Distributed Server Migration for Scalable Internet Service Deployment

GEORGIOS SMARAGDAKIS[†] NIKOLAOS LAOUTARIS[‡] KONSTANTINOS OIKONOMOU[¶]
georgios@net.t-labs.tu-berlin.de nikos@tid.es okon@ionio.gr

IOANNIS STAVRAKAKIS[§] AZER BESTAVROS[⋆]
ioannis@di.uoa.gr best@cs.bu.edu

***Abstract*— The effectiveness of service provisioning in large-scale networks is highly dependent on the number and location of service facilities deployed at various hosts. The classical, centralized approach to determining the latter would amount to formulating and solving the *uncapacitated k-median* (UKM) problem (if the requested number of facilities is fixed – k), or the *uncapacitated facility location* (UFL) problem (if the number of facilities is also to be optimized). Clearly, such centralized approaches require knowledge of global topological and demand information, and thus do not scale and are not practical for large networks. The key question posed and answered in this paper is the following: “How can we determine in a distributed and scalable manner the *number* and *location* of service facilities?”.**

In this paper, we develop a scalable and distributed approach that answers our key question through an iterative re-optimization of the location and the number of facilities within network neighborhoods. We propose an innovative approach to migrate, add, or remove servers within limited-scope network neighborhoods by utilizing only local information about the topology and demand. We show that even with limited information about the network topology and demand, within one or two hops, our distributed approach achieves performance, under various synthetic and real Internet topologies and workloads, that is comparable to that of optimal, centralized approaches requiring full topology and demand information. We also show that it is responsive to volatile demand. Our approach leverages recent advances in virtualization technology towards an automated placement of services on the Internet.

***Index Terms*— Service deployment, server migration, content delivery, facility location.**

I. INTRODUCTION

Imagine a large-scale bandwidth/processing-intensive service such as real-time distribution of software updates and patches [2], and content delivery that relies on distributed datacenters [3] or a cloud computing platform [4], [5], [6], to name some. Such services must cope with the typically *voluminous* and *bursty* demand — both in terms of overall load and geographical distribution of the sources of demand — due to flash crowd phenomena [7]. To deploy such services, decisions must be made on the time scale of minutes to a few hours based on: (1) the location, and optionally, (2) the number of nodes (or hosting infrastructures) used to deliver the service. Two well-known formulations of classic *Facility Location Theory* [8] can be used as starting points for addressing decisions (1) and (2), respectively. The *uncapacitated k-median (UKM)* problem prescribes the locations for instantiating a fixed number of service facilities, k, so as to minimize the distance between users and the closest facility capable of delivering the service. In the *uncapacitated facility location (UFL)* problem, the number of facilities is not fixed, but *jointly* derived along with the locations as part of a solution that minimizes the combined service hosting and access costs.

Limitations of existing approaches: Even though it provides a solid basis for analyzing the fundamental issues involved in the deployment of network services, facility location theory is not without its limitations. First and foremost, proposed solutions for UKM and UFL are centralized, so they require the gathering and the transmission of the entire topological and demand information to a central point, which is not possible (not to mention practical) for large networks. Second, such solutions are not adaptive in the sense that they do not allow for easy reconfiguration in response to local changes in the topology and the intensity of the demand for service. Moreover, over-reaction to local changes in demand using a centralized solution may lead to costly service deployment. To address these limitations we propose distributed versions of UKM and UFL, which we use as means of constructing an autonomic service deployment scheme.

A scalable approach to autonomic service deployment: We develop a scheme in which an initial set of service facilities are allowed to migrate adaptively to the best network locations, and optionally to increase/decrease in number so as to best service the current demand. Our scheme is based on developing distributed versions of the UKM problem (for the case in which the total number of facilities must remain fixed) and the UFL problem (when additional facilities can be acquired at a price or some of them be closed down). Both problems are combined under a common framework with the following characteristics. An existing facility gathers the topology of its local network neighborhood. The facility also monitors the

[†] Deutsche Telekom Laboratories and Technical University of Berlin, Germany.
[‡] Telefónica Research, Barcelona, Spain.
[¶] Dept of Informatics, Ionian University, Corfu, Greece.
[§] Dept of Informatics and Telecommunications, University of Athens, Greece.
[⋆] Computer Science dpt., Boston University, MA, USA.

G. Smaragdakis is supported in part by the EU projects BigFoot (FP7-ICT-317858) and CHANGE (FP7-ICT-257422), and by an IKY-DAAD award (54718944). N. Laoutaris is supported in part by the NANODATACENTERS program (FP7-ICT-223850) of the EU. K. Oikonomou was supported in part by the FP6 project Autonomic Network Architecture (ANA, IST-27489), which is funded by IST FIRE Program of the European Commission. I. Stavrakakis was supported in part by the FP7 project SOCIALNETS (IST, IST-217141), which is funded by IST FET Program of the European Commission. A. Bestavros is supported in part by a number of NSF Awards including CISE/CNS #1012798, CISE/CSR #0720604, ENG/EFRI #0735974, CISE/CNS #0524477, CISE/CNS #0520166, CNS/ITR #0205294, and CISE/EIA RI #0202067. Parts of this work appeared in the proceedings of the 26th IEEE INFOCOM Conference [1].

demand of the nodes that have it as closest facility. This is the demand that the facility has to satisfy. It keeps an exact representation of the demand of nodes in its local network neighborhood and an approximate representation for all the nodes that are outside its neighborhood but have the facility as the closest one. The observed local topology and demand information is then used to re-optimize the current location (and optionally the number of) facilities by solving the UKM (or the UFL) problem in the local network neighborhood.

Reducing the amount of topological information that needs to be gathered and processed centrally at any point (i.e., at facilities that re-optimize their positions) is a plus for scalability. On the other hand, reducing this harms the overall performance as compared to centralized solutions that consider the entire topological information. In this paper we examine this trade-off by developing a distributed facility location algorithm and by evaluating it under various topologies and workloads. Our results show that even with limited information about the network topology and demand, within one or two hops, our distributed approach achieves performance that is comparable to that of optimal, centralized approaches requiring full topology and demand information. Our results also show that our distributed solutions allows an autonomic service deployment to be responsive to volatile demand.

Addressing the distributed Software as a Service placement problem: Our approach leverages recent advances in virtualization technology and flexible billing models such as pay-as-you-go [4] as well as the availability of cloud resources on the Internet [9], [10], [11] towards a fully automated and scalable service deployment. In particular, it provides a distributed solution for the placement of replicas of software in the emerging field of Software as a Service (SaaS) that is currently limited, typically, to centralized hosting.

Outline: The remainder of this paper is structured as follows. Section II provides a brief background on facility location. Section III overviews the general architecture of our approach and provides the connection to the facility location theory. Section IV presents our distributed facility location approach to autonomic service deployment. Section V examines analytically issues of convergence and accuracy due to approximate representation of the demand of nodes outside the local network neighborhoods. Section VI evaluates the performance of our schemes on synthetic topologies. Section VII presents results on real-world (AS-level) topologies. Section VIII looks at the effects of volatile demand and addresses technical challenges for the deployment of our approach. Section IX presents previous related work. Section X concludes the paper with a summary of findings.

II. Background on Facility Location

Let $G = (V, E)$ represent a network defined by a node set $V = \{v_1, v_2, \ldots, v_n\}$ and an undirected edge set E. Let $d(v_i, v_j)$ denote the length of a shortest path between v_i and v_j, and $s(v_j)$ the (user) service demand originating from node v_j. Let $F \subseteq V$ denote a set of facility nodes – i.e., nodes on which the service is instantiated. If the number of available facilities k is given, then the specification of their exact locations amounts to solving the following uncapacitated k-median problem:

Definition 1: (UKM) Given a node set V with pair-wise distance function d and service demands $s(v_j)$, $\forall v_j \in V$, select up to k nodes to act as medians (facilities) so as to minimize the service cost $C(V, s, k)$:

$$C(V, s, k) = \sum_{\forall v_j \in V} s(v_j) d(v_j, m(v_j)), \qquad (1)$$

where $m(v_j) \in F$ is the median that is closer to v_j.

On the other hand, if instead of k, one is given the costs $f(v_j)$ for setting up a facility at node v_j, then the specification of the facility set F amounts to solving the following uncapacitated facility location problem:

Definition 2: (UFL) Given a node set V with pair-wise distance function d and service demands $s(v_j)$ and facility costs $f(v_j)$, $\forall v_j \in V$, select a set of nodes to act as facilities so as to minimize the joint cost $C(V, s, f)$ of acquiring the facilities and servicing the demand:

$$C(V, s, f) = \sum_{\forall v_j \in F} f(v_j) + \sum_{\forall v_j \in V} s(v_j) d(v_j, m(v_j)), \qquad (2)$$

where $m(v_j) \in F$ is the facility that is closer to v_j.

For general graphs, both UKM and UFL are NP-hard problems [12]. A variety of approximation algorithms have been developed under metric distance using a plethora of techniques, including rounding of linear programs [13], local search [14], [15], and primal-dual methods [16].

III. Description of the Architecture

Large scale software systems, e.g., software update systems such as Microsoft Windows Update [2] or applications such as Apple iCloud and Google Apps, rely more and more on an on-demand software delivery model, that is referred to as Software as a Service (SaaS). In SaaS, software and associated data are hosted in cloud infrastructures. Thanks to the elasticity of the cloud, supported by virtualization, it is possible to expand or shrink the software installation on-demand and minimize the overall cost [4]. Such software systems not only deliver Terabytes of data daily to millions of users, but also have to incorporate complex decision processes for customizing the delivered content to the peculiarities of different clients with respect to localization, previously-installed applications, compatibilities, and optional components, among others. For scalability issues and to improve the end-user experience, a number of replica of the software has to be installed in different locations on the Internet [17]. We refer to the complex process of configuring, placing, and delivering software as the *Software as a Service placement problem*. The nature of this problem goes beyond the dissemination of a single large file, where a peer-to-peer approach is an obvious solution [18]. Moreover, it is unlikely that software providers are willing to trust intermediaries with such processes. Rather, we believe that such applications are likely to rely on dedicated or virtual hosts, e.g., servers offered for lease through third-party overlay networks – *a la* Akamai or PlanetLab, or the newest breed of Cloud Computing platforms e.g., Amazon Web Services.

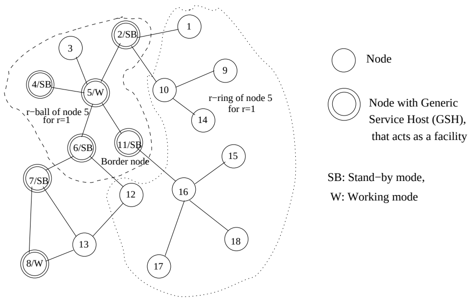

Fig. 1. A snapshot of the operation of a service that utilizes our proposed architecture to dynamically deploy servers in a distributed cloud environment.

We propose a *distributed* solution to address the Software as a Service placement problem. We believe that the use of our distributed facility location approach presents significant advantages in terms of optimizing the operational cost and efficiency of deploying such applications, and improve end-user experience [19]. In the remainder of this section, we present the general architecture of our approach and provide a mapping from the aforementioned software distribution service to our abstract UKM and UFL problems.

Our architecture relies on the observation that cloud resources are available today in many locations on the Internet [4], [9], [10] and that these resources can be reserved on-demand by utilizing recent advances of virtualization technology and open standards, such as OpenStack (http://www.openstack.org). In our architecture we consider the availability of a set of network hosts upon which specific functionalities may be installed and instantiated on demand that is also recommended by major network providers [11]. We use the term *Generic Service Host* (GSH) to refer to the software and hardware infrastructure necessary to host a service. For instance, a GSH could be a well-provisioned Linux server, a virtual machine (VM) slice similar to that used in PlanetLab or a set of resources in a Cloud Computing platform (e.g., an Amazon Machine Image (AMI) in the context Amazon Web Services).

A GSH may be in Working (`W`) or Stand-By (`SB`) mode. In `W` mode, the GSH constitutes a service facility that is able to respond to client requests for service, whereas in `SB` mode, the GSH does not offer the actual service, but is ready to switch to `W` if it is so directed. Switching to `W` might involve the transfer of executable and configuration files for the service from other GSHs or from the service provider. Thus, the set of facilities used to deliver a service is precisely the set of GSHs in `W` mode. By switching back and forth between W mode and `SB` mode, the *location* as well as the *number* of facilities used to deliver the service could be controlled in a distributed fashion. In particular, a GSH in `W` mode monitors the topology and the corresponding demand in its vicinity and is capable of re-optimizing the location of the facility. Third-party Autonomous Systems (AS) may host the GSHs of service providers, possibly for a fee. In particular, the hosting AS may charge the service provider for the resources it dedicates to the GSHs, including the software/hardware infrastructure supporting the GSHs as well as the bandwidth used to carry the traffic to/from GSHs in `W` mode. The implementation of the above-sketched scenaria requires each GSH to be able to construct its surrounding AS-level topology within r hops. This can be achieved through standard topology discovery protocols, e.g., CAIDA Skitter (http://www.caida.org/tools/measurement/skitter).

Communication: Each GSH has also to be aware of all the GSH that participate in the service and its status, `W` or `SB`. This is easily achieved by having a bootstrap server of the service where periodically all the GSHs report their status. The status of each GSH is also communicated to all the other GSHs through a broadcast message. Summaries about the network location and assignment of clients, and client demands have to be exchanged between neighboring GSHs. GSHs are neighbors if they are at most r network hops away, where r is a parameter of the system. Periodically, GSHs that belong to the same neighborhood exchange information about the current demand.

Decision rounds: Through a leader election protocol each working GSH becomes leader and makes a local decision, by solving a local UKM or UFL, on which neighboring GSH (including itself) should be in `W` mode. This decision is then communicated to the bootstrap server and to all the other GSHs via broadcasting. The details of the algorithm are presented in Section IV. As we will show in Section V, a stable set of GSHs is selected in no more than a logarithmic number of rounds to the size of the network times the ratio of the maximum to minimum distance of pair of nodes. We also show in Section V how we can bound the number of rounds before convergence. Our experiments, see Sections VI and VII, show that in practical settings this may require only a small number of rounds (iterations of our algorithm, presented in Section IV), typically between 5 and 20 for AS-level Internet graphs. Our experiments also show that the first rounds yields most of the cost reduction.

Synchronization: To avoid instabilities that can cause lack of convergence, we assume that all the GSHs are synchronized. This can be easily achieved by using the standard Network Time Protocol (NTP). Each GSH should also be aware of the demand of its clients and their location in the network. This can be easily achieved by maintaining and analyzing the connection logs. Synchronization of GSH is necessary but not sufficient condition. The sufficient condition is that, at each point of time, only one node is solving the distributed facility location problem (see Section IV-C).

Figure 1 illustrates a snapshot of the operation of the aforementioned software distribution when utilizing our proposed architecture. Nodes 2, 4, 5, 6, 7, 8, and 11 serve GSHs. Nodes 5 and 8 are in `W` mode and all the others in `SB` mode. Node 5 serves the demand of nodes 2, 3, 4, 6, and 11 that are within 1 hop away. It also serves the demand of nodes 1, 9, 10, and 14 that are closer to node 2 (a node with GSH), as well as the demand of nodes 15, 16, 17, and 18 that are closer to node 11 with GSH, and node 12 closer to node 6 (a node with GSH). Node 5 has to periodically solve the UKM or UFL problem to decide either to migrate the server to node 2, 4, 6, or 11 in order to minimize the service deployment and operational cost as well as to better serve the volatile demand. It does so by taking into account the total demand it serves, the source of the demand and the local view of the network topology. Node 5 can also well decide that more instances of the service are needed in the neighborhood, thus, it can change the mode of any of the facilities 2, 4, 6 or 11 from SB to W by solving the UFL problem. Clients of the service should be able to locate the facility closest to it, and it requires a GSH to be able to inform potential clients of the service regarding its W or SB mode. Both of these could be achieved through standard resource discovery mechanisms like DNS re-direction [20], [21] (appropriate for application-level implementations of our distributed facility location approach) with appropriate TTL values [22] or proximity-based anycast routing [23] (appropriate for network layer implementations). We show in Section VIII-C that the performance of our scheme degrades gracefully as re-direction becomes more imprecise.

IV. Distributed Facility Location Scheme with Local Information

In this section we develop distributed versions of UKM and UFL by utilizing a limited horizon approach in which GSHs have exact knowledge of the topology of their local topology within r hops, exact knowledge of the demand of each client or node in the local topology, and approximate knowledge of the aggregate demand from nodes in the r-ball outside this local topology. For the rest of the paper we are using the terms GSH and candidate facility interchangeably. If the GSH is in W mode we refer to the node that hosts the GSH as *open* facility, or simply facility.

A. Definitions

We start by formally defining the local network neighborhoods of a facility, that we will refer as r-balls.

Definition 3: (r-ball) An r-ball of a facility is a sub-graph that includes all the nodes that are reachable within r-hops and all physical edges that connect any of these nodes.

Naturally, the r-balls of two different facilities can overlap. For optimization reasons, we can join r-balls that overlap to create a larger graph that we refer as r-shape. If more than two r-balls overlap then there should be at least one common facility for all r-balls.

Definition 4: (r-shape) An r-shape is the union of two or more r-balls that overlap and have at least one common facility.

As mentioned before, exact information about the topology and demand is maintained for the nodes in the r-ball. Approximate information is maintained for the demand of nodes outside the r-ball that is served by the facility. To achieve the demand of those nodes outside the r-balls, their demand is attached to the particular node that is closer to them and is within the r-ball of the facility. We refer to them as border nodes and all the nodes that are potentially border nodes constitute the skin of the r-ball.

Definition 5: (r-border node and r-skin) An r-border node is any node that is r hops away from a facility. The set of border nodes constitute the r-skin of the r-ball of a facility.

Definition 6: (r-ring) The r-ring of a facility is the set of nodes outside the r-ball that it serves. The demand of each node in the ring is attached to its closest r-border node.

In Figure 1 we provide a setting where we annotate nodes according to the introduced definitions. The sub-graph included within the dashed line is the r-ball of node 5 for r=1. The sub-graph included within the dotted line annotates the ring for node 5. Nodes 2, 3, 4, 6, and 11 are all border nodes for 5 for r=1 and they constitute the skin of the 1-ball. Nodes 2, 4, 6 and 11 are candidate facilities (GSHs in SB mode). When $r = 2$, the 2-balls of node 5 and 8 can be merged to shape a new 2-shape because facilities 6 and 7 are common.

B. Notations

Our distributed approach will be based on an iterative method in which the location and the number of facilities (in the case of UFL only) may change between iterations.

We make use of the following notations to explain our distributed algorithm. Most of the notations are superscripted by m, the ordinal number of the current iteration. Let $H \subseteq V$ denote the set of candidate facility nodes and $F^{(m)} \subseteq H$ the set of facility nodes at the mth iteration. Let $V_i^{(m)}$ denote the r-*ball* of facility node v_i. Let $U_i^{(m)}$ denote the *ring* of facility node v_i. The *domain* $W_i^{(m)} = V_i^{(m)} \bigcup U_i^{(m)}$ of a facility node consists of its r-ball and the surrounding ring. From the previous definitions it is easy to see that $V = V^{(m)} \bigcup U^{(m)}$, where $V^{(m)} = \bigcup_{v_i \in F^{(m)}} V_i^{(m)}$ and $U^{(m)} = \bigcup_{v_i \in F^{(m)}} U_i^{(m)}$.

C. The Distributed Algorithm

Our distributed algorithm starts with an arbitrary initial batch of facilities, which are then refined iteratively through relocation and duplication until a (locally) optimal solution is reached. In a nutshell, the algorithm starts with the initial set of facilities, and iteratively runs local UKM or UFL by re-evaluating the r-balls of the facilities one by one. Once a facility is evaluated, it is marked as processed. When all the facilities are processed, the algorithm examines if the set of facilities remains the same. If it is not, it continues evaluating all the r-balls in the same manner until the set of facilities is the same with the one in the last iteration. Formally, it includes the following steps:

Initialization: Pick randomly an initial set $F^{(0)} \subseteq H$ of $k_0 = |F^{(0)}|$ to act as facilities. Let $\mathcal{F} = F^{(0)}$ denote a temporary variable containing the "unprocessed" facilities from the current batch. Also, let $\mathcal{F}^- = F^{(0)}$ denote a variable containing this current batch of facilities.

Iteration m**:** Pick an unprocessed facility $v_i \in \mathcal{F} = F^{(m)}$ and process it by executing the following steps:

1) Construct the topology of its surrounding r-ball by using an appropriate neighborhood discovery protocol (see [24] for such an example).

2) Test whether its r-ball can be merged with the r-balls of other nearby facilities (see Section IV-A). Let $J \subseteq F^{(m)}$ denote a set composed of v_i and the facilities that can be merged with it. J induces an r-*shape* $G_J = (V_J, E_J)$, i.e., the sub-graph of G composed of the facilities of J, their neighbors up to distance r, and the edges between them. We can place constraints on the maximal size of r-shapes to guarantee that it is always much smaller than n, i.e., we do not want to end up

solving the centralized problem. In our algorithm, we restrict that no r-shapes contains more half of the total nodes.

3) Re-optimize the r-shape G_J. If the original problem is UKM, solve the $|J|$-median within the r-shape by considering all the candidate facilities in the r-shape. This can produce new locations for the $|J|$ facilities. If the original problem is UFL, solve the UFL within the r-shape by considering all the candidate facilities in the r-shape. This can produce new locations as well as change the number of facilities, i.e., make it smaller or larger than $|J|$. In both cases the local re-optimization is conducted by using one of the UKM or UFL solutions (the details regarding the optimization of r-shapes are given in Section IV-D). Numerical results can be obtained by using Integer Linear Programming (ILP) formulations [13] and local search heuristics [15] for solving UKM and UFL within r-shapes. Since both perform very closely in all our experiments [25], [26], we don't discriminate between the two.

4) Remove processed facilities, both the original v_i and the ones merged with it, from the set of unprocessed facilities of the latest batch, i.e., set $\mathcal{F} = \mathcal{F} \setminus (J \bigcap \mathcal{F}^-)$. Also update $F^{(m)}$ with the new locations of the facilities after the re-optimization.

5) Test for convergence. If $\mathcal{F} \neq \emptyset$ then some facilities from the latest batch have not yet been processed, so perform another iteration. Otherwise, if the configuration of facilities changed with respect to the initial one for the latest batch, i.e., $F^{(m)} \neq \mathcal{F}^-$, then form a new batch by setting $\mathcal{F} = F^{(m)}$ and $\mathcal{F}^- = F^{(m)}$, and perform another iteration. Else (if $F^{(m)} = \mathcal{F}^-$), then no beneficial relocation or elimination is possible, so terminate by returning the (locally) optimal solution $F^{(m)}$.

Our distributed algorithm guarantees that all the users are connected to one facility and the demand of each user is satisfied. In this paper we focus on the UKM and UFL with unsplittable demands (high DNS TTL values or anycast). With our approach it is possible to assign a user and its demand to more that one facility if we considered the related facility location problems with splittable demands [14]. It is expected that fractional assignment of demand can provide even better results as the load in servers can be better balanced (low DNS TTL values). Our solution is general enough to address also the case of capacitated facility location (CFL) [13] where each facility can satisfy a maximum number of users or demand.

D. *Optimizing r-shapes*

As discussed in Section II, the input of a UKM problem is defined completely by a tuple $\langle V, s, k \rangle$, containing the topology, the demand, and the number of allowed medians. A UFL problem is defined by a tuple $\langle V, s, f \rangle$, similar to the previous one, but with facility creation costs instead of a fixed constraint on the number of allowed facilities. For the optimization of an r-shape, we set $V = V_J$, and $k = |J|$ (for the case of UKM) or $f = \{f(v_j) : \forall v_j \in V_J\}$ (for the case of UFL).

Regarding service demand, a straightforward approach would be to set $s = \{s(v_j) : \forall v_j \in V_J\}$, i.e., retain in the re-optimization of the r-shape the original demand of the nodes of the r-shape. Such an approach would, nonetheless, be inaccurate since the facilities within an r-shape serve the demand of the nodes of the r-shape, as well as those in the corresponding ring of the r-shape. Since there are typically a few facilities, each one has to serve a potentially large number of nodes, e.g., of order $O(n)$), and thus the rings are typically much larger than the corresponding r-shapes. Note that r is intentionally kept small to limit the size of the individual re-optimizations. Re-optimizing the arrangement of facilities within an r-shape without considering the demand that flows-in from the ring would, therefore, amounts to disregarding too much information (as compared to the information considered by a centralized solution). Including the nodes of the ring into the optimization is, of course, not an option, as the ring can be arbitrarily large ($O(n)$) and, therefore, considering its topology would contradict our prime objective — to perform facility location in a scalable, distributed manner.

Our solution for this issue is to consider the demand of the ring implicitly by mapping it into the local demand of the nodes that constitute the r-skin. This intermediate approach bridges the gap between absolute disregard for the r-ring, and full consideration of its exact topology. The details of the mapping are as follows. Let v_i denote a facility inside an r-shape G_J. Let $v_j \in U$ denote a node in the corresponding ring, having the property that v_i is v_j's closest facility. Let v_k denote a node on the r-skin of G_J, having the property that v_k is included in a shortest path from v_j to v_i. To take into consideration the demand from v_j while optimizing the r-shape G_J, we map that demand onto the demand of v_k, i.e., we set: $s(v_k) = s(v_k) + s(v_j)$.

Note, the assignment of nodes demand is done after each re-optimization. We do require synchronization of individual facilities to avoid parallel re-optimizations. Facility time synchronization is easy to achieve using the Network Time Protocol (NTP). Before optimizing an r-shape, a lock message with an identifier is sent by the facility of the r-shape to all other facilities. This indicates that for this round this node is the candidate leader. When the re-optimization is finished an unlock message is sent by (one of) the new facility with the same identifier to all the other facilities. No further information exchange is required between nearby facilities because each facility can monitor the demand it serves. Our algorithm is robust to mapping error, non-stationary demand, and imperfect redirection of users to facilities as we will elaborate in Sections V-B and VIII.

V. A More Detailed Examination of Distributed Facility Location

The previous section has provided an overview of the basic characteristics of the proposed distributed facility location approach. This section sheds light to some important albeit more complex properties of the proposed solution.

A. *Convergence of the Iterative Method*

We start with the issue of convergence. First we show that the iterative algorithm of Section IV-C converges in a finite number of iterations. Then we show how to control the

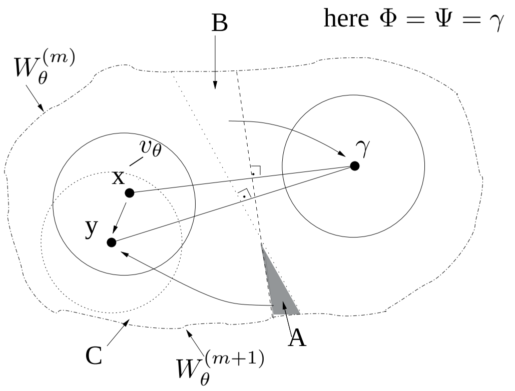

Fig. 2. Analysis of the migration of a facility from X to Y.

convergence speed so as to adapt it to the requirements of practical systems.

Proposition 1: The iterative local search approach for distributed facility location converges in a finite number of iterations.

Proof: Since the solution space is finite, it suffices to show that there cannot be loops, i.e., repeated visits to the same configuration of facilities. A sufficient condition for this is that the cost (either Equation (1) or (2) depending on whether we are considering distributed UKM or UFL) be monotonically decreasing between successive iterations, i.e., $c^{(m)} \geq c^{(m+1)}$. Below, we show that this is the case for the UKM applied to r-shapes with a single facility. The cases of UKM applied to r-shapes with multiple facilities, and of UFL follow from straightforward generalizations of the same proof.

Suppose that during iteration $m+1$ facility v_θ is processed and that between iteration m and $m+1$, v_θ is located at node x, whereas after iteration $m+1$, v_θ is located at node y. If $x \equiv y$, then $c^{(m)} = c^{(m+1)}$. For the case that $x \neq y$, we need to prove that $c^{(m)} > c^{(m+1)}$.

For the case in which $W_\theta^{(m)} \equiv W_\theta^{(m+1)}$, it is easy to show that $c^{(m)} > c^{(m+1)}$. Indeed, since the facility moves from x to y it must have been that this reduces the cost of the domain of v_θ, i.e., $c(W_\theta^{(m)}) > c(W_\theta^{(m+1)})$, which implies $c^{(m)} > c^{(m+1)}$, since no other domain is affected.

The case in which $W_\theta^{(m)} \neq W_\theta^{(m+1)}$ is somewhat more involved. It implies that there exist sets of nodes A, B: $A \cup B \neq \emptyset$, $A = \{z \in V : z \notin W_\theta^{(m)}, z \in W_\theta^{(m+1)}\}$ and $B = \{z \in V : z \in W_\theta^{(m)}, z \notin W_\theta^{(m+1)}\}$. A is actually the set of nodes that were not served by facility v_θ before the $m+1$ iteration and are served after the $m+1$ iteration. Similarly, B is the set of nodes that were served by facility v_θ before the $m+1$ iteration and are not served after the $m+1$ iteration. Let $C = \{z \in V : z \in W_\theta^{(m)}, z \in W_\theta^{(m+1)}\}$ be the set of nodes that remained in the domain of v_θ after its move from x to y (Figure 2 depicts the aforementioned sets). Since $W_\theta^{(m)} = B \cup C$ (B, C disjoint) and the re-optimization of $W_\theta^{(m)}$ moved the facility v_θ from x to y, it must be that:

$$c(B,x) + c(C,x) > c(B,y) + c(C,y) \tag{3}$$

where $c(B,x)$ denotes the cost of servicing the nodes of B from x (similar definitions for $c(C,x)$, $c(C,y)$).

Let Φ denote the set of facilities that used to service the nodes of A before they entered the domain of v_θ at $m+1$. Similarly, let Ψ denote the set of facilities that get to service the nodes of B after they leave the domain of v_θ at $m+1$. From the previous definitions it follows that the necessary conditions for migration are:

$$c(A,y) < c(A,\Phi) \tag{4}$$

$$c(B,y) > c(B,\Psi) \tag{5}$$

Using Equation (5) in Equation (3) we obtain:

$$c(B,x) + c(C,x) > c(B,\Psi) + c(C,y) \tag{6}$$

Applying Equations (6) and (4) to the difference $c^{(m)} - c^{(m+1)}$, we can now show the following:

$$c^{(m)} - c^{(m+1)} =$$
$$\Big(c(B,x) + c(C,x) + c(A,\Phi)\Big) - \Big(c(A,y) + c(C,y) + c(B,\Psi)\Big) =$$
$$\Big(c(B,x) + c(C,x) - c(B,\Psi) - c(C,y)\Big) + \Big(c(A,\Phi) - c(A,y)\Big) > 0$$

which proves the claim also for the $W_\theta^{(m)} \neq W_\theta^{(m+1)}$ case, thus completing the proof. ■

We can control the convergence speed by requiring each turn to reduce the cost by a factor of α, in order for the turn to be accepted and continue the optimizing process; i.e., accept the outcome from the re-optimization of an r-shape at the mth iteration, only if $c^{(m)} \geq (1+\alpha)c^{(m+1)}$. In this case, where an at least α improvement is achieved at each turn, the following proposition describes the convergence speed.

Proposition 2: The iterative local search approach for distributed facility location converges in $O(\log_{1+\alpha} n \cdot max(s(v))/min(s(v)))$ steps.

Proof: Let $c^{(0)}$, $c^{(M)}$, c^* denote the initial cost, a locally minimum cost obtained at the last (Mth) iteration, and the minimum cost of a (globally) optimal solution, respectively. Here we consider M to be the number of "effective" iterations, i.e., ones that reduce the cost by the required factor. The total number of iterations can be a multiple of M up to a constant given by the number of facilities. Since we are interested in asymptotic complexity we can disregard this and focus on M.

For $m < M$ we have required that $c^{(m)} \geq (1+\alpha)c^{(m+1)}$, or equivalently, $c^{(0)} \geq (1+\alpha)^m c^{(m)}$. Thus when the iteration converges we have:

$$c^{(0)} \geq (1+\alpha)^M c^{(M)} \Rightarrow$$

$$M \leq \log_{1+\alpha} \frac{c^{(0)}}{c^{(M)}} \leq \log_{1+\alpha} \frac{c^{(0)}}{c^*} \tag{7}$$

$c^{(m)}$ is upper bounded by the number of node times the maximum distance to median times the maximum demand, i.e., $O(n^2 \cdot max(s(v)))$. $c^{(0)}$ is lower bounded by the number of nodes times the minimum distance to median times the min demand, i.e., $\Omega(n \cdot min(s(v)))$.

Substituting in Equation (7) gives the claimed upper bound for the number of iterations. Thus, the number of iterations is bounded by $log_{1+a} n \cdot max(s(v))/min(s(v))$. ■

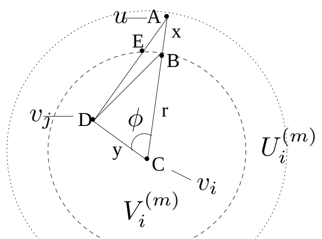

Fig. 3. Example of a possible facility migration from node v_i to node v_j with respect to a particular node $u \in U_i$.

B. The Mapping Error and its Effect on Local Re-Optimizations

In this section we discuss an important difference between solving a centralized version of UKM or UFL (Definitions 1, 2) applied to the entire network and our case where these problems are solved within an r-shape based on the demand that results from a fixed mapping of the ring demand onto the skin. In the centralized case, the amount of demand generated by a node is not affected by the particular configuration of the facilities within the graph, since all nodes in the network are included and considered with their original service demand. In our case, however, the amount of demand generated by a skin node can be affected by the particular configuration of facilities within the r-shape. In Figure 3 we illustrate why this is the case. Node u on the ring has a shortest path to facility node v_i that intersects the skin of v_i's r-ball at point B, thereby increasing the demand of a local node at B by $s(u)$. As the locations of the facilities may change during the various steps of the local optimizing process (e.g., the facility moves from C to D, see Figure 3), the skin node along the shortest path between u and the new location of the facility may change (node/point E in Figure 3). Consequently, a demand *mapping error* is introduced by keeping the mapping fixed (as initially determined) throughout the location optimization process. Let $\Delta_i(r,j,u)$ denote the amount of mapping error attributed to ring node u with respect to a move of the facility from v_i to v_j under the aforementioned fixed mapping and radius r. Then the *total mapping error* introduced in domain W_i under radius r is given by:

$$\Delta_i(r) = \sum_{\substack{v_j \in V_i \\ v_j \neq v_i}} \sum_{u \in U_i v_j \neq v_i} \Delta_i(r,j,u). \qquad (8)$$

The mapping error in Equation (8) could be eliminated by re-computing the skin mapping at each stage of the optimizing process i.e., for each new intermediate facility configuration). Such an approach not only would add to the computational cost but – most important – would be practically extremely difficult to implement as it would require the collection of demand statistics under each new facility placement, delaying the optimization process and inducing substantial overhead. Instead of trying to eliminate the mapping error one could try to assess its magnitude (and potential impact) on the effectiveness of the distributed UKM/UFL.

The example depicted in Figure 3 helps derive an expression for the mapping error $\Delta_i(r,j,u)$, assuming a two-dimensional plane where nodes are scattered in a uniform and continuous manner over the depicted domain. $\Delta_i(r,j,u)$ corresponds to the length difference of the two different routes between node u (point A) and node v_j (point D). Therefore,

$$\Delta_i(r,j,u) = |AB| + |BD| - |AD|. \qquad (9)$$

Note that for those cases in which the angle $\hat{\phi}$ between AC and CD, is 0 or π, $|AB| + |BD| = |AD|$, and therefore, $\Delta_i(r,j,u) = 0$. For any other value of $\hat{\phi}$, AB, BD and AD correspond to the edges of the same triangle and therefore, $|AB| + |BD| - |AD| > 0$ or $\Delta_i(r,j,u) > 0$.

Based on Equation (9), it is possible to derive an upper bound regarding the total mapping error $\Delta_i(r)$ for this particular environment. In [25, Appendices E, F], we prove that:

$$\Delta_i(r) \leq 2\pi^2 r^3 (R^2 - r^2), \qquad (10)$$

where R is the radius of the particular domain W_i.

According to Equation (10), the upper bound for $\Delta_i(r)$ is close to 0, when $r \to 0$ or $r \to R$. We are interested in those cases where the r-ball is small. This corresponds to small values of r for the particular (two-dimensional continuous) environment. Therefore, a small radius r in addition to being preferable for scalability reasons has the added advantage of facilitating the use of a simple and practical mapping with small error and expected performance penalty. Indeed, in the following sections we show that small values for r yield both fast and accurate results in different network graphs and demands.

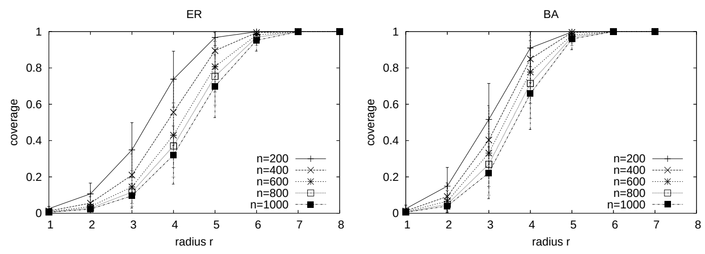

Fig. 4. Average coverage of a node for different size of ER and BA graphs.

VI. Synthetic Results on ER and BA Graphs

In this section we evaluate our distributed facility location approach on synthetic Erdös-Rényi (ER) [27] and Barabási-Albert (BA) [28] graphs generated using the BRITE generator [29]. For ER graphs, BRITE uses the Waxman model [30] in which the probability that two nodes have a direct link is $P(u,v) = \alpha \cdot e^{-d/(\beta L)}$, where d is the Euclidean distance between u and v, and L is the maximum distance between any two nodes. We maintain the default values of BRITE $\alpha = 0.15$, $\beta = 0.2$ combined with an incremental model in which each node connects to $m = 2$ other nodes. For BA graphs, i.e., random scale-free graphs using a preferential attachment mechanism, we also use incremental growth with $m = 2$. This parameterization creates graphs in which the number of (undirected) links is almost double the number of vertices (as also observed in real AS traces that we use later in the paper).

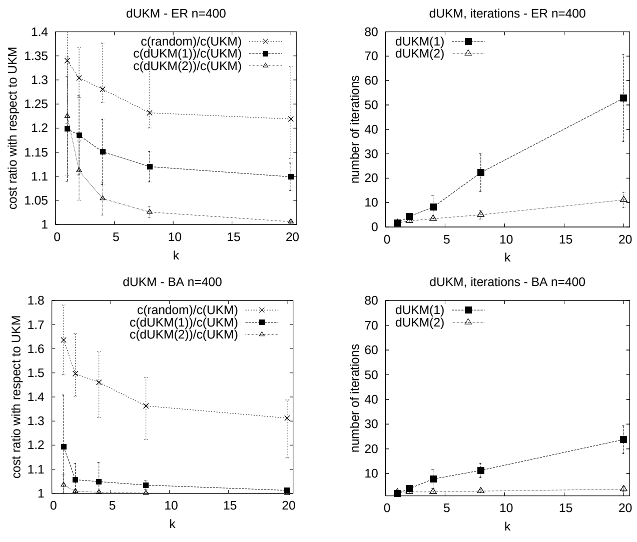

Fig. 5. The relative performance between random and UKM, the dUKM(r), UKM, and the number of iterations for the convergence of the dUKM, for $r = 1$ and $r = 2$, and different facility densities $k/n = 0.1\%, 0.5\%, 1\%, 2\%$, and 5% under ER and BA graphs.

A. Node Coverage with Radius r

Figure 4 depicts the fraction of the total node population that can be reached in r hops starting from a certain node in ER and BA graphs, respectively. We plot the mean and the 95^{th} percentile confidence interval of each node under different network sizes $n =$ $400,$ $600,$ $800,$ 1000, representing typical populations of core ASes on the Internet as argued later on. The figures show that a node can reach a substantial fraction of the total node population by using a relatively small r. In ER graphs, $r = 2$ covers $2\% - 10\%$ of the nodes, whereas $r = 3$ increases the coverage to $10\% - 32\%$, depending on network size. The coverage is even higher in BA graphs, where $r = 2$ covers $4\%-15\%$, whereas $r = 3$ covers $20\%-50\%$, depending again on network size. These observations are explained by the fact that in larger networks the density of nodes within r hops varies in BA and ER graphs. In BA graphs, owing to the preferential attachment, it is expected that the density of nodes that are reachable within r hops increases as the population of the nodes increases. On the other hand, in ER graphs the density of the nodes within r hops does not increase as fast as in BA graphs as the node population increases.

B. Performance of Distributed UKM

In this section we examine the performance of our distributed UKM of radius r, hereafter referred to as dUKM(r), when compared to the centralized UKM utilizing full knowledge. For the rest of the paper, unless otherwise mentioned, all the nodes in the network are candidate facilities. The set of facilities ($|F| = k$, where $F \in V$, see Section II) and the service cost in UKM can be obtained by solving an Integer Linear Program (ILP). In this work we use the TOMLAB/CPLEX ILP solver. For the ILP formulation of the k-median problem see [13]. Note that the solution of ILP yields the optimal cost. We fix the network size to $n = 400$ (matching measurement data on core Internet ASes that we use later on) and assume that all nodes generate the same amount of service demand $s(v) = 1, \forall v \in V$. To ensure scalability, we don't want our distributed solution to encounter r-shapes that involve more than 10% of the total nodes, and for this we limit the radius to $r = 1$ and $r = 2$, as suggested by the node coverage results of the previous section. We let the fraction of nodes that are able to act as facilities (i.e., service hosts) take values $k/n = 0.1\%, 0.5\%, 1\%, 2\%,$ and 5%. We perform each experiment 100 times to reduce the uncertainty due to the initial random placement of the k facilities.

The plots on the left-hand-side of Figure 5 depict the cost of our dUKM(r) approach normalized over that of the optimal but centralized UKM, with the plot on top for ER graphs and the plot on the bottom for BA graphs. We also plot the normalized cost of open uniformly at random k facilities, denoted as *random*. For both ER and BA graphs, the performance of our distributed solution tracks closely that of the centralized one, with the difference diminishing fast as r and k are increased. The normalized performance for BA graphs converges faster (i.e., at smaller k for a given r) to ratios that approach 1. This owes to the existence of highly-connected nodes (the so called "hubs") in BA graphs — building facilities in few of the hubs is sufficient for approximating closely the performance of the centralized UKM. The two plots on the right-hand-side of Figure 5 depict the number of iterations needed for dUKM(r) to converge. A smaller value of r requires more iterations as it leads to the creation of a large number of small sub-problems (re-optimizations of many small r-shapes). BA graphs converge in fewer iterations, since for the same value of r BA graphs induce larger r-shapes. Again, it is the hubs that create large r-shapes. Even under a small r, a hub will be close to the facility that re-optimizes its location, and this will bring many of the hub's immediate neighbors into the r-shape. We conclude that less re-optimizations are needed in BA graphs than in ER graphs when the number of nodes and demand profile are the same. The cost of opening facilities uniformly at random is much higher than this of our distributed UKM. This supports our argument that distributed UKM yields significant cost reduction in both ER and BA graphs.

C. Performance of Distributed UFL

In order to evaluate the performance of distributed UFL of radius r, henceforth referred to as dUFL(r), we need to decide how to set the facility acquisition costs $f(v_j)$ which constitute part of the input of a UFL problem (see Definition 2). This is a non-trivial task, essentially a pricing problem for network services. Although pricing is clearly out of scope for this paper, we need to use some form of $f(v_j)$'s to demonstrate our point that, as with UKM, the performance of the distributed version of UFL tracks closely that of the optimal but centralized UFL (obtained by solving the ILP). To that end, we use two types of facility costs: *uniform*, where all facilities cost the same independently of location (i.e., $f(v_j) = f$, $\forall v_j \in V$) and, *non-uniform*, where the cost of a facility at a given node depends on the location of that node. The uniform cost model is more relevant when the dominant cost is that of setting up the service on the host, whereas the non-uniform cost model is more relevant when the dominant cost is that of operating

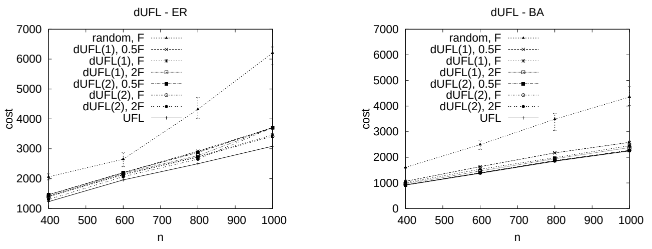

Fig. 6. Cost comparison between random, dUFL(r) and UFL, for $r = 1$ and $r = 2$, and different network sizes under ER and BA graphs and degree-based facility cost $f(v_j) = d(v_j)^{1+\alpha_G}$.

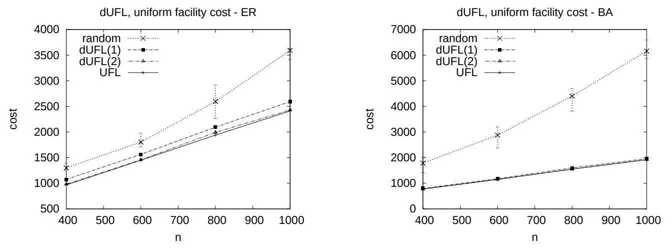

Fig. 7. Cost comparison between random, dUFL(r) and UFL, for $r = 1$ and $r = 2$, and different network sizes under ER and BA graphs and uniform facility cost.

the facility (implying that this operating cost is proportional to the desirability of the host, which depends on topological location). The later cost model is general enough to capture the congestion associated with each facility.

For the non-uniform case we will use the following rule: we will make the cost of acquiring a facility proportional to its degree, i.e., proportional to the number of direct links it has to other nodes. The intuition behind this is that a highly connected node will most likely attract more demand from clients, as more shortest-paths will go through it and, thus, building a facility there will create a bigger hot-spot, and therefore the node should charge more for hosting a service. Note, as sketched in the Introduction, a node may correspond to an AS that charges for allowing network services to be installed on its local GSH. In [31],[32] the authors showed that the "coverage" of a node increases super-linearly with its degree (or alternatively, the number of shortest paths that go through it). We, therefore, use as facility cost $f(v_j) = d(v_j)^{1+\alpha_G}$, where $d(v_j)$ is the degree of node $v_j \in V$ and α_G is the skewness of the degree distribution of the graph G. In order to estimate the value of α_G, we use the Hill estimator: $\hat{\alpha}^{(Hill)}_{k,m} = 1/\hat{\gamma}_{k,m}$, where: $\hat{\gamma}_{k,m} = \frac{1}{k}\sum_{i=1}^{k} \log \frac{X_{(i)}}{X_{(k+1)}}$, $X_{(i)}$ denotes the i-th largest value in the sample $X_1, ..., X_n$. We prefer the Hill estimator since it is less biased than linear regression for fitting power-law exponents.

In Figure 6 we plot the cost of dUFL(1), dUFL(2), and centralized UFL, in ER and BA graphs under the aforementioned degree-based facility cost. For dUFL, we present three lines for each radius r, corresponding to different initial number of facilities used in the iterative algorithm of Section IV-C. We use $k_0 = 0.5 \cdot F$, F, and $2 \cdot F$, where F denotes the number of facilities opened when applying the corresponding centralized UFL. As evident from the results, the cost of dUFL is close to that of UFL (around 5-15% for both types of graphs). As with dUKM, the performance improves with r and is slightly better for BA graphs (see the explanation in Section VI-B). Also we observe a tendency for lower costs when starting the distributed algorithm with a higher number of initial facilities. Under the non-uniform (degree-based) cost model, both dUFL and UFL open facilities in 2-8% of the total nodes, depending on the example.

We also evaluate the performance of dUFL under uniform facility cost f; the cost is set at a value that leads to building the same number of facilities as the corresponding degree-based example. Both the distributed and centralized UFL build the same number of facilities, and the performance of dUFL is very close to the centralized one, as is illustrated in Figure 7. Again, we emphasize that our goal here is not to evaluate performance under different pricing scheme, but rather to show that the performance of distributed UFL tracks well that of the centralized, optimal approach. Moreover, under both cost models, the random placement of facilities yields high cost.

The number of iterations for dUFL(r) to converge is similar to this of the dUKM(r), for the same graph (ER or BA, and size of graph), in both the case of degree-based and the uniform facility cost. The number of iterations can be significantly reduced, especially for large graphs, if no migration takes place unless the service cost is decreased by a factor α, as discussed in Section V-A.

VII. Results for Real AS-level Topologies

To further investigate the performance of our distributed approach and to better support our sketched application scenario described in the introduction, we include in this section performance results on real AS-level maps under non-uniform service demand from different clients. We choose the AS-level to evaluate our approach as many infrastructure providers such as content distributors, data-centers and cloud providers are located within an AS that peers with other ASes [33], [34], or maintaining a highly distributed hosting infrastructure in a large number of ASes [17], [35], [9] to better satisfy end-user demand and control their operational cost. We assume that a candidate facility is present in each AS.

A. Description of the AS-level Dataset

We use the relation-based AS map of the Internet obtained using the measurement methodology described in [36]. The dataset includes two kinds of relationships between ASes.
Customer-Provider: The customer is typically a smaller AS that pays a larger AS for providing it with access to the rest of the Internet. The provider may, in turn, be a customer of an even larger AS. A customer-provider relationship is modeled using a directed link from the provider to the customer.
Peer-Peer: Peer ASes are typically of comparable sizes and have mutual agreements for carrying each other's traffic. Peer-peer relationships are modeled using undirected links.

Overall the dataset includes 12,779 unique ASes, 1,076 peers and 11,703 customers, connected through 26,387 directed and 1,336 undirected links. Since this AS graph is not connected, we chose to present results based on its largest connected component, which we found to include a substantial part of the total AS topology at the peer level: 497 peer ASes connected with 1,012 undirected links. There are smaller

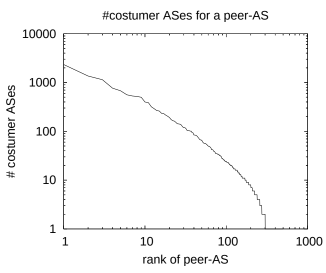

Fig. 8. Number of customer ASes for each peer-AS in decreasing order according to rank.

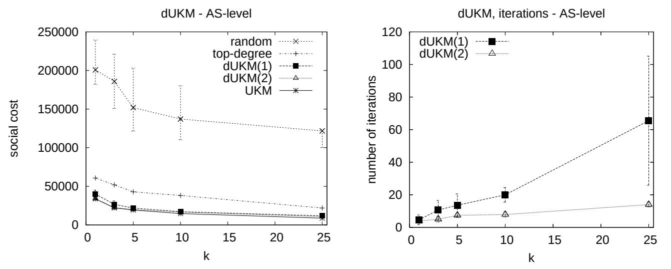

Fig. 9. The cost of random, top-degree, dUKM(r) and UKM, and the number of iterations for the convergence of the dUKM, for $r = 1$ and $r = 2$, and different facility densities $k/n = 0.1\%, 0.5\%, 1\%, 2\%$, and 5% in the AS graph.

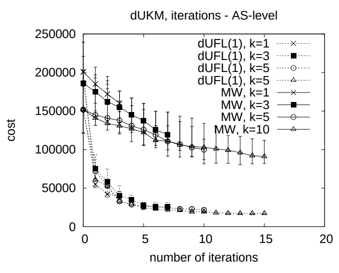

Fig. 10. The cost of dUKM(1) and MW distributed algorithm in the AS graph as the number of iterations increases.

connected components (2-8 ASes) that are formed by small regional ISPs with peering relationships. We verified that this component contains all the 20 largest peer ASes reported in [36]. Since it would be very difficult to obtain the real complex routing policies of all these networks, we did not consider policy-based routing, but rather assumed shortest-path routing based on the connected component.

We exploit the relationships between ASes in order to derive a more realistic (non-uniform) service demand for the peer ASes that we consider. Our approach is to count for each peer AS the number of customer ASes that have it as provider, either directly or through other intermediary ASes. We then set the service demand of a peer AS to be proportional to this number. In Figure 8 we plot the demand profile of peer ASes (in decreasing order using log-log scale). As evident from this plot, the profile is power-law like (with slight deviation towards the tail), meaning that few core ASes carry the majority of the demand that flows from client ASes. In the sequel we present performance results in which nodes correspond to peer ASes that generate demand that follows the aforementioned power-law like profile. We seek to identify the peer ASes for building service facilities.

B. Distributed UKM on the AS-level Dataset

The plots on the left-hand-side of Figure 9 show the cost of random, top-degree, dUKM(1), dUKM(2), and the centralized UKM, under the AS-level graph. Top-degree is a heuristic where k facilities are placed at ASes with the highest peer degree and requires full knowledge of the AS-level topology which is difficult to get as links may be missing [37]. Our manual investigation shows that the footprints of ASes with high peer degree are concentrated in different geographical locations. Clearly, even for small values of r, the performance of our distributed approaches track closely that of the optimal but centralized approach. The random opening of facilities again yields high cost. On the other hand, the top-degree heuristic yields low cost that is approximately 15-20% higher than this of the dUKM and centralized UKM.

Regarding the number of iterations needed for convergence, the same observations apply as with the synthetic topologies, i.e., they increase with smaller radii. The substantial benefit from knowledge of only local neighborhood topologies ("neighbors of neighbor") has been observed for a number of applications, including [24] which has also investigated and quantified implementation overhead in an Internet setting. In Figure 10 we plot the cost of dUFL(1) as the number of iterations increase. We notice that only two or three iterations are sufficient to get the cost reduction that is close to the optimal but centralized UKM.

We also compare with the only distributed facility location work we are ware of, the one by Moscibroda and Wattenhofer [38], henceforth refereed as *MW*. This work explores a trade-off between the amount of communication and the resulting approximation ration. The authors showed that it is possible to achieve an $O(\sqrt{\mu}(m\rho)^{1/\sqrt{\mu}}\log(m+n))$ approximation in $O(\mu)$ communication rounds (in our algorithm this refers to iterations), where the message size is bounded by $O(\log n)$ bits, m is the number of facilities, n is the number of clients, and ρ is a coefficient that depends on the cost values. We present the performance of MW algorithm in Figure 10. The performance of the MW algorithm is poor, especially when the number of rounds is small. This is expected as WM algorithm tries to approximate the solution of the centralized facility location with a minimal number of rounds of communication and minimal message size.

C. Distributed UFL on the AS-level Dataset

Table I presents the performance of dUFL and random opening of facilities on the AS-level dataset. Again, it is verified that dUFL is very close in performance to UFL, even for small values of r (within 4% for $r = 2$, under both examined facility cost models). The cost of randomly open facilities is, in some cases, more than six times the cost of our distributed UFL. The top-degree heuristic yields low cost under the degree-based cost model but not under the uniform cost model. Notice also that the number of facilities in the case of random and top-degree placement has to be estimated offline. We also noticed that again a small number of rounds is sufficient to reduce the dUFL cost close to this of the optimal but centralized UFL. The performance of MW is far from optimal (R=10 rounds) but improves the initial cost with minimal communication cost.

VIII. Non-stationary demand and imperfect redirection

Up to now, our performance study has been based on assuming (1) stationary demand, and (2) perfect redirection of each client to its closest facility node. The stationary demand assumption is not justified for relatively large time-scales (hours or days), and perfect redirection can be either too

	dUFL(1)	dUFL(2)	top-degree	random	MW
degree -based	1.22 (1.20)	1.04 (1.03)	1.35 (1.35)	6.67 (6.68)	6.20 (6.28)
uniform	1.01 (1.01)	1.01 (1.01)	1.80 (1.80)	6.35 (6.33)	6.01 (6.04)

TABLE I
MEAN (MEDIAN) COST RATIO BETWEEN dUFL(r), TOP-DEGREE, RANDOM, MW, AND UFL IN THE AS-LEVEL TOPOLOGY.

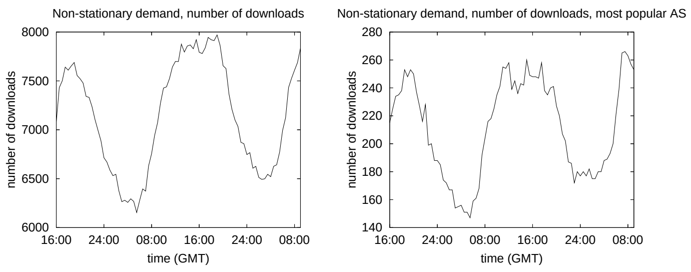

Fig. 11. The number of concurrent downloads from all ASes and from the most popular AS in the torrent of an on-line multi-player game at each measurement point.

costly to implement or too difficult to enforce due to faults or excessive load. In this section we look at the performance of distributed facility location scheme when dropping the aforementioned assumptions. First, we present a measurement study for obtaining the non-stationary demand corresponding to a multi-player on-line game and then use this workload to derive a performance comparison between dUFL and UFL. Then, we assume that mapping a client to its closest facility node has to incur some time lag and study the performance implications of such an imperfect redirection scheme.

A. Measuring the Demand of a Popular Multi-player Game

We used the Mininova web-site to track all requests for joining a torrent corresponding to a popular on-line multi-player game. By tracking the downloads of the game client, which is possible to do due to the use of BitTorrent, we can obtain a rough idea about the demographics of the load put on the game servers, to which we do not have direct access. We then use this workload to quantify the benefits of instantiating game servers dynamically according to dUFL. Hereafter, we use the term facility and server interchangeably.

More specifically, we connected periodically at 30-minute intervals to the tracker serving this torrent, over a total duration of 42 hours. At each 30-minute interval, we got all the IPs of participating downloaders by issuing to the tracker multiple requests for neighbors until we got all distinct downloaders at this point in time. In Figure 11 (left) we plot the number of concurrent downloads at each measurement point. Overall, we were able to capture a sufficient view of the activity of the torrent and detect expected profiles, e.g., diurnal variation over the course of a day. In total, we saw 34,669 unique users and the population varied from 6,000 to 8,000 concurrent users, i.e., the population variance was close to 25%.

Moving on, we used Routeviews dataset [39] to map each logged IP address to an AS. The variance in the number of concurrent users from a particular AS was even higher. Focusing on the most popular AS, we found out that the variance in the number of concurrent users was as high as 50%, as it is shown in Figure 11 (right). Last, we looked at churn at the AS level by counting the number of new ASes joining and existing ASes leaving the torrent over time [40]. Formally, we defined $churn(t) = \frac{U_{t-1} \ominus U_t}{\max\{|U_{t-1}|,|U_t|\}}$, where U_t is the set of ASes at time t, and $\ominus$ is the set difference operator. In Figure 12 we plot the evolution of churn. One can observe that AS-level churn is quite high, ranging from 6% to 11%, with no specific pattern. This serves our purpose which is to study the performance of dUFL under non-stationary demand.

B. Distributed UFL under Non-stationary Demand

We consider a distributed server migration scheme given by dUFL with radius $r = 1$. The pricing model for starting a server at an AS is the aforementioned degree-based one of Section VI-C. The evaluation assumes an AS-level topology obtained from Routeviews. The demand originating from each AS at each particular point in time is set equal to the value we obtained from measuring the downloads going to the torrent of the game client. We compare the cost of UFL, dUFL(1), with this of two static placement heuristics: static-min and static-max. Static-min is a simple heuristic that maintains the same placement across time. The number of maintained facilities is equal to the minimum number of facilities that UFL opened in the duration of the experiment. This is used as a baseline for the performance of an under-provisioned static placement of servers according to minimum load. Static-max captures the cost of an over-provisioned placement according to peek load. Obviously, static-max suffers from a high purchase cost of buying a maximum number of servers (in this case 100), whereas static-min suffers from high communication cost to reach the smaller number of used servers (in this case 70).

We report the average cost in the duration of the experiment (42 hours) for each one of the aforementioned policies. For each policy we repeated the experiment 100 times to remove the effect of the initial random opening of facilities. In Figure 15 we plot the resulting average costs along with 95^{th} percentile confidence intervals. One can see that dUFL(1) achieves 4 to 7 times lower cost compared to static-min and static-max. Looking at the close-up, it can also be seen that dUFL(1) is actually pretty close, within 10-20%, of the performance of the centralized UFL computed at each point in time. Taken together, these results indicate that dUFL(1) yields a high performance also under non-stationary demand.

Next, we quantify the number of server migrations required by dUFL(1), between consecutive intervals, to track the offered non-stationary demand. In Figure 13 we plot the percentage of servers that are migrated, henceforth referred as migration ratio, along with 95^{th} percentile confidence intervals based on 100 runs. Evidently, migrations are rather rare, typically 0%-3%, after the servers stabilize from their initial random positions, to where dUFL(1) will have them at each point in time. In Figure 14 we show the number of utilized servers over time when applying our distributed algorithm (dUFL(1)) under non-stationary demand. The number of utilized servers varies a lot throughout the experiment, but the migration ratio between two consecutive intervals is small. There are two noticeable exceptions, at around 16:00 GMT and 08:00 GMT, when churn

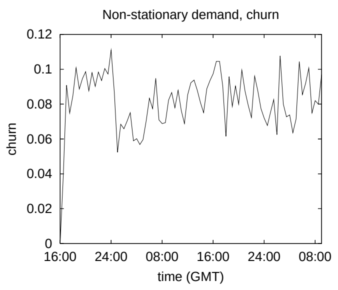

Fig. 12. Churn evolution in the AS-level in the torrent of a popular on-line multi-player game at each measurement point.

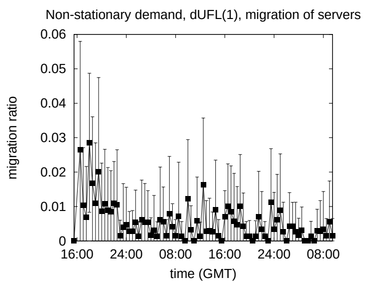

Fig. 13. Migration ratio of dUFL(1) in the torrent of a popular on-line multi-player game at each measurement point.

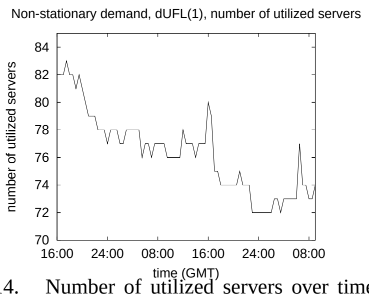

Fig. 14. Number of utilized servers over time when applying dUFL(1) under non-stationary demand.

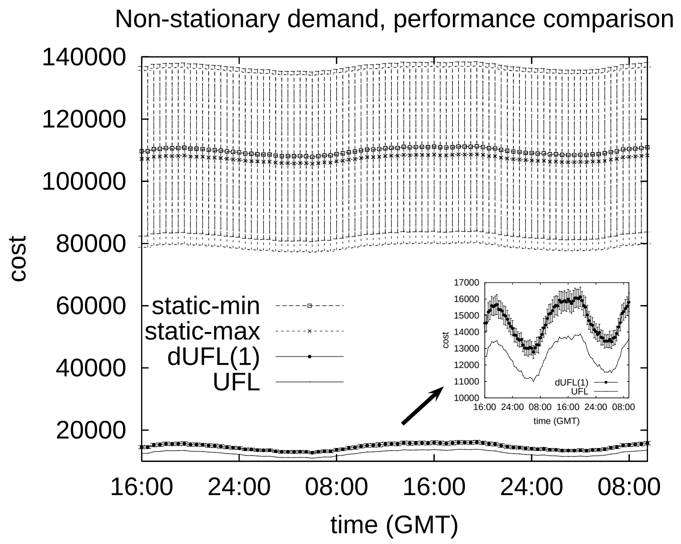

Fig. 15. Average cost of static-min, static-max, dUFL(1), and UFL in the torrent of a popular on-line multi-player game at each measurement point.

is high (up to 7%). This is to be expected as the evening peak hour (in different parts of the globe) starts at the around that time. These results suggest that dUFL(1) is relatively robust to demand changes and can typically address them without massive numbers of migrations that are of course costly in terms of bandwidth and management. Of course, the number of migrations can be reduced further by trading performance with laziness in triggering a migration.

C. The Effect of Imperfect Redirection

We now move on to dropping the assumption that clients are always redirected to their closest facility, which pretty much implies that there are no performance penalties for them due to server migrations. In many cases it has been shown that perfect redirection is indeed feasible using route triangulation and DNS [21]. In this section, however, we relax this assumption, and study the effects of imperfect redirection. We do so to cover cases in which perfect redirection is either too costly to implement, or exists, but performs sub-optimally due to faults or excessive load.

To this end, we assume that there exists a certain amount of *lag* between the time a server migrates to a new node and the time that the migration is communicated to the affected clients. During this time interval, a client might be receiving service from its previously closest facility which, however, may have

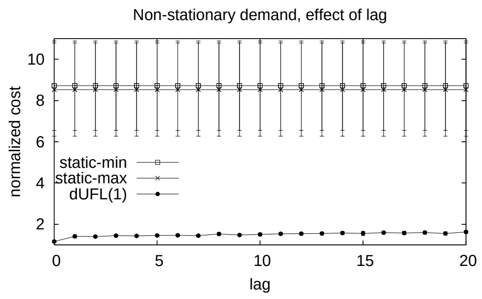

Fig. 16. Normalized cost of static-min, static-max and dUFL(1) with respect to the cost of UFL in the torrent of a popular on-line multi-player game under various levels of lag.

ceased to be optimal due to one or several migrations. Since we assume that migrations occur at fixed time intervals, we measure the lag in terms of number of such intervals (1 facility migration at each interval). Notice that under the existence of lag, even with stationary demand, the optimization is no longer guaranteed to be loop-free as the server that is about to migrate is still active (see Section V-A). We solve this by stopping the iterative re-optimization if it reaches a certain high number of iterations.

In Figure 16 we plot the cost ratio between dUFL(1) and dUFL and the 95^{th} percentile confidence interval under various levels of lag that range from 0 up to 20 As expected, lag puts a performance penalty on dUFL. The degradation, however, is quite smooth, while the performance always remains superior to static-min and static-max.

IX. Related Work

There is a rich literature on facility location theory. Initial results are surveyed in the book by Mirchandani and Francis [8]. A large number of subsequent works focused on developing centralized approximation algorithms [13], [14], [15], [16]. The authors of [41] have proposed an alternative approach for approximating facility location problems based on a continuous “high-density” model. Recently, generalizations of the classical centralized facility location problem have appeared in [42], [43]. The first mention of a distributed facility location algorithm is by Jain and Vazirani [16] while commenting on their primal-dual approximation method, but they do not pursue the matter further. To the best of our knowledge, the only work in which distributed facility location has been the focal point seems to be the recent work of

Moscibroda and Wattenhofer [38]. This work explores a trade-off between the amount of communication and the resulting approximation ration. The authors showed that it is possible to achieve a non-trivial approximation with constant number communication rounds where the message size is bounded. The online version of facility location, in which request arrive one at a time according to an arbitrary pattern, has been studied by Meyerson [44] that gave a randomized online $O(1)$-competitive algorithm for the case that requests arrive randomly and a $O(\log n)$-competitive algorithm for the case that arrival order is selected by an adversary. Andreev et al. [45] very recently proposed approximation algorithms to select locations for sources in a capacitated graph such that a given set of demands can be satisfied simultaneously, with the goal of minimizing the number of locations chosen. Their solution is centralized. Oikonomou and Stavrakakis [46] have proposed a fully distributed approach for service migration — their results, however, are limited to a single facility (representing a unique service point) and assume tree topologies.

Several application-oriented approaches to distributed service deployment have appeared in the literature, e.g., Yamamoto and Leduc [47] (deployment of multicast reflectors), Rabinovich and Aggarwal [48] (deployment of mirrored web-content), Chambers et al. [49] (on-line multi-player network games), Cronin et al. [50] (constrained mirror placement), Krishnan et al. [51] (cache placement), Qureshi et al. [52] (energy cost-aware server selection), and Frank et al. [53] (on-demand server deployment in microdatacenters). The aforementioned works are strongly tied to their specific applications and do not have the underlying generality offered by the distributed facility location approach adopted in our work. Relevant to our work are also the works of Oppenheimer et al. [54] on systems aspects of a distributed shared platform for service deployment, Aggarwal et al. [55] on automated data placement for geo-distributed cloud services, Wendell et al. [56] on decentralized server selection for cloud services, and Loukopoulos et al. [57] on the overheads of updating replica placements under non-stationary demand.

X. Conclusion

We have described a distributed approach for the problem of placing service facilities in large-scale networks. We overcome the scalability limitations of classic centralized approaches by re-optimizing the locations and the number of facilities through local optimizations which are refined in several iterations. Re-optimizations are based on exact topological and demand information from nodes in the immediate vicinity of a facility, assisted by concise approximate representation of demand information from neighboring nodes in the wider domain of the facility. Using extensive synthetic and trace-driven simulations we demonstrate that our distributed approach is able to scale well by utilizing limited local information without making serious performance sacrifices as compared to centralized optimal solutions. We also demonstrate that our distributed approach yields a high performance under non-stationary demand and imperfect redirection. Our approach leverages recent advances in virtualization technology and the flexibility of billing models, such as pay-as-you-go, towards a fully automated Internet-scale service deployment.

References

[1] N. Laoutaris, G. Smaragdakis, K. Oikonomou, I. Stavrakakis, and A. Bestavros, “Distributed Placement of Service Facilities in Large-Scale Networks,” in *IEEE INFOCOM ’07*.

[2] C. Gkantsidis, T. Karagiannis, P. Rodriguez, and M. Vojnovic, “Planet Scale Software Updates,” in *ACM SIGCOMM ’06*.

[3] N. Laoutaris, P. Rodriguez, and L. Massoulie, “ECHOS: Edge Capacity Hosting Overlays of Nano Data Centers,” *ACM SIGCOMM CCR*, vol. 38, no. 1, pp. 51–54, 2008.

[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, “A view of cloud computing,” *CACM*, vol. 53, no. 4, pp. 50–58, 2010.

[5] S. Ghemawat, H. Gobioff, and S.-T. Leung, “The Google file system,” *SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 29–43, 2003.

[6] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, “Dynamo: Amazon’s Highly Available Key-value Store,” in *ACM SOSP ’07*.

[7] P. Wendell and M. J. Freedman, “Going Viral: Flash Crowds in an Open CDN,” in *ACM IMC ’11*.

[8] P. Mirchandani and R. Francis, *Discrete Location Theory*. John Wiley and Sons, 1990.

[9] B. Ager, W. Mühlbauer, G. Smaragdakis, and S. Uhlig, “Web Content Cartography,” in *ACM IMC ’11*.

[10] I. Poese, B. Frank, G. Smaragdakis, S. Uhlig, A. Feldmann, and B. Maggs, “Enabling Content-aware Traffic Engineering,” *ACM SIGCOMM CCR*, vol. 42, no. 5, pp. 21–28, 2012.

[11] “Network Functions Virtualisation,” Network Operators White Paper, appeared in SDN and OpenFlow World Congress, Oct. 2012.

[12] O. Kariv and S. Hakimi, “An Algorithmic Approach to Network Location Problems, Part II: p-medians,” *SIAM Journal on Applied Mathematics*, vol. 37, pp. 539–560, 1979.

[13] M. Charikar, S. Guha, D. B. Shmoys, and E. Tárdos, “A Constant Factor Approximation Algorithm for the k-median Problem,” in *ACM STOC ’99*.

[14] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman, “Analysis of a Local Search Heuristic for Facility Location Problems,” in *ACM SODA ’98*.

[15] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit, “Local Search Heuristics for k-Median and Facility Location Problems,” *SIAM J. on Computing*, vol. 33, no. 3, pp. 544–562, 2004.

[16] K. Jain and V. V. Vazirani, “Primal-Dual Approximation Algorithms for Metric Facility Location and k-Median Problems,” in *IEEE FOCS ’99*.

[17] T. Leighton, “Improving Performance on the Internet,” *CACM*, vol. 52, no. 2, pp. 44–51, 2009.

[18] D. Kostić, A. Rodriguez, J. Albrecht, and A. Vahdat, “Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh,” in *ACM SOSP ’03*.

[19] R. Kohavi, R. M. Henne, and D. Sommerfield, “Practical Guide to Controlled Experiments on the Web: Listen to Your Customers not to the HiPPO,” in *ACM KDD ’07*.

[20] J. Pan, Y. T. Hou, and B. Li, “An Overview DNS-based Server Selection in Content Distribution Networks,” *Comp. Netw.*, vol. 43, no. 6, 2003.

[21] N. Faber and R. Sundaram, “MOVARTO: Server Migration across Networks using Route Triangulation and DNS,” in *VMworld ’07*.

[22] B. Ager, W. Mühlbauer, G. Smaragdakis, and S. Uhlig, “Comparing DNS Resolvers in the Wild,” in *ACM IMC ’10*.

[23] V. Lenders, M. May, and B. Plattner, “Density-based vs. Proximity-based Anycast Routing for Mobile Networks,” in *IEEE INFOCOM ’06*.

[24] M. Naor and U. Wieder, “Know Thy Neighbor’s Neighbor: Better Routing for Skip-Graphs and Small Worlds.” in *IEEE IPTPS ’04*.

[25] G. Smaragdakis, “Overlay Network Creation and Maintenance with Selfish Users,” *PhD Dissertation, CS dpt., Boston U.*, September 2008.

[26] G. Smaragdakis, N. Laoutaris, V. Lekakis, A. Bestavros, J. W. Byers, and M. Roussopoulos, “Selfish Overlay Network Creation and Maintenance,” *IEEE/ACM Tran. on Netw.*, vol. 19, no. 6, pp. 1624–1637, Dec. 2011.

[27] P. Erdös and A. Rényi, “On random graphs I,” *Publ. Math. Debrecen*, vol. 6, pp. 290–297, 1959.

[28] A.-L. Barábasi and R. Albert, “Emergence of Scaling in Random Networks,” *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[29] A. Medina, A. Lakhina, I. Matta, and J. Byers, “BRITE: An Approach to Universal Topology Generation,” in *MASCOTS ’01*.

[30] B. M. Waxman, “Routing of multipoint connections,” *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 9, pp. 1617–1622, 1988.

[31] S. Jin and A. Bestavros, "Small-World Internet Topologies: Possible Causes and Implications on Scalability of End-System Multicast," CS dpt, Boston U., Tech. Rep. BUCS-TR-2002-004.
[32] ——, "Small-world characteristics of internet topologies and implications on multicast scaling," *Comp. Netw.*, vol. 50, no. 5, 2006.
[33] D. Antoniades, E. Markatos, and C. Dovrolis, "One-click Hosting Services: A File-Sharing Hideout," in *ACM IMC '09*.
[34] K. Rup, H. V. Madhyastha, S. Srinivasan, S. Jain, A. Krishnamurthy, T. Anderson, and J. Gao, "Moving Beyond End-to-end Path Information to Optimize CDN Performance," in *ACM IMC '09*.
[35] E. Nygren, R. K. Sitaraman, and J. Sun, "The Akamai Network: A Platform for High-performance Internet Applications," *SIGOPS Operatings Systems Review*, vol. 44, pp. 2–19, August 2010.
[36] L. Subramanian, S. Agarwal, J. Rexford, and R. H. Katz, "Characterizing the Internet Hierarchy from Multiple Vantage Points," in *IEEE INFOCOM '02*.
[37] B. Ager, N. Chatzis, A. Feldmann, N. Sarrar, S. Uhlig, and W. Willinger, "Anatomy of a Large European IXP," in *ACM SIGCOMM '12*.
[38] T. Moscibroda and R. Wattenhofer, "Facility Location: Distributed Approximation," in *ACM PODC '05*.
[39] University of Oregon Route Views Project, http://www.routeviews.org/.
[40] P. B. Godfrey, S. Shenker, and I. Stoica, "Minimizing Churn in Distributed Systems," in *ACM SIGCOMM '06*.
[41] C. W. Cameron, S. H. Low, and D. X. Wei, "High-density Model for Server Allocation and Placement," in *ACM SIGMETRICS '02*.
[42] M. Mahdian and M. Pal, "Universal Facility Location," in *ESA '03*.
[43] N. Garg, R. Khandekar, and V. Pandit, "Improved Approximation for Universal Facility Location," in *ACM SODA '05*.
[44] A. Meyerson, "Online Facility Location," in *IEEE FOCS '01*.
[45] K. Andreev, C. Garrod, B. Maggs, and A. Meyerson, "Simultaneous Source Location," *ACM Trans. on Algor.*, vol. 6, no. 1, pp. 1–17, 2009.
[46] K. Oikonomou and I. Stavrakakis, "Service Migration: The Tree Topology Case," in *Med-Hoc-Net '06*.
[47] L. Yamamoto and G. Leduc, "Autonomous Reflectors Over Active Networks: Towards Seamless Group Communication," *AISB*, vol. 1, no. 1, pp. 125–146, 2001.
[48] M. Rabinovich and A. Aggarwal, "RaDaR: A Scalable Architecture for a Global Web Hosting Service," in *WWW '99*.
[49] C. Chambers, W. chi Feng, W. chang Feng, and D. Saha, "A Geographic Redirection Service for On-line Games," in *ACM MULTIMEDIA '03*.
[50] E. Cronin, S. Jamin, C. Jin, A. R. Kurc, D. Raz, and Y. Shavitt, "Constraint Mirror Placement on the Internet," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 7, 2002.
[51] P. Krishnan, D. Raz, and Y. Shavit, "The Cache Location Problem," *IEEE/ACM Trans. on Networking*, vol. 8, no. 5, pp. 568–581, 2000.
[52] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, "Cutting the Electric Bill for Internet-Scale Systems," in *ACM SIGCOMM '09*.
[53] B. Frank, I. Poese, Y. Lin, R. Weber, J. Rake, G. Smaragdakis, A. Feldmann, S. Uhlig, and B. Maggs, "Pushing CDN-ISP Collaboration to the Limit," *ACM SIGCOMM CCR*, vol. 43, no. 3, 2013.
[54] D. Oppenheimer, B. Chun, D. Patterson, A. C. Snoeren, and A. Vahdat, "Service Placement in a Shared Wide-area Platform," in *USENIX ATC '06*.
[55] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan, "Volley: Automated Data Placement for Geo-Distributed Cloud Services," in *USENIX/ACM NSDI '10*.
[56] P. Wendell, J. W. Jiang, M. J. Freedman, and J. Rexford, "DONAR: Decentralized Server Selection for Cloud Services," in *ACM SIGCOMM '10*.
[57] T. Loukopoulos, P. Lampsas, and I. Ahmad, "Continuous Replica Placement Schemes in Distributed Systems," in *ACM ICS '05*.

Georgios Smaragdakis is a Senior Researcher at Deutsche Telekom Laboratories and the Technical University of Berlin. He received the Ph.D. degree in computer science from Boston University, the Diploma in electronic and computer engineering from the Technical University of Crete, and he interned at Telefonica Research. His research interests include the measurement, performance analysis, and optimization of content distribution systems and overlay networks with main applications in service deployment, server selection, distributed replications and caching, ISP-Application collaboration, and overlay network creation and maintenance. Dr. Smaragdakis received the ACM IMC 2011 best paper award for his work on web content cartography.

Nikolaos Laoutaris is a senior researcher at the Internet research group of Telefonica Research in Barcelona. Prior to joining the Barcelona lab he was a postdoc fellow at Harvard University and a Marie Curie postdoc fellow at Boston University. He got his PhD in computer science from the University of Athens in 2004. His general research interests are on system, algorithmic, and performance evaluation aspects of computer networks and distributed systems. Current projects include: Efficient inter-datacenter bulk transfers, energy-efficient distributed system design, content distribution for long tail content, scaling of social networks, pricing of broadband services and ISP interconnection economics.

Konstantinos Oikonomou received his M.Eng. in Computer Engineering and Informatics from University of Patras, Greece, in 1998. In September 1999 he received his M.Sc. in Communication and Signal Processing from Imperial College (London) and his Ph.D. degree in 2004 from the Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Greece. His Ph.D. thesis focuses on Topology-Unaware TDMA MAC Policies for Ad Hoc Networks. Between December 1999 and January 2005 he was employed at INTRACOM S.A, as a research and development engineer. He is an Assistant Professor in Computer Networks with the Department of Informatics of the Ionian University, Corfu, Greece. His current research interests involve among others, performance issues for ad hoc and sensor networks, autonomous network architectures, efficient service discovery (placement, advertisement and searching) in unstructured environments, scalability issues in large networks.

Prof. Ioannis Stavrakakis , *IEEE Fellow*: Diploma in Electrical Engineering, Aristotelian University of Thessaloniki, (Greece), 1983; Ph.D. in EE, University of Virginia (USA), 1988; Assistant Professor in CSEE, University of Vermont (USA), 1988-1994; Associate Professor of ECE, Northeastern University, Boston (USA), 1994-1999; Associate Professor of Informatics and Telecommunications, University of Athens (Greece), 1999-2002 and Professor since 2002. Teaching and research interests are focused on resource allocation protocols and traffic management for communication networks, with recent emphasis on: peer-to-peer, mobile, ad hoc, autonomic, delay tolerant and future Internet networking. His research has been published in over 170 scientific journals and conference proceedings and was funded by NSF, DARPA, GTE, BBN and Motorola (USA) as well as Greek and European Union (IST, FET, FIRE) Funding agencies. He has served repeatedly in NSF and EU-IST research proposal review panels and involved in the TPC and organization of numerous conferences sponsored by IEEE, ACM, ITC, and IFIP societies. He is the chairman of IFIP WG6.3 and has served as an elected officer for the IEEE Technical Committee on Computer Communications (TCCC). He is an associate editor for the ACM/Kluwer Wireless Networks and Computer Communications journals and has served in the editorial board of the IEEE/ACM transactions on Networking and the Computer Networks Journals.

Azer Bestavros received the Ph.D. degree in computer science from Harvard University in 1992. He is Founding Director of the Hariri Institute for Computing and a Professor in the Computer Science Department at Boston University, which he joined in 1991 and chaired from 2000 to 2007. Prof. Bestavros' research interests are in the broad areas of networking and real-time embedded systems. His contributions include his pioneering the distribution model adopted years later by CDNs, his seminal work on Internet and web characterization, and his work on compositional certification of networked systems and software. He is the chair of the IEEE Computer Society TC on the Internet, served on the program committees and editorial boards of major conferences and journals in networking and real-time systems, and received distinguished service awards from both the ACM and the IEEE. He received the United Methodist Scholar Teacher Award at B.U. and the 2010 ACM SIGMETRICS Inaugural Test of Time Award (with M. Crovella).