Placing Servers for Session-Oriented Services

Sumi Choi[1] and Yuval Shavitt[2]

Abstract— The provisioning of dynamic forms of services is becoming the main stream of today's network. In this paper, we focus on services assisted by network servers and different forms of associated sessions. We identify two types of services: transparent, where the session is unaware of the server location, and configurable, where the sessions need to be configured to use their closest server. For both types we formalize the problem of optimally placing network servers and introduce approximated solutions. We present simulation result of approximations and heuristics. We also solve the location problem optimally for a special topology. We show, through a series of examples, that our approaches can be applied to a variety of different services.

I. Introduction

Active networks bring the ability to place services anywhere in the network. This will enable a model in which companies sell the usage of the service instead of the software that perform the service itself. This frees users from managing the software (installation, regular updates, etc.) and purchasing an expensive one when the service needs are limited, and enables them to evaluate various competitors in a short time period.

Another advantage of the service-in-the-network model is the ability of Internet service providers (ISPs) to deploy transparent services in their networks. Such services may include caching, authentication, security related snooping, DOSA alarms, etc.

While active networks give us the freedom to deploy services anywhere in the network, we would like to limit the number of such service centers, due to maintenance costs and the overhead associated with each deployed service (extra filters in the routers, extra delay for filtered packets, etc.). Thus, one would like to place servers wisely, either to make the maximum gain out of the number of servers it can afford to deploy, or to deploy the minimal number of servers needed to achieve certain level of service. This paper is the first, we are aware of, that looks and this problem.

The interaction in the Internet is done in sessions between end points, and thus, it is natural to associate service with sessions. A service performed on a session should bring some quantifiable gain, and its usage entails some cost. The same service may have different gain and cost according to the deployment economic model. For example, if an ISP deploys a caching service, the gain can be either the saving in traffic inside its network, or the reduction of delay for the clients. The optimal placement of the service need not be the same for these two optimization criteria, though the service performed is the same.

One can classify session-oriented services according to several criteria, which influence the mathematical modeling of the optimization problem. In the following we discuss: symmetry, the number of interacting parties, and transparency.

In some sessions, the interaction of the parties with the server is symmetric, while in others it is asymmetric. 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 compression 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

[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.

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 obtain 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.

A. 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 overlayed on top of the underlying network and configure their application sessions with associated 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 that applications try to optimize.

In the example of the conference center where the time constraint is important, one can see the direct relationship between the location of the servers and the latency of the data stream because data coming from sending end-points has to reach the server before delivered to the receivers. In other cases, like in the content adaptation and the video compression, the service changes the bandwidth consumption of the sessions, and thus, the location of the servers determines the total bandwidth consumption, which becomes the minimization criterion. 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 of the optimal server placement in the overlay service network 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 overall 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 II. 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 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.

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 associated with each individual session. 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 VI.

B. Applicability

The algorithms presented in this paper works on gathered statistics of service traffic, and should result in good placements based on the time of the day, or the day of the week. Active networks enable the easy migration of servers according to the usage pattern, and thus can improve the overall performance.

We believe usage statistics is quite stable and back this by the study performed by Krishnan et al. [1] 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 high between the weekdays. As a result, the cost of placing servers (caches in [1]) based on the overall statistics proved effective and resulted in a penalty of 1-10% (compared to the optimal daily placement) for most of the weekdays.

While we are not aware of similar statistics for other services, it is not inconceivable that it will follow the same pattern: low client return rate but fairly constant service demands from sites or regions. For example, large corporations have different cultures: some encourage the use of video conferences, some require data translation/conversion due to remote sites with different language/equipment, etc. The same argument holds for regions of the world where differences in culture create different demand for services and difference in the times service is required.

C. Organization

In the next section, the background of this work is detailed related to theoretical problems as well as to practical service networks. Then, we illustrate the network model and the metric formation in section III, proceed onto the overlay service networks along with the k-median problem in section IV and section V, and also describe the transparent service networks and the session cover problem in section VI. In the last section we raise a few issue for future research and summarize the paper contribution.

II. Related Works

The problem of placing servers for a group of single clients has several well-known variants that were all studied extensively. One is the facility location problem, which is an optimization problem in a set of n points with an associated cost for opening a facility (a server in our case). The goal is to find a set of locations among the n points to place facilities so as to minimize the sum of the distances from each of the n points to the nearest facility location and the cost of opening the facilities. The k-median problem is another related problem similar to the facility location problem, but but here the number of locations is given, k, and we minimize only the sum of the distances between each of the n points and the nearest facility location. The k center problem resembles the k-median problem, only the optimization criteria is to minimize the maximum distance between any of the n points and the nearest server.

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 in the context of network applications and discuss the corresponding server placement problems.

Another optimization problem which we find useful for transparent servers is the set cover problem. In this problem, we are given sets of elements and a separate group of elements, and we are required to find a minimal subset such that every element in the group appears at least once in the subset.

Although all the above problems are intractable, 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 [2]), 3.16 [3], 2.41 [4], and more recently 1.74 [5]. 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 [6] approximated the cost within a factor of $(1+\epsilon)$ of optimal by relaxing the restriction of the number of servers to be up to $(1+\frac{1}{\epsilon})(\ln n+1)k$. Arora, Raghavan, and Rao [7] 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, see Fig. 1). Suppose we have two sessions (depicted as arcs in Fig. 1): 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 [1], [8], but when multiple sources are considered only heuristics were suggested [1], [9]. As we show in section IV-A, 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 [1], [8].

Choi et al. [10] study a configuration problem for

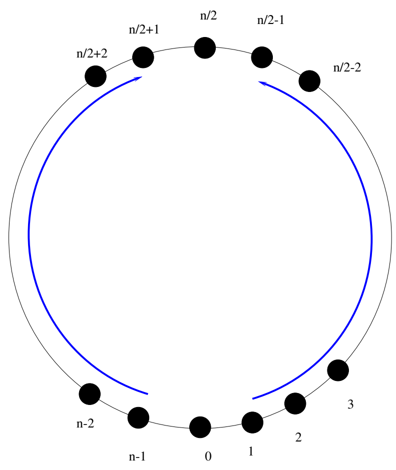


Fig. 1. An example for the difference between the session server location problem and the k-median and k-center problems.

a session through a series of servers that are already placed in the network. In their problem an optimal route and servers are in search for a session given a network and the locations of the servers. This study was done in the context of active network, a paradigm that enables the fast deployment of services inside the network. Thus, active network research, in general, can benefit from the results of our work where the servers need to be placed optimally.

III. 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 $v_i \in V$. 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 $s_i = (v_{i_1}, ..., v_{i_n})$ 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, that can be used, 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 Equation (1) as

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

This is the case of asymmetric cost function to the server. Note that the same application is symmetric for one criterion and asymmetric for another.

While the costs associated with applications allow one to determine better locations for placing a server for each session, the transparent servers add other constraints, a route for each session. So, for a unicast session $s_i = (v_{i_1}, v_{i_2})$, a route is given as $r_i = \{v_{i_1} = u_{i_1}, u_{i_2}, \ldots u_{i_{l(i)}} = v_{i_2}\}$. The server for the session is to be placed on the route.

IV. 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 budget for $k(< n)$ servers and 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 serving 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:

$$\begin{aligned}
&\text{minimize} && \sum_{s_i\in S}\sum_{j\in V} c_{ij}x_{ij} && (3)\\
&\text{subject to} && && (4)\\
\forall i\in S, &\sum_{j\in V} x_{ij} &=& \ 1 && (5)\\
&\sum_{j\in V} y_j &\le& \ s && (6)\\
\forall i\in S, \forall j\in V\ & x_{ij} &\le& \ y_j && (7)\\
\forall i\in S, \forall j\in V\ & x_{ij},\ y_j &\in& \ \{0,1\} && (8)
\end{aligned}$$

Solving this program is $\mathcal{NP}$-hard thus it is usually done by relaxing condition (8) 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 [6] which finds locations at the cost not more than $(1+\epsilon)$ of the optimal cost, but it might need a factor of 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 [7].

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. [11] 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. If one wishes to assume cost symmetry and that cost obey the triangle inequality, better approximations exist [12], [7].

Also, it is important to note that our formulation above holds for all the other non-transparent cases discussed in the introductory part of the paper. This includes asymmetric costs, e.g., different cost for end-points such as transmitter and receiver in case of bandwidth optimization with a compression server. (This is not to be confused with the asymmetric link cost mentioned in the previous paragraph.) For the two server case, when we need to place two types of servers such as an encoder and a decoder, we can use an indicator variable for each server pair and each session and obtain the same IP structure.

A. A simulation study

We generated networks based on the newly discovered power-log law [13]. Our generator is based on the algorithm suggested by Albert and Barabasi [14]. In all the generated network we picked the parameter to be $m_0=4, m=3, p=0.1, q=0$. For each instance we generated $10n$ sessions that are randomly generated according to two models, 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 was found to reflect better the distribution of service provider, say a web server, in the network [15]. Each point in our simulations represents 10 random session distribution on three different networks, total of 30 runs.

As the service model, we considered unicast sessions which requires one server. We set the session cost metric to be the hop count, i.e., for session $s_i=(v_{i_1},v_{i_2})$ the cost is given by

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

Here, $d(u,v)$ is the hop count. Clearly, the optimization goal is to minimize the total cost.

We simulated three algorithms:[1]

- Random where the k session servers are placed randomly in the network nodes with even probability.
- Greedy where we greedily select the location that reduces the total session cost the most. We iterate this process k times.
- 1-Greedy [1] where we first check the best locations for two servers together and then at each additional step we check all the possibility to remove one server and add two new.
- ϵ-Approx [6] first converts the 0-1 integer program formulation into a fractional linear program by allowing fractional assignments of servers, and computes a solution. Secondly, using the fractional server assignments and the value of ϵ, it bounds the distance

[1] We did not simulated the approximation algorithm presented above due to its high computational requirement.

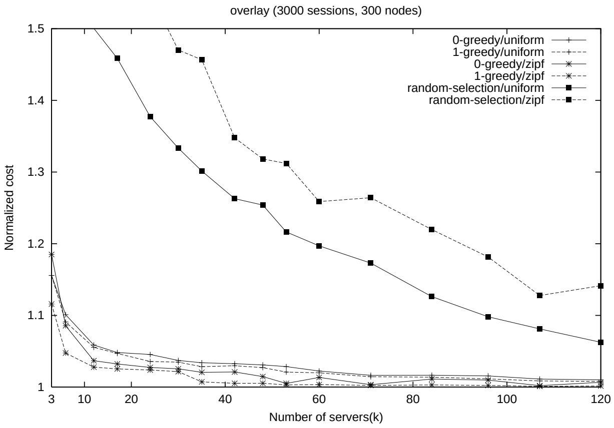


Fig. 2. The cost as a function of the number of servers, in a network of 300 nodes

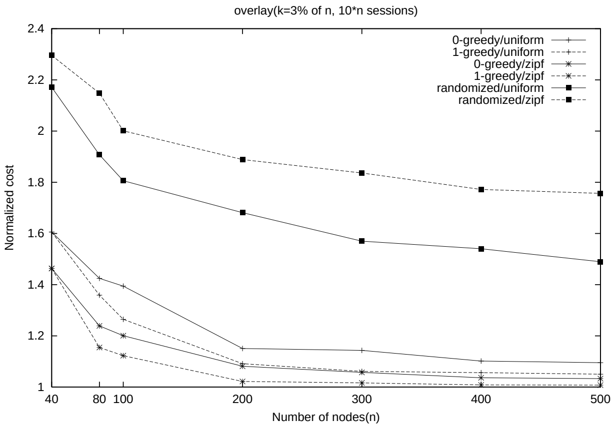


Fig. 4. The cost as a function of the network size, when the number of servers is fixed to 3% of the network size

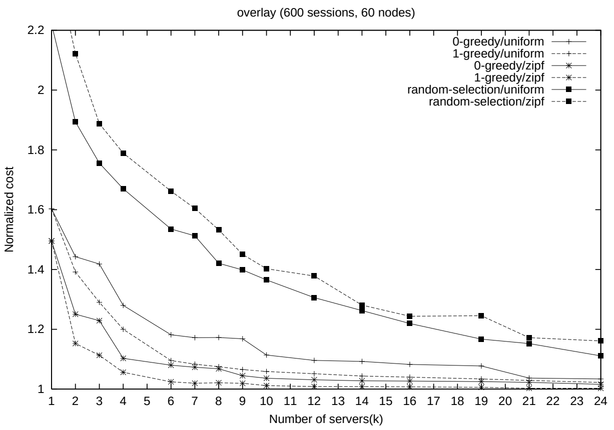


Fig. 3. The cost as a function of the number of servers, in a network of 60 nodes

between a session and a server, and forms a set cover problem where the cover of each session is defined as a set of servers satisfying the bound. Lastly, it applies the greedy set cover algorithm to the set cover problem to obtain a final set of servers.

In the first set of simulations, we measured the resulted cost as a function of the number of servers. Figures 2 and 3 present the cost for networks of 300 and 60 nodes, respectively. 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 1.0 represents no detours due to server placement. It is clear that Random is performing much worse than Greedy and is not a suitable candidate as a placement algorithm. The difference between Greedy and 1-Greedy is quite large for small networks but diminishes for larger networks. For the mirror placement problem Jamin et al. [9] report almost no difference between these two algorithms (and 2-Greedy, as well) maybe because they fail to look at smaller networks. All algorithms demonstrate a diminishing return curve: for the first servers we gain much in performance but as the number of servers increases this gain diminishes. The same phenomena was observed for placement of mirrors by Jamin 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 hold for all three algorithms. The reason for this is that when one end-point is selected using the Zipf distribution, by placing servers near the few nodes that participate in many sessions (say, popular web sites), we get a good cover of most of the sessions.

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. Since we found in figures 2 and 3 that the reasonable cost-performance trade-off is around 3-4% we selected κ=0.03. The results where we vary the number of nodes between 40 and 500 are depicted in Figure 4. 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 [16] with the number of nodes while we increase the number of servers linearly. It is also visible, that 1-Greedy is improving performance over Greedy quite significantly even for the 500 node network.

Finally we present comparison of the ϵ-Approx al-

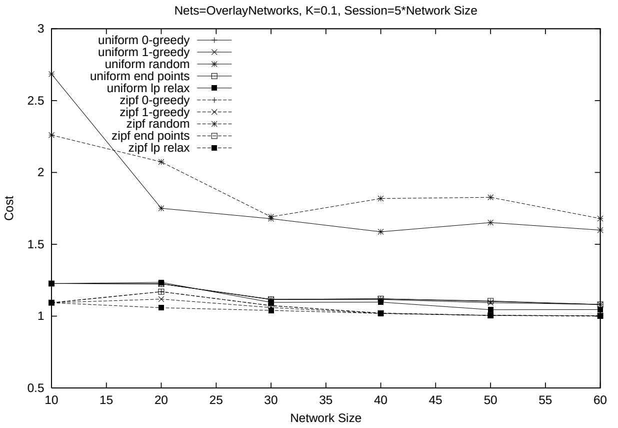


Fig. 5. The costs of overlay networks including results of the ϵ-Approx method with $\epsilon = 2$ (networks of size varied from 10 through 60, where the number of servers is fixed to 10% of the entire network nodes and the session end points follow the Zipf or the uniform distribution.)

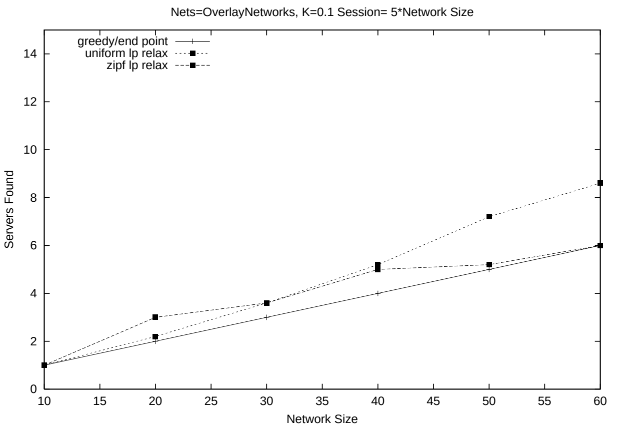


Fig. 6. The number of servers found by the ϵ-Approx method $\epsilon = 2$ compared to the fixed number of servers in other algorithms.

gorithm against the other methods. Due to the computational complexity of this algorithm we bring here (see Figure 5) simulations for rather small networks. As can be expected, the ϵ-Approx performs better from 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. Figure 6 compares the number of servers placed by the ϵ-Approx algorithm to the fixed bound $k = 0.1n$ in other algorithms (Note that the approximation may place up to $1.5 \ln(n)k$ servers).

The increase in the number of servers placed by the ϵ-Approx for the Zipf distribution for networks with more than 40 nodes becomes negligible but so is the gain compared to greedy in the performance. For the uniform distribution ϵ-Approx places more session-servers which might explain it performance advantage over greedy. Thus, we conclude that the greedy algorithms are more efficient and simpler to implement and seem to perform comparably to ϵ-Approx.

B. Extra constraints

We can incorporate in the IP formulation for placement problem described above additional constraints on the individual sessions. The two most practical are to limit the maximum increase in the cost of a session or to limit the maximum cost for any single session. 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 and assuming c_{ij} represent delay we can simply add a condition

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

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 VI

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

We presented an IP formulation for the server placement problem of overlay networks in the previous section. Although it is an intractable problem for general networks, for which we gave practical heuristics, we show in this section an optimal solution for the optimal 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 $O(n^2|S|)$:

- A session whose both end-points are inside the segment contribute to the cost the shortest detour between one of the session end-point and the segment edge. Thus, if we have $l_o \geq v_{i_1} > v_{i_2} \geq l_i$ the cost of session s_i is given by $2\min\{d(l_o, v_{i_1}), d(v_{i_2}, l_i)\}$.

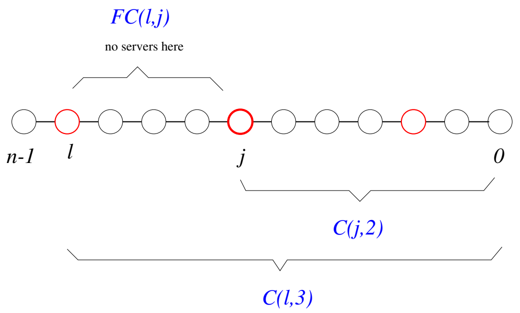


Fig. 7. The definition of $C(j,\ k')$.

- A session that has exactly one end-point inside the segment is not detoured since it is served by either l_o or l_i.
- A session that contains the segment (whose end-point are at both sides of the segment) is also served by either l_o or l_i, and thus is not detoured.
- A sessions whose end-points are on either side of the segment is not detoured through the segment since, in the worst case, it can be served by the closest of l_o and l_i. This session does not contribute to the cost of detours in this segments.

Thus, we only need to consider in the calculation of $FC(l_o,\ l_i)$ sessions whose both end points are inside the segment.

If $l_o = n-1$ we assume in the calculation that no server exists at node $n-1$, and similarly if $l_i = 0$ we assume no server at node 0. This is due to the observation that a solution with a server at node 0 cannot be better than a solution with this server moved to node 1, since sessions are, at least, one hop long, and thus a session that has an end-point at node 0 must pass through or terminate at node 1. The same reasoning works for the location of a server at node $n-1$.

We use dynamic programming to build an optimal solution to a segment from the optimal solution for shorter segments. 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. Figure 7 shows an example of such a segment. Note that $n-1 \geq l_o > 0$, and we do not need to consider the case where $k' > j$.

The minimal extra cost for detours is given by $C(n-1,\ k+1)$, and what we seek is the location of the k servers in this case. For the base case, it is easy to see that for all $n-1 \geq l \geq 2$, $C(l,2) = \min_{l \geq l' > 0} FC(l,l') + FC(l',0)$. Note, that in the calculation of $FC(l',0)$ we assume no server at node 0, the base case calculation of $C(l,2)$ indeed calculate the optimal position of two servers, such that one is inside the segment and one is at node l. For $k' > 2$, we have:

Claim 1 For $k' > 2$ and $l > k'$,

$$C(l,\ k') = \min_{k > l' \geq k'-1} \{C(l',\ k'-1) + FC(l,\ l')\} \quad (9)$$

Proof: The optimal placement of $k-1$ servers in the segment $[0, l-1]$ and a server at l, must have the closest server to l be placed somewhere in the segment $[0, l-1]$ and the rest of the servers optimally distributed between this server and node 0. Therefore the optimal cost is the minimum cost of these $l-k$ cases. ■

Theorem 2 $C(n-1, k+1)$ is the optimal cost of placing k servers in the line $[0, n-1]$.

Proof: We showed in claim 1 above that $C(l, k')$ position optimally k' servers is the segment $[0, l]$. The calculation of $FC(n-1,\ l)$, assume no server at location $n-1$ thus applying claim 1 for $C(n-1, k+1)$ will result is placing only k servers, hence the theorem holds. ■

The algorithm now is straight forward: first calculate $C(l, 2)$ for $n-1 \geq l \geq 2$. Next for each $l' > 2$ compute $C(l',\ k')$, for all $k \geq k' > 2$. The complexity of this algorithms is $O(n^2|S|)$ to compute the $FC()$ values, and $O(n^2 \cdot k)$ to compute $C(l',\ k')$. Thus overall, the algorithm complexity is $O(n^2|S|)$. [2]

This formulation can also be used to improve the dynamic programming of the cache location problem suggested by Krishnan et al. [1] from $O(n^3k)$ to $O(n^2k)$.

VI. 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 part of the input for this problem, and what we seek is a minimum number of servers satisfying the constraint. This approach for solving the server placement problem 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

[2] Note that if $|S| < k$ there is a trivial solution, thus the lack of a max term in the complexity

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 constraint, sc_i contains the sessions whose constraint can be accommodated by the node v_i. Note that the constraint can be determined in the same method as the *cost* metric was defined in Section III.

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 [17] showed a $1 + \ln n$ approximation to the general problem. The approximation is achieved by a greedy algorithm which starts with a maximal cover for a group of elements and successively adds remaining maximal covers for the uncovered elements. This approximation cannot be improved even if we know an upper bound on the set size [18].

A. Simulation results

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

- Random: Where at each iteration a server is placed randomly in the network nodes that are part of some session route, with even probability. Once a session is covered, its route is removed from the session pool, to avoid selection of a node due to a session which is already covered.
- End-point Placement (EPP): We find the session end point which is most frequently used and place the first server at that node. We remove all the session that were covered, and repeat iteratively.
- Greedy [17]: Where we greedily select the location that optimizes the increase of the cover, the algorithm appears in [19].
- 1-Greedy [1]: Where we first pick the best two locations and then at each additional step we check all the possibility to remove one server and add two new.

Figure 8 shows how the number of servers found by the algorithms vary as a function of the network size. Unlike with 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 servers required grows sub-linearly with the number of nodes. Random fails miserably and requires about half 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.

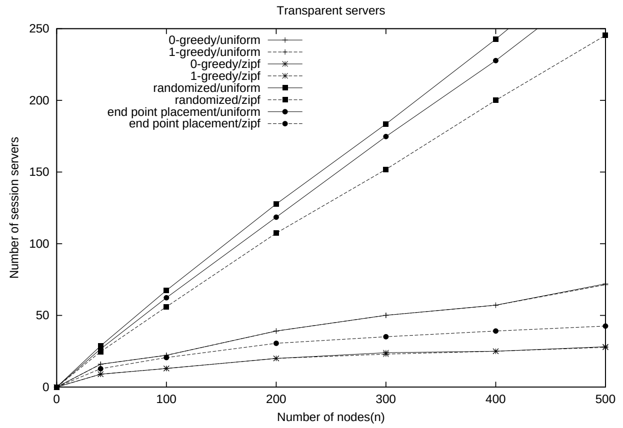


Fig. 8. The number of session servers as a function of the network and session size for the greedy, the end-point placement and the randomized algorithms

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

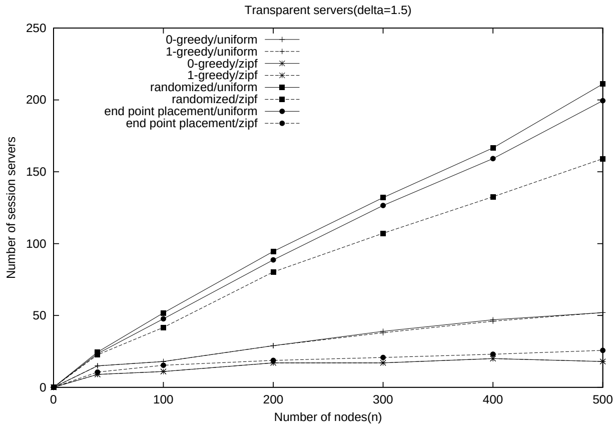
$$cost(s_i) \leq \delta \times d(v_{i_1}, v_{i_2})$$

where δ is a stretch factor. Note, that for this constraint the transparent location problem is a special case where $\delta = 1$. We set δ to be 1.5 for our simulation and the result is shown in Figure 9. Comparing the curves for Greedy and 1-Greedy in Figure 8 and Figure 9(a) we see that we need less servers for $\delta = 1.5$ since the constraint is relaxed. Figure 9(b) shows how the number of required servers changes with δ for a network of 300 nodes. The large decrease at $\delta = 1.5$ and $\delta = 2$ are since we used minimum hop as our cost and the average session length for the tested network was 2.

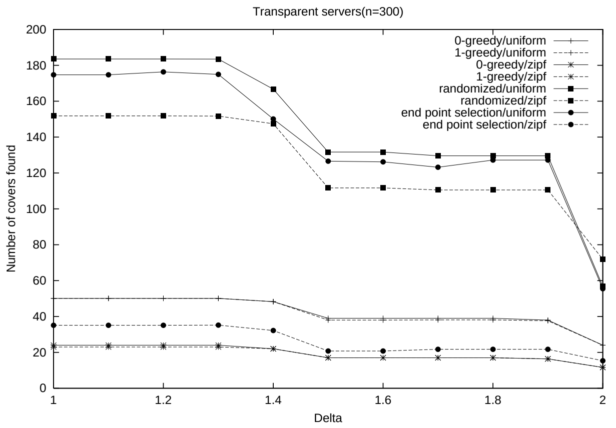
Figure 9(c) shows the ratio of the cover cost where $\delta = 1.5$ to the cost where $\delta = 1$. Interestingly, the additional overhead from relaxing the constraint of the server placement is minuscule, less than 1%, while the gain in reducing the number of servers is noticeable (about 20% in Fig. 9(b)).

VII. Concluding Remarks

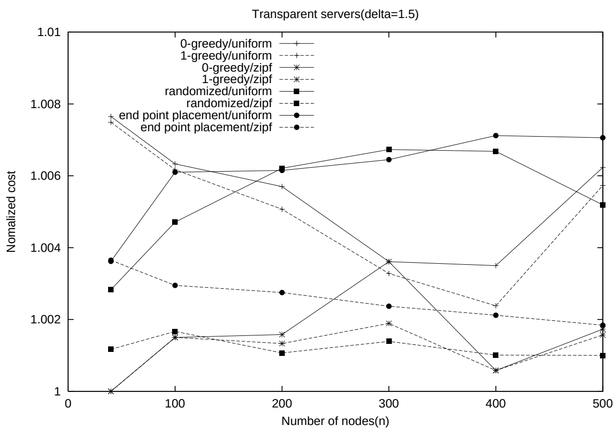
We note, that in our formulation we did not consider the processing delay at the servers. This was



(a) The number of servers found when the constraint on the session hop distance is relaxed by the factor of $\delta = 1.5$ to the minimum distance



(b) The number of servers found with different hop distance restrictions (δ values)



(c) The cost of the case of $\delta = 1.5$ compared to the transparent case where $\delta = 1$

Fig. 9. Result with constraints on the length of the sessions using δ

omitted since it is constant and thus does not effect the location problems. However, in some cases we need to take this into account, e.g., when we define a stretch factor on the delay, one needs to add the processing delay to the calculation of *cost*.

Another issue we did not touched 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.

To summarize, this work is the first to identify the importance of the location problems for session servers. We formulize two different classes of this problem and identify many extensions. We presented approximations, heuristics, and exact solutions to several variants of the problem, and simulated some of them.

Several architectures have been proposed for deploying and dynamically configuring services in active networks [20] and active services [21]. Placing services or servers is a critical issue particularly in such networks, which should benefit the most from our work. We also believe that there is more room for further research in this area.

References

[1] P. Krishnan, Dan Raz, and Yuval Shavitt, "The cache location problem," IEEE/ACM Transactions on Networking, vol. 8, no. 5, pp. 568–582, Oct. 2000.

[2] D.S. Hochbaum, "Approximation algorithms for combinatorial problems," Mathematical Programming, vol. 22, pp. 148–162, 1982.

[3] David B. Shmoys, Eva Tardos, and Karen Aardal, "Approximation algorithms for facility location problems (extended abstract)," in ACM Symposium on Theory of Computing, 1997, pp. 265–274.

[4] S. Guha and s. Khuller, "Greedy strikes back: Improved facility location algorithms," in ACM–SIAM Symposium on Discrete Algorithms (SODA), 1998.

[5] F.A. Chudak, "Improved approximation algorithms for uncapacitated facility location," in IPCO: 6th Integer Programming and Combinatorial Optimization Conference, 1998.

[6] Jyh-Han Lin and Jeffrey Scott Vitter, "ϵ-approximations with minimum packing constraint violation," in STOC'92, Victoria, B.C., Canada, 1992.

[7] Sanjeev Arora, Prabhakar Raghavan, and Satish Rao, "Approximations schemes for euclidean k-medians and related problems," in STOC'98, Dallas, TX, USA, 1998, pp. 106–113.

[8] Bo Li, Mordecai J. Golin, Giuseppe F. Italiano, Xin Deng, and Kazem Sohraby, "On the optimal placement of web proxies in the internet," in IEEE INFOCOM'99, Mar. 1999, pp. 1282 – 1290.

[9] S. Jamin, C. Jin, A. Kurc, D. Raz, and Y. Shavitt, "Constrained mirror placement on the internet," in IEEE Infocom 2001, Anchorage, AK, USA, Apr. 2001.

[10] Sumi Choi, Jonathan Turner, and Tilman Wolf, “Configuring sessions in programmable networks,” in IEEE INFOCOM’01, Anchorage, AK, USA, Apr. 2001.

[11] Stefan Savage, Andy Collins, Eric Hoffman, John Snell, and Thomas Anderson, “The end-to-end effects of internet path selection,” in ACM SIGCOMM’99, Boston, MA, USA, Aug./Sept. 1999.

[12] Jyh-Han Lin and Jeffrey Scott Vitter, “Approximation algorithms for geometric median problems,” Information processing Letters, vol. 44, pp. 245–249, 1992.

[13] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos, “On power-law relationships of the internet topology,” in ACM SIGCOMM 1999, Boston, MA, USA, Aug./Sept. 1999.

[14] Réka Albert and Albert-László Barabási, “Topology of evolving networks: local events and universality,” Physical Review Letters, vol. 85, no. 24, pp. 5234–5237, 11 Dec. 2000.

[15] 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.

[16] Réka Albert, Hawoong Jeong, and Albert-László Barabási, “Diameter of the world wide web,” Nature, vol. 401, pp. 130–131, 1999.

[17] D.S. Johnson, “Approximation algorithms for combinatorial problems,” Journal of Computer and System Sciences, vol. 9, pp. 256–278, 1974.

[18] Uriel Feige, “A threshold of $\ln n$ for approximating set cover,” Journal of the ACM, vol. 45, no. 4, pp. 634–652, July 1998.

[19] Vijay Vazirani, Approximation Methods, Springer-Verlag, 2001.

[20] Jonathan M. Smith, Kenneth L. Calvert, Sandra L. Murphy, Hilarie K. Orman, and Larry L. Peterson, “Activating networks: A progress report,” IEEE Computer, vol. 32, no. 4, pp. 32–41, Apr. 1999.

[21] E. Amir, S. McCanne, and R. Katz, “An active service framework and its application to real-time multimedia transcoding,” in ACM SIGCOMM’98, Sept. 1998.