Distributed Server Migration for Scalable Internet Service Deployment

GEORGIOS SMARAGDAKIS[†] georgios.smaragdakis@telekom.de
NIKOLAOS LAOUTARIS[‡] nikos@tid.es
KONSTANTINOS OIKONOMOU[¶] okon@ionio.gr
IOANNIS STAVRAKAKIS[§] ioannis@di.uoa.gr
AZER BESTAVROS[⋆] 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), 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?" We propose an innovative approach in which topology and demand information is limited to neighborhoods, or balls of small radius around selected facilities, whereas demand information is captured implicitly for the remaining (remote) clients outside these neighborhoods, by mapping them to clients on the edge of the neighborhood; the ball radius regulates the trade-off between scalability and performance. We develop a scalable, distributed approach that answers our key question through an iterative re-optimization of the location and the number of facilities within such balls. We show that even for small values of the radius (1 or 2), 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.**

***Index Terms*— Server migration, resource allocation, facility location, service deployment.**

I. INTRODUCTION

Motivation: Imagine a large-scale bandwidth/processing-intensive service such as the real-time distribution of software updates and patches [2], a distributed data-center [3], a cloud computing platform [4], [5], [6], *etc.* 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 recently observed flash-crowd phenomena. To deploy such services, decisions must be made 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* [7] 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 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 changes in the topology and the intensity of the demand for service. To address these limitations we propose distributed versions of UKM and UFL, which we use as means of constructing an automatic service deployment scheme.

A scalable approach to automatic 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 immediate surrounding area, which is defined by an *r-ball* of neighbors – nodes that are within a *radius* of r hops from the facility. The facility also monitors the demand that it receives from the nodes that have it as closest facility. It keeps an exact representation of demand from within its r-

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

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 CISE/CSR Award #0720604, ENG/EFRI Award #0735974, CISE/CNS Award #0524477, CISE/CNS Award #0520166, CNS/ITR Award #0205294, and CISE/EIA RI Award #0202067. Parts of this work appeared in the proceedings of the 26^{th} IEEE INFOCOM Conference [1].

ball, and an approximate representation for all the nodes on the *ring* of its r-ball (nodes outside the r-ball that receive service from it). In the latter case, the demand of nodes on the "*skin*" of the r-ball is increased proportionally to account for the aggregate demand that flows in from outside the r-ball through that node. When multiple r-balls intersect, they join to form more complex r-*shapes*. The observed 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 vicinity of the r-shape.

The trade-off between scalability and performance: Reducing the radius r decreases 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). This is a plus for scalability. On the other hand, reducing r harms the overall performance as compared to centralized solutions that consider the entire topological information. This is a minus for performance. We examine this trade-off experimentally using synthetic (Erdös-Rényi [8] and Barabási-Albert [9]) and real (AS-level [10]) topologies. We show that even for very small radii, *e.g.*, $r = 1$ (*i.e.*, facility migration is allowed only to first-hop neighbors), or $r = 2$ (*i.e.*, facility migration is allowed only up to second-hop neighbors), the performance of the distributed approach tracks closely that of the centralized one. Thus, increasing r much more is not necessary for performance, and might also be infeasible since even for relatively small r, the number of nodes contained in an r-shape increases very fast (owing to the small, typically $O(\log n)$, diameter of most networks, including the aforementioned ones).

A case study — large-scale timely distribution of customized software: Consider a large scale software update system, similar to that used for *Microsoft Windows Update*[1] . Such a system not only delivers terabytes of data to millions of users, but also it has to incorporate complex decision processes for customizing the delivered updates to the peculiarities of different clients [2] with respect to localization, previously-installed updates, compatibilities, and optional components, among others. This complex process goes beyond the dissemination of a single large file, where a peer-to-peer approach is an obvious solution [11]. Moreover, it is unlikely that software providers will be 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 Planet Lab, or the newest breed of Cloud Computing platforms (*e.g.*, Amazon EC2[2]). To that end, 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.[3] In the remainder of this section, we provide a mapping from the aforementioned software distribution service to our abstract UKM and UFL problems.

[1] http://update.microsoft.com

[2] http://aws.amazon.com/ec2

[3] It is important to note that the large-scale timely distribution of customized content is hardly unique to the dissemination of software updates, as it could be equally instrumental for "Virtual Product Placement" in live content as well as in video-on-demand services, to mention two examples.

Service providers, hosts, and clients: We envision the availability of a set of network hosts upon which specific functionalities may be installed and instantiated on demand. 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 Planet Lab[4] or that envisioned in GENI[5] , or a set of resources in a Cloud Computing platform (*e.g.*, an Amazon Machine Image (AMI) in the context of EC2).

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.[6] 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 *number* as well as the *location* of facilities used to deliver the service could be controlled in a distributed fashion. In particular, a GSH in W mode (*i.e.*, a facility) monitors the topology and the corresponding demand in its vicinity and is thus 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.[7] In particular, the hosting AS may charge the service provider for the assets 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 scenarios requires each GSH to be able to construct its surrounding AS-level topology up to a radius r. This can be achieved through standard topology discovery protocols[8] . Also, it requires a client to 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 status. Both of these could be achieved through standard resource discovery mechanisms like DNS re-direction [12], [13] (appropriate for application-level realizations of our distributed facility location approach) or proximity-based anycast routing [14] (appropriate for network layer realizations). Furthermore, we show in Section VII-C that the performance of our scheme degrades gracefully as re-direction becomes more imprecise.

Outline: The remainder of this paper is structured as follows. Section II provides a brief background on facility location. Section III presents our distributed facility location approach to automatic service deployment. Section IV examines analytically issues of convergence and accuracy due to approximate representation of the demand of nodes outside r-shapes. Section V evaluates the performance of our schemes on synthetic topologies. Section VI presents results on real-world (AS-

[4] http://www.planet-lab.org

[5] http://www.geni.net/GDD/GDD-06-08.pdf

[6] Switching to W might involve the transfer of executable and configuration files for the service from other GSHs or from the service provider.

[7] Notice that each AS (or a smaller organizational unit therein) is also a client of the service, with demand proportional to the aggregate number of requests originating from its end-users (*e.g.*, number of downloads of a service pack).

[8] http://www.caida.org/tools/measurement/skitter

level) topologies. Section VII looks at the effects of non-stationary demand and imperfect redirection. Section VIII presents previous related work. Section IX concludes the paper with a summary of findings and on-going work.

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 $k = |F|$ 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 $C(V, s, k)$:

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

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:

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

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 [15]. A variety of approximation algorithms have been developed under metric distance using a plethora of techniques, including rounding of linear programs [16], local search [17], [18], and primal-dual methods [19].

III. A Limited Horizon Approach to Distributed Facility Location

In this section we develop distributed versions of UKM and UFL by utilizing a natural limited horizon approach in which facilities have exact knowledge of the topology of their r-ball (surrounding topology up to r-hop neighbors), exact knowledge of the demand of each node in their r-ball and approximate knowledge of the aggregate demand from nodes on the ring surrounding their r-ball. 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.

A. Definitions

We make use of the following definitions, most of which are superscripted by m, the ordinal number of the current iteration. Let $F^{(m)} \subseteq V$ denote the set of facility nodes at the mth iteration. Let $V_i^{(m)}$ denote the *r-ball* of facility node v_i, *i.e.*, the set of nodes within radius r from v_i. Let $U_i^{(m)}$ denote the *ring* of facility node v_i, *i.e.*, the set of nodes not contained in $V_i^{(m)}$, but are being served by facility v_i, or equivalently, the nodes that have v_i as their closest facility. 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 $V^{(m)} = \bigcup_{v_i \in F^{(m)}} V_i^{(m)}$, $U^{(m)} = \bigcup_{v_i \in F^{(m)}} U_i^{(m)}$.

B. 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. It includes the following steps:

Initialization: Pick randomly an initial set $F^{(0)} \subseteq V$ of $k_0 = |F^{(0)}|$ nodes 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 a facility $v_i \in \mathcal{F}$ 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 [20] for such an example).
2) Test whether its r-ball can be merged with the r-balls of other nearby facilities. We say that two or more facilities can be merged (to actually mean that their r-balls can be merged), when their r-balls intersect, *i.e.*, when there exists at least one node that is within distance r from all the facilities . Let $J \subseteq F^{(m)}$ denote a set composed of v_i and the facilities that can be merged with it.[9] J induces an *r-shape* $G_J = (V_J, E_J)$, defined as 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 $O(n)$.
3) Re-optimize the r-shape G_J. If the original problem is UKM, solve the $|J|$-median within 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 — this can produce new locations as well as change the number of facilities (make it smaller or larger than $|J|$). In both cases the re-optimization is conducted by using a centralized algorithm.[10] The details regarding the optimization of r-shapes are given in Section III-C.

[9] The merging operation is recursive. When an initial r-ball merges with a second one, then additional facilities that can merge with the second one merge as well, and so on.

[10] The numerical results presented in Sections V and VI are obtained by using Integer Linear Programming (ILP) formulations [7] and local-search heuristics [18] for solving UKM and UFL within r-shapes. Since both perform very closely in all our experiments, 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 $F^{(m)}$.

C. 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 service 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 service 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.[11] *Re-optimizing the arrangement of facilities within an r-shape without considering the demand that flows-in from the ring would, therefore, amount 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* skin *of the r-shape*. The skin consists of nodes on the border (or edge) of the r-shape, *i.e.*, nodes of the r-shape that have direct links to nodes of the ring. This intermediate approach bridges the gap between absolute disregard for the 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 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: $s(v_k) = s(v_k) + s(v_j)$.

[11] Notice that r is intentionally kept small to limit the size of the individual re-optimizations.

IV. 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. The section goes beyond that to look closer at 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 III-B converges in a finite number of iterations. Then we show how to control the 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 Eq. (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 (Fig. 1 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:

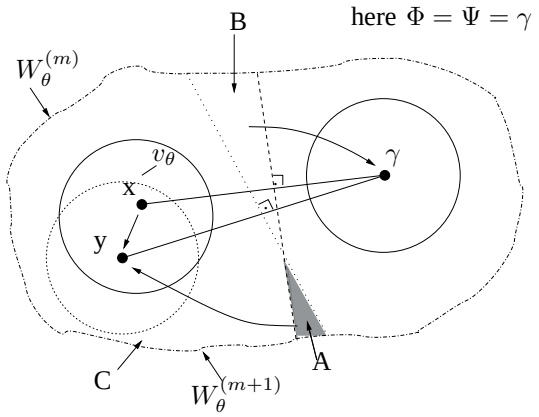

Fig. 1. Depiction of the move of a facility from X to Y and of the sets A, B, and C.

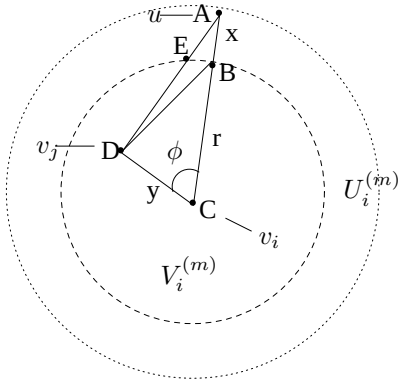

Fig. 2. Example of a possible facility movement from node v_i to node v_j with respect to a particular node $u \in U_i$.

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

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:

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

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

Using Eq. (5) in Eq. (3) we obtain:

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

Applying Eqs (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, the following proposition describes the convergence speed.

Proposition 2: The iterative local search approach for distributed facility location converges in $O(\log_{1+\alpha} n)$ 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:

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

Given the definition of the cost and the fact that node service demands ($s(v)$'s) are constants with respect to the size of the input (n), it is easy to see that $c^{(0)}$ can be upper bounded by $O(n^2)$ and c^* be lower bounded by $\Omega(n)$. This leads to an $O(n)$ upper bound for $\frac{c^{(0)}}{c^*}$. Substituting in Eq. (7) gives the claimed upper bound for the number of iterations. ■

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 (Defs 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 Fig. 2 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, Fig. 2), the skin node along the shortest path between u and the new location of the facility may change (node/point E in Fig. 2). 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:

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

The mapping error in Eq. (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. This is explored next.

The example depicted in Fig. 2 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,

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

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 Eq. (9), it is possible to derive an upper bound regarding the total mapping error $\Delta_i(r)$ for this particular environment. In Appendix I, we prove that,

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

where R is the radius of the particular domain W_i (for simplicity we assume that the domain is also a circle).

According to Eq. (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.

V. Synthetic Results on ER and BA Graphs

In this section we evaluate our distributed facility location approach on synthetic Erdös-Rényi (ER) [8] and Barabási-Albert (BA) [9] graphs generated using the BRITE generator [21]. For ER graphs, BRITE uses the Waxman model [22] 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 we also use incremental growth with $m = 2$. This parametrization 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).

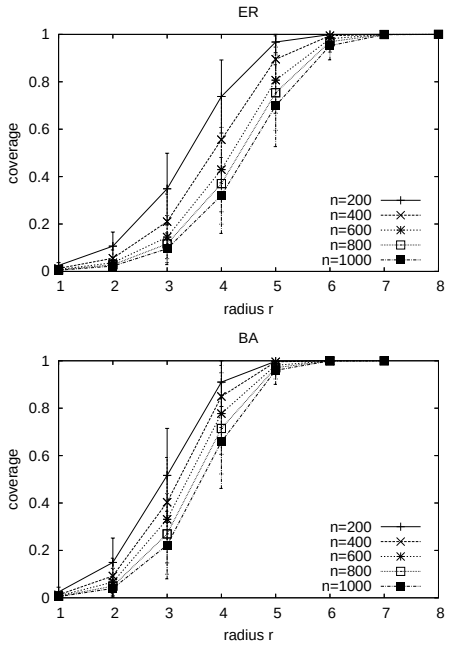

Fig. 3. Average coverage of a node for different size of ER and BA graphs.

A. Node Coverage with Radius r

Fig. 3 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% 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 larger networks exhibit longer shortest paths and diameters and also because BA graphs, owing to their highly skewed (power-law) degree distribution, possess shorter shortest paths and diameters than corresponding ER graphs of the same link density.

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. We fix the network size to $n = 400$ (matching measurement data on core Internet ASes that we use later

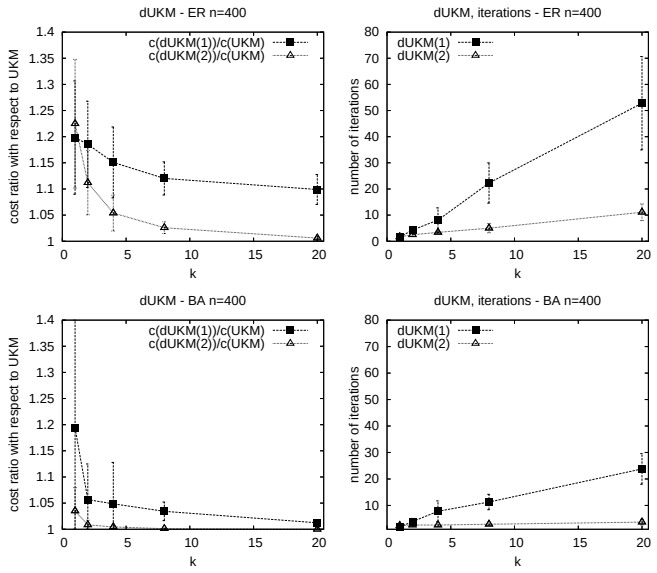

Fig. 4. The relative performance between dUKM(r) and UKM, and the number of iterations for the convergence of the former, for $r = 1$ and $r = 2$, and different facility densities $k/n = 0.1\%, 0.5\%, 1\%, 2\%$, and 5% under ER and BA graphs.

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 that 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 10 times to reduce the uncertainty due to the initial random placement of the k facilities.

The plots on the left-hand-side of Fig. 4 depict the cost of our dUKM(r) approach normalized over that of the centralized UKM, with the plot on top for ER graphs and the plot on the bottom for BA graphs. 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 call "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 Fig. 4 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[12] and, thus, fewer re-optimizations.

C. Performance of distributed UFL

In order to evaluate the performance of dUFL(r), we need to decide how to set the facility acquisition costs $f(v_j)$,

[12] 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.

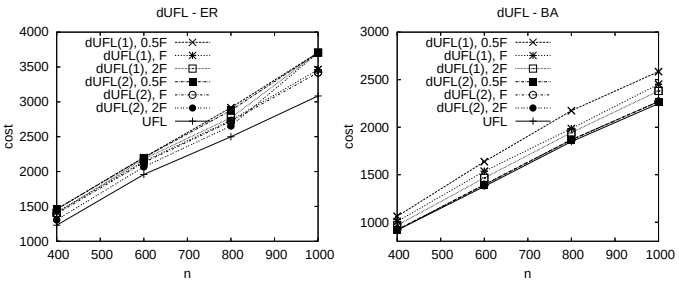

Fig. 5. Cost comparison between dUFL(r) and UFL, for $r = 1$ and $r = 2$, and different network sizes under ER and BA graphs and degree-based facility cost $f(v_j) = d(v_j)^{1+\alpha_G}$.

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 centralized one. 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 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.[13] In [23],[24] 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: $\hat{\alpha}_{k,m}^{(Hill)} = 1/\hat{\gamma}_{k,m}$, 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 Fig. 5 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 III-B. We use $k_0 = 0.5 \cdot F$, F, and $2 \cdot F$, where F denotes the

[13] 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.

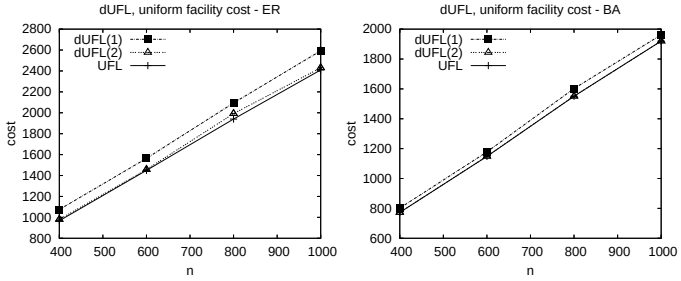

Fig. 6. Cost comparison between dUFL(r) and UFL, for $r = 1$ and $r = 2$, and different network sizes under ER and BA graphs and uniform facility cost.

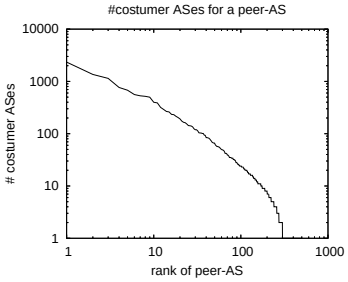

Fig. 7. Number of customer ASes for each peer-AS in decreasing order according to rank.

number of facilities opened by 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 V-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 Fig. 6.

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.

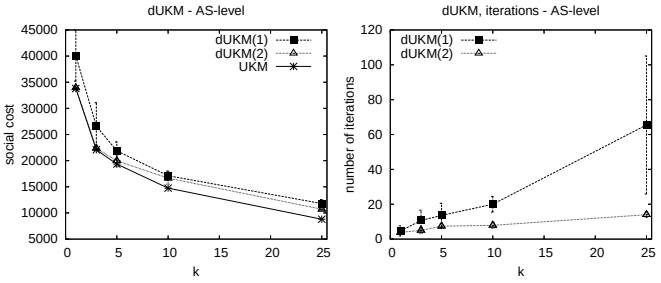

Fig. 8. The cost of dUKM(r) and UKM, and the number of iterations for the convergence of the former, for $r = 1$ and $r = 2$, and different facility densities $k/n = 0.1\%, 0.5\%, 1\%, 2\%$, and 5% under the AS graph.

VI. Results for Real AS-level Topologies

To further investigate the performance of our distributed approach, as well as 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.

A. Description of the AS-level Dataset

We use the relation-based AS map of the Internet from December 2001[14] obtained using the measurement methodology described in [10]. 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,[15] 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; we verified that this component contains all the 20 largest peer ASes reported in [10]. 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 aforementioned 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 Fig. 7 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 ASs that generate demand that follows the aforementioned power-law like profile. We seek to identify the peer ASes for building service facilities.

[14] http://www.cc.gatech.edu/∼mihail/ASdata.html

[15] There are smaller connected components (2-8 ASes) that are formed by small regional ISPs with peering relationships.

	cost ratio dUFL(1)/UFL		cost ratio dUFL(2)/UFL	
	mean	median	mean	median
degree-based	1.22	1.20	1.04	1.03
uniform	1.01	1.01	1.01	1.01

TABLE I
COST RATIO BETWEEN DUFL(r) AND UFL IN THE AS-LEVEL TOPOLOGY.

B. Distributed UKM on the AS-level Dataset

The plots on the left-hand-side of Fig. 8 show the cost of dUKM(1), dUKM(2), and the centralized UKM, under the AS-level graph. Clearly, even for small values of r, the performance of our distributed approaches track closely that of the centralized approach. 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 [20] which has also investigated and quantified implementation overhead in an Internet setting.

C. Distributed UFL on the AS-level Dataset

Table I presents the performance of dUFL 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).

VII. 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 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 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[16] 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.

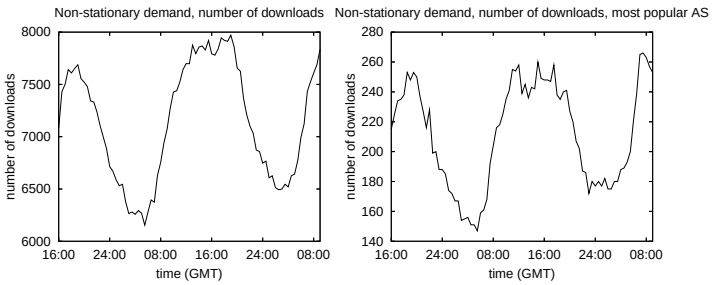

Fig. 9. The number of concurrent downloads from all ASes and from the most popular AS in the torrent of an on-line multi-player game at each measurement point.

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[17] . In Fig. 9 (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[18] 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 Fig. 9 (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 [25]. 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 Fig. 10 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 V-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,

[16] http://www.mininova.org

[17] Tracker is a server that maintains the set of distinct downloaders of a torrent. Upon a neighbor set request, the tracker replies with a random subset of the distinct downloaders set. We requested the size of the distinct downloaders set, and then we repeatedly requested for a new neighbor set until we reach the same number of distinct IPs.

[18] http://www.routeviews.org

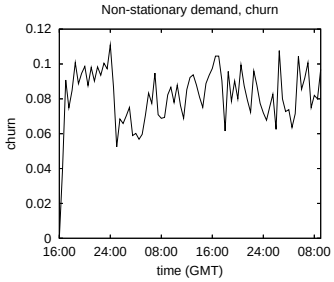

Fig. 10. Churn evolution in the AS-level in the torrent of a popular on-line multi-player game at each measurement point.

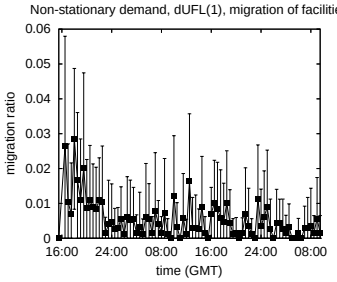

Fig. 11. Migration ratio of dUFL(1) in the torrent of a popular on-line multi-player game at each measurement point.

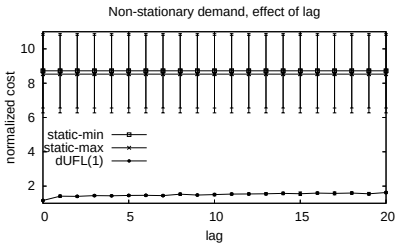

Fig. 13. Normalized cost of static-min, static-max and dUFL(1) with respect to the cost of UFL in the torrent of a popular on-line multi-player game under various levels of lag.

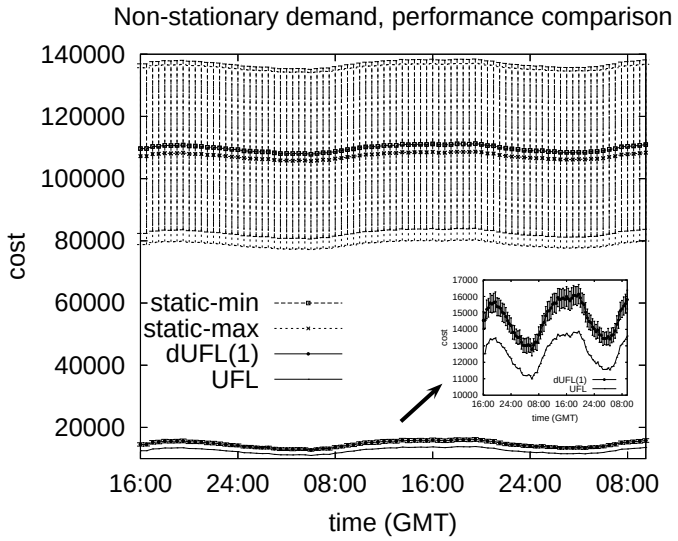

Fig. 12. Average cost of static-min, static-max, dUFL(1) and UFL in the torrent of a popular on-line multi-player game at each measurement point.

dUFL(1), 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 few bought 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 Fig. 12 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 Fig. 11 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. 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, etc. 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 [13]. 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 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 in Section IV-A). We solve this by stopping the iterative re-optimization if it reaches a certain high number of iterations.

In Fig. 13 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 (which means that clients of

facility i hear about i's migration after i+lag has completed its migration). 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.

VIII. Related Work

There is a huge literature on facility location theory. Initial results are surveyed in the book by Mirchandani and Francis [7]. A large number of subsequent works focused on developing centralized approximation algorithms [16], [17], [18], [19]. The authors of [26] 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 [27], [28]. The first mention of distributed facility location seems to have been from Jain and Vazirani [19] 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 [29]. This work, however, is mostly focused on deriving worst-case performance bounds for distributed facility location. It is based on primal-dual techniques that are amenable to such analysis, but which are too complicated for practical implementation purposes, as compared to our work. Furthermore, [29] does not include any experimental results or implementation guidelines of practical purposes. The online version of facility location, in which request arrive one at a time according to an arbitrary pattern, has been studied by Meyerson [30] 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. Oikonomou and Stavrakakis [31] 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 [32] (deployment of multicast reflectors), Rabinovich and Aggarwal [33] (deployment of mirrored web-content), Chambers et al. [34] (on-line multi-player network games), Cronin et al. [35] (constrained mirror placement), and Krishnan et al. [36] (cache placement). 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. [37] on systems aspects of a distributed shared platform for service deployment, and Loukopoulos et al. [38] on the overheads of updating replica placements under non-stationary demand.

IX. 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 by utilization 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.

References

[1] N. Laoutaris, G. Smaragdakis, K. Oikonomou, I. Stavrakakis, and A. Bestavros, "Distributed Placement of Service Facilities in Large-Scale Networks," in *Proceedings of IEEE INFOCOM '07*, Anchorage, AK, 2007.

[2] C. Gkantsidis, T. Karagiannis, P. Rodriguez, and M. Vojnovic, "Planet Scale Software Updates," in *Proc. of ACM SIGCOMM '06*, Pisa, Italy, 2006.

[3] N. Laoutaris, P. Rodriguez, and L. Massoulie, "Echos: Edge capacity hosting overlays of nano data centers," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 1, pp. 51–54, 2008.

[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, Feb 2009.

[5] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," *SIGOPS Operatings 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 *Proc. of ACM SOSP '07*, Stevenson, WA, 2007.

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

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

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

[10] L. Subramanian, S. Agarwal, J. Rexford, and R. H. Katz, "Characterizing the Internet Hierarchy from Multiple Vantage Points," in *Proc. of IEEE INFOCOM '02*, New York City, NY, 2002.

[11] D. Kostić, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh," in *Proc. of SOSP '03*, Bolton Landing, NY, USA, 2003.

[12] J. Pan, Y. T. Hou, and B. Li, "An Overview DNS-based Server Selection in Content Distribution Networks," *Computer Networks*, vol. 43, no. 6, 2003.

[13] N. Faber and R. Sundaram, "MOVARTO: Server Migration across Networks using Route Triangulation and DNS," in *Proc. of VMworld'07*, San Francisco, CA, 2007.

[14] V. Lenders, M. May, and B. Plattner, "Density-based vs. Proximity-based Anycast Routing for Mobile Networks," in *Proc. of IEEE INFOCOM '06*, Barcelona, Spain, 2006.

[15] 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.

[16] M. Charikar, S. Guha, D. B. Shmoys, and E. Tárdos, "A constant factor approximation algorithm for the k-median problem," in *Proc. of ACM STOC '99*, Atlanta, GA, 1999.

[17] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman, "Analysis of a Local Search Heuristic for Facility Location Problems," in *Proc. of ACM-SIAM SODA '98*, San Francisco, CA, 1998.

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

[19] K. Jain and V. V. Vazirani, "Primal-Dual Approximation Algorithms for Metric Facility Location and k-Median Problems," in *Proc of IEEE FOCS '99*, New York City, NY, 1999.

[20] M. Naor and U. Wieder, "Know Thy Neighbor's Neighbor: Better Routing for Skip-Graphs and Small Worlds." in *Proc. of IPTPS*, 2004.

[21] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: An Approach to Universal Topology Generation," in *Proc. of MASCOTS '01*, Cincinnati, OH, 2001.

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

[23] S. Jin and A. Bestavros, "Small-World Internet Topologies: Possible Causes and Implications on Scalability of End-System Multicast," CS Department, Boston University, Tech. Rep. BUCS-TR-2002-004, January 30 2002.

[24] ——, "Small-world characteristics of internet topologies and implications on multicast scaling," *Computer Networks*, vol. 50, no. 5, pp. 648–666, 2006.

[25] P. B. Godfrey, S. Shenker, and I. Stoica, "Minimizing Churn in Distributed Systems," in *Proc. of ACM SIGCOMM '06*, Pisa, Italy, 2006.

[26] C. W. Cameron, S. H. Low, and D. X. Wei, "High-density Model for Server Allocation and Placement," in *Proc. of ACM SIGMETRICS '02*, Marina Del Rey, California, 2002.

[27] M. Mahdian and M. Pal, "Universal Facility Location," in *Proc. of ESA '03*, Budapest, Hungary, 2003.

[28] N. Garg, R. Khandekar, and V. Pandit, "Improved Approximation for Universal Facility Location," in *Proc of ACM-SIAM SODA '05*, Vancouver, British Columbia, 2005.

[29] T. Moscibroda and R. Wattenhofer, "Facility Location: Distributed Approximation," in *Proc. of ACM PODC '05*, Las Vegas, NV, USA, 2005.

[30] A. Meyerson, "Online Facility Location," in *Proc. of FOCS '01*, Washington, DC, USA, 2001.

[31] K. Oikonomou and I. Stavrakakis, "Service Migration: The Tree Topology Case," in *Proc. of Med-Hoc-Net '06*, Lipari, Italy, 2006.

[32] L. Yamamoto and G. Leduc, "Autonomous Reflectors Over Active Networks: Towards Seamless Group Communication," *AISB*, vol. 1, no. 1, pp. 125–146, 2001.

[33] M. Rabinovich and A. Aggarwal, "RaDaR: A Scalable Architecture for a Global Web Hosting Service," in *Proc. of WWW '99*, Toronto, Canada, 1999.

[34] C. Chambers, W. chi Feng, W. chang Feng, and D. Saha, "A Geographic Redirection Service for On-line Games," in *Proc. of ACM MULTIMEDIA '03*, Berkeley, CA, USA, 2003.

[35] 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.

[36] P. Krishnan, D. Raz, and Y. Shavit, "The Cache Location Problem," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 568–581, 2000.

[37] D. Oppenheimer, B. Chun, D. Patterson, A. C. Snoeren, and A. Vahdat, "Service Placement in a Shared Wide-area Platform," in *Proc. of USENIX '06*, Boston, MA, 2006.

[38] T. Loukopoulos, P. Lampsas, and I. Ahmad, "Continuous Replica Placement Schemes in Distributed Systems," in *Proc. of ACM ICS '05*, Boston, MA, 2005.

APPENDIX I
DERIVATION OF AN UPPER BOUND FOR $\Delta_i(r)$

For the rest, a two-dimensional space is considered over which nodes are scattered in a uniform and continuous manner. The r-ball is considered as a circle with radius r and the entire domain also as a circle with radius R (see Fig. 2).

Suppose that a node $u \in U_i$ is served by its closest facility node v_i. This case is depicted in Fig. 2 where u is located at point A and the corresponding facility node v_i is located at point C. Note that line AC intersects with the periphery (skin) of the r-ball at a particular point denoted by B. Clearly, line AC corresponds to the shortest distance between points A and C (nodes u and v_i, respectively). Denoting as x the length of AB, $|AB|$ (the distance of node u from the skin of the r-ball) we can write $AC = x + r$. Line AC may be regarded as the path over which node u uses the resources of the facility located at node v_i.

Suppose that a node $u_j \in V_i$ is considered as a possible alternative facility location. Let D be the point denoting the location of v_j and let y denote the distance between node v_i and node v_j (i.e., the length of CD, $|CD|$). The mapping error, $\Delta_i(r, j, u) = |AB| + |BD| - |AD|$, is always positive since $|AB| + |BD| > |AD|$ (AB, BD and AD correspond to edges of the same triangle) when $A\hat{B}D \neq 0$ and $A\hat{B}D \neq \pi$. The mapping error becomes zero only in the exceptional cases where $A\hat{B}D = 0$ and $A\hat{B}D = \pi$ (corresponding to $\hat{\phi} = \pi$ and $\hat{\phi} = 0$, respectively, as concluded from Fig. 2).

Let $\Delta_i(r, j)$ be the summation of $\Delta_i(r, j, u)$, $\forall u \in U_i$. Since we have assumed the network area as a two-dimension continuous space, all nodes $u \in U_i$ correspond to the ring area U_i, depicted in Fig. 2. Consequently, $\Delta_i(r, j)$ is given by the following integral,

$$\Delta_i(r, j) = \int_{U_i} \Delta_i(r, j, u) du. \tag{11}$$

Let $\Delta_i(r)$ denote the total mapping error, or the summation of $\Delta_i(r, j)$ for all nodes $j \in V_i$. Therefore,

$$\Delta_i(r) = \int_{V_i} \Delta_i(r, j) dj. \tag{12}$$

In Appendix II we derive the following analytical expression for $\Delta_i(r, j, u)$ as a function of parameters x, y, r and $\hat{\phi}$:

$$\begin{aligned} \Delta_i(r, j, u) = \quad & x + \sqrt{r^2 + y^2 - 2yr\cos\hat{\phi}} \\ & - \sqrt{(x + r)^2 + y^2 - 2y(x + r)\cos\hat{\phi}}. \end{aligned} \tag{13}$$

$\Delta_i(r, j, u)$ as it is given by Eq. (13) is difficult to be analyzed. In addition, an analytical expression regarding $\Delta_i(r)$ is not easy to be derived since it is hard to obtain the corresponding integrals. Therefore, in the sequel we obtain an upper bound $\Delta_i(r)$ by using a simple upper bound for $\Delta_i(r, j, u)$ as explained below.

It is easy to see that $r^2 + y^2 - 2yr\cos\hat{\phi} \leq r^2 + y^2 + 2yr = (r + y)^2$, since $-1 \leq \cos\hat{\phi} \leq 1$. Also, $(x + r)^2 + y^2 - 2y(x + r)\cos\hat{\phi} \geq (x + r)^2 + y^2 - 2y(x + r) = (x + r - y)^2$ (note that $y \leq r$).

Based on Eq. (13), it is concluded that $\Delta_i(r, j, u) \leq x + \sqrt{(r + y)^2} - \sqrt{(x + r - y)^2} = x + r + y - x - r + y$. Therefore, $\Delta_i(r, j, u) \leq 2y$. Given that $y \leq r$,

$$\Delta_i(r, j, u) \leq 2r. \tag{14}$$

In order to derive $\Delta_i(r, j)$, according to Eq. (11), an analytical expression has to be derived for the integral $\int_{U_i} \Delta_i(r, j, u) du$. Note that $0 \leq \Delta_i(r, j, u) \leq 2r$, $\int_{U_i} \Delta_i(r, j, u) du \leq \int_{U_i} 2r du$ and R corresponds to the radius of the $U_i \cup V_i$ area (note that $R \geq r$). Eventually,

$$\Delta_i(r, j) \leq 2\pi r(R^2 - r^2), \tag{15}$$

since the area of the ring U_i is $\pi(R^2 - r^2)$.

In order to derive $\Delta_i(r)$, according to Eq. (12), an analytical expression has to be derived for the integral $\int_{V_i} \Delta_i(r, j) dj$.

Note that $0 \leq \Delta_i(r,j) \leq 2\pi r(R^2 - r^2)$ and $\int_{V_i} \Delta_i(r,j)dj \leq \int_{V_i} 2\pi r(R^2 - r^2)dj$. Eventually,

$$\Delta_i(r) \leq 2\pi^2 r^3 (R^2 - r^2), \tag{16}$$

since the r-ball area is πr^2.

APPENDIX II
DERIVATION OF AN ANALYTICAL EXPRESSION FOR $\Delta_i(r,j,u)$

When one of the angles of a triangle ($\hat{\phi}$) is known as well as the length of both adjacent edges (r and y), then the length of the third edge is possible to be derived as a function of $\hat{\phi}, r, y$. Two different cases may be distinguished with respect to the triangle's particular form, as depicted in Fig. 14.

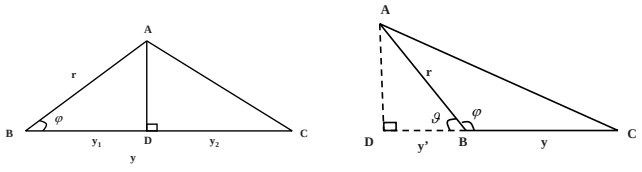

Fig. 14. The two distinguished cases studied to derive the analytical expression for $\Delta_i(r,j,u)$.

For the case depicted in Fig. 14.a, $\cos\hat{\phi} = \frac{y_1}{r}$. Since $y = y_1 + y_2$, $y_2 = y - y_1 = y - r\cos\hat{\phi}$. Furthermore, $\sin\hat{\phi} = \frac{|AD|}{r}$ and $|AD| = r\sin\hat{\phi}$. It holds that $|AC|^2 = |AD|^2 + y_2^2$, or $|AC| = \sqrt{|AD|^2 + y_2^2}$, or $|AC| = \sqrt{r^2\sin^2\hat{\phi} + y^2 + r^2\cos^2\hat{\phi} - 2yr\cos\hat{\phi}}$. Eventually,

$$|AC| = \sqrt{r^2 + y^2 - 2yr\cos\hat{\phi}}. \tag{17}$$

The same result is also derived for the case depicted in Fig. 14.b, where $\hat{\theta} = \pi - \hat{\phi}$. For this case, $|AC| = \sqrt{|AD|^2 + (y + y')^2}$. However, $|AD| = r\sin\hat{\theta}$ and $y' = r\cos\hat{\theta}$. Since, $\sin\hat{\theta} = \sin\hat{\phi}$ and $\cos\hat{\theta} = -\cos\hat{\phi}$, $|AD| = r\sin\hat{\phi}$ and $y' = -r\cos\hat{\phi}$. Eventually, Eq. (17) holds for this case as well.

Georgios Smaragdakis received the Diploma in electronic and computer engineering from the Technical University of Crete, Chania, Greece, the Ph.D. degree in computer science from Boston University, MA, and he interned at Telefónica Research, Barcelona, Spain.

He is a Senior Research Scientist at Deutsche Telekom Laboratories and the Technical University of Berlin, Berlin, Germany. His research interests include the design and analysis of computer networks and content distribution systems with main applications in overlay network creation and maintenance, service deployment, network storage, distributed caching, and network security.

Nikolaos Laoutaris is a 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 with emphasis on content distribution, overlay networks, P2P, and multimedia communications.

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 currently holds an academic position as Lecturer 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; Assist. Prof. in CSEE, University of Vermont (USA), 1988-1994; Assoc. Prof. of ECE, Northeastern University, Boston (USA), 1994-1999; Assoc. Prof. of Informatics and Telecommunications, University of Athens (Greece), 1999-2002 and Prof. 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, including: organizer of the 1999 IFIP WG6.3 workshop, the COST-NSF NeXtworking'03, the Workshop on Autonomic Communications (WAC2005); co-organizer of the 1996 ITC Mini-Seminar, the IEEE Autonomic Opportunistic Communications (AOC'07 &'08); technical program co-chair for the IFIP Networking'00, EWC'04, IFIP WiOpt'05, COST-NSF NeXtworking'07; general co-Chair for Networking'2002, IFIP MedHocNet'07. 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 PhD degree in computer science from Harvard University in 1992. He is a professor in and former chairman of the Computer Science Department at Boston University. His research interests are in networking and in real-time systems. Prof. Bestavros' research contributions include his pioneering of the push content distribution model adopted years later by CDNs, his seminal work on traffic characterization and reference locality modeling, his work on various network transport, caching, and streaming media delivery protocols, his work on e2e inference of network caricatures, his work on identifying and countering adversarial exploits of system dynamics, his work on game-theoretic approaches to overlay and P2P networking applications, his generalization of classical rate-monotonic analysis to accommodate uncertainties in resource availability/usage, his use of redundancy-injecting codes for timely access to periodic broadcasts, his work on verification of network protocol compositions, including the identification of deadlock-prone arrangements of HTTP agents, and his work on virtualization services and programming environments for embedded sensor networks. His work has culminated so far in 13 PhD theses, more than 80 masters and undergraduate student projects, five US patents, two startup companies, and over 3,000 citations. His research has been funded by more than $15 million of government and industry grants. He has served as the general chair, program committee chair, officer, or PC member of most major conferences in networking and in real-time systems. Prof. Bestavros has received distinguished service awards from both the ACM and the IEEE, and is a senior member of both the IEEE and ACM. He is the Chair of the IEEE Computer Society Technical Committee on the Internet and a distinguished speaker of the IEEE.