Edge Provisioning with Flexible Server Placement

Hao Yin, Xu Zhang, Hongqiang Harry Liu, Yan Luo, *Member, IEEE*, Chen Tian, Shuoyao Zhao, and Feng Li

Abstract—We present Tentacle, a decision support framework to provision edge servers for online services providers (OSPs). Tentacle takes advantage of the increasingly flexible edge server placement, which is enabled by new technologies such as edge computing platforms, cloudlets and network function virtualization, to optimize the overall performance and cost of edge infrastructures. The key difference between Tentacle and traditional server placement approaches lies on that Tentacle can discover proper unforeseen edge locations which significantly improve the efficiency and reduce the cost of edge provisioning. We show how Tentacle effectively identifies promising edge locations which are close to a collection of users merely with inaccurate network distance estimation methods, e.g., geographic coordinate (GC) and network coordinate systems (NC). We also show how Tentacle comprehensively considers various pragmatic concerns in edge provisioning, such as traffic limits by law or ISP policy, edge site deployment and resource usage cost, over-provisioning for fault tolerance, etc., with a simple optimization model. We simulate Tentacle using real network data at global and county-wide scales. Measurement-driven simulations show that with a given cost budget Tentacle can improve user performance by around 10-45 percent at global scale networks and 15-35 percent at a country-wide scale network.

Index Terms—Edge provisioning, optimization, decoupling, unforeseen locations

1 Introduction

The latency experienced by end users to access online services is essential for keeping customers and the reputation of the services' brand. For example, Bing found that a 2 second slowdown changed queries/user by -1.8 percent and revenue/user by -4.3 percent [1].

In this respect, *edge servers*, the computation, network and storage resources deployed close to end users, have become an indispensable infrastructure to guarantee the low user-facing latency in online services nowadays. This is because they can cache content near the users' accesses, boost users' TCP throughput and recovery speed with short round trip time (RTT), and avoid the peering point congestion and network failures in core networks.

To ensure that users can be served by edge servers close by with sufficient capacity, an online service provider (OSP) with distributed, evolving user crowds must also provision its edge infrastructure regularly. Despite that different applications might be of various specific concerns on the edge infrastructure, there are two basic requirements to be met generally by all edge provisions. The first one is the good *proximity* between the edge servers and the users they serve; and the second is the sufficient *capacity* on the edge servers. Nevertheless, due to the inherently widely-distributed architecture of edge infrastructures, people are facing many pragmatic complexities in edge provisioning:

Many Deployment Options. The first source of complexity, abundant deployment options, is both an opportunity and a challenge. OSPs today have a broad spectrum of provisioning choices. For instance, they can deploy new servers in known locations, e.g., Internet eXchange Points (IXPs) and data centers in clouds, or upgrade capacity in their existing deployment footprint [2]. Recent developments of edge computing, cloudlets, CDN-i and NFV (network functions virtualization) offer even greater degrees of freedom to place edge servers in new locations. In a given location, an OSP also has practical choices in rolling out its own server deployments, peering their datacenters with local ISPs with their own cables [3], or simply renting third-party servers in colocation facilities. Such flexibility in server placement results in not only a great opportunity but also a huge complexity for optimizing the performance and cost in edge provisions.

Diverse Requirements. The second source of complexity is simply that there are too many practical considerations which edge operators have to take into account in this process. The simplest tradeoff is between user expectations on performance and deployment costs; e.g., ideally we need edge servers with short delay and high bandwidth to all users but this also entails a large number of servers and high costs. Also, the edge infrastructure should tolerate the edge site outages, therefore over-provisioning is also

- *H. Yin, X. Zhang, S. Zhao and F. Li are with the Research Institute of Information Technology, Tsinghua University, Beijing 100084, China. E-mail: h-yin@mail.tsinghua.edu.cn, {xzhang12, lifeng13}@mails.tsinghua.edu.cn, zsyintl@126.com.*
- *H.H. Liu is with the Mobility and Networking Research Group, Microsoft Research, Redmond, WA 98052. E-mail: harliu@microsoft.com.*
- *Y. Luo is with the Department of Electrical and Computer Engineering, University of Massachusetts Lowell, Lowell, MA 01852. E-mail: yan_luo@uml.edu.*
- *C. Tian is with State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu 210093, China. E-mail: tianchen@nju.edu.cn.*

Manuscript received 6 Dec. 2015; revised 18 July 2016; accepted 16 Aug. 2016. Date of publication 31 Aug. 2016; date of current version 15 Mar. 2017. Recommended for acceptance by A. Benoit.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2016.2604803

important. In addition, nowadays ISPs also provide low-cost and high-performance server sites in their own networks, and the usage of these sites is restricted to only the users in the specific domains [4].

To the best of our knowledge, there are no previous known studies on how to systematically inform these different tradeoffs and provision the edge with flexible server placement as commercial edge infrastructure providers have not yet provided any details on their solutions. The most related work is the canonical "server placement" formulations from the literature (e.g., [5], [6], [7], [8], [9], [10], [11]) which simply model the provisioning problem as selecting k locations from n candidate locations. However, such simplified formulations are not sufficiently expressive to meet the operational requirements. First, they cannot capture the range of new deployment opportunities available to the edge operators today; i.e., we may indeed have new possible server locations that fall outside the fixed set of n locations. Second, they cannot comprehensively cover the constraints on costs, quality of user experience, and traffic properties.

In this paper, we conduct a pioneering and exploratory study on how OSPs could provision edge infrastructures with flexible server placements while considering various pragmatic requirements. With analytical and experimental reasoning and comparisons of multiple design choices, we develop Tentacle, a decision support framework that comprehensively considers cost budgets, performance requirements, traffic constraints and existing server deployments, and shows the remarkable benefits by placing the edge servers at the right locations with the right deployment methods and the appropriate server capacity, and serving the right users.

The first challenge to design Tentacle is how to accommodate various practical considerations when selecting server locations and capacity. Enumerating all possible locations and deployment options to find a solution satisfying all requirements is usually intractable or infeasible due to the huge number of candidate edge locations and the unforeseen locations. We design an effective heuristics to address this challenge with the key insight of decoupling the limitations on realistic edge location selection from other concerns. We first assume that we could place servers anywhere, and find out the ideal edge locations which respect the constraints and achieve the goals of the provision. Then, we select or discover real server sites which can best approach the ideal locations. Finally, we model all provisioning decisions with a single integer linear programing (ILP) formulation.

The second challenge to design Tentacle is how to discover ideal and actual server locations which ensure the proximity between users and edge servers. Since there could be numerous known and even more unforeseen candidate locations, it is extremely difficult and expensive to directly measure the network latencies between all optional edge locations and users. Geographic coordinate (GC) and network coordinate (NC) [12], [13], [14], [15], [16], [17], [18] provide a lightweight approach to estimate the network delays. Nevertheless, the predictions from either GC or NC have remarkable inaccuracy [19], [20], [21], which can significantly undermine the effectiveness of the server location search. To overcome this challenge, we performed a global-scale measurement and evaluation on both GC and NC. Our key observation is that despite the fact that the absolute values of network delays predicted by GC and NC are largely inaccurate, the relative ranking of network delays given by NC is highly trustworthy. In other words, if a pair of network hosts have a smaller delay than another pair in NC, there is a large chance that this pair's real network delay is also smaller than that of the other pair. Thus, Tentacle merely leverages the delay ranking provided by NC to search the ideal edge locations.

We simulate Tentacle using realistic user bases both in a global-scale network and a country-wide one. The provision plans derived with Tentacle outperform those with other alternative approaches in both deployment costs and quality of user experiences (QoE). For a global scale case, Tentacle can improve the QoE by about 10-45 percent within a given cost budget and about 15-35 percent improvement for a country-wide case. In both scenarios, Tentacle can provision edges within a limited budget to satisfy QoE even when other alternative approaches fail to do so.

Our contributions are summarized as follows:

- We present Tentacle, the first framework to achieve a systematic and comprehensive edge provisioning for online service providers with various practical considerations. The novelty lies in the decoupling of ideal edge location identification and mapping ideal locations to actual locations in physical networks, thus discovering unforeseen better edge sites in the network.
- We develop a novel method to take advantage of the latency ranking, which is much less sensitive to delay prediction errors in NC. This method achieves more effective edge provisioning than directly using the inaccurate delay predictions by NC.
- We extensively simulate Tentacle using both global-scale and country-wide, realistic network data, and the results reveal the promising performance of Tentacle over existing approaches. We also explore the design choices of coordinate systems and clustering algorithms in Tentacle and how Tentacle can achieve fault tolerance in the provision plans.

2 Background and Motivation

2.1 Growing Demands on Edge Infrastructure

Internet users are widely distributed in thousands of heterogeneous access networks. Due to some fundamental challenges, e.g., congestion at peering points between two networks, poor TCP performance caused by long RTTs, etc., it is far from satisfactory to host online services and directly serve users with only dozens of datacenters.

Edge servers are placed in strategic locations which are close (in networks) to end users. They cache contents close to users' accesses and terminate TCP or HTTP connections. The low network latency between an edge server and a user can typically ensure a high ramp-up and recovery speed of TCP throughput and a low server response delay. Therefore, edge servers have been an indispensable infrastructure of services with high bandwidth and/or low latency requirements, such as cache servers in CDNs and frontend proxies in search engines and online games.

The demands on edge infrastructure have been reaching an era of rapidly growth. One driving force of this growth is

the business expansions of existing online services and the migrations of interactive applications from offline to online. For instance, it has been reported that, in recent three years, CDN providers like Akamai, LimeLight, Level 3 and MaxCDN all made large investments to extend their edge footprints to emerging markets like Latin America, Australia, Africa and Asia [22]. For another example, to support Office Online, Microsoft has deployed edge servers in approximately 100 locations around the world in recent years [23]. Another potentially bigger driving force of the growth is the computation off-loading demands from devices to edge servers, *a.k.a.* edge computing. Many leading tech companies, e.g., Intel, Qualcomm, IBM, Cisco and Microsoft, have mentioned such visions in their white papers [24], [25], [26], [27], [28] that a huge growth in the network's edge will come in the near future due to the developments of smart phone applications, wearable devices, Internet of things and Big Data.

For maintaining the proximity between edge servers and users, edge provisioning will be frequently performed by online services whose user crowds are growing and evolving.

2.2 Flexible Edge Server Placement

In this paper, as suggested by Akamai [3], we define an *edge location* as a tuple of $\langle city, AS(\text{Autonomous System})\rangle$. An edge location could include multiple server clusters and we call each one of them as an *edge site*. Edge server placement decides how to select edge sites.

At present, there exists a broad spectrum in how to select edge sites and expand server capacities. Besides Internet Exchange Points, datacenters, and server clusters in ISP server sites, OSPs can also deploy servers quickly to thousands of edge locations with the edge computing services offered by many traditional CDN providers such as Akamai, EdgeCast and LimeLight. Additionally, an increasing number of ISPs have opened their edge locations to other service providers with cloudlets, network functions virtualization (NFV) and CDN federations (CDNi) [29]. Moreover, building a new server site in a selected edge location is also an option, and many companies, including e.g., Amazon, Google and Microsoft etc., are continuously expanding their own edge infrastructure by doing so. Sometimes they even lay or buy private fibers to connect some local networks with their datacenters [3].

The flexibility in edge server placement does not only offer a chance to make edge servers closer to users, but also create a huge opportunity for edge operators to optimize the performance and the cost in edge provisioning.

2.3 Opportunities and Complexities

There are two major sources of complexity to realize a systematic edge provision. First, due to the flexibility in server placement, there are always both known and unforeseen edge locations in edge provisioning. Traditional solutions, where servers are selected from a fixed pool of candidates, cannot take advantage of this flexibility. An effective edge provision should have the ability to discover new edge locations which can significantly improve the edge infrastructure. For example, in Fig. 1, $S1$ and $S2$ are two pre-known edge locations and $S3$ and $S4$ are two unforeseen edge locations. If we only provision with $S1$ and $S2$, the result could

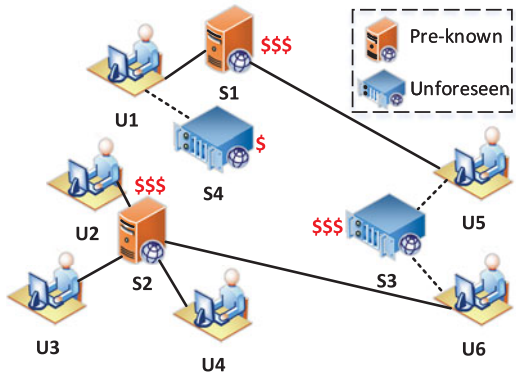


Fig. 1. Discovering unforeseen edge locations can potentially improve performance and save cost.

be using $S1$ to serve $U1, 5$ and use $S2$ to serve $U2, 3, 4, 6$. However, $S3$ can serve $U5, 6$ with better proximity and the same cost as $S1$ and $S2$, and $S4$ can serve $U1$ with lower cost and the same proximity as $S1$. An edge location selection with improvement in proximity and reduction in cost should be $S2, 3, 4$. Lacking the ability to discover $S3$ or $S4$ means missing the potential improvement in proximity and the opportunity of saving cost.

However, since discovering the unforeseen edge sites has to involve human efforts to investigate and interact with other business organizations, it is intractable to enumerate all tens of thousands of edge locations and potentially hundreds of thousands of edge sites. Also, because the costs, edge site performance, and feasible edge locations are changing all the time with the network developments, discovering proper edge sites is not a one-shot either.

In addition, there are many pragmatic trade-offs in edge provisioning. Besides achieving the goal of the proximity between users and edge servers, edge operators also consider the cost budgets, the capacity of edge sites, the provisioning for edge site outages, and the constraints on which users an edge location can serve according to the regional laws or the ISPs' policies. These considerations usually couple tightly together, resulting in a huge search space. Edge operators should comprehensively consider these trade-offs and explore the best provisioning plan under multiple constraints.

3 Overview

In this section, we model the basic problem in edge provisioning and provide an overview of our methods to solve the problem.

3.1 The Basic Problem in Edge Provisioning

There are two basic requirements in edge provisioning. The first is the proximity between users and edge servers and the second is the sufficient capacity each user can obtain from nearby servers. Formally, we define the basic problem in edge provisioning as follows:

Edge Provisioning Problem. Let D_{max} be the maximum tolerated distance between users and edge servers. Given a user set $U = \{u\}$,[1] we find a set of edge sites $S = \{s\}$. Denote

1. Backend servers are treated as special users in U.

S_u as the candidate edge site pool of user u, and we require that $S_u = \{s | s \in S, d_{u,s} \leq D_{max}\}$[2] where $d_{u,s}$ is the distance between u and s. We have the following two requirements on the server set S_u:

- *Proximity*: for all (or p percent) of the users, there are at least m sites in S_u. Usually $m > 1$ for fault tolerance under outages in individual edge sites.
- *Capacity*: there is always (or with probability q percent) at least one server in S_u having the capacity to serve u.

This problem is one variant of "facility location problem" (FLP) with server capacity constraints. FLP is proved to be NP-hard [30], and it is widely studied in operation research field [31], [32], [33], [34]. However, we cannot directly apply the existing solutions of FLP in the edge provisioning. This is because (1) all available solutions to FLP assume that the cost or distance between users and servers are known or easy to obtain, which is usually not true in edge provisioning scenarios; (2) existing solutions to FLP can only choose a server subset from a fixed pool of pre-known candidates, lacking the ability to discover unforeseen edge locations, which means missing the potential improvement in proximity and the opportunity of saving cost; and (3) in practice we have additional concerns on, for instance, deployment cost, fault tolerance, and traffic limitations due to the regional laws or ISP policies. Therefore, we design our own heuristics to solve the edge provisioning problem with the pragmatic considerations, and.

3.2 Modeling Edge Provisioning Problem

The primary goal of Tentacle is to find proper edge sites and determine the capacity of them to achieve the proximity and capacity goals simultaneously with the lowest cost. However, we will never know the edge capacity and cost we can have until we locate the unforeseen edge sites. Therefore, Tentacle separates the provisions of edge locations (for proximity) from edge capacity planning. It first performs an edge localization step to find proper edge locations, which are the ideal places one should deploy edge servers, but might not be feasible due to physical limitations. Tentacle then searches available edge sites in or close to the identified ideal edge locations.

Edge Localization. Taking advantage from the flexible server placement, our key idea in searching the good edge locations is that we first assume we can deploy edge sites anywhere and find the ideal edge locations. Then we use both pre-known and unforeseen edge sites surrounding the ideal edge locations to approach them.

To quantify the edge locations, as the first step, we need to map all real network hosts to points in a virtual coordinate space $\mathcal{S}$ and the distance between two points could be computed by their coordinates. In practice, S can be a metric space like geographical (longitude and latitude) coordinates, or network coordinates [13], [17], [18].

Suppose we consider to find k edge locations $L = \{l_1, \ldots, l_k\}$ to serve the given user set $U = \{u\}$. l is used interchangeably for both edge locations and its corresponding point in $\mathcal{S}$ and so does u. The distance from a user u to the closest edge location in L is

$$\forall u \in U : d_{u,L} = \min_{\forall l \in L} \|u - l\|. \quad (1)$$

Our objective is to find the coordinates of the locations in L, which minimizes the distance from an arbitrary user u to L

$$d_{max} = \min \max_{\forall u \in U} d_{u,L}. \quad (2)$$

When k is large enough, the objective d_{max} will finally be smaller than D_{max} because in the extreme case when $k = |U|$, there will be at least one server to place side-by-side with each user. Our goal is to find a modestly large k to achieve $d_{max} \leq D_{max}$ which means we have found k edge locations which can meet the distance requirement to users. The proximity requirement can be met if we eventually find or build at least m edge sites in or close to the edge locations in the solution L.

When $k = 1$, Eqn. (1) is equivalent to

$$\forall u \in U : d_{u,L} = \|u - l\|, \quad (3)$$

so that Eqns. (2) and (3) form a convex optimization problem, which could be efficiently solved [35]. Unfortunately, however, when $k > 1$, the problem becomes a "minimax facility location" problem which is NP-hard again [31].

Tentacle takes a simple two-step heuristics to solve the problem when $k > 1$. It first clusters the users, and makes sure that the users in the same cluster are close enough. For instance, in the user clustering if Tentacle can ensure that the distances between two users from the same cluster is no more than $2 \times D_{max}$, it is most likely to find a single server position whose distance to the users in the cluster is less than D_{max}. Then it solves Eqns. (2) and (3) to find a *single* edge location for each cluster independently, which is the ideal edge location for the cluster.

There are two questions behind the edge localization model of Tentacle. The first question is how to select the coordinate space $\mathcal{S}$. Almost all previous work [13], [17], [18] focused on evaluating and improving the relative prediction error of GC and NC in network distance prediction. Nevertheless, we find that the correctness and accuracy of the user clustering and the solution of Eqns. (2) and (3) are much more dependent on the *consistency* of $\mathcal{S}$ in distance ranking than how close their predictions are to real network distances. We define the *consistency* of a network distance estimation space $\mathcal{S}$ as following:

Definition 3.1 (Distance Consistency). *Given a pair of host pairs $((A, B), (C, D))$, denote the distance between A and B in the physical network as $d^*_{A,B}$ and in $\mathcal{S}$ as $d_{A,B}$, similar for host pair (C, D). If $\mathcal{S}$ can guarantee that*

$$d_{C,D} \leq d_{A,B} \Longleftrightarrow d^*_{C,D} \leq d^*_{A,B},$$

then the pair of host pairs $((A, B), (C, D))$ is consistent between $\mathcal{S}$ and the physical networks.

Definition 3.2 ($\mathcal{S}$'s Consistency). *$\mathcal{S}$'s consistency c is defined as the probability that a randomly-given pair of host pairs, e.g.,*

2. In this paper we assume a homogeneous D_{max} among users and backend servers. It is simple for our model to be extended to heterogeneous D_{max}.

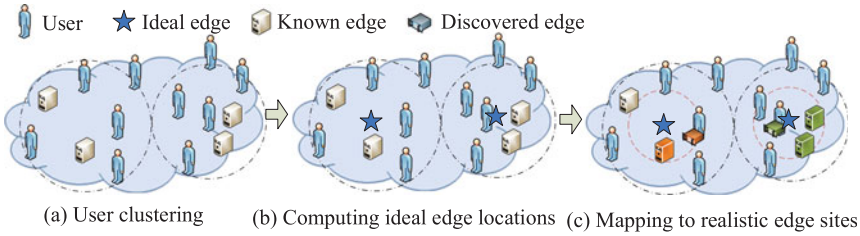


Fig. 2. The entire process of edge localization in Tentacle.

$((A, B), (C, D))$, *is consistent between* $\mathcal{S}$ *and the physical networks.*

If $c = 100$ percent, we say $\mathcal{S}$ is fully consistent. We will use analytic (Section 4.1.1) and experimental (Section 6.4) results to demonstrate that choosing $\mathcal{S}$ with high consistency can guarantee a high quality of edge localization.

The second question is how to discover unforeseen edge sites with an ideal edge location in $\mathcal{S}$. Since we define an edge location as a tuple of ⟨ city, AS ⟩, such reversing mapping is not straightforward no matter Tentacle adopts NC or GC. The approach Tentacle leverages to solve this problem is to learn physical edge locations from the users. After obtaining an ideal edge location (l), Tentacle will check the physical locations of the users which are most close to l in $\mathcal{S}$, and try to discover unforeseen edge sites locations of these selected users. Note that if $\mathcal{S}$ is highly consistent, these users should also be the closest to l in physical networks.

Fig. 2 illustrates the whole process of edge localization in Tentacle. Finally, Tentacle obtains a collection of candidate edge sites, including both pre-known and discovered unforeseen. Next, Tentacle will take the users who have enough qualified candidate servers ($|S_u| \geq m$) into the capacity provisioning process.

Edge Capacity Provisioning. The central question in capacity provisioning is how to decide the capacity reservation in each edge server. In this paper, we assume that for an OSP, only one type of server resource is bottleneck, so that we simply use *the number of concurrent users an edge server can support* as the metric of server capacity.

There are multiple factors to consider in edge capacity provisioning. First of all, the edge servers should have sufficient capacity to serve their users, even if under the cases that some edge sites lose their capacity due to outage. Second, the capacity reservation significantly impacts the cost of edge infrastructure. There can be a one-time investment to use an edge site, and there will also be long-term bandwidth cost to satisfy users' requirements.

Tentacle formulates all of the considerations into a single integer linear programming model (see Section 4.2) which can find an optimized capacity provisioning plan. Especially, inspired from the concept and the formulation of forward fault correction (FFC) in [36], Tentacle uses a similar formulation and methodology to guarantee that each user u can get enough capacity from S_u even if there are up to n ($n < |S_u|$) edge sites in S_u fails simultaneously. The details of the formulation will be presented in Section 4.2.1.

4 Design

This section elaborates how Tentacle performs edge provisioning, starting from only the IP addresses of users and ending with a comprehensive provisioning plan. The overall work flow is shown in Fig. 3.

4.1 Edge Localization

As shown in Fig. 3, Tentacle takes four steps in the phase of edge localization. The goal of this phase is to discover effective edge locations and obtain the set of candidate edge locations (S_u) which satisfy the proximity requirement for each user u.

4.1.1 Mapping Users to a Metric Space

There are two advantages to map users in physical networks into a metric space $\mathcal{S}$. First of all, the distances among users and edge sites could be estimated with a lightweight way. Because there could be millions of users which are distributed globally, it is difficult and expensive to directly measure the network distances. Second, the edge locations can be quantified, so that it can be much easier to compute the ideal edge locations.

GC and NC are two metric spaces that are widely used in the state of the art. However, according to previous work and our measurements, both of them have remarkable prediction errors. For instance, with a one-month-long latency measurement from 360 PlanetLab nodes to about 2,272,000 independent/24 subnets widely distributed in the world, we established a NC-space based on GNP. We also leverage commercial IP-GeoLocation services to build a Geo-space for the same subnet set. We randomly select 100,000 host pairs, that is, (A_i, B_i), where A_i is randomly sampled from the Planetlab nodes, while B_i is randomly sampled from the subnets, and $1 \leq i \leq 100,000$. Fig. 4a shows the relative estimation errors [37] from the global NC-space and Geo-space. Despite that NC-space is generally better than Geo-space in latency prediction, it still has about 10 percent chance to get 50 percent or larger prediction error. In addition, Fig. 4b shows the relative prediction errors of NC-space and Geo-space from a country-wide measurement with 38 vantage

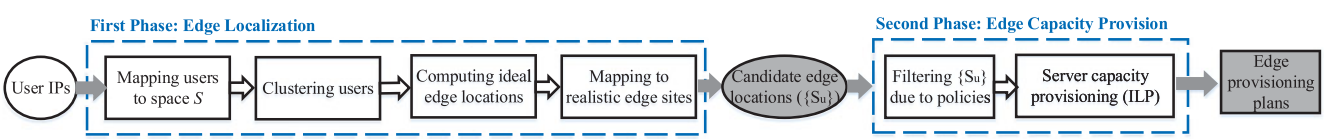


Fig. 3. The overall working flow of Tentacle.

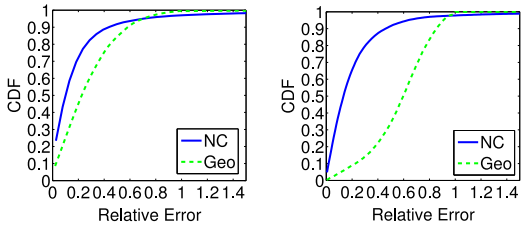


Fig. 4. The comparison for the relative estimation error between GC and NC in different network scales.

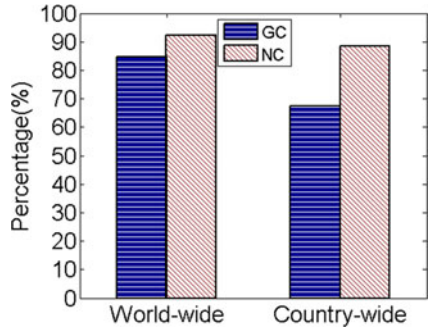


Fig. 5. The comparison for the consistent extent between GC and NC in different network scales.

points and 140,000 independent/24 subnets, in which we can draw a similar conclusion for NC-space, yet Geo-space performs even worse.

Nevertheless, the central question in the selection of metric space is: how does the inaccuracy of a metric space impact the quality of edge localization (e.g., the ratio proximity violations in users)? In the development of Tentacle, we observe that what matters is how *consistent* in distance ranking $\mathcal{S}$ is, rather than how close its prediction is to the real network distance. Theoretically, we have the following lemma:

Lemma 4.1. *If $\mathcal{S}$ is fully consistent, and the ideal edge location l given by the edge localization model of Eqns. (2) and (3) has a corresponding location l^* in physical networks, l^* also minimizes the maximum distance to users in physical networks.*

Proof. Let $d_{l,u}$ be the distance between l and u in $\mathcal{S}$, and u_l be the farthest user to l in $\mathcal{S}$. Therefore, we have

$$\forall u \in U : d_{l,u_l} \geq d_{l,u},$$

According to the consistency of $\mathcal{S}$, we also have

$$\forall u \in U : d^*_{l,u_l} \geq d^*_{l,u},$$

where $d^*_{l,u}$ is the distance between l and u in physical networks. Hence, u_l is also the farthest user to l in physical network. Given an arbitrary location l', and its farthest user in physical network (and thus also in $\mathcal{S}$) $u_{l'}$, we have

$$d_{l,u_l} \leq d_{l',u_{l'}},$$

because l minimizes the maximum distance to users in U among all locations in $\mathcal{S}$. Again, according to the consistency of $\mathcal{S}$, we derive

$$d^*_{l,u_l} \leq d^*_{l',u_{l'}},$$

which finishes the proof because l' is an arbitrary location in physical networks. □

According to the Definition of $\mathcal{S}$'s consistency c, there will be at least a fraction of c users whose distances to l in physical networks are less than d^*_{l,u_l} due to the Law of Large Numbers, that is, at most a fraction of $1-c$ users whose distances to l in physical networks are larger than d^*_{l,u_l}. For instance, in Fig. 4b, GC's average relative prediction error is about 6 times of NC's, but the ratio of user proximity violation in the provision plan from GC is only about 15 percent higher than NC in our experiments at country-wide scale. This can be explained by Fig. 5 in which the GC's consistency is only about 15 percent lower than NC's. For Fig. 5, we randomly select 100,000 pairs of host pairs both for the global case and the countrywide case, that is, $((A_i, B_i), (C_i, D_i))$, where A_i and C_i are randomly sampled from the Planetlab nodes for the global case and from the vantage points for the country-wide case, while B_i and D_i are randomly sampled from the subnets in each case, and $1 \leq i \leq 100,000$. Then we observe and compare the percentage of the 100,000 pairs of host pairs that are consistent with each other, as illustrated in Fig. 5.

Algorithm 1. ClusterUsersFPC(P, D)

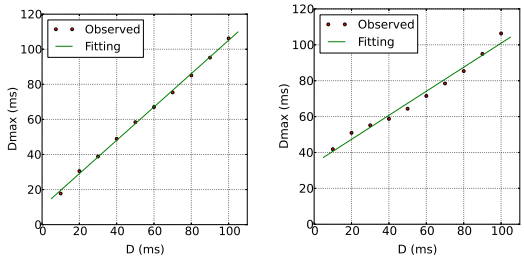
```
// P is the set of points.
// D is the maximum distance between two points in one cluster.
p ← a point randomly picked from P;
Q ← {p} ;
P ← P - {p};
d_max ← +∞;
// d_{p,Q}: the shortest distance between p and the points in Q.
while d_max > D/2 do
    // Pick the point in P which is the farthest to Q.
    p ← select from P where d_{p,Q} = max{d_{p,Q} | ∀p ∈ P} ;
    P ← P - {p}; Q ← Q + {p}; d_max ← d_{p,Q};
foreach q ∈ Q do
    // Putting points in P which are closest to q together.
    P_q ← select from P where q is the closest point in Q to p ;
    C_q ← P_q + {q} ;
return {C_q | ∀q ∈ Q};
```

From Fig. 5 we can see that NC (GNP) is better than GC in consistency, so we adopt GNP in Tentacle. It is out of the scope of this paper to compare the consistency across all known NC or other virtual metric spaces for networks. However, if there exists more consistent $\mathcal{S}$, Tentacle is flexible to plugin it.

4.1.2 User Clustering

Despite there are numerous clustering algorithms, we adopt the ones which only need distance rankings to group the users, e.g., farthest point clustering (FPC) and hierarchical clustering (HC), to take advantage of the high consistency of $\mathcal{S}$. As shown in Algorithm 1, we choose to use FPC algorithm to cluster the users, because it has been proven to be more efficient in "minimax facility location" problems [33].

After mapping all users to point set P in $\mathcal{S}$, this algorithm first picks one random point out of P and put it into an empty point set Q. Define the distance between a point p to point set



(a) Country-wide (b) World-wide

Fig. 6. Explore the relationship between D and D_{max}.

Q ($d_{p,Q}$) as the shortest distance between p and the points in Q. Every step (Lines 9-11), the point in P that is farthest to Q is removed from P and added in Q until the maximum distance from points in P to Q is not larger than $D/2$. Then we cluster P into several clusters based on Q. For each point q in Q, we put the points in P which are closest to q together and generate a cluster. The number of clusters equals to the number of points in Q. If $\mathcal{S}$ is a metric space (both NC and GC are), according to the triangle inequality, the distance between an arbitrary pair of points inside one cluster generated by Algorithm 1 is no more than D in $\mathcal{S}$.

Given the GNC's intrinsic consistent extent, Tentacle can deal with OSP's requirements by adjusting the value of D, that is, the maximum distance between two points in one cluster. For example, we can adjust the value of D and explore the corresponding value for D_{max} (e.g., $>$ 95 percent users served by at least one edge within D_{max}). By adjusting D, we can get the observed values of D_{max} under different values of D, as shown in Fig. 6, and then we can get the relationship between them by fitting the observed data. From the fitting curves, it can be concluded that D and D_{max} are nearly linear with each other in both cases, thus we can choose the suitable D based on OSP's requirement on D_{max}.

4.1.3 Computing Ideal Edge Locations

After clustering the users, Tentacle tries to find the ideal location l for each cluster C. The ideal location l is a point in $\mathcal{S}$ that minimizes the distance from an arbitrary user in C to it, that is

$$l = \arg\min_{p} \max_{\forall u \in C} \|p - u\|. \quad (4)$$

The problem is a convex optimization problem, which could be efficiently solved [35].

4.1.4 Mapping to Realistic Edge Sites

After obtaining the ideal edge location l for each user cluster C (by solving optimization problem Eqns. (2) and (3)), Tentacle first searches pre-known edge sites and users which are the closest to l. Since Tentacle knows the real edge locations of these edge sites and users, it suggests OSPs to investigate edge site deployment opportunities inside or close to these locations.

There are generally four types of deployment methods in a given edge location:

- *Reusing* the existing edge capacity if there is already an edge site in the location. This method usually has the lowest one-time construction cost.
- *Adding* new edge capacity if there is already an edge site in the location, but the existing capacity is not enough.
- *Renting* edge capacity if any third parties, like cloud providers edge computing providers or ISPs, have deployed edge sites at the location.
- *Developing* a new edge capacity if the location is suitable to build server sites from scratch, available to be connected with datacenters with private cables, or feasible to create edge servers by nano-DC or NFV (network function visualization). This method usually costs the most one-time construction cost.

Note that one candidate edge location can have multiple feasible deployment methods, so that we can have multiple edge sites in a single edge location.

We put "reusing" together with other capacity expansion methods to naturally support incremental edge deployments. Nevertheless, in edge provisioning Tentacle does not explicitly prefer using existing edge sites, because one of its goals is to improve the proximity. If existing sites have good performance and low cost, the final provision plan should pick them. Otherwise, Tentacle will suggest to abandon some existing sites because it finds better ones.

For each cluster C, all feasible edge sites in the cluster will be added into the candidate site pool S_u for each user u in the cluster. If Tentacle could not find any unknown available edge sites, it would use the pre-known edge sites directly. This would affect the candidate edge pool S_u for some user u. If S_u is empty, it shows that Tentacle could not find an edge site that satisfies the proximity requirement for the user u in the round. If there are residual users after the edge capacity provisioning process, Tentacle will collect all the residual users from all groups, relax the performance requirements for them, and go through the basic process until they are served or no more edge capacity can be added.

4.1.5 User Constraints for Edge Sites

Due to the local laws (e.g., personal data cannot be sent outside the European Economic Area) or ISP policies (e.g., a node in CDN-i provided by an ISP can only serve users inside this ISP), the candidate site pool S_u which Tentacle obtains from preceding process can have some invalid server sites for some particular users. Tentacle will finally filter such invalid sites out for each user.

4.2 Edge Capacity Provisioning

The candidate site pool S_u obtained by Tentacle lists all edge sites which satisfies the proximity requirement to user u. The final step of Tentacle is to decide how to provision the capacity of these edge sites. Table 1 lists the key notations we uses in Tentacle's edge capacity provisioning model.

First of all, Tentacle merges all users with the same S_u into a single group g, that is, users in a cluser C would be merged into several groups. Let S_g denotes the candidate site pool of user group g, and N_g is the number of user in g.

Define the output variable as $\boldsymbol{n_{g,s}}$ which is the number of users in g served by the edge site s. The total number of users assigned from g should be no more than the total number of users in g

TABLE 1
The Key Notations in Tentacle Model

Inputs	
S	Metric space.
$G = \{g\}$	Set of user groups.
N_g	The number of users in g
S_g	Set of candidate edge sites of g
M_s	The capacity of edge site s.
p_s	The price of unit resource in edge site s.
c_s	The one time cost to open edge site s.
Outputs	
$\boldsymbol{n_{g,s}}$	The number of users from g served by s
$\boldsymbol{r_{g,s}}$	The capacity reserved on s for g.
$\boldsymbol{y_s}$	$\boldsymbol{y_s} = 1$ if s is chosen, $\boldsymbol{y_s} = 0$ otherwise.

$$\forall g \in G : \sum_s \boldsymbol{n}_{g,s} \leq N_g. \tag{5}$$

Let $\boldsymbol{r_{g,s}}$ be the capacity reserved on edge site s for user group g, and we have that the capacity reservation should be no less than the user allocation

$$\forall g \in G : \boldsymbol{r_{g,s}} \geq \boldsymbol{n}_{g,s}. \tag{6}$$

As mentioned in Section 3, we use the maximum number of concurrent users as a metric of the capacity of an edge site, we have the capacity constraint

$$\forall s : \sum_g \boldsymbol{r}_{g,s} \leq M_s, \tag{7}$$

where M_s is the maximum concurrent users the edge site s is able to serve.

Let y_s be a binary indicator, and $y_s = 1$ means the edge site s is finally deployed and $y_s = 0$ otherwise. $\sum_g \boldsymbol{r}_{g,s}$ is the overall capacity needed at the edge sit s, where $\sum_g \boldsymbol{r}_{g,s} \geq 0$. So we have

$$y_s = \begin{cases} 0 & \text{if} \quad \sum_g \boldsymbol{r}_{g,s} = 0 \\ 1 & \text{if} \quad \sum_g \boldsymbol{r}_{g,s} > 0. \end{cases} \tag{8}$$

In order to make the conditional constraint linear, we introduce a large constant N, which can make $0 \leq \frac{\sum_g \boldsymbol{r}_{g,s}}{N} < 1$. It's feasible to make it as $\frac{\sum_g \boldsymbol{r}_{g,s}}{N} \to 0^+$ when $\sum_g \boldsymbol{r}_{g,s} N > 1$ and $N \to +\infty$. Then We can use following linear constraints to express the relationship between y_s and $\boldsymbol{r_{g,s}}$

$$\forall s : \frac{\sum_g \boldsymbol{r}_{g,s}}{N} \leq y_s \leq \sum_g \boldsymbol{r}_{g,s} \tag{9}$$

$$\forall s : y_s \in \{0, 1\}. \tag{10}$$

When $\sum_g \boldsymbol{r}_{g,s} = 0$, Eqn. (9) means $0 = \frac{\sum_g \boldsymbol{r}_{g,s}}{N} \leq y_s \leq \sum_g \boldsymbol{r}_{g,s} = 0$, that is, $y_s = 0$. When $\sum_g \boldsymbol{r}_{g,s} \geq 0$, Eqn. (9) means $0 < \frac{\sum_g \boldsymbol{r}_{g,s}}{N} \leq y_s \leq \sum_g \boldsymbol{r}_{g,s}$, so that it must be $y_s = 1$ combined with Eqn. (10). So we can use Eqn. (9) and (10) to replace Eqn. (8).

The total cost of the deployment should be the sum of resource usage cost and one-time deployment cost (c_s) for an edge site s. Denote p_s as the cost to support one user on s, we have the total cost as

$$\sum_s \left(p_s \sum_g \boldsymbol{r}_{g,s} + c_s y_s \right).$$

Our final goal is to maximize the total number of satisfied users, as well as reducing the cost. Therefore, the objective function we derive is

$$\max. \sum_{g,s} \boldsymbol{n}_{g,s} - \eta \sum_s \left(p_s \sum_g \boldsymbol{r}_{g,s} + c_s y_s \right), \tag{11}$$

where $\eta > 0$ is a selected constant to balance the importance of the number of users and infrastructure cost.

After solving the ILP model given by Eqns. (5), (6), (7), (8), (9), (10), and (11), the capacity of edge site s would be $\sum_g \boldsymbol{r}_{g,s}$. The value of $\boldsymbol{r}_{g,s}$ provides an upper-bound for the number of users from g to be served by s.

4.2.1 Over-Provision for Fault Tolerance

There are two levels of fault tolerances in edge provisioning [3]: intra-site level which handles load balancing among servers in individual edge sites, and inter-site level which handles the outages of an entire edge site. The former is simple because we only need to add some redundant servers in each site and rely on intra-site load balancers to make the individual server failures transparent to users.

As to inter-site fault tolerance, for each g, the edge sites in S_g should have sufficient capacity to support all users in g when an arbitrary site in S_g goes down. This requires the reserved capacity on each edge site to satisfy

$$\forall g \in G, \forall s' \in S_g : \sum_{s:s \neq s'} \boldsymbol{r}_{g,s} \geq \sum_s \boldsymbol{n}_{g,s}. \tag{12}$$

Generally, we can also require the residual sites in S_u have sufficient capacity after arbitrary k ($k < |S_u|$) sites fail, with the Forward Fault Correction (FFC) formulation in [36].

Let λ be a vector which encodes a specific failure case, such that its element $\lambda_s = 1$ if edge site s is down, and $\lambda_s = 0$ otherwise. Denote $\Lambda_k = \{\lambda | \sum_s \lambda_s \leq k\}$ be the set of all failure cases λ which has no more than k edge site failures. We require that

$$\forall g \in G, \lambda \in \Lambda_k : \sum_s \boldsymbol{r}_{g,s} \times \lambda_s \geq \sum_s \boldsymbol{n}_{g,s}, \tag{13}$$

which can be efficiently encoded and solved with "sorting network" method in [36].

5 Practical Issues

In this section, we discuss a couple of practical issues in implementing Tentacle including backend server provisioning, and the variance of online user distribution.

5.1 Edge Provisioning with Backend Servers

Many online services need an integrated infrastructure with both edge (frontend) servers and backend servers. For instance, CDNs use edge servers as proxies and caches. When an edge server's cache does not have a requested

content object, it will download the object from an original server which typically sits in a small number of locations like datacenters. The overall user performance of such online services also depends on the proximity and the capacity of backend servers.

In the case that backend servers are given, e.g., many CDNs have external original servers from their customers without their controls, Tentacle makes sure that the edge sites it picks up satisfy the proximity requirement to the backend servers. It removes any edge server whose delay to the closest backend server is larger than the maximum tolerate delay. In this case, because Tentacle cannot decide the capacity of backend servers, it only ensures the satisfaction in proximity.

In the case that backend servers are also under Tentacle 's control, Tentacle will perform the same process to provision the location and the capacity of backend servers as it provisions the edge servers. For backend servers, edge servers are their conceptual users. The workload from edge servers to backend servers can be obtained from the workload generated from users to each edge sites and the cache hit rates on each edge sites. Tentacle will first discover backend locations from the distribution of edge servers, and then provision the backend server' capacity with the same ILP format in edge provisioning.

5.2 The Variance of User Distribution

After a system have been deployed, OSP can adopt reactive techniques to cope with highly dynamic and unpredictable users' requests. However, the system must have the capacity to handle all the user distribution. In the edge provisioning process shown in Section 4, we assume that the distribution of online users is relative stable. However, in reality the users might be evolving hour by hour, so that the edge provisioning needs to guarantee that the edge infrastructure can achieve the proximity and capacity requirement for online users all the time.

First of all, Tentacle should not distinguish users who are identical from the OSPs' points of view. For instance, it is common to treat all users within the same "/24" IP subnet as identical. Therefore, Tentacle only counts the number of online users within each /24 subnet at a moment as the online user distribution.

One strategy to deal with the variance of online user distribution is to make sure the infrastructure can satisfy some representative user distributions sampled at different period of time. From history, we observe many representative user distributions. We can merge all these representative user distributions into a worst-case user distribution by counting the largest number of users in each /24 subnet. Then by satisfying the worst user distribution, Tentacle can handle any real time user distribution if only the number of online users in each /24 subnet is no more than the worst-case user distribution.

By adopting the worst-case strategy, the system deployed by Tentacle is capable to handle any real time user distribution if the number of online users is no more than the worst-case user distribution. Moreover, the over-provisioning capacities can be used to accommodate the increasing user demand in the future. In order to reduce the overhead, Tentacle can also satisfy e.g., 70 percent of the peak user distribution. In this way, the system deployed by Tentacle costs less but some real time user distribution will exceed the capacity of the system. Its a trade-off between cost and performance.

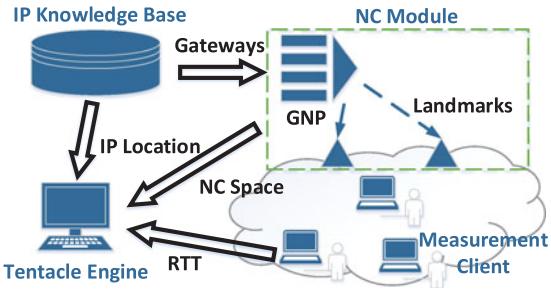


Fig. 7. The system architecture of Tentacle.

6 Evaluation

To compare it with existing approaches of edge provisioning, we simulate Tentacle using both a world-wide and a country-wide (Mainland China) realistic user bases. We show in the following sections that Tentacle outperforms other approaches in reducing the overall costs and improving the percentage of users whose QoE requirements are satisfied. Moreover, we also leverage the measurement data from real networks to explore Tentacle's design choices.

6.1 Experiment Setup

As shown in Fig. 7, there are four key components. *IP knowledge base* is used by the learning approach which maps ideal edge locations from NC spaces to physical networks. It stores the geographical locations, the home ISPs and the NC coordinates of the IP subnets with /24 mask. The IP knowledge base is built on top of HBase. In *NC module*, we adopt GNP and reuse the source code of [13] to construct a NC space for all the network hosts. The landmarks ping the gateway IP address of each subnet in our IP knowledge base (e.g., IP 1.2.3.1 is the gateway address of subnet 1.2.3.0/24). The NC coordinates of these gateways are computed, and all IP addresses within a subnet share the same NC coordinate with the subnet's gateway. After collecting the information from the above three modules, Tentacle*Engine* tries to provisioning edges for the users. We implement the *edge provisioning algorithms* in Section 4 with 1,000+ lines of Python code. Gurobi Optimizer 5.6 is used for solving the ILP in Section 4.2.

Besides Tentacle itself, we also implement a *Measurement client* system which is used to justify the proximity of the unforeseen edge sites picked by Tentacle with the actual delay measurements. This measurement client system is deployed at the domestic scale at present.

6.1.1 Global Scale

IP Knowledge Base. We analyze a one-day log from a commercial global scale online service and extract more than 6,990,000 independent IP addresses, which include 2,272,460 different IP subnets with /24 mask. We obtain each IP's geographical location (country, state, city, longitude, latitude, AS, and home-ISP) from a commercial world-wide IP lookup service.[3]

3. http://www.ipaddresslabs.com/

TABLE 2
The Construction Cost and Bandwidth Price for Some Major Cities in China

City	BJ	GZ	SH	HF	JJ	NC
Construction Cost ($ per year)	10,432	8,114	11,015	9,109	13,036	9,109
Bandwidth Price ($ per year & per 100Mbps)	14,487	12,242	15,818	10,429	8,636	10,429

NC Module. In order to construct a global NC system, we take advantage of 11 online planetlab nodes from Planetlab platform to act as landmarks. About 1,116,576 subnets whose gateways regularly respond to the pings are assigned NC coordinates.

Measurement Client. In order to evaluate an edge provisioning plan, we use the 360 online planetlab nodes to act as the potential edge sites. We randomly sample from the potential edge sites and obtain 50 edges as the known edge sets, leaving the remaining ones as the set to be discovered. After an edge provisioning plan is made, we use the planetlab nodes to measure the actual performance in global Internet.

6.1.2 Country-Wide Scale

IP Knowledge Base. In this experiment, we obtain each IP's geographical location at city-level and home-ISP from a domestic IP lookup service.[4]

NC Module. By now 38 GNP vantage points geographically distributed in China have been deployed and we also select 11 from them to act as landmarks. About 140,000 subnets whose gateways regularly respond to the pings are assigned NC coordinates.

Measurement Client. We have measurement clients installed on about 12,500 end hosts. Because of the resource limits in the end hosts, we only use the clients occasionally to evaluate the proximity of an edge provisioning plan. The evaluation process is as follows: after an edge provisioning plan is made, we try to identify one measurement client whose location is close to an edge site A in the provisioning plan. Then we wake up a probing process inside the client and instruct it to ping the users that are assigned to edge site A. If any of the users do not respond to pings, the probing process will ping their gateways instead. The RTT measured by the client is a good approximation of A's user-server RTT. We perform such measurement infrequently (e.g., once per hour) but for a long period (e.g., one week), to extra out the noises in individual RTT due to queuing delays or temporary routing changes.

6.2 Experiments in the Wilds

We first simulate Tentacle with a user base at a global scale and use the measurement data obtained through the globally-distributed Planetlab nodes and the measurement nodes that we have deployed in Mainland China. In such a way, we strive to simulate Tentacle and other provisioning approaches in a realistic setting where a service provider tries to reach its customers worldwide. After that, we also utilize the data sets from the country-wide network to study the effectiveness of Tentacle on planning services to be deployed and optimized in a specific region. In both scenarios, we use the actual pricing models on edge host buildout and network bandwidth to compute the deployment costs.

Scenario. Given a set of users and their bandwidth demands, we plan a new edge infrastructure for a network service provider to serve the users under different cost budgets and users' requirements on proximity. We compare the deployment costs and the percentage of users whose requirements are satisfied.

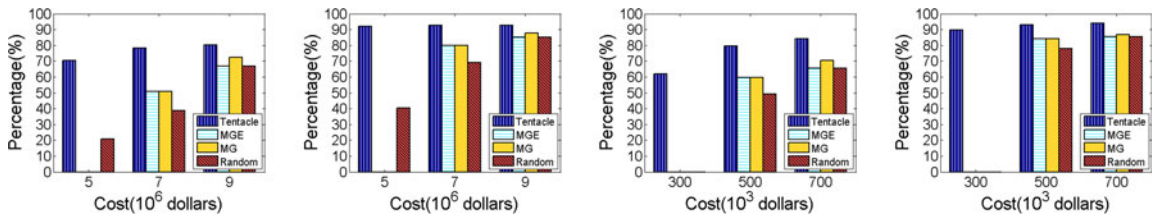
Users of Online Services. For the global-scale user base, we collect ~1,116,000 independent user IPs from a search engine. These users are widely distributed over the world. For the country-wide scale, ~20,000 independent user IPs across Mainland China are collected from a prominent CDN provider. These IPs are active and usually online during our experimental period.

Users' Requirements. Users' QoE requirements on the service are assumed to differ based on the network scales. For example, we suppose that a user expects at least 1 (or 3) edge site(s) whose distance is within 50 (or 90) ms in the world-wide networks but only 20 (or 40) ms in the country-wide networks. The average long-term download bandwidth demand of each user is 50 Kbps and the distribution of individual user demands is learnt from real CDN traces.

Edge Sites. In the world-wide network, we select a part of the online planetlab nodes across the world as pre-known potential edge sites, while Tentacle can possibly discover other new sites beyond these candidates, which consist of the remaining planetlab nodes. For Mainland China network, we select 500 edge locations around China as known candidates to deploy edge sites, while Tentacle can also possibly deploy new edge sites in new locations beyond these candidates. The clients deployed near by these new locations are used to estimate their latency to users in the network.

Edge Cost Model. The cost to acquire bandwidth at an edge location includes two parts: one-time construction cost and long-term bandwidth cost. We assume that an edge provisioning plan should accommodate all users' demand on the bandwidth, that is, the overall capacity for a plan should not be less than a threshold. The bandwidth price across the world is referred from [38]. Moreover, we only have one-time cost when we deploy a new edge site in a location. In order to merge the one-time cost with the long-term cost, we split the one-time cost to each year that a server can work. In our experiment, the construction cost for a single physical server in a given city includes two parts: 1) the servers depreciation ($ per year), which is equal to the average cost to buy a server divides the average time a server can work; 2) the servers maintenance cost ($ per year), which is the average cost for holding the server in the given city, including the rent for a computer room, the electrical cost and so on. Besides, we studied the construction and bandwidth prices in all major cities in China. Table 2 shows the CDN edge deployment cost in some of the cities.

4. http://developer.baidu.com/map/index.php?title=webapi/ip-api



(a) world-wide, 50ms (b) world-wide, 90ms (c) country-wide, 20ms (d) country-wide, 40ms

Fig. 8. Given cost budgets, the percentage of users with at least 1 edge within the expected distance.

The listed construction cost is not for a datacenter, but for a single server.

Alternative Provisioning Approaches. To evaluate Tentacle on provisioning edges for OSPs with flexible server placements, we compare it with two alternative approaches in existing work for server deployment [7], [8], [39]. Moreover, we extend the most frequently used one Marginal-Greedy (MG) to support flexible server placements. These approaches are described as follows:

(1) *Marginal-Greedy*: This method only selects edge sites within the given known candidate sites. It uses a greedy algorithm to select the edge sites from the candidate pool one by one. At the kth round, it selects a site which achieves the minimum maximum user-server RTT together with the $k-1$ edge sites selected in last $k-1$ rounds. This selection process ends when all users' bandwidth demands are satisfied by the selected edge sites. This method is proposed in [39], while the similar ideas also appear in [7], [8].

(2) *Marginal-Greedy-Extension(MGE)*: This method first selects edge sites within given known candidate sites, and then tries to find new edges if a part of users' demands could not be satisfied. Similar to MG, MGE uses a greedy algorithm to select the edge sites from the candidate pool one by one. However, if all the pre-known sites are used but users' demands are still not satisfied, it regards all the known users as the candidate and continues the selection processes. After that, it tries to find new edge around the selected users, similar with Tentacle.

(3) *Random*: This method only selects edge sites within the given known candidate sites. It randomly chooses several edge sites that can satisfy the total bandwidth demands from the users. Then, it assigns users to the closest edge site with enough capacity.

Performance Metrics. We compare different provisioning approaches using two metrics: (1) the percentage of users whose demands are satisfied and (2) the total deployment cost.

6.3 Experimental Results

Fig. 8 compares the percentage of users with satisfactory QoE obtained with various provisioning methods under different cost budgets and users' expectations, i.e., at least 1 edge site within a given distance. The subfigures show the cases for different users' expectations and different user bases, respectively. It is intuitive that the percentage of users having satisfactory delay time grows as the provider invests more deployment cost on building more edge sites, and decreases when users' expectations become more demanding, i.e., requiring more edges within smaller user-server RTT. The results show that, in all cases, Tentacle outperforms the other alternatives.

1) If the provider's investment is limited, Tentacle is more likely to find an edge provisioning solution with enough capacity to accommodate all users' requests. For example, when the cost budgets is 300 thousand dollars in Mainland China, shown in Figs. 8c and 8d, Tentacle can guarantee that 60 percent of the users have at least 1 edge within 20 ms while 90 percent within 40 ms. In contrast, no solutions given by other alternatives can satisfy the capacity requirements and the cost budgets simultaneously. This is mainly because Tentacle has the ability to discover new edge locations with lower bandwidth prices and good proximity to users. As Fig. 9 shows, 57 percent of the edges provisioned by Tentacle are unforeseen beforehand, when the cost budget is limited to 300 thousand dollars and users' requirements is demanding (at least 1 edge within 20 ms).
2) Tentacle achieves much higher percentage of satisfactory QoE than other provisioning approaches, especially when the deployment budget is moderate. For example, in Fig. 8a, at 7 million dollars in world-wide cases, the percentage led to by Tentacle is generally 45 percent higher than the "random" approach, and 30 percent higher than MG and MGE. In this case, existing edge sites can satisfy the capacity requirements, so MGE has the same percentage of satisfied users with MG. As to the 500 thousand dollars in country-wide cases, shown in Fig. 8c, the percentage led to by Tentacle is 35 percent higher than the "random" approach, and 25 percent higher than MG and MGE.
3) Tentacle outperforms MGE even if the cost budget is enough and MGE begin to discover unforeseen sites as well. For example in Fig. 8c, at 7 million dollars in country-wide cases, the percentage led to by Tentacle is 15 percent higher than MGE, and

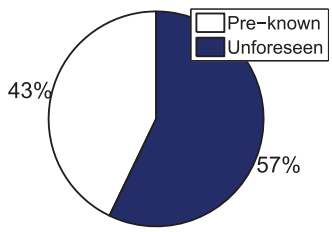
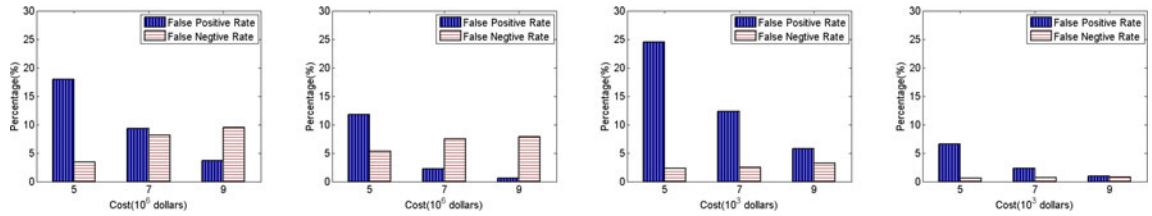
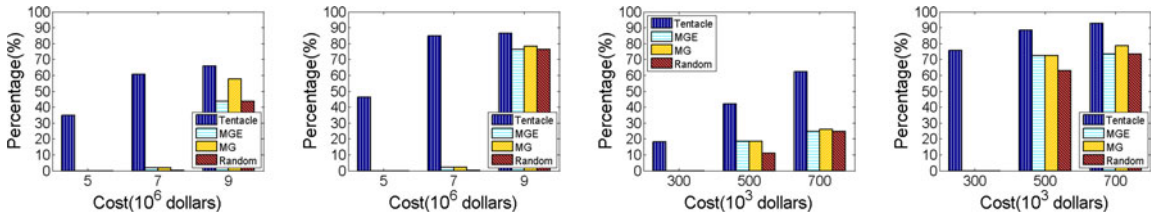


Fig. 9. The percentage of pre-known and unforeseen edge locations provisioned by Tentacle with a limited cost budget (300 thousand dollars) and demanding users' requirements (at least 1 edge within 20 ms) in Mainland China.



(a) world-wide, 50ms (b) world-world, 90ms (c) country-wide, 20ms (d) country-wide, 40ms

Fig. 10. Given cost budgets, the false positive/negative rate for users with at least 1 edge with the expected distance.



(a) world-wide, 50ms (b) world-world, 90ms (c) country-wide, 20ms (d) country-wide, 40ms

Fig. 11. Given cost budgets, the percentage of users with at least 3 edge within the expected distance.

20 percent higher than MG and "random" approach. This is because MGE tries to discover unforseen edges after using the existing edge sites. In this case, MG has the same percentage of satisfied users with "random" approach because every pre-known edge sites has been selected by both methods.

Fig. 10 shows the false positive rate and the false negative of Tentacle for users with at least 1 edge with the expected distance when given cost budgets, where the false positive rate is defined as the percentage of users reported by Tentacle as being with at least 1 edge within the expected distance, but actually NOT; the false negative rate is defined as the percentage of users reported by Tentacle as NOT being with at least 1 edge within the expected distance, but actually YES. As shown in the figure, the false positive decreases as the cost budget increases while the false negative rate the opposite. Combined with Fig. 8, it can be concluded that the gap between the expected performance and the real performance is mainly caused by the cost budget when the cost budget is limited; while by the latency estimation error when the cost budget is enough.

Fig. 11 compares the percentage of users with satisfactory QoE for various provisioning methods under different cost budgets, where users expects at least 3 edges within the given distance.We observe the similar results: Tentacle delivers much higher QoE with the same cost budgets than all other alternative provisioning approaches. Especially in the world-wide cases where cost budgets is limited to 5 million dollars and users expect at least three edge sites within the given distances, shown in Figs. 11a and 11b, Tentacle can find solutions satisfying 35 or 48 percent of the users whereas the alternative approaches can hardly find a viable solution.

Tentacle can take fault tolerance into consideration with relatively small cost increment. Fig. 12 compares the costs whether Tentacle takes fault tolerance into consideration when provisioning edges under a user requirement (at least 1 edge within 50 ms). When 70 percent of users' requirements are satisfied, the edge provisioning plan without fault tolerance would cost about $ 5 million dollars. However, if we expect the percentage of satisfied users keeps the same level even when one edge site fails, the provisioning plan with fault tolerance would cost 18.1 percent more capital of the OSPs. To ensure 80 percent of users' requirements are satisfied, an OSP needs to invest 15.3 percent more than the base case.

6.4 Evaluation of Tentacle Design Choices

There are a couple of coordinate systems and clustering algorithms available for Tentacle to map hosts and group users, respectively. The possible coordinate systems include NC and GC while the clustering algorithms can be either FPC or K-Means [40]. We are interested in finding out the best suitable combinations for Tentacle to generate cost-effective deployment plans. We perform experiments with both a global-scale network and a country-wide network, and show the results in Fig. 13.

To compare the performance of these combinations of coordinate systems and grouping methods, we modify Tentacle's basic process by replacing Algorithm 1 with KMeans/NC, KMeans/GC, FPC/NC and FPC/GC, and replay the experiment scenario in Section 6.2. The expected user-server RTT is set to be 50 and 90 ms in the global scale network, and 20 and 40 ms in the country-wide networks, respectively. We require that for any user there exists at least one edge site that meets such QoE requirement. We can see from the results that (1) at a global scale, the

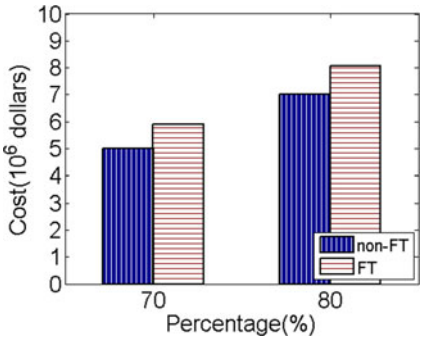
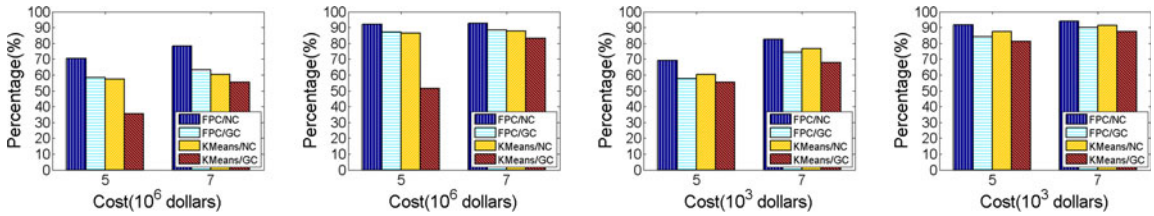


Fig. 12. Given the percentage of satisfied users (at least 1 edge within 50 ms), comparison between the costs whether Tentacle takes fault tolerance into consideration.



(a) world-wide, 50ms (b) world-wide, 90ms (c) country-wide, 20ms (d) country-wide, 40ms

Fig. 13. Given the cost constraint, the percentage of users which have at least 1 edge within a given distance.

FPC/NC has significant advantages than other methods when the budget for edge deployment is moderate (e.g. $5 M) and the QoE requirement is high (at least one edge within 50 ms). As we relax the QoE requirement, the FPC/GC and KMeans/NC show some improvement but still lag behind FPC/NC. KMeans/GC is the worst combination. In Figs. 13c and 13d, we observe a similar trend except that KMeans/NC performs better than FPC/GC. In all the cases, FPC/NC is the best combinations that yields the best QoE given a budget of edge deployment.

7 Related Work

We build upon the rich line of work on edge provisioning problems. Researchers have studied various aspects of this problem, including i) how to select locations from a set of candidate locations for placing cache servers to optimize user performance [5], [7], [8], [9] and ii) how to evaluate the performance of edge server locations via measurements [6], [10] or human comments in social networks [11] to facilitate the provisioning. The major differences between Tentacle and these previous work lie in that: (1) Tentacle opens the possibility to deploy new server sites in locations which are unknown in advance; (2) Tentacle does not only consider user performance, but also comprehensively takes into account the deployment cost, traffic properties and fault tolerance. Therefore, Tentacle uncovers and explores a broader spectrum in server placement problems.

In recent years, how to make online service delivery cost-efficient and ISP-friendly becomes an important research field. Liu et al. [41] study how to optimize the edge infrastructure cost and user performance by redirecting user requests to proper edge nodes. These existing works either stand at an OSP's point of view to design frameworks to facilitate edge provisioning, or stand at ISPs' point of view to deploy protocols to ease the server deployments in ISPs' networks. Tentacle is complementary to them, because it stands at edge infrastructure providers' point of view and its provisioning process benefits from the collaboration between OSPs and ISPs. It optimizes the infrastructure deployment cost which is different from the infrastructure usage cost in [41].

There are other loosely related placement works. Transparent cache placement problem has attracted some academic interests [42], [43]. Cohen et al. [44] use joint mirror placement and request routing to achieve traffic engineering purpose. Our paper focuses on OSP edge nodes provisioning.

Network coordinate [12], [13], [14], [15], [16], [17], [18] has been used in many fields before. Ball et al. [45] use network coordinate for request routing in CDN. Agarwal et al. [46] match P2P users in a hybrid coordinate space. Armitage et al. [47] use network coordinate for service discovery. Tentacle makes a novel usage of NC for computing and locating the edge server locations which are close to a group of users.

8 Conclusion

Latency experienced by end users is critical to online service providers. In practise, edge provisioning is extremely complicated due to the abundant deployment options and many practical considerations. Tentacle is the first framework to achieve a systematic and comprehensive edge provisioning for online service providers with various practical considerations. Its power lies in the novel idea of decoupling the limitations on realistic edge location selection from other concerns. By first finding out the ideal edge locations, then mapping these ideal locations to actual locations in physical networks, Tentacle can even discover unforseen potential edge sites. Via large scale real world deployment and evaluation, we demonstrate that Tentacle can improve the QoE by up to 10-45 percent within a given cost budget.

Acknowledgments

This work was supported in part by the National Basic Research Program of China (973 Program) under Grant no. 2012CB315801, in part by the National Key Research and Development Program under Grant no. 2016YFB1000102, in part by the National Natural Science Foundation of China under Grant nos. 61672318 and 61602194, in part by the independent research project of Tsinghua University under Grant no. 20131089304, in part by the projects of Tsinghua National Laboratory for Information Science and Technology (TNList), in part by the European Seventh Framework Programme (FP7) under Grant no. PIRSES-GA-2012-318939, and in part by the National Science Foundation of United States under Grant nos. ACI-1547428, ACI-1541434, ACI-1440737, and ACI-1450996. Xu Zhang is the corresponding author.

References

[1] The user and business impact of server delays, additional bytes, and http chunking in web search. [Online]. Available: http://velocityconf.com/velocity2009/public/schedule/detail/8523

[2] R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, "Multicloud deployment of computing clusters for loosely coupled MTC applications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 6, pp. 924–930, Jun. 2011.

[3] E. Nygren, R. K. Sitaraman, and J. Sun, "The Akamai network: A platform for high-performance internet applications," *ACM SIGOPS Oper. Syst. Rev.*, vol. 44, no. 3, pp. 2–19, 2010.

[4] D. Rayburn, "Comcast launches commercial CDN service allowing content owners to deliver content via the last mile," 2014. [Online]. Available: http://blog.streamingmedia.com/2014/05/comcast-launches-commercial-cdn-service-allowing-content-owners-to-deliver-content-inside-the-last-mile.html

[5] B. Li, M. J Golin, G. F Italiano, X. Deng, and K. Sohraby, "On the optimal placement of web proxies in the internet," in *Proc. 18th Annu. Joint Conf. IEEE Comput. Commun. Soc.*, 1999, vol. 3, pp. 1282–1290.

[6] B. Krishnamurthy, C. Wills, and Y. Zhang, "On the use and performance of content distribution networks," in *Proc. 1st ACM SIGCOMM Workshop Internet Meas.*, 2001, pp. 169–182.

[7] L. Qiu, V. N Padmanabhan, and G. M Voelker, "On the placement of web server replicas," in *Proc. IEEE INFOCOM Conf.*, 2001, vol. 3, pp. 1587–1596.

[8] E. Cronin, S. Jamin, Cheng Jin, A. R. Kurc, D. Raz, and Y. Shavitt, "Constrained mirror placement on the internet," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 7, pp. 1369–1382, Sep. 2006.

[9] P. Radoslavov, R. Govindan, and D. Estrin, "Topology-informed internet replica placement," *Comput. Commun.*, vol. 25, no. 4, pp. 384–392, 2002.

[10] C. Huang, A. Wang, J. Li, and K. W Ross, "Measuring and evaluating large-scale CDNS," in *Proc. 8th ACM SIGCOMM Conf. Internet Meas.*, 2008, vol. 8, pp. 15–29.

[11] Y. Zhang, D. Li, and M. Tatipamula, "The Freshman handbook: A hint for server placement in online social network services," in *Proc. IEEE 18th Int. Conf. Parallel Distrib. Syst.*, 2012, pp. 588–595.

[12] P. Francis, et al., "IDMaps: A global internet host distance estimation service," *IEEE/ACM Trans. Netw.*, vol. 9, no. 5, pp. 525–540, Oct. 2001.

[13] T. E. Ng and H. Zhang, "Predicting internet network distance with coordinates-based approaches," in *Proc. 21st Annu. Joint Conf. IEEE Comput. Commun. Soc.*, 2002, vol. 1, pp. 170–179.

[14] M. Pias, J. Crowcroft, S. Wilbur, T. Harris, and S. Bhatti, "Lighthouses for scalable distributed location," in *Proc. 2nd Int. Workshop Peer-to-Peer Syst. II*, 2003, pp. 278–291.

[15] L. Tang and M. Crovella, "Virtual landmarks for the internet," in *Proc. 3rd ACM SIGCOMM Conf. Internet Meas.*, 2003, pp. 143–152.

[16] Y. Shavitt and T. Tankel, "Big-bang simulation for embedding network distances in euclidean space," *IEEE/ACM Trans. Netw.*, vol. 12, no. 6, pp. 993–1006, Dec. 2004.

[17] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," in *Proc. Conf. Appl. Technol. Archit. Protocols Comput. Commun.*, 2004, vol. 34, pp. 15–26.

[18] M. Costa, M. Castro, A. Rowstron, and P. Key, "PIC: Practical internet coordinates for distance estimation," in *Proc. 24th Int. Conf. Distrib. Comput. Syst.*, 2004, pp. 178–187.

[19] B. Wong, A. Slivkins, and E. G. Sirer, "Meridian: A lightweight network location service without virtual coordinates," in *Proc. Conf. Appl. Technol. Archit. Protocols Comput. Commun.*, 2005, pp. 85–96.

[20] H. Zheng, E. K. Lua, M. Pias, and T. G Griffin, "Internet routing policies and round-trip-times," in *Proc. 6th Int. Workshop Passive Active Netw. Meas.*, 2005, pp. 236–250.

[21] J. Ledlie, P. Gardner, and M. I Seltzer, "Network coordinates in the wild," in *Proc. 4th USENIX Conf. Netw. Syst. Des. Implementation*, 2007, vol. 7, pp. 299–311.

[22] Global CDN delivery market research report. [Online]. Available: http://www.micromarketmonitor.com/market-report/cdn-delivery-reports-6023250418.html

[23] Where is my data? [Online]. Available: http://www.microsoft.com/online/legal/v2/en-us/MOS_PTC_Geo_Boundaries.htm

[24] ETSI, "Mobile edge computing," 2014. [Online]. Available: https://networkbuilders.intel.com/docs/Mobile-edge_Computing_Introductory_Technical_White_Paper_V1.pdf

[25] Intel, "Intel's big data policy: A vision for big data," 2014. [Online]. Available: http://www.intel.com/content/www/us/en/big-data/intel-corp-big-data-policy-position-paper.html

[26] Microsoft, "Cloudlets for mobile computing," 2014. [Online]. Available: http://www.intel.com/content/www/us/en/big-data/intel-corp-big-data-policy-position-paper.html

[27] QualComm, "The view from the new network edge," 2014. [Online]. Available: https://developer.qualcomm.com/blog/uplinq-2014-recap-view-new-network-edge

[28] IBM, "Akamai and IBM team up for edge computing," 2014. [Online]. Available: http://www.computerweekly.com/news/2240045587/Akamai-and-IBM-team-up-for-edge-computing

[29] Cisco, "Content delivery network (CDN) federations," 2011. [Online]. Available: www.cisco.com/web/about/ac79/docs/sp/CDN-PoV_IBSG.pdf

[30] T. Feder and D. Greene, "Optimal algorithms for approximate clustering," in *Proc. 20th Annu. ACM Symp. Theory Comput.*, 1988, pp. 434–444.

[31] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto, "Optimal packing and covering in the plane are NP-complete," *Inf. Process. Lett.*, vol. 12, pp. 133–137, 1981.

[32] N. Megiddo and A. Tamir, "On the complexity of locating linear facilities in the plane," *Oper. Res. Lett.*, vol. 1, no. 5, pp. 194–197, Nov. 1982.

[33] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Comput. Sci.*, vol. 38, pp. 293–306, 1985.

[34] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit, "Local search heuristic for K-median and facility location problems," in *Proc. 33rd Annu. ACM Symp. Theory Comput.*, 2001, pp. 21–29.

[35] E. Welzl, "Smallest enclosing disks (balls and ellipsoids)," in *Results and New Trends in Computer Science*. Berlin, Germany: Springer-Verlag, 1991, pp. 359–370.

[36] H. H. Liu, S. Kandula, R. Mahajan, M. Zhang, and D. Gelernter, "Traffic engineering with forward fault correction," in *Proc. ACM Conf. SIGCOMM*, 2014, pp. 527–538.

[37] Y. Liao, W. Du, P. Geurts, and G. Leduc, "DMFSGD: A decentralized matrix factorization algorithm for network distance prediction," *IEEE/ACM Trans. Netw.*, vol. 21, no. 5, pp. 1511–1524, Oct. 2013.

[38] ITU-D, "Measuring the Information Society Report," 2014. [Online]. Available: http://www.itu.int/en/ITU-D/Statistics/Documents/publications/mis2014/MIS2014_without_Annex_4.pdf

[39] Y. Wang, C. Huang, J. Li, and K. W Ross, "Estimating the performance of hypothetical cloud service deployments: A measurement-based approach," in *Proc. IEEE INFOCOM*, 2011, pp. 2372–2380.

[40] H. Yin, X. Zhang, T. Zhan, Y. Zhang, G. Min, and D. O. Wu, "Etclust: A framework for scalable and Pareto-optimal media server placement," *IEEE Trans. Multimedia*, vol. 15, no. 8, pp. 2114–2124, Dec. 2013.

[41] H. H. Liu, Y. Wang, Y. R. Yang, H. Wang, and C. Tian, "Optimizing cost and performance for content multihoming," in *Proc. ACM SIGCOMM Conf. Appl. Technol. Archit. Protocols Comput. Commun.*, 2012, pp. 371–382.

[42] P Krishnan, D. Raz, and Y. Shavitt, "The cache location problem," *IEEE/ACM Trans. Netw.*, vol. 8, no. 5, pp. 568–582, Oct. 2000.

[43] J. Xu, B. Li, and D. L. Lee, "Placement problems for transparent data replication proxy services," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 7, pp. 1383–1398, Sep. 2002.

[44] R. Cohen and G. Nakibly, "A traffic engineering approach for placement and selection of network services," *IEEE/ACM Trans. Netw.*, vol. 17, no. 2, pp. 487–500, Apr. 2009.

[45] N. Ball and P. Pietzuch, "Distributed content delivery using load-aware network coordinates," in *Proc. ACM CoNEXT Conf.*, 2008, Art. no. 77.

[46] S. Agarwal and J. R Lorch, "Matchmaking for online games and other latency-sensitive P2P systems," in *Proc. ACM SIGCOMM Conf. Data Commun.*, 2009, vol. 39, pp. 315–326.

[47] G. Armitage and A. Heyde, "Reed: Optimizing first person shooter game server discovery using network coordinates," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 8, no. 2, 2012, Art. no. 20.

Hao Yin received the BS, ME, and PhD degrees from Huazhong University of Science and Technology, Wuhan, China, in 1996, 1999, and 2002, respectively, all in electrical engineering. He is a professor in the Research Institute of Information Technology, Tsinghua University. His research interests span broad aspects of multimedia communication and computer networks. He is also the secretary-general with Industry Innovation Alliance of Future Internet, China.

Xu Zhang received the bachelor's degree in communication engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 2012. He is working toward the PhD degree in the Department of Computer Science and Technology, Tsinghua University. He was a visiting student in the Department of Electrical & Computer Engineering, University of Florida, from Nov. 2015 to May 2016. His research interests include content delivery networks, network measurement, and multimedia communications.

Hongqiang Harry Liu received the BS and MS degrees from Tsinghua University, Beijing, China, in 2007 and 2010, respectively, and the PhD degree in computer science from Yale University, in 2014. He is a researcher in Mobility and Networking Research Group, Microsoft Research, Redmond. His research interests include traffic engineering, software-defined networking, data center networks, content delivery networks, peer-to-peer applications, and cloud infrastructures for big data.

Yan Luo received the BE and ME degrees from Huazhong University of Science and Technology and the PhD degree in computer science from the University of California, Riverside, in 2005. He is a professor in the Department of Electrical and Computer Engineering, University of Massachusetts Lowell. His research spans broadly computer architecture and network systems. His current projects focus on heterogeneous architecture and systems, software defined networks, and deep learning. He has served on the program committee of numerous international conferences and as a guest editor and referee of premier journals. He is a member of the IEEE and the ACM.

Chen Tian received the BS, MS, and PhD degrees from the Department of Electronics and Information Engineering, Huazhong University of Science and Technology, China, in 2000, 2003, and 2008, respectively. He is an associate professor in State Key Laboratory for Novel Software Technology, Nanjing University, China. He was previously an associate professor in the School of Electronics Information and Communications, Huazhong University of Science and Technology, China. From 2012 to 2013, he was a postdoctoral researcher in the Department of Computer Science, Yale University. His research interests include data center networks, network function virtualization, distributed systems, Internet streaming, and urban computing.

Shuoyao Zhao received the bachelor's degree in computer science from Institute for Interdisciplinary Information Science, Tsinghua University, Beijing, China, in 2014. His research interests include network measurement and network science.

Feng Li is now working toward the PhD degree in the Department of Computer Science and Technology, Tsinghua University. His main research interests include network measurement and big data.