n_i \in T_j$ and 0 otherwise. The objective here is still to minimize the number of selected sets but with the additional constraints of:

$$\sum_{j=1}^{j=m} b_{ij}x_j \geq 2 \quad \text{for } i = 1 \dots n \quad (5.3)$$

Since all nodes in the primary set are also in the backup set centered at the same server, $b_{ij} = 1$ if $a_{ij} = 1$; but we can not have a primary server also service the same node as the backup server — the constraint in (5.3) ensures the selection of a different server as the backup.

5.1.1 LP Relaxation-based Methods

The above formulation can be approximated by first solving the LP relaxation of the problem optimally and then rounding the fractional values to integers. The LP relaxation of the problem is to allow the selection variables x_j to take fractional values between $[0, 1]$. The LP relaxation can be solved in polynomial time and the rounding can be done in $O(n)$. Reference [33] introduced a rounding algorithm which is a p -approximation algorithm, where $p = \max_i\{\sum_j a_{ij}\}$ is the maximum number of sets covering an element. Although this worst case result is not very promising, we are more interested in the average case performance. We refer to the rounding algorithm in [33] as the *fixed-rounding (FR)* algorithm:

Step 0: Solve the LP relaxation of the problem

and let $\{x_j^*\}$ be the optimal solution;

Step 1: Output sets = $\{j | x_j^* \geq \frac{1}{p}\}$.

The intermediate solution for the LP relaxation naturally provides a lower bound $= \sum_j x_j^*$ for the set cover problem, since the fractional solution is an optimal solution and the LP relaxation is a super set of the set cover problem. We will use this lower bound to evaluate the quality of the solutions produced by our algorithms.

We have also devised an *incremental-rounding (IR)* algorithm that imposes more restricted rules while selecting sets based on the value of x_j^* . Whenever we select a set, we remove all the elements that satisfy the covering constraint in (5.2) due to the newly

selected set. Let M denote the union of all elements covered after each step. For the remaining uncovered elements in a set S_j , we compute $p_j = \max_i \{\sum_j a_{ij}\}$ for $i \in S_j \setminus M$. Among all the sets that have selection variables greater than the inverse of p_j , we choose the set that has the largest number of nodes that have not been covered.

- Step 1: Solve the LP relaxation of the problem
and let $\{x_j^*\}$ be the optimal solution;
- Step 2: Select set S_j such that :
- 2(a) S_j has the largest number of uncovered element;
 - 2(b) $x_j^* \geq \frac{1}{p_j}$;
- Step 3: Repeat step 2 until all elements are covered.

The correctness of the algorithm holds: for each uncovered node, at least one set has $x_j^* \geq \frac{1}{\sum_j a_{ij}}$ and $p_j \geq \sum_j a_{ij}$. By selecting all sets whose values satisfy 2(b), we are guaranteed to cover all the nodes. Further more, since p_j is non-increasing in each repetition and $p_j \leq p$, the set selection criterion is more restrictive than that in the FR algorithm, which in turn reduces the number of sets selected. Although the worst case bound is the same for both algorithms, we observe from our simulations that the IR algorithm typically performs much better than the FR algorithm.

An alternative to rule 2(a) is to select the set with the greatest x_j^* value, since the larger the value of the selection variable, the more “essential” the set may be. For example, if a node is covered by a single set, then the selection variable of this set must be 1 and the set must be selected. However, most of our simulations show that rule 2(a) generally performs better than this alternative rule. One plausible explanation is that rule 2(a) is more objective in attempting to include as many uncovered nodes as possible, while the alternative rule first selects those more “essential” sets, which may not contain many nodes.

It is easy to see that both of the algorithms can still have redundant sets in the final solution. To prune these unnecessary extra sets, we use a simple pruning algorithm as the final step to complete the set selection:

- Step 1: Sort all selected sets in increasing order of set size;
- Step 2: Starting from the smallest set, check if it can be removed without leaving any of its nodes uncovered.
- Step 3: Repeat Step 2 until all sets are checked.

5.1.2 Greedy Heuristics

A greedy algorithm is usually attractive due to its simplicity. In [41, 47], Johnson and Lovász introduced a greedy algorithm for the set cover problem with an $O(\log n)$ approximation ratio. The basic greedy attribute of the algorithm is to select a set at every step that contains the maximum number of uncovered elements. For the backup problem variant, we extend the algorithm by treating any node that has not satisfied the constraints of (5.2) and (5.3) as equally uncovered. At each step, we select a set that has the largest number of remaining uncovered nodes and repeat till there are no more uncovered nodes.

5.1.3 Comparison of the FR, IR and the Greedy Algorithms

We first compare our *incremental rounding (IR)* algorithm with the *fixed rounding (FR)* algorithm proposed in [33] and with the greedy algorithm. The results are further compared with the lower bound obtained as the optimal solution from the LP method. The LP solver we used, is called PCx [18] which is an interior-point based linear programming package.

We use a simple setup to investigate the relative performance of these algorithms. The underlying network graph is a single graph of randomly distributed nodes on a 100 by 100 unit length map. The service range of a server is 20 units. Ideally, if nodes are perfectly positioned, this will give a solution of $\lceil \frac{100}{40} \rceil \times \lceil \frac{100}{40} \rceil = 9$ selected servers regardless of the node density. The lower bound we obtained is indeed not far from the ideal and stays constant with increasing node density as shown in Figure 5.1.

We show the performance of the rounding algorithms with and without the pruning routine in Figure 5.1. As expected, the FR algorithm performs badly with increasing node density, since the largest number of sets covering a node p , also increases with node

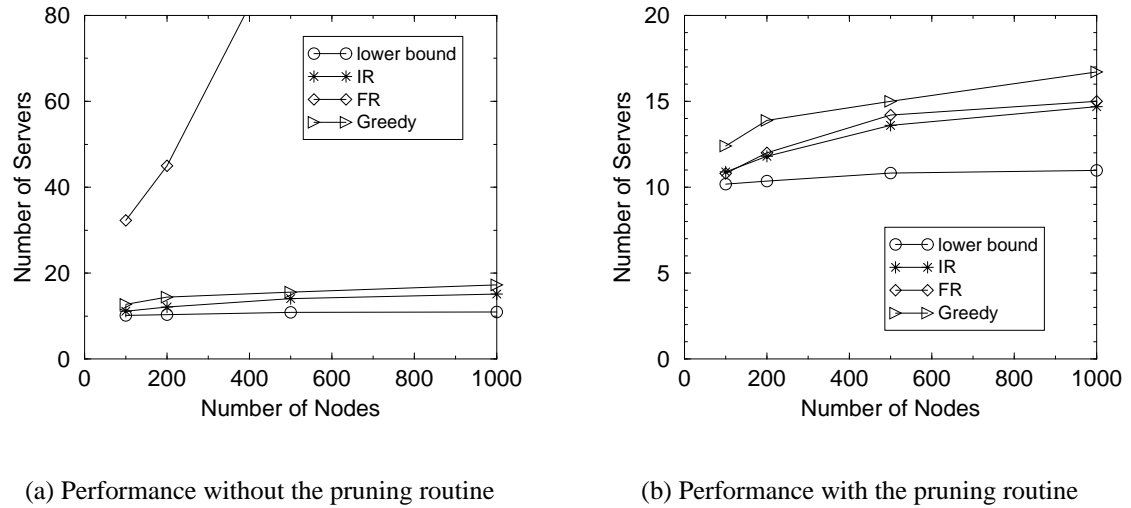


Figure 5.1: Performance Comparison of the FR and IR Algorithms

density which makes the selection criterion less strict. On the other hand, the IR algorithm is always the closest to the lower bound. The FR algorithm does benefit greatly from the pruning routine, achieving performance closer to the lower bound, and is only slightly worse than the IR algorithm, but better than the greedy algorithm. This relative performance holds for other settings we have tried as well. In the rest of the Chapter, we will mainly focus on the IR algorithm to evaluate the placement methods in more complicated network configurations.

5.2 Network Models

We model the networks using two types of graphs: random graphs and geographic graphs. The latter consists of network nodes within each of the 50 largest US metropolitan areas. For inter-domain network connectivities, we specify a set of parameters to determine the location and density of network peering points. For intra-domain network connectivities, as ISPs are not willing to disclose fully their network topology, we assume that they are able to engineer and operate their own networks with little or no congestion internally so that the delays between the routers are dominated by the link propagation delay. Consequently, we

model the intra-domain network as a complete graph. We assume the “hot-potato” routing policy at the inter-domain level, which minimizes the number of network domains crossed. Hence, traffic destined to another domain is always sent to the nearest peering points from the originator towards the destination domain. Although such policy does not result in the best global routes, it is widely used by the current inter-domain routing protocol: the Border Gateway Protocol (BGP) [66].

Our modeling choices do not correspond directly to the current Internet since the information needed to model geographically at the AS-level and the router-level networks is not generally available. The Netgeo tool [80] from CAIDA is probably the best mechanism available for capturing such data. It extracts information from the whois [30] database and attempts to map Internet hosts according to their domain names. However, the accuracy of this method is not at all clear since large IP address blocks can be assigned to a single network entity, and there is the possibility of inconsistency among whois databases. Additionally, it is not possible to determine all the locations of network peering points as many ISPs have private peering links in addition to their point of presence at the public peering points such as the MAEs and NAPs. We detail our parameter choices for the two models below and summarize the parameters in Table 5.1.

Table 5.1: Parameters for Generating Network Graphs

Parameters	Meanings
n	network size as # of nodes
$scale$	size of the network graph
N_p	probability of a city in a network
TX_p	interconnection probability between two networks
TX_{scope}	scope of a region possible for interconnection
TX_{ds}	interconnection density
$vicinity$	vicinity feasible for nodes co-location

Random Graph

In the random graph model, nodes are randomly distributed over a plane of size $scale \times scale$. The number of nodes in each network is uniformly drawn from the interval on $[min,$

$max]$. We divide the plane into fixed size regions according to the parameter TX_{scope} . The interconnection probability TX_p decides if a pair of networks interconnect; we choose TX_p based on the size of the two networks:

$$TX_p = \alpha e^{\beta \frac{\sqrt{n_1 n_2}}{max}}$$

where n_1 and n_2 are the number of nodes in the two networks, α and β determine the scale and shape parameters of the probability distribution, respectively. So, two large networks are more likely to interconnect than two smaller networks.

If two networks interconnect, we randomly select a number of regions to interconnect according to the interconnection density TX_{ds} . If there are multiple nodes from each network in the same region, we select the closest pair of nodes; if a region is selected, but one of the network does not have any node in that region, we choose another region until we meet the peering density criterion, or we have considered all regions. We allow co-location if nodes from different networks are within a geometric distance (the *vicinity* parameter) of each other. A server placed at a co-location can send traffic to all these networks with no additional cost.

Geographic Graph

In the geographic model, we use the 50 largest metropolitan areas as node locations. We then divide the US continent into 5 regions, namely northeast, northcentral, southeast, southcentral and west, and categorize nodes into regions with a certain amount of overlap. Unlike the random graph model where all networks share the same geometric space, the geographic model consists of two types of networks: regional networks and national networks. Each city joins the network with probability N_p : the selection of nodes for a regional network considers only nodes that belong to that region; while a national network considers all 50 cities. As before, we interconnect two networks with probability TX_p . The values of TX_p may be different depending on the types of the two networks. For example, two national networks will have $TX_p = 1$, since they are almost always interconnected;

while two regional networks are less likely to peer with each other directly but to transit through a national network. We allow interconnections only if two network nodes are in the same city and use TX_{ds} to decide the number of peering points of two networks.

Geographic Categorization of Metropolitan Areas

Region north_central[17] =	"Chicago, IL", "Minneapolis, MN", "Cincinnati, OH", "Indianapolis, IN", "Nashville, TN", "Grand Rapids, MI",	"Detroit, MI", "St. Louis, MO", "Kansas City, MO", "Columbus, OH", "Memphis, TN", "Louisville, KY" ;	"Cleveland, OH", "Denver, CO", "Milwaukee, WI", "Salt Lake City, UT", "Oklahoma City, OK",
Region north_east[21] =	"New York, NY", "Philadelphia, PA", "Cleveland, OH", "Milwaukee, WI", "Columbus, OH", "Buffalo, NY", "Rochester, NY",	"Chicago, IL", "Boston, MA", "Pittsburgh, PA", "Virginia Beach, VA", "Charlotte, NC", "Hartford, CT", "Raleigh, NC",	"Washington, DC", "Detroit, MI", "Cincinnati, OH", "Indianapolis, IN", "Greensboro, NC", "Providence, RI", "Richmond, VA" ;
Region west[10] =	"Los Angeles, CA", "Phoenix, AZ", "Portland, OR", "Salt Lake City, UT";	"San Francisco, CA", "San Diego, CA", "Sacramento, CA",	"Seattle, WA", "Denver, CO", "Las Vegas, NV",
Region south_central[12] =	"Dallas, TX", "Phoenix, AZ", "San Antonio, TX", "Austin, TX",	"Houston, TX", "St. Louis, MO", "New Orleans, LA", "Memphis, TN",	"Atlanta, GA", "Kansas City, MO", "Nashville, TN", "Oklahoma City, OK";
Region south_east[15] =	"Atlanta, GA", "Virginia Beach, VA", "New Orleans, LA", "Memphis, TN", "West Palm Beach, FL",	"Miami, FL", "Orlando, FL", "Greensboro, NC", "Raleigh, NC", "Louisville, KY",	"St. Petersburg, FL", "Charlotte, NC", "Nashville, TN", "Jacksonville, FL", "Richmond, VA";

5.3 Simulation Results

It is challenging to select a representative set of simulations that demonstrate the relationships between the methodologies, the configurations and the results, given the vast number of parameters. In order to concentrate on a few aspects which we considered interesting, we have mostly used small and uniform settings in the simulations presented in this section. We do not claim our network models capture all the fundamental characteristics of the Internet, but we believe that the combination of random networks and the geographic networks provides a wide enough spectrum to give us some confidence in the general utility of the methods. Throughout the section, readers are referred to Table 5.1 for the definitions of the parameters. Unless otherwise mentioned, we use the following default parameter values: the random graph scale is 100 by 100 units; the probability of network interconnection is 1.0 for both network configurations; the region size is 10 units and the co-location vicinity is two units in the random graph; the probability of including cities in the regional networks is 0.6 and 0.8 for national networks in the geographic networks.

5.3.1 Single Network

We first present results on a single network for both the random graph and the geographical graph. Each result for the random graph is averaged over 10 runs, while the result for the geographic graph is from a single run, since the node locations are all fixed.

Figure 5.2 shows the number of required servers when varying service range. In general, both the IR and the greedy algorithm performs closely with the lower bound. With increasing service range, the number of servers needed drops significantly. In most cases, the IR algorithm outperforms the greedy algorithm.

Figure 5.3 shows the performance against the different service requirements. We perform simulations on the following three scenarios: (a) $k = 1$, with only one primary server required to cover each node; (b) $k = 1$ and requiring one backup server; (c) $k = 1$ with relaxation on the server to client distance. The last scenario allows a compromise on

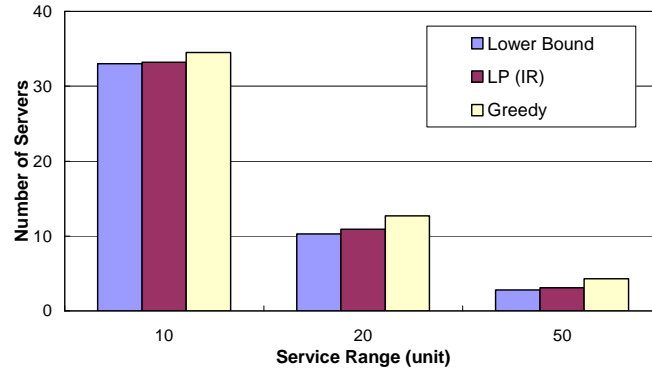
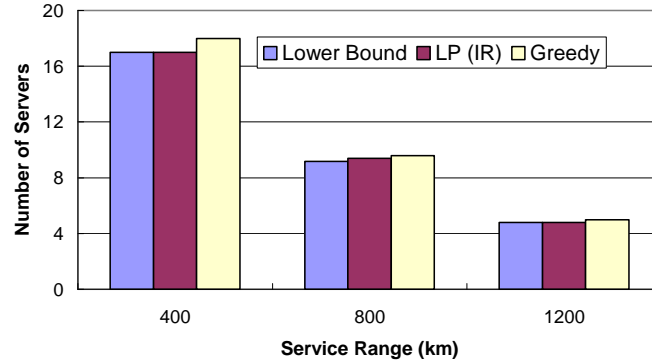
(a) Single Random Network: $n = 100$, scale = 100, $k = 1$ (b) Single Geographical Network: $n = 50$, $k = 1$

Figure 5.2: Variation on Server Service Range

the service standard for a limited number of clients. This allows service providers to be more cost effective and not to install servers just for a few remotely located nodes. The backup server range is twice that of the primary server for both networks. The relaxation is done by including every node in at least l server sets, even if it is not within the range of l servers. That is, if a node is in the range of $l' < l$ servers, we add it to the sets for the next $l - l'$ servers that it is closest to.

Figure 5.3 shows that the addition of the backup servers only increases the number of total servers slightly. The service range relaxation is also effective in reducing number of servers to about 75% and 55% of the original number, with $l = 3$ and $l = 5$ respectively. However, the relaxation is useful only when the service range is small. In Figure 5.3,

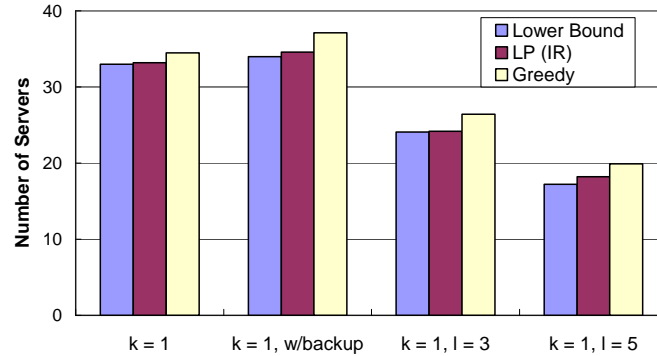
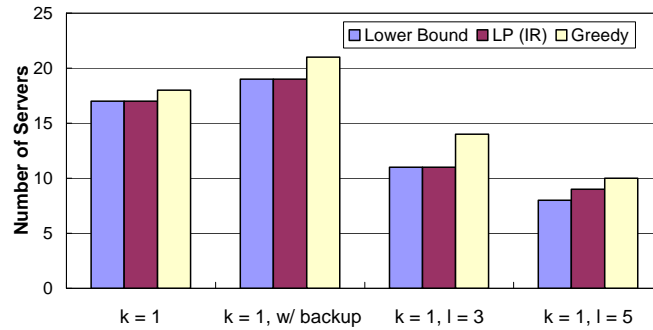
(a) Single Random Network: $n = 100$, scale = 100, range = 10(b) Single Geographical Network: $n = 50$, range = 400 km

Figure 5.3: Variation on Different Service Requirement

the service range is 10 units and 400 km in the two network configurations, which covers about 7% and 8% of the maximum distance in their respective networks. If we double the service range, we find that the relaxation becomes irrelevant since each node is likely to be included in multiple sets already. Table 5.2 shows the average node to server distance with and without the service range relaxation. Since there may be multiple servers covering a node, we select the closest server when computing the distance. The result for the random graph are the worst case among 10 runs. These results quantify the compromises made to service quality in order to reduce the server cost.

Table 5.2: Average Client to Server distance

	Random Graph (unit) range = 10			Geographic Graph (km) range = 400 km		
	mean	std.	max	mean	std.	max
$l = 0$	4.94	3.74	9.83	167.49	140.11	398.79
$l = 3$	6.30	4.47	14.82	259.78	222.65	1013.42
$l = 5$	8.45	5.56	25.36	304.69	238.59	1114.78

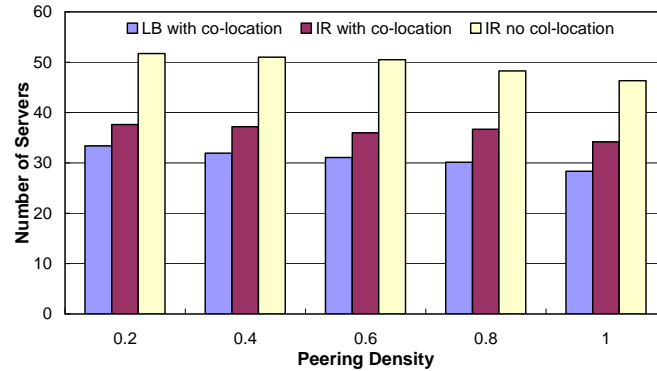
5.3.2 Multiple Networks

In this section, we study the relationships between server placement and the density of network peering links. By “peering links”, we mean both the peering and transit relationship between two ISPs. As these links aggregate and transport traffic from one domain to another, their limited capacities contribute significantly to the user experienced network congestion. Additionally, these network exchange points maybe located off the optimal path, resulting in longer and more circuitous routes. One way to circumvent these congestion points is to use co-location services, where servers can access multiple networks and can route traffic directly to these networks without going through the exchange points. We demonstrate the relative performance with and without server co-location in Figure 5.4.

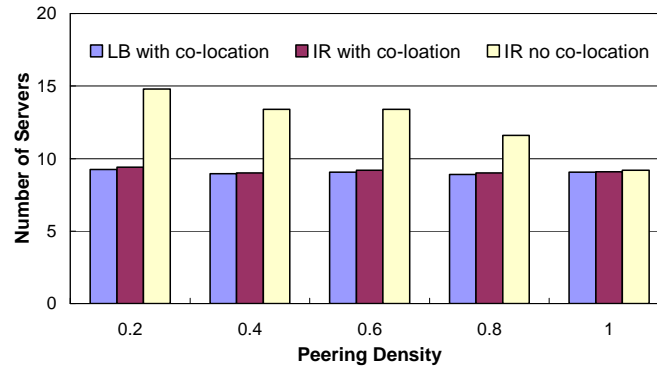
Table 5.3: Number of Peering Links In Use

Density	Random Network			Geographic Network		
	5 networkss, 100 nodes per network			5 regional, 1 national network, total 96 nodes		
	total links	co-location	no co-location	total links	co-location	no co-location
0.2	83.05	22.16%	38.77%	11.6	17.24%	80.17%
0.4	158.05	23.28%	37.52%	18.9	10.05%	82.54%
0.6	238.60	21.42%	38.98%	27.4	6.20%	82.48%
0.8	317.75	22.19%	35.22%	36.3	8.82%	90.36%
1.0	410.1	21.24%	35.94%	45.2	7.30%	96.90%

For this simulation, we use two network configurations: one is constructed from 5 random graphs, the other is constructed from 5 regional networks and 1 national network in the geographic model. Hereafter, we will use the term m - n to denote geographic networks consisting of m regional networks and n national networks. We use $TX_p = 1$, so every pair of networks is always interconnected, unless they do not have any common presences



(a) Random Network: 5 networks, 100 nodes per network, range = 20, $TX_{scope} = 10$, vicinity = 2



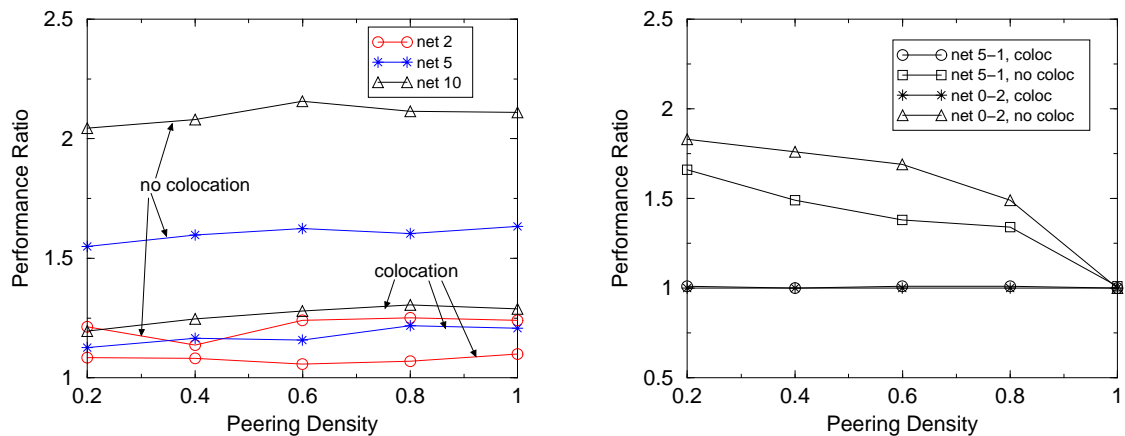
(b) Geographic Networks: 5 regional networks, 1 national networks, total nodes = 96, range = 800 km

Figure 5.4: Variation on Network Peering Density

in any of the peering regions. The peering density in Figure 5.4 determines the number of peering points of two networks.

Figure 5.4 shows the number of required servers for the lower bound with co-location and the IR algorithm with and without co-location. The co-location service reduces the number of required servers by as much as 50% when peering is sparse. The geographic graph appears to be more sensitive to the peering density, as the performance of IR with no collocation approaches the lower bound when the peering density approaches 1.

This is because the peering link is “cheaper” in a geographic network than in a random network. In the geographic network, two networks peer only if they both have presences in the same metropolitan area, which results in the delay on the peering link being zero. Contrarily, the peering link has a positive delay in the random graphs which adds to the server to client delay. Additionally, the “hot-potato” routing policy always selects the closest peering link rather than the one with the lowest delay. These combined effect determines that the increased peering density in random networks does not help much in reducing the server cost. Table 5.3 summarizes the number of links used in the two scenarios. In the random graph, the percentage of peering links in use stays almost constant; even more of them are available with increasing peering density. On the other hand, large percentages of peering links are in use in the geographic network when no co-location is allowed. The results for the case of co-location show that only a very small number of peering links are needed if servers are able to access multiple metropolitan area networks.



(a) Random Network: range = 20, vicinity = 2, 100 nodes per network.

(b) Geographic Networks: range = 800 km.

Figure 5.5: Relative Performance Ratio Against Lower Bound

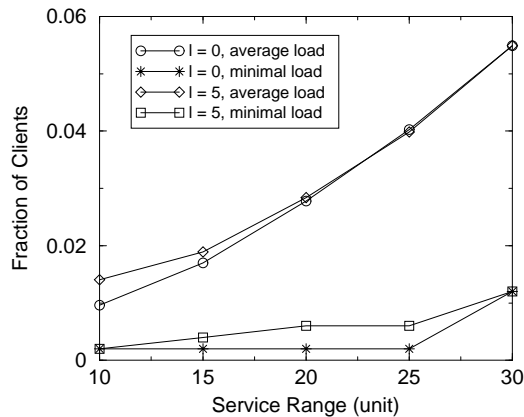
Figure 5.5 shows the relative performance ratio of the IR algorithm, with and without co-location, against the lower bound. This is the same measure as those in Figure 5.4 but with more varieties on the network configurations. We varied the number of random networks as 2, 5 and 10 and used two configurations for the geographic networks: 5-1 and

0-2. The results are mostly consistent with that in Figure 5.4. Additionally, it suggests that the performance of non co-located servers can degrade greatly with the larger number of networks.

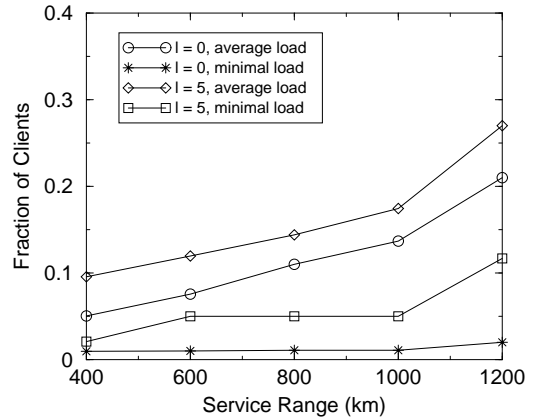
5.3.3 Server Load

Throughout this chapter, we assumed that the cost of installing and operating a separate server far out weights the cost of buying more bandwidth at a location. We are also interested in the distribution of load among the servers, particularly the minimum load on a server, since it is not very cost-effective to install a server only to operate under very small load.

Figure 5.6 shows the average and minimum server load distribution under the IR algorithm with co-location. As expected, the average server load increases with the increase of service range. The load distribution is more even on the random networks than on the geographic networks due to the different node distribution: the geographic networks have significantly higher population density on both coastal areas which creates higher load for servers.



(a) Random Network, 5 nets, 100 nodes per network, peering density = 0.4.



(b) Geographic Networks, 5-1 nets, total 96 nodes, peering density = 0.4.

Figure 5.6: Characteristics of Server Load

However, the minimal load does not lift as much for both networks. It is easier to understand this effect in the geographic networks, since a few metropolitan areas, such as Seattle and Portland, are more segregated from the rest of the areas. Consequently, it takes at least one server to cover and only cover these two areas. Although it is less obvious in the random networks, it seems that there are always a small number of nodes that are particularly far away from the rest of the group. The curves labeled $l = 5$ indicate the average and minimal fraction of clients served by a single server when we enforce each node to be included in at least 5 server sets. The increase of the minimal server load from the relaxation is more visible in the geographic networks than in the random networks.

5.4 Related Work

The application of the set cover problem in network design has two variants: the facility location problem and the k -median problem. The facility location problem minimizes the joint cost of server installation and the cost of connecting each client to its designated server. This problem has been applied to designing and placing network concentrators. The k -median problem minimizes the cost of connections between clients and servers under the constraint that no more than k servers can be used. Both problems are NP-Hard. The best known approximation algorithms can achieve constant ratio [28, 38, 77], if the connection cost is symmetric and satisfies the triangle inequality. For arbitrary cost, the worst case bound is $O(\log n)$. However, neither of the problems can be applied directly to the design of overlay networks, since in the overlay model the communication channels between clients and servers are over the commodity Internet and do not incur any cost to service providers. Rather, the major cost is the number of servers needed to service all clients and the access bandwidth required at each server's network interface.

Our model of server placement more closely resembles the set cover problem. The classic greedy algorithm for solving the set cover problem [41, 47] achieves an $O(\log n)$ performance ratio. In geometric spaces, the problem is easier. In [35], Hochbaum proposed

a shifting strategy that gives an $O(1 + \varepsilon)$ performance ratio. Unfortunately, the interconnections between networks dictate that the network propagation delay no longer exhibits the geometric properties of distance.

References [59,60] studied the problem of placing cache replicas in the network and formulated it as the k -median problem: given a specific number of servers, what is the best placement that achieves the highest average service level to clients, where service level is indicated by access delay from a client to its nearest replica. In [59], Qiu et al. proposed several placement strategies including: a greedy strategy that incrementally places replicas to achieve highest service quality; a hot-spot strategy that places replicas near the clients that generate the greatest load. In [60], the authors also proposed a max degree strategy by placing replicas in decreasing order of nodes' degrees. By simulating over several synthetic and real network graphs, they concluded that the greedy strategy performs remarkably well, achieving within 1.1 to 1.5 of the lower bound.

Our approach to network design is from a different angle. we are more interested in examining the necessary cost, in this case the number of servers, if we want to provide all clients the guaranteed service. This gives service providers insight into the relation of network cost and the achievable service quality. Then, they can make adjustments to reflect their revenue stream, such as eliminating servers that are only serving small numbers of clients. Contrarily, the work in [59, 60] measures the average service quality which masks the number of unhappy customers. Additionally, the performance of our approach, which is the number of required servers, is not susceptible to the cost metric of connections, since we only use it to categorize clients as serviceable or not by a server; while theirs is achieved for a specific cost metric, namely the access delay. Since the connection cost metric depends heavily on the application, it is not known if the same ratio could be achieved.

Another difference is that we model the network geographically and consider server co-locations. As networks overlap geographically, the number of potential server locations is much fewer in number than the number of network nodes that need to be considered. In [59,60], they used network graphs consisting of tens of thousands nodes for router-level graphs and thousands of nodes for AS-level graphs. Consequently, the optimal algorithm

based on LP relaxation is too expensive for their models. We think considering the geographical locations of servers is important and corresponds to the current trend in the Internet, where servers are typically located in data centers in different geographic areas. This reduces the problem size and enables us to solve it more optimally. In Section 5.3, we compare the performance of our algorithm both with co-location and without, and show that with co-location we can reduce the number of required servers to approximately half of that with no network co-locations.

5.5 Summary

We have presented a server placement method in overlay networks as an application of the set cover problem. The placement satisfies constraints on the server to client paths, which indicate the achievable service quality along the path. We expect that network provisioning for quality of service will become more common as the Internet continues to grow; and such an automated methodology is useful for service providers to analyze the potential cost of network provisioning.

We solved the set cover problem using methods based on linear programming relaxation as well as greedy heuristics. We also presented an incremental integer rounding algorithm for the LP-relaxation based method. Our network settings model explicitly the presence of co-location services, which have become increasingly popular for business corporations to out source their data servers. Our results indicate co-location can save up to 50% of the server installation cost. We also presented variants of the simple set cover problem to allow backup servers and to allow distance relaxation. These variances brings opportunities to provide more cost effective services. Through simulation, we studied the behavior of the algorithms under various network settings and observed the implication of network peering density and the characteristics of server load distributions. Although, LP-relaxation based methods are traditionally considered as too expensive and complex to use in more time-critical contexts, we found that it is suitable and effective for overlay network design where the solution quality is more important than the running time.