Proxy Location problems and their generalizations

Sumi Choi[1] and Yuval Shavitt[2]

Abstract

In this paper, we suggest a new angle to look on proxy location problems. We view the proxy as a service and generalize the problem of optimally placing proxies to optimally placing any service in the network. We differentiate between two cases: transparent services, such as transparent caches (TERCs), and non-transparent services where the end stations, such as hosts using a cache, need to be configured with respect to the service locations. This formulation enables us to study in a unified framework the location problems related to active networks, overlay peer-to-peer networks, as well as, CDNs and caching. For both cases we formalize the problem of optimally placing network servers and introduce approximation algorithms. We present simulation results of approximations and heuristics. We also present an optimal solution for the location problem for a special topology. We show, through a series of examples, that our approach can be applied to a variety of different services.

1. Introduction

Caches and proxies have been used for a while to improve server reaction times for web users, to reduce the traffic in the network, and to relieve load from servers. Not surprisingly the problem of optimally placing caches have been studied extensively in the past [14, 13, 9]. Similar placement problems have been studied in the context of content delivery networks (CDNs) [12, 5] and in the context of electronic content [8].

In this paper, we suggest a new angle to look on these location problems. We view the proxy, cache, or any other replica server as a service and generalize the problem of placing them to placing any service in the network. We differentiate between two cases: transparent services, such as transparent caches (TERCs), and non-transparent services where the end stations, such as hosts using a cache, need to be explicitly configured with the location of the cache. This formulation enables us to study in a unified framework the location problems related to active networks, overlay peer-to-peer networks, as well as, CDNs and caching.

Active networks [18] bring the ability to place services anywhere in the network. Such services may include caching, authentication, compression, protocol conversion, etc. Due to maintenance costs and the overhead associated with each deployed service, we would like to limit the number of such service centers. Given the situation, we consider two optimization problems for placing servers: finding a limited number of servers to obtain the maximum gain, and finding the minimum number of servers needed to achieve a given level of service. This paper is the first, we are aware of, that looks and this problem.

Most interactions in the Internet form sessions between end points. Thus, we particularly consider the cases, where services are associated with sessions. A service performed on a session should bring some quantifiable gain, and naturally, its usage entails some cost. The same service may have different gain and cost according to the economic model. For example, if an ISP deploys a caching service, the gain can be either the saving in the traffic volume of the network, or the reduction of the delay for its clients. The optimal placement of the service needs not be the same for these two optimization criteria, though the service performed is the same. Session-oriented services can be classified according to several criteria, which influence the mathematical modeling of the optimization problem. In the following we discuss two different criteria: symmetry and transparency.

The symmetry of an interaction between end parties is determined by the cost incurred by the traffic. Note that when a service is involved at a midpoint, the cost may change depending on the processing applied to the traffic. When the cost does not change, the interaction is symmetric. Symmetric interaction occur either because the same type of information flows both ways, like in a full duplex conference gateway, or because the 'cost' of the flow does not change when passing through the server, e.g., when the cost is bandwidth and the service is a protocol conversion. An example of an asymmetric interaction is a video com-

[1] Sumi Choi is with the Dept. of Computer Science, Washington University in St. Louis, USA.

[2] Yuval Shavitt is with the Dept. of Electrical Engineering - Systems, Tel-Aviv University, Israel.

[3] This work was done in part while the authors were in Bell Labs, Lucent Technologies.

pression server where the cost in the consumed bandwidth. The cost of getting the data to the server is much higher than the cost of transmitting the data to the player.

Naturally, many of the services in the networks are between two parties: protocol conversions, application layer conversion (a language translation service), compression of a videophone session, and many more. Multi-party services are also growing in numbers: conference calls, games, etc. However, there are also some services which seem to involve only one party, such as caching. We claim that these are special cases of a session where one end point, the content provider, is fixed and the other end point, the client, is dynamic. A content caching service is one that benefits the requesters with the delivery of locally cached content. However, the content origin needs also to interact with the cache to deliver the content. This is not more than the case of an asymmetric session where the 'cost' of delivery from the server (cache) to one side (the client) is higher than the cost of delivery of the content from the other side (content origin) to the server (cache). It is important to note that although caching is a special case of a session service, results obtained for caching cannot be applied to the general case of session servers, at least not in all its forms.

Finally, servers may be transparently placed in the network such that user need not be aware of their location, or even about their existence. Or, they may be placed at known locations requiring clients to demand a service directly to them. In the sequel we will elaborate about the modeling of these two distinct cases.

1.1. Service Networks and Server Placement

Firstly, we consider overlay networks for providing services. In this types of networks, service providers are in control of a network overlaid on top of the underlying network and configure their application sessions with appropriate servers. Session configurations usually aim at optimizing a certain metric value, which we refer to as cost, such as the performance or the efficiency of the sessions. In cases where the time constraint is crucial as in interactive media transmissions, server locations have more impact on the performance, i.e., the latency. This is easy to see as end-to-end paths has to pass, at least, one of the server locations. In other cases, as in the content adaptation and the video compression, the service changes the bandwidth consumption of the sessions. For an economic reason, one may want to place servers at certain locations such that the total required bandwidth is minimized While the cost for the session configuration and the criteria for the optimization change with the application, the server placement problem is generalized into a couple of formations.

One formation is placing servers in overlay service networks, and is described in the following situation. Suppose there is fixed budget, k, that limits the number of servers to be placed in a network. The goal of the problem is to find k locations for the servers that optimize the average value of the session configuration cost. The problem is closely related to the k-median problem where one needs to place k medians such that the sum of the distances between the nodes and their closest median is minimized. We discuss the relationship between the two problems in Section 2. In another form of the problem the goal is to identify the minimum number of locations that guarantee certain constraints on the session configuration cost, which is specified by the target application, e.g., maximum delay between any pair of clients. This form of the problem is closely related to the k-center problem.

The second case is the location of **transparent servers**. Transparent servers are servers whose location, or even their existence, is kept unknown to the end-points (users). The main advantage of transparent servers is that they require no configuration of the session end-points, yet the server can only serve traffic that flows through it usually by intercepting it. In order to provide transparent services, one has to place servers so that at least one of them is located on each of the session routes. Therefore, transparent servers enforce stronger constraints on the server locations. In this case, the goal is to identify the minimum set of server locations that satisfies the constraint of each session. We view the problem of transparent server placement as an instance of the set cover problem and discuss it in Section 5.

1.2. Applicability

In our study, we assumed static traffic usage for placing servers. This assumption on traffic usage is supported by the study performed by Krishnan *et al.* [13] for caching. In their study, they checked the daily client population for a medium size web server and found that the day-to-day correlation was minuscule: 2.7-7.5% of the clients user population appeared in any two out of the 14 days sampled. However, when the overall demand from a network region was compared the correlation was very strong between the weekdays. As a result, placing servers (caches in [13]) based on the overall statistics proved effective and resulted in a penalty of 1-10% in the gain (compared to the optimal daily placement) for most of the weekdays.

2. Related Works

The problem of placing servers for a group of single clients has several well-known variants that were all studied extensively [19]: the facility location problem, the k-median problem, and the k center problem. In these problems, the service that the facilities provide is in an abstract and simple form which is described as an assignment of each end point to a server (a facility location) in a metric space or a graph. Then, the objective term to be optimized is given with the end-point-to-server distances and, in case of the facility location problem, with the costs of opening facilities. In this work, we expand the concept of services to groups of nodes and discuss the corresponding server placement problems.

Although all the above problems are intractable, in general graphs, there are approximation methods that find in polynomial time a solution which is guaranteed to be within some ratio from the optimal. The practicality of the approximations vary, which is part of this study. Approximation algorithms for the facility location problem have been proposed with factors of $O(\log(n))$ (for the greedy algorithm [10]), and 1.74 [7]. The k-median problem is harder to approximate due to the restriction on the number of servers (k), however, there are approximations with an extra relaxation on that restriction. For general graphs, Lin and Vitter [16] suggest a $(1+\epsilon)$ approximation that relaxes the restriction of the number of servers to be up to $(1+\frac{1}{\epsilon})(\ln n+1)k$. Arora, Raghavan, and Rao [3] extended the techniques of Lin and Vitter to achieve a polynomial-time approximation for the Euclidean space.

To show the difference between the k-median problem and our k-session server problem, consider a ring network comprised of n nodes numbered 0 to $n-1$ (n is even). Suppose we have two sessions, one between node 1 and node $n/2-1$ and one between node $n-1$ and node $n/2+1$. We would like to place two servers in the network. For the 2-median problem, the optimal locations for these two servers are at node 0 and node $n/2$, with gives a cost of 4, since each of the four nodes is exactly one hop away from its nearest server. This is also the optimal location in the 2-center problem. However, this choice of two locations is bad for the session server problem because it requires both sessions to make a detour of two hops, while the two servers can be placed on the two session routes, requiring no detours.

For some special cases, the session server placement problem was studied in the past. For the case where there is one sender and a large number of receivers such as when objects need to be cached, the problem was studied extensively. Polynomial solutions for trees and other regular topologies exist [13, 14], but when multiple sources are considered only heuristics were suggested [13, 9]. As we show in Section 4.1, results obtained for the cache location problem are not always indicative to the general case. Needless to say, not all the variants studied in this paper are covered by previous work, in particular, transparent server placement, which is studied in its general form here, was studied in the past only for the simple tree and ring cases [13, 14].

3. Model and Metric

The network is represented by a graph $G=(V,E)$ where V is the set of nodes, and $E \subset V \times V$ is the set of links. The set of sessions, S, is a collection of tuples $s_i=(v_{i_1}, v_{i_2}, \ldots)$, where $v_{i_j} \in V$. Each tuple s_i represents a session whose end-points are the nodes v_{i_j}. In cases where the sessions are limited to unicast connections s_i contain exactly two nodes. We give some modeling examples in this section.

Consider a group communication application such as a conference call. In this case, the session cost is determined by the average latency of the data delivery from speakers to listeners. Assuming $d(u,v)$ is the latency from u to v, the value of the cost given to a configuration of a conference call session $s_i=(v_{i_1}, \ldots, v_{i_n})$ with the center at v_c is

$$cost(s_i) = \sum_{v_{i_k} \in s_i} w_i \cdot 2d(v_{i_k}, v_c)$$

where w_i is the weight of session s_i. E.g., if one would like to give equal weight for all sessions regardless the number of participants, we can set $w_i = 1/|s_i|$. Remember that s_i is only one group out of the large group of sessions, S, that are served by the same servers.

To demonstrate the difference between symmetric and asymmetric cost consider unicast sessions that uses compression server. When the optimization criterion is the end-to-end delay the cost of a session s_i passing through a server v_c is given by

$$cost(s_i) = d(v_{i_1}, v_c) + d(v_c, v_{i_2})$$

where $d(v_i, v_j)$ is the transmission delay between node v_i and node v_j. However, if the optimization criterion is the bandwidth consumption the cost of session s_i passing through a server v_c is given by

$$cost(s_i) = bw_{in} d(v_{i_1}, v_c) + bw_{out} d(v_c, v_{i_2}) \qquad (1)$$

where $d(v_i, v_j)$ is the hop-distance between node v_i and node v_j, and bw_{in} and bw_{out} are the data bandwidth rates into and out of the compression server, respectively. One may define $d_{in}(v_i, v_j) = bw_{in} d(v_i, v_j)$ and $d_{out}(v_i, v_j) = bw_{out} d(v_i, v_j)$ and rewrite Eq. (1) as

$$cost(s_i) = d_{in}(v_{i_1}, v_c) + d_{out}(v_c, v_{i_2}) \qquad (2)$$

The cost is symmetric when $bw_{in} = bw_{out}$, asymmetric otherwise. For transparent servers, there is an additional constraint. A server has to be placed on the route of each session.

4. Servers in Overlay Networks

In this section, we consider an overlay network where individual sessions can be configured with a server dynamically. The problem is stated as follows. We assume that there is a budget for $k(< n)$ servers given a set of sessions. The goal is to find the k locations for the servers in the network, that minimize the costs involved with the sessions.

Let c_{ij} be the cost of serving session i with a server located at node j. When the length of session routes is used for the cost, the cost is defined as $c_{ij} = \sum_{v_{i_k} \in s_i} d(v_{i_k}, v_j)$. The c_{ij} values can be calculated efficiently after running an all-pair shortest-path algorithm. Now, let x_{ij} be a variable indicating whether session i is using a server at node j. The following 0-1 integer program (IP) solves the server placement problem for the sessions:

$$\text{minimize} \quad \sum_{s_i \in S} \sum_{j \in V} c_{ij} x_{ij} \qquad (3)$$

subject to

$$\forall i \in S, \ \sum_{j \in V} x_{ij} = 1 \qquad (4)$$

$$\sum_{j \in V} y_j \leq s \qquad (5)$$

$$\forall i \in S, \forall j \in V \ \ x_{ij} \leq y_j \qquad (6)$$

$$\forall i \in S, \forall j \in V \ \ x_{ij}, \, y_j \in \{0, 1\} \qquad (7)$$

Solving this program is $\mathcal{NP}$-hard, thus it is usually done by relaxing condition (7) and allowing the x_{ij} indicators to take rational values between 0 and 1. Surprisingly, this integer program has the exact same structure as the standard program for the k-median problem. Note that here, we have $n|S|$ x_{ij} variables instead of n^2 variables which are used in the original k-median problem. In the worst case, this is only a factor of n more variables. Thus, results obtained for the k-median problem are also valid for our problem with proper adjustment. Specifically, we can use the ϵ-approximation suggested by Lin and Vitter [16] which finds locations at the cost not more than $(1 + \epsilon)$ of the optimal cost, but it might need up to $(1 + \frac{1}{\epsilon})(\ln n + 1)$ more service centers. This is most likely the best one can hope to achieve for the k-median problem if an ϵ-approximation is desired for the cost [3].

It is important to note that the result above does not assume symmetry in the graph distances, i.e., $d(v_i, v_j)$ may be different from $d(v_j, v_i)$. In addition, the result above does not require the triangle inequality to hold, i.e., $d(v_i, v_j)$ may be larger than $d(v_i, v_k) + d(v_k, v_j)$. Savage et al. [17] showed that for about 50% of the routes in the Internet there exists other routes through some other node which are shorter, or in other words, 50% of the routes in the Internet participate in some triangle that does not obey the triangle inequality. It is worth mentioning that in most cases the inequality is not violated by a large percentage. For cost metrics obeying the triangle inequality, better approximations exist [15, 3].

4.1. A simulation study

We generated Internet-like power-law networks based on an algorithm suggested by Albert and Barabasi [1]. The

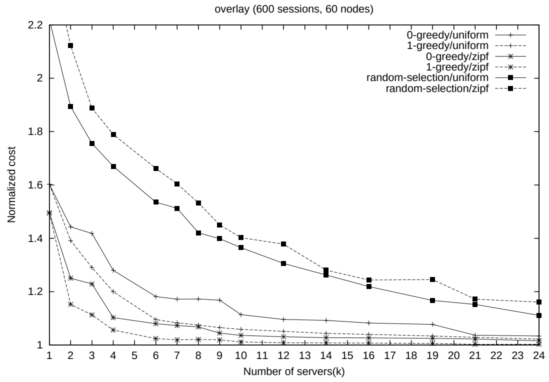

Figure 1. The cost as a function of the number of servers, in a network of 60 nodes

following parameters were used in the network generation: $m_0 = 4, m = 3, p = 0.1, q = 0$. We describe the simulation set up here. For each network we generated $10n$ sessions that are randomly generated according to the two models, *uniform* and *Zipf*, where n is the number of nodes in the network. In the *uniform* model, both session end points were selected uniformly from the network nodes. In the *Zipf* model, we selected one end-point uniformly and the other according to the Zipf distribution (with parameter 0.8) which reflects better the distribution of service provider, say a web server [4]. Each point in our simulations represents 10 random session distributions on three different networks, total of 30 runs. As the service model, we considered unicast which requires one server. We set the session cost metric to be the hop count. Clearly, the optimization goal is to minimize the total cost. We simulated the following algorithms:

- **Random:** k session servers are placed randomly in the network nodes.
- **Greedy:** k servers that reduce the total session cost most are selected in an iterative fashion.
- **1-Greedy** [13]: servers are selected in each greedy step, where one server is removed and two servers are added so that the total cost is reduced most.
- **ϵ-Approx** [16]: Lin and Vitter's ϵ-approximation, based on fractional linear programming combined with greedy set cover solution.

In the first set of simulations, we measured the resulted cost as a function of the number of servers. Figure 1 presents the cost for networks of 60 nodes. The number of sessions was ten times the number of nodes, and we varied the number of servers, k. The cost is normalized such that

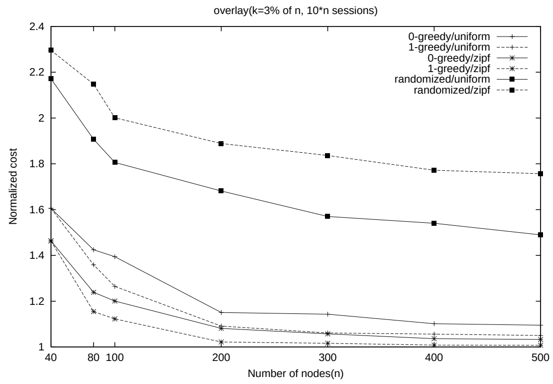

Figure 2. The cost as a function of the network size, when the number of servers is fixed to 3% of the network size

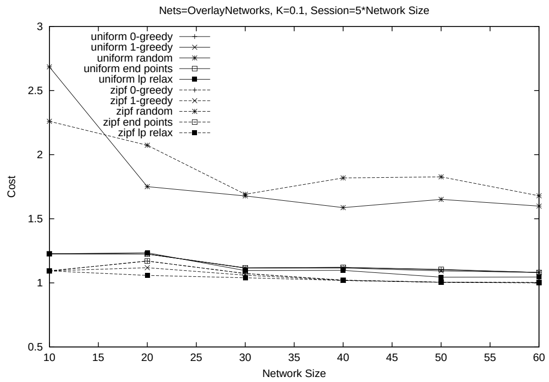

Figure 3. Costs of overlay networks including results of the ϵ-*Approx* with $\epsilon = 2$ (with network of 10 to 60 nodes, 10% of nodes as servers, and Zipf/uniform distributions.)

1.0 represents no detours due to server placement. Overall, the *Greedy* algorithms perform better than *Random* providing smaller costs under both distribution models. The *1-Greedy* algorithm particularly shows better results, which may vary depending on the size of networks. More details are given in [6]. All algorithms demonstrate a diminishing return curve: for the first servers we gain much in performance but this gain diminishes as the number of servers increases. Diminishing return was also observed for placement of mirrors by Cronin *et al.* [9]. It is surprising, though, that for only a small number of servers, even just twelve, we already improve performance such that the detour overhead is less than 20%. We note that the knee point where the cost curves flatten is around 3-4% of the number of nodes.

It is clear from the figures that under the Zipf distribution we get much better improvement than when the session end-points are uniformly selected. This holds for all three algorithms. This is followed from the observation that the Zipf distribution concentrates load on a few nodes, and we get a good cover of most of the sessions by placing servers near the few nodes that participate in many sessions (say, popular web sites).

In another set of simulations, we set the number of servers, κ, to be a constant factor of the network size and varied the number of nodes in the network. From Figure 1, the reasonable cost-performance trade-off is around 3-4%. Thus, we selected κ=3%. The results where we vary the number of nodes between 40 and 500 are shown in Figure 2. As before, the cost is normalized such that 1.0 represents no detours due to server placement. We can observe here that for all the tested algorithms the performance improves with the network size. This is because the diameter of these networks increases logarithmically [2] with the number of nodes as we increase the number of servers linearly.

Finally we present comparison of the ϵ-*Approx* algorithm against the other methods. Due to the computational complexity of this algorithm the simulations were limited to small networks, and the results are shown in Figure 3. As can be expected, the ϵ-*Approx* performs better than all the rest for both uniform and Zipf distributions. For the Zipf distribution the difference diminishes quickly, for as little as 40 nodes it becomes close to none; for uniform distribution the gap remains evident in all the simulated network sizes. However, remember that ϵ-*Approx* may place more servers (see [6]), thus, we conclude that the greedy algorithms are more efficient and simpler to implement and seem to perform comparably to ϵ-*Approx*.

4.2. Extra constraints

We can incorporate additional constraints for individual sessions in the IP formulation described earlier. We consider one of the most practical constraints in this paper, which is to set the upper bound on the cost of sessions such as to limit the maximum delay of sessions. This can be done by filtering out all the x_{ij} variables that violate the required condition. For example, to constraint the end-to-end delay of a session we can simply add a condition

$$\forall i \in S, j \in V, \text{ s.t. } c_{ij} < T_D \quad x_{ij} = 0$$

where c_{ij} is the delay for the session i given the server j. Of course, this type of formulation should be used only if the constraint is not very restrictive. Too restrictive constraint can render the problem unsolvable. In this case, one should use different formulation, based on set cover, that is presented in Section 5.

4.3. Optimal placement of non-transparent servers on a line

We stated that solving the IP formulation for the server placement problem of overlay networks is an intractable problem for general networks. We present here an optimal solution for the server placement problem in the special case of line networks.

Consider a line of n nodes numbered from 0 to $n-1$. The input is the set of sessions $s_i \in S$, such that $s_i = (v_{i_1}, v_{i_2})$. A node can accommodate both a session end-point and a server. Let $FC(l_o,\ l_i)$ be the extra detour cost on the segment $(l_o,\ l_i)$, where two servers are located at l_o, and l_i, and no server is located inside the segment. For $n-1 \geq l_o > l_i \geq 0$ this cost can be easily computed from the input in $O(n^2|S|)$. Let $C(l_o,\ k')$ be the overall extra-cost of detours in the segment $[0, l_o]$, when k' servers are located optimally in it, while one is forced to be at l_o. We can write for $k' > 2$ and $l > k'$,

$$C(l,\ k') = \min_{k > l' \geq k'-1} \{C(l',\ k'-1) + FC(l,\ l')\} \quad (8)$$

and solve it using dynamic programming with $O(n^2|S|)$ operations. Due to lack of space we omit the details, which appear in [6], and only state the results.

Theorem 1 $C(n-1, k+1)$ *is the optimal cost of placing* k *servers in the line* $[0, n-1]$.

5. Transparent Servers

For service networks with transparent servers, we have a stronger constraint on the server locations: a server has to be located on the route of every session. Thus the routing should be a part of the input for this problem, and what we seek is a minimum number of servers satisfying the constraint. This approach can also be applied to the situation when the main goal of an application is to guarantee strong constraints or quality of services to their sessions, e.g., end-to-end delay.

We formulate both problems as a set cover problem. Given a network graph $G = (V, E)$ and a set of sessions S, we define a session cover sc_i for each node $v_i \in V$ as the set of sessions in S, such that the corresponding node v_i satisfies the constraints of the sessions in sc_i. In case of the transparent servers, sc_i contains the sessions whose route include the node v_i. For the applications with per-session constraints, sc_i contains the sessions whose constraint can be accommodated by the node v_i. By selecting a set of session covers C such that $\cup_{sc_i \in C} sc_i = S$, we find a set of locations that satisfy the constraint of all the session in S. The goal of our problem is to minimize the set C, thus minimize the number of servers. This is an instance of the set cover problem. Although it is $\mathcal{NP}$-complete there are good and simple approximations that are practical in the context of networks. Johnson [11] showed a $1 + \ln n$ approximation to the general problem. In the next section, we describe some heuristic algorithms for our problem formulation, and show the simulation results.

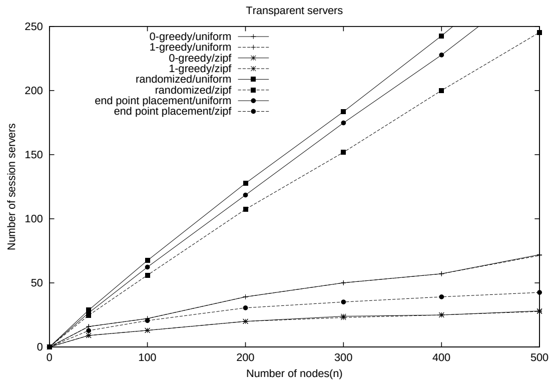

Figure 4. Number of session servers as a function of the network and session size for the greedy, the end-point placement and the randomized algorithms

5.1. Simulation results

We generated the networks as described in Section 4.1, and the routes were selected to be shortest paths. We simulated the following algorithms

- **Random:** A server is placed randomly in the network nodes that are part of some session route. Once a session is covered, its route is removed from the session pool. Repeat the above until every session is covered.
- **End-point Placement (*EPP*):** Find the session end point which is most frequently used and place the first server at that node. Remove all the sessions that were covered, and repeat the above.
- **Greedy [11]:** Select the location that covers most sessions, the algorithm appears in [20].
- **1-Greedy [13]:** First pick the best two locations, and then at each additional step check all the possibility to remove one server and add two new.

Figure 4 shows how the number of servers found by the algorithms varies as a function of the network size. Unlike in the overlay network case, here the difference between *Greedy* and *1-Greedy* are negligible to non-existent. For both session selection processes, the number of required servers grows sub-linearly with the number of nodes. *Random* shows a very poor performance and requires about half

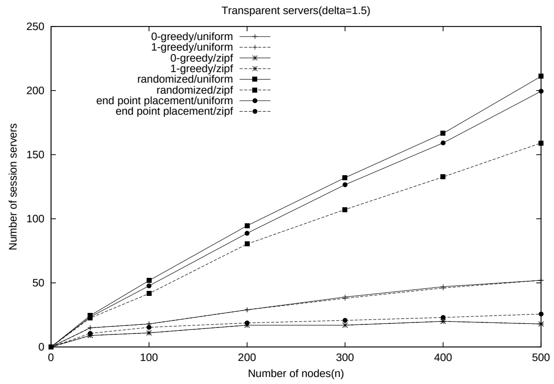

Figure 5. Number of servers found with the session path length relaxed by the factor of $\delta = 1.5$ to the minimum

of the nodes to be servers. Here too, the Zipf distribution requires less servers, about half for *Greedy*. For the Zipf distribution, *EPP* performs very close to *Greedy* and much better than *Random* while for the uniform distribution *EPP* is only slightly better than *Random*.

We also simulated the session cover problem for per-session constraint using the same service model with a stretch factor constraint on session hop distance. Formally, the constraint can be stated as

$$cost(s_i) \leq \delta \times d(v_{i_1}, v_{i_2})$$

where δ is the stretch factor. Note, that for this constraint the transparent location problem is a special case where $\delta = 1$. We set δ to 1.5 for our simulation and the result is shown in Figure 5. Figure 4 and Figure 5 shows that we need less servers for $\delta = 1.5$ since the constraint is relaxed. More simulation results are presented in [6].

6. Concluding Remarks

We note that the processing delay at the servers was not considered in our problem formulation because it is constant and thus does not affect the location problems. However, in some cases we may need to take this into account, e.g., when we define a stretch factor on the delay, the processing delay must be added to the calculation of $cost$.

Another issue we did not touch in this paper is the load on the servers. For example, one can add the constraint $\sum_{i \in S} x_{ij} < M$ to the optimization problem of Eq. (3) to limit the maximum load of any server by M.

References

[1] R. Albert and A.-L. Barabási. Topology of evolving networks: local events and universality. *Physical Review Letters*, 85(24):5234–5237, 11 Dec. 2000.

[2] R. Albert, H. Jeong, and A.-L. Barabási. Diameter of the world wide web. *Nature*, 401:130–131, 1999.

[3] S. Arora, P. Raghavan, and S. Rao. Approximations schemes for euclidean k-medians and related problems. In *STOC'98*, pages 106–113, Dallas, TX, USA, 1998.

[4] L. Breslau, P. Cao, L. Fan, G. Philips, and S.Shenker. Web caching and zipf-like distributions: evidence and implications. In *INFOCOM'99*, New York, NY, USA, Mar. 1999.

[5] Y. Chen, R. H. Katz, and J. D. Kubatowicz. Dynamic replica placement for scalable content delivery. In *First International Workshop on Peer-to-Peer Systems (IPTPS 2002)*, Mar. 2002.

[6] S. Choi and Y. Shavitt. Placing servers for session-oriented services. Technical Report WUCS-01-41, Washington University at St. Louis, Dept. of Computer Science, 2001.

[7] F. Chudak. Improved approximation algorithms for uncapacitated facility location. In *IPCO: 6th Integer Programming and Combinatorial Optimization Conference*, 1998.

[8] I. Cidon, S. Kutten, and R. Soffer. Optimal allocation of electronic content. *Computer Networks*, 40(2):205–218, Oct. 2002.

[9] E. Cronin, S. Jamin, C. Jin, A. Kurc, D. Raz, and Y. Shavitt. Constrained mirror placement on the Internet. *IEEE Journal on Selected Areas in Communications*, 20(7):1369–1382, Sept. 2002.

[10] D. Hochbaum. Approximation algorithms for combinatorial problems. *Mathematical Programming*, 22:148–162, 1982.

[11] D. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.

[12] J. Kangasharju, J. Roberts, and K. W. Ross. Object replication strategies in content distribution networks. In *The 6th International Web Caching Workshop and Content Delivery Workshop*, Boston, MA, USA, June 2001.

[13] P. Krishnan, D. Raz, and Y. Shavitt. The cache location problem. *IEEE/ACM Transactions on Networking*, 8(5):568–582, Oct. 2000.

[14] B. Li, M. J. Golin, G. F. Italiano, X. Deng, and K. Sohraby. On the optimal placement of web proxies in the internet. In *IEEE INFOCOM'99*, pages 1282 – 1290, Mar. 1999.

[15] J.-H. Lin and J. S. Vitter. Approximation algorithms for geometric median problems. *Information processing Letters*, 44:245–249, 1992.

[16] J.-H. Lin and J. S. Vitter. ϵ-approximations with minimum packing constraint violation. In *STOC'92*, Victoria, B.C., Canada, 1992.

[17] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The end-to-end effects of internet path selection. In *ACM SIGCOMM'99*, Boston, MA, USA, Aug./Sept. 1999.

[18] J. M. Smith, K. L. Calvert, S. L. Murphy, H. K. Orman, and L. L. Peterson. Activating networks: A progress report. *IEEE Computer*, 32(4):32–41, Apr. 1999.

[19] B. C. Tansel, R. L. Francis, and T. J. Lowe. Location on networks: a survey. part I: The p-center and p-median problems. *Management Science*, 29(4):482–497, Apr. 1983.

[20] V. Vazirani. *Approximation Methods*. Springer-Verlag, 2001.