Tradeoffs Between Cost and Performance for CDN Provisioning Based on Coordinate Transformation

Hao Yin, Xu Zhang, Shuoyao Zhao, Yan Luo, *Member, IEEE*, Chen Tian, and Vyas Sekar

***Abstract*—Today's content delivery is characterized by key trends such as converged media delivery over HTTP, increasing volumes of multimedia content delivered over IP, and elevated user expectations on quality-of-experience. In this respect, server provisioning is a critical phase of CDN management, which affects both incumbent and entrant CDN operators as well as internet service providers. However, existing tools and approaches to solve server placement problems have serious shortcomings: they offer only coarse tuning knobs and limit servers to a set of candidate sites given *a priori*. Our conversations with CDN operators reveal that a new provisioning mechanism is necessary to take advantage of emerging opportunities such as faster speed to roll out new locations and more access networks. In this paper, we present the design of DISC, a decision support system to help CDN operators systematically investigate different design tradeoffs and evaluate what-if scenarios. The key enabler underlying DISC is a network coordinate-based data analysis workflow that can flexibly embed different cost, performance, and workload characteristics without sacrificing the fidelity. We describe practical use cases and experiences in applying DISC to a large country-wide deployment. The results show that DISC significantly reduces average latency, deployment cost, and interdomain traffic.**

***Index Terms*—Clustering, decision support system, server provisioning, transform.**

I. Introduction

USER-PERCEIVED quality-of-experience (QoE) plays a critical role in Internet video applications [1]. In order to improve QoE with lower cost, operators usually outsource videos to content delivery networks, which keeps multimedia contents on the servers close to end users, thus speeding up the access time and avoiding global network bottlenecks [2]. According to Cisco's survey [3], over half of all Internet video traffic will be delivered through CDN by 2019. Unsurprisingly, we see a lot of renewed interest in *CDN provisioning*, with both incumbent and new entrants in the CDN market. For instance, a recent study maps the dramatic growth of the Google content delivery infrastructure [4]. Similarly, content providers like Netflix are also creating standards such as OpenConnect to invigorate server deployments inside internet service providers (ISP) [5]. And, traditional ISPs are also re-entering the CDN marketplace with both individual and federated offerings [6], [7].

CDN provisioning in today's network and application landscape raises both new opportunities as well as new challenges for different players in the content delivery ecosystem. In terms of opportunities, we see that CDN operators and content providers now have more flexibility in placing delivery server nodes with the advent of modular datacenters [8], increasing deployment of internet exchange points (IXP) [9], and research enablers for nano-datacenters [10]. Furthermore, operators can even roll out their own server infrastructure when needed, and as evidenced by Google's new growth, it is viable to expand the reach deeper into access ISPs [4].

At the same time, the growing user expectations of video quality [11] and the increasing "long tail" of content [12] also raise new challenges for content delivery. In some sense, we see that traditional delivery server deployments were largely driven by network constraints (e.g., where ISP Point-of-Presences or collocation services were available). In contrast, deployments today and in the future will largely be driven by user expectations and workload demands. Moreover, the new dimensions of flexibility in placement described earlier also raise practical challenges in terms of carefully understanding the space of design tradeoffs across user experience, cost considerations, and the overall network impact (e.g., how much inter-domain traffic).

However, existing server placement approaches (e.g., [13]–[19]), fall woefully short of meeting requirements on several fronts. At a high level, these approaches typically model the provisioning problem as some variant of the traditional facility location problem, where we need to choose k out of M given candidate sites. While this modeling paradigm suffices for the traditional model where the set of candidate server locations was fixed and known a priori, it does not capture the new dimensions

Manuscript received May 27, 2016; revised September 8, 2016 and February 28, 2017; accepted April 13, 2017. Date of publication April 24, 2017; date of current version October 13, 2017. This work was supported in part by the National Key Research and Development Program under Grant 2016YFB1000102, in part by the National Natural Science Foundation of China under Grant 61672318, Grant 61631013, and Grant 61602194, in part by the United States National Science Foundation under Grant 1547428, Grant 1541434, Grant 1440737, and Grant 1450996, in part by the European Seventh Framework Programme under Grant PIRSES-GA-2012-318939, in part by the projects of Tsinghua National Laboratory for Information Science and Technology, and in part by a grant from the Intel Corporation. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Zhu Li. *(Corresponding author: Xu Zhang.)*

H. Yin, X. Zhang, and S. Zhao are with the Research Institute of Information Technology, Tsinghua University, Beijing 100084, China (e-mail: h-yin@mail.tsinghua.edu.cn; xzhang12@mails.tsinghua.edu.cn; zsyintl@126.com).

Y. Luo is with the Department of Electrical and Computer Engineering, University of Massachusetts Lowell, Lowell, MA 01854 USA (e-mail: yan_luo@uml.edu).

C. Tian is with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210000, China (e-mail: tianchen@nju.edu.cn).

V. Sekar is with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: vsekar@andrew.cmu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2017.2696309

of flexibility of rolling out new sites as required. In some sense, the space of such new "sites" is potentially infinite as it need not fit inside a "discrete" set of M locations, and artificially imposing such a discrete model may make the problem scale simply intractable. Furthermore, such a provisioning model implicitly combines all the cost-performance considerations into a single tunable "knob", namely the number of locations k. Thus, it may not adequately capture subtle effects in user QoE, inter-domain traffic footprint, and variable deployment costs (e.g., power costs at different geographical locations [20]).

Our contribution in this work is the design and implementation of DISC, a *decision support system* that helps operators to systematically understand the cost vs. performance tradeoffs on multiple dimensions. At the heart of DISC is a *network coordinates* based approach that allows us to flexibly capture the various cost and performance considerations. In contrast to the discrete M candidate location approach, a coordinates-based approach gives the flexibility to consider the entire IP space as candidate locations. Moreover, operators can then flexibly overlay different cost/performance metrics over the network coordinate space to analyze different deployment tradeoffs.

In order to seize the emerging opportunities mentioned above, DISC takes advantage of the idea of "Transform" by transforming network locations from the *IP space* (Network Space) into a *coordinate space* (Coordinate Space), solving a clustering problem in the Coordinate Space and inverse-transforming the solution in the *coordinate space* (Coordinate Space) to the *physical space* (Physical Space). We show analytically that our clustering algorithm in Coordinate Space preserves crucial performance properties such as the latency. Having obtained the clustering solution in the Coordinate Space, we then use a data-driven machine learning method to translate the solution back into the Physical Space, which determines the optimal physical locations and total deployment cost. In addtion, DISC allows CDN operators to tune the parameters such as cost models based on their own existing infrastructure and technology so that DISC's outputs reflect accurate deployment trade-offs for the operators.

We fully implement the DISC system in China, and use it in a number of scenarios to validate our framework's effectiveness and correctness. The results show that DISC is instrumental in clearly quantifying the trade-offs among performance, cost and inter-domain traffic, and mechanisms enabled by the framework are easy to use and flexible enough to accommodate a variety of deployment strategies.

The paper is organized as follows. Section II briefly summarize the related work. Section III introduces the motivation of the work. Section IV describes the system architecture of DISC. Section V presents the analysis of the components of DISC. Section VI provide examples of how to use DISC solve various provisioning problems. Finally the paper is concluded in Section VII.

II. Related Work

Server/mirror placement for content distribution have been extensively analyzed before [13]. Li *et al.* [14] minimize the average latency from clients to the deployed proxies using dynamic programming, which obtains the optimal solution but under the assumption that the underlying network topology is a tree. Qiu *et al.* [15] compare different algorithms for several realistic topologies, and conclude that greedy algorithms could achieve near-optimal results. Jamin *et al.* [17] propose greedy strategies, which deploy servers in decreasing order of node degrees. Radoslavov *et al.* [16] expand the same idea to a two-level approach. Huang *et al.* [18] propose to take the locations of other CDNs' servers as candidate sites to deploy servers, while Hasan *et al.* [21] propose to take ASes as the candidate sites. Similarly, Zhang *et al.* [19] use existing online social networks to solve placement problem of new entrants. All these works require prior knowledge of the candidate sites. Even if some works consider deployment cost, it is modelled as a one-time cost and irrelevant to the service volume; also, none of them considers inter-domain traffic.

There are loosely related placement works. Transparent cache placement problem has attracted some academic interests [22], [23]. Cohen *et al.* [24] use joint mirror placement and request routing to achieve traffic engineering purpose. Our paper focus on CDN nodes placement.

Network coordinate has been used in many fields before. Ball *et al.* [25] use network coordinate for request routing in CDN. Agarwal *et al.* [26] match P2P users in a hybrid coordinate space. Armitage *et al.* [27] use network coordinate for service discovery. DISC uses network coordinate to expand the space of the candidate sites.

III. Motivation

As discussed earlier, deploying new CDN nodes is a complex multidimensional optimization problem that includes considerations such as infrastructure and operating cost, impact on end-user perceived quality of experience, network-level impact w.r.t. inter-domain traffic, and the resilience of the system to future requirements. In this section, we motivate these requirements using public datasets, measured evidence, and detailed conversations with CDN operators.

Cost vs. Performance: The cost of deploying CDN nodes varies quite significantly across different locations. For example, Fig. 1 shows data from a public survey conducted in 2012 on the cost of bandwidth in different metropolitan regions.[1] There are several social, political, and economic factors that go into deciding the cost-per-unit bandwidth across different locations; our goal is not to identify these but rather create a system that works around such practical realities.

The end result is that operators may have to be judicious in the choice of deployment locations. For instance, even though a given metropolitan area X might have a lot of potential clients, the cost of rolling out a new site at X may be prohibitively high. This claim is not merely hypothetical. Our conversations revealed that such decisions happen in real life. For instance,

[1] "How much do businesses pay for a bandwidth of 100 mbit/s?," [Online]. Available: http://www.rezopole.net/images/stories/pdf/100mbit-united-minds-120305.pdf

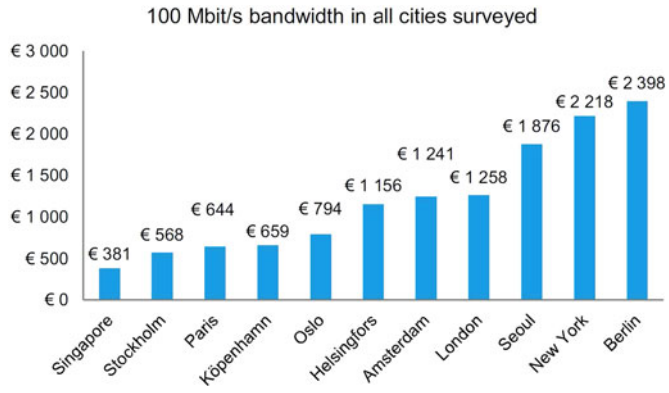

Fig. 1. Deployment costs are extremely different among cities.[1]

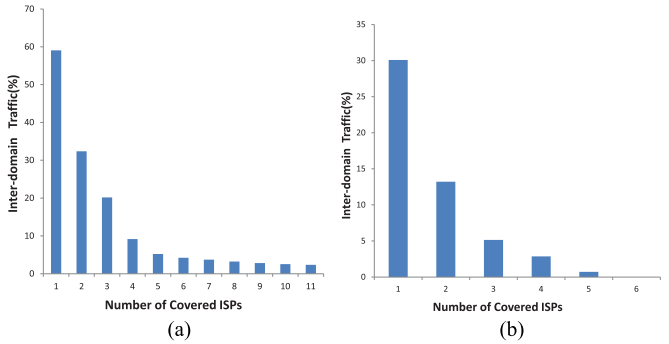

Fig. 2. Diminishing effect of building more sites for reducing inter-domain traffic.

TABLE I
LINK STATE OF INTERDOMAIN TRAFFIC

Link State	Percentage (%)
Healthy (<90 ms)	17.41
Warning (>90 ms & <180ms)	49.73
Critical (>180 ms)	32.85

CDN A choose to serve their clients of SH (a large metropolitan region) from HZ because the deployment cost of HZ is just one-third of that in SH, but the potential increase in latency is $\leq$5%. Similarly, another CDN B we interviewed also confirmed that they chose to serve their clients in SH from nodes in QH, where the cost was one-fifth and latency increase was $\approx$20%.

Coverage vs. inter-domain traffic: Another consideration for CDN operators is to minimize the total amount of inter-domain traffic. This is an important metric on two fronts. First, from a global perspective is a more network-friendly solution that reduces the overall stress on the network. Second, from a more CDN-centric view, minimizing the number of inter-domain links the traffic traverses can improve the user-perceived performance. To elaborate on the second aspect, Table I shows the measured inter-domain latency in China during peak hours. The traffic in the "Healthy" state has latency less than 90 ms almost 80% of the time during these busy hours, traffic in "Warning" state has latency between 90 ms and 180 ms, while traffic in "Critical" state has the latency more than 180 ms. We observe that only 17.41% of the inter-domain links are in the "Healthy" state while up to 32.85% are in the "Critical" state.

To minimize the impact of such congested inter-domain links on their customers' QoE, CDN operators would naturally like to avoid such inter-domain links. Now, there is a natural cost-benefit analysis at play here: Ideally we want to cover all ISPs, but that might require a much larger-scale deployment. Fig. 2 shows the reduction in the total amount of inter-domain traffic vs. the number of ISPs covered for different metro regions BJ and WH. Here, an ISP is marked as covered if there is some CDN node deployed there; i.e., we are not concerned about the capacity here in this simple analysis. In both cases, we see a dramatic "diminishing returns" effect; in the case of BJ there is a sharp drop once the top-6 ISPs are covered while in the case of WH the inter-domain traffic reduces by 90% after just two ISPs are covered. Again, such tradeoffs play a crucial role in determining (a) the number of sites and (b) the specific locations at which a CDN should deploy nodes to meet demand in a given region.

Operator survey: Our conversations with the CDN managers to understand their current decision processes reveals that the approach they employ today is quite ad-hoc and experience based. Moreover, they feel that rather capture placement as a simple k-out-of-M optimization problem, they would really like to systematically understand the fundamental tradeoffs. Thus, our goal is to provide a framework that allows operators to consider a broader search space of placement options and explore these tradeoffs.

IV. DISC SYSTEM OVERVIEW

As discussed in the previous section, existing approaches have a rigid placement model that is inadequate to capture the requirements. Furthermore this is a restrictive model as it precludes locations that fall outside the given "discrete" candidate set. Fig. 3 shows an overview of the DISC system.

Inputs: There are three broad inputs into the DISC decision support system:

1) The first kind of input is network performance measurement data obtained either in-house or from other public Internet datasets[2] or commercial services.[3] This input serves as the basis for extrapolating the user QoE for different node deployment. Our current DISC implementation largely focuses on network latency measurements obtained from our implemented system.
2) The second input to DISC is real world data or models that capture the setup and operating cost of adding some specific capacity in a specific physical location. For instance, this can be a function of local taxes, local power pricing, bandwidth costs, and the availability of existing co-location services. Our current implementation uses proprietary cost data obtained from a large CDN.

[2]"Caida," [Online]. Available: http://www.caida.org/data/
[3]"Keynote," [Online]. Available: http://www.keynote.com/

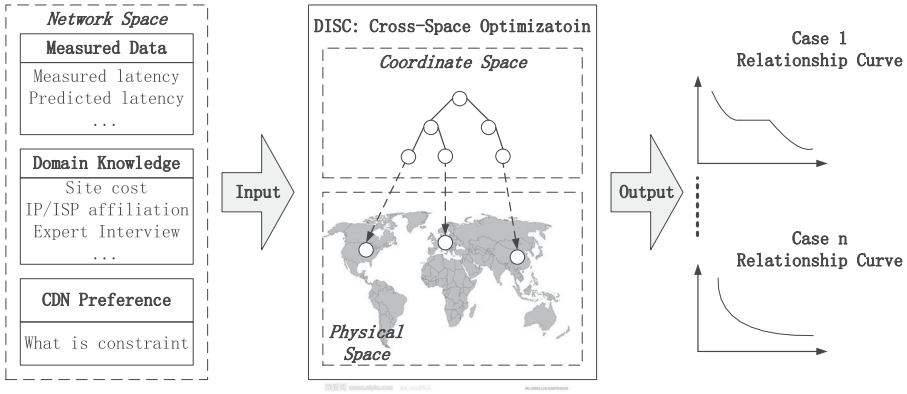

Fig. 3. DISC Architecture.

3) Finally, we have the user facing input to DISC where the CDN operator[4] provides "what-if" deployment queries to understand the cost-performance tradeoffs of different design choices.

Output of DISC: Rather than provide a single opaque parameter like the number of nodes or their locations, we envision the output to be a set of *relationship curves* that allows the CDN operator to visualize the tradeoff between different factors. As discussed earlier this includes considerations such as deployment cost (C) vs. performance (P) and also cost vs. inter-domain traffic footprint (T). For instance, the output of DISC will be a set of tradeoff curves as shown: 1) tradeoff between P and C assuming T is unconstrained; or 2) use an upper bound on T and analyze the P vs. C tradeoff; or 3) given a constraint on P due to SLAs, understand the tradeoff between T and C.

Approach overview: Conceptually, the operation of the DISC system can be divided into four stages:

1) In the first stage DISC uses the network performance data to build a scalable *network coordinate* mapping. This is the role of the *Transform Engine*. One of the properties of this step is to ensure that the subsequent analysis will perform comparably to running the system on the original physical space.
2) In the next stage, *Clustering Engine* clusters the points in the network coordinate space using a bottom-up hierarchical clustering approach. The idea here is that the "clustriod" of each cluster is the location where a node should be deployed. Hierarchical clustering is a natural solution here as it allows us to flexibly tune the granularity of clustering depending on the number nodes to be deployed. For instance, if we want to deploy more nodes then we will choose a more fine-grained clustering granularity but if we are more constrained then we choose cluster clustriods that are higher up in the hierarchy.
3) After determining the coordinate of the candidate node locations via clustering, we need to then map this back into an actual physical realization. This is the goal of the *i-Transform Engine* that conceptually does a inverse transform from the coordinate space to the physical space.
4) Finally, we envision a frontend *Knowledge Center* that allows the operators to visualize these tradeoffs and customize input parameters to inform their provisioning decisions.

[4]We use the term CDN broadly to incorporate both third-party CDNs as we know today such as Akamai/Limelight as well as provider- and ISP-operated CDNs.

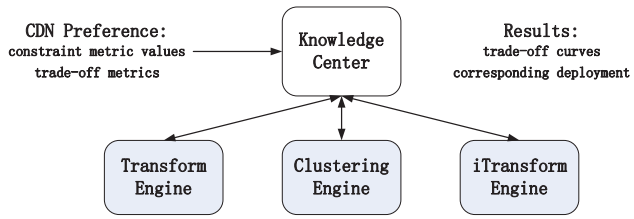

Fig. 4. Framework of DISC.

Fig. 4 shows the framework of DISC. In the next section, we describe the design of the core technical components: *Transform Engine*, *Clustering Engine*, the *i-Transform Engine* and *Knowledge Center*.

V. DISC System Design

A. Transform Engine

As the CDN node deployment is a one-time deal and the location of a node cannot be altered easily once it has been deployed, it's necessary to collect and rely on relatively stable *static network factors* when deploying the nodes, leaving network dynamics to content delivery and request routing [28]. It is however challenging to capture such stable metrics without incurring prohibitive measurement expenses.

To reduce the measurement cost, *Transform Engine* adopts existing network coordinate techniques [29]–[31] to collect information and transforms the infrastructure deployment problem in the physical space to a more manageable problem in Coordinate Space. Network coordinate technique treats the whole network (hereinafter called Network Space) as a geometric space (hereinafter called Coordinate Space), and regards hosts in the network as points in Coordinate Space. The latency between any two hosts is predicted by the distance between the corresponding two points in Coordinate Space.

Transform Engine requires the latencies between arbitrary two users in order to provide necessary information for node

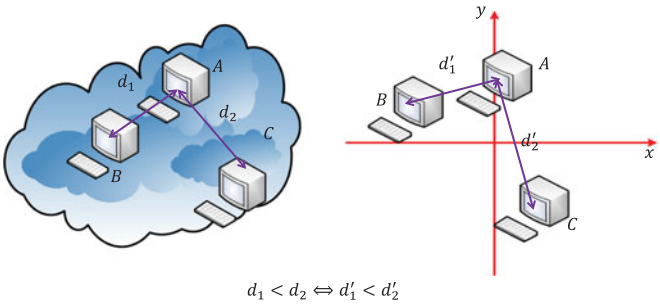

Fig. 5. Consistency between edges in network space and corresponding edges in coordinate space.

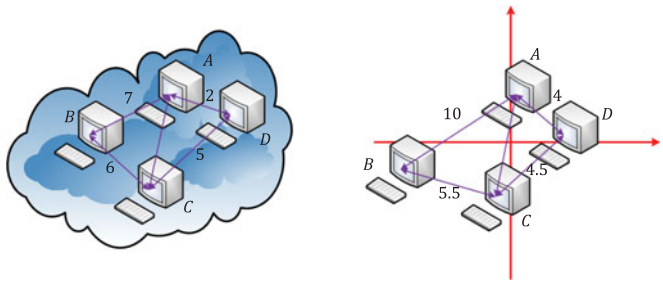

Fig. 6. Example of mismatching using a K-means-based clustering algorithm, where node A is chosen in network space but node C is chosen in coordinate space.

deployment. iPlane[32] provides predicted delay between any IP pairs, however, the cost of measuring all pair-wise latencies for large-scale CDN is prohibitive. Vivaldi [30], a fully distributed scheme to build the positioning system, relies on cooperation from end hosts, which is not always possible. *Transform Engine* adopts the Global Networking Positioning (GNP) [29] technique for its landmark-based property because it incurs low cost to establish and tolerates short term instability. What's more, *Transform Engine* does not rely on GNP's absolute value of predicted latency, instead it takes advantage of the *consistency property*, which means the relative length relationship between two edges in Coordinate Space is consistent with their relationship of network delay in Network Space.

To quantify the GNP's consistent property, we firstly define a metric called *consistency* to describe the consistent extent between two edges. Assume edge i has a measured latency d_i in Network Space and a predicted latency $d_i^{'}$ in Coordinate Space. The consistent extent $c_{i,j}$ between edge i and edge j equals to 1 if and only if the length relationship between d_i and d_j in Network Space is consistent with the relationship between $d_i^{'}$ and $d_j^{'}$ in Coordinate Space, that is to say, if $d_i > d_j$, then $d_i^{'} > d_j^{'}$, and vice verse, which can be stated as following:

$$c_{i,j} = \begin{cases} 1 & \text{if } (d_i - d_j)(d_i^{'} - d_j^{'}) > 0 \\ 0 & \text{if } (d_i - d_j)(d_i^{'} - d_j^{'}) < 0. \end{cases} \quad (1)$$

Fig. 5 illustrates a simple case, in which edge AB established by host A and host B has a measured latency d_1 in Network Space and predicted latency $d_1^{'}$ in Coordinate Space, while edge AC established by host A and host C has a measured latency d_2 in Network Space and predicted latency $d_2^{'}$ in Coordinate Space. If the consistency of edge AB and AC is 1, $d_1 < d_2$ is equivalent to $d_1^{'} < d_2^{'}$, and vice verse.

Then we use the consistent extent of two edges to define the consistent extent c_w of the whole network. c_w means that if we select the shorter one from two arbitrary edges according to the predicted latencies, the latency result will be the same as the one based on the measured distance with the possibility c_w. The consistency extent c_w of the whole network can be defined as

$$c_w = \frac{\sum_{i,j \in E; i<j} c_{i,j}}{\binom{|E|}{2}} \quad (2)$$

where E is the edge set and $\binom{|E|}{2}$ is the number of selections when choosing two edges from the network.

To verify the property, we choose the original data used in [29] as the test set, which were collected through 19 *probes* to 869 *targets* and have been published,[5] and re-constructed the eight-dimensional coordinate system used by [29]. Among the 19 probes, 15 probes act as *landmarks*. We call the other 4 probes *vantage hosts*, which acted as ordinary hosts in Internet to verify the system.

Once building the coordinate system, we can obtained two edge sets: set1 (13,035 edges from landmarks to targets) and set2 (3,476 edges from vantage hosts to targets). The length of any edge from set1 or set2 can be measured using ICMP ping packages from the probes in physical space and predicted using the network coordinate in Coordinate Space. Then we compare the measured latencies with the predicted latencies of arbitrary two hosts, and statistically derive the consistency of the network coordinate system. Our study shows that the percentage of edge pairs with consistency property is 92.56% when the two edges belong to set1, while 92.70% when the two edges belong to set2. In other words, if we choose two edges randomly, with high probability their length relationship is consistent between the physical space and coordinate space. We believe that the probability can be improved further with enhanced network coordinate techniques.

B. Clustering Engine

The goal of *Clustering Engine* is to determine the coordinate of each node to be deployed. The *Clustering Engine* groups neighboring nodes and choose the "center" of a cluster as the ideal location to place the server. It is a nontrivial task because a clustering algorithm may cause mismatch between Coordinate Space and Network Space if not chosen carefully. For example, the classical K-means algorithm finds the node with the smallest distance sum as the center of a cluster. We could repeat the process until the local optimal solution is found. Fig. 6 illustrates such a case where node A is chosen using K-means as the center of the cluster in Network Space whereas node C is chosen in Coordinate Space. Such mismatch causes errors which would

[5]"Global networking positioning," [Online]. Available: http://www.cs.cmu.edu/ eugeneng/research/gnp/

Algorithm 1: PlaceNodes(U)

```
// U is the set of users
// C is the set of clusters in Coordinate Space
// c_i is the i_th cluster in C
C ← {{u_i}|∀u_i ∈ U}
R ← {}
while |C| > 1 do
    c_i, c_j ← the closest two clusters in C ;
    c ← c_i ∪ c_j ;
    C.delete(c_i) ;
    C.delete(c_j) ;
    C.add(c) ;
    R.add(Record(C))
return R ;
```

Algorithm 2: Record(C)

```
// C is the set of clusters in Coordinate Space
S ← {}
for each c in C do
    Find a point (p) whose maximum distance to points
    in cluster c is minimum
    S ← S ∪ {p}
return S ;
```

be accumulated in the iterative process. Subsequently the final result deviates greatly from the ideal solution.

To address the mismatch problem, we choose a hierarchical clustering algorithm to group hosts into clusters and merges the "closest" two clusters step by step. Specifically, each ordinary host itself constitutes a single cluster at the beginning, and then the closest two clusters are merged into a new cluster, with the number of clusters decreased by one serving the cluster members. Repeat the above merging process until all the hosts are clustered into one set. For each cluster, its *center* is used as the appropriate location in Coordinate Space to deploy the node. During the merging process, we record the intermediate results and thus quantify the relationship between various factors and the number of nodes.

Formally we define the "center" of a cluster (aka "clustroid") as the point whose maximum Euclidean distance to other points in a Euclidean space is minimal. We claim that if the consistency property holds, the clustroid of each cluster determined in Network Space happens to have the coordinates of the clustroid in Coordinate Space. We provide the proof in Appendix A. This ensures that the node location determined in Coordinate Space is equivalent in Network Space without introducing errors.

The clustering process is described with Algorithm 1 and Algorithm 2. Algorithm 1 addresses a key question: how to define the distance between two clusters. As Algorithm 1 shows, we find the closest two clusters and merge them into a new one in each loop of the clustering process (Line $8-12$). The distance between two clusters c_i and c_j, $dist(c_i, c_j)$, is the maximum possible distance between any two points from these two clusters. Such distance definition tends to make points in one cluster as concentrated as possible. The distance $dist(c_i, c_j)$ is formally defined as

$$dist(c_i, c_j) = \max\{dist(h_l, h_m), \forall h_l \in c_i, \forall h_m \in c_j\}.$$

One important property of $dist(c_i, c_j)$ is that if two clusters c_i, c_j merge into a new cluster, the distance between the new cluster and a third cluster c_k can be deduced directly from $dist(c_i, c_k)$ and $dist(c_j, c_k)$, that is

$$dist(c_i \cup c_j, c_k) = \max\{dist(c_i, c_k), dist(c_j, c_k)\}.$$

This property is very useful in reducing the algorithm's complexity.

In order to quantify these relationships among metrics, Algorithm 1 records the intermediate results using Algorithm 2. After the completion of Algorithm 1, R is set of all possible deployment strategies generated by DISC, and each element $S \in R$ corresponds to a CDN node deployment strategy, which consists of the nodes' coordinates.

C. i-Transform Engine

After *Clustering Engine* determines different deployment strategies, and gives each node's network coordinate of a strategy in Coordinate Space, *i-Transform Engine* is responsible for inversely transforming the coordinate to an physical location in Physical Space. A node's physical location not only refers to its geographical location, but also includes its ISP property. As to the granularity of geographical location, we leverage the granularity of current online address lookup API.[6,7]

An IP's corresponding physical place information looked up from these online address databases,[6,7] includes the ISP and geographical location (country, state, and city). We use a tuple to denote a host in the Physical Space, that is "IP, physical location, network coordinate". Using such tuples, *i-Transform Engine* can deduce a point's physical location whose network coordinate is known.

Due to the existence of network coordinate's consistency, two points close to each other in the network coordinate space are also close to each other in the physical space. Based on this idea, *i-Transform Engine* adopts a voting mechanism in a point's neighborhood to determine its physical location. The neighborhood of a point consists of the closest k points to it, which is also called k nearest neighborhood (k-NN) [33]. The mechanism can be stated as Algorithm 3.

With k-NN as the basis, we apply a voting mechanism to ensure that the chosen location of a node does not violate any other constraints. Fig. 7 gives such an example, where the blue ellipses are points which locate at Location 1, the red triangles are points which locate at Location 2 and the purple star is the point whose physical place is to be determined. The closest 5 points to the purple star constitute its 5 nearest neighborhood (5-NN). There are 4 points in Location 1 but only 1 point in

[6]"Taobao online IP query interface," [Online]. Available: http://ip.taobao.com/ipSearch.php

[7]"Sina online IP query interface," [Online]. Available: http://int.dpool.sina.com.cn/iplookup/iplookup.php

Algorithm 3: DetermineLocation(B, p, k)

```
// B is the database of all known IPs, represented by a
   tuple "IP, physical place, network coordinate"
// p is the network coordinate of a given point
// k is the number of points in p's neighborhood
N_p ← the closest k points to p in B
s ← the majority location of N_p
return s ;
```

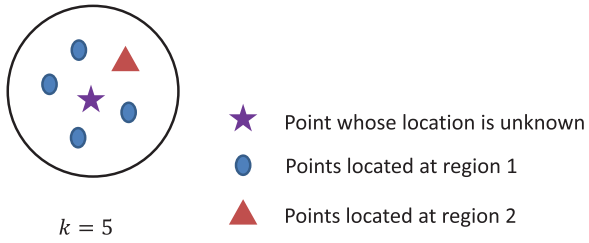

Fig. 7. Example for the voting mechanism in a point's neighborhood.

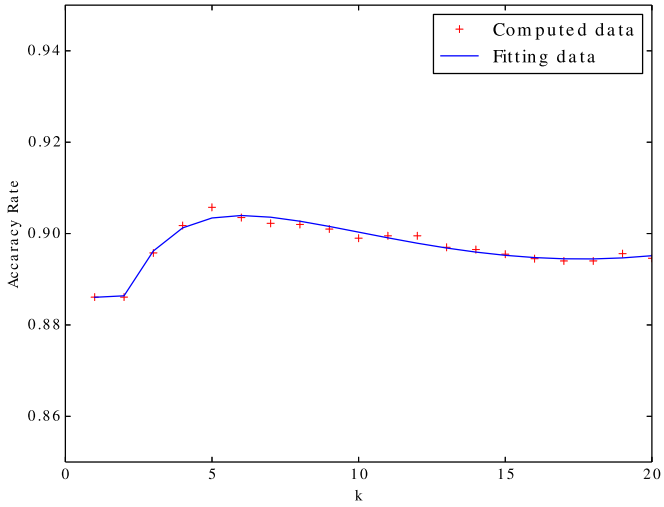

Fig. 8. Relationship between the predicted accuracy rate and the point number k in neighborhood.

Location 2, thus we take the majority place in the 5-NN, that is, Location 1 as the physical place of the star.

We use an experiment to verify the rationality of the voting mechanism and determine the optimal neighborhood size, although [33] has proved that the probability of error of the nearest neighbor rule is bounded by twice the Bayes probability of error. After constructing the network coordinate space (detailed in Section VI), we choose 11,000 hosts randomly from the total user set, among which 10,000 hosts belong to the training set and the rest to the test set. Then we infer the physical location of each IP from the test set using Algorithm 3 and look up its physical addresses from a public database.[6] Next we compare the inferred location with the practical location and determine the accuracy of the inversely mapping algorithm, which is defined as the percentage of IPs whose inferred locations are in accordance with their physical locations.

Fig. 8 shows the relationship between the predicted accuracy rate and the neighborhood size k. The accuracy rate varies with the number of points in the neighborhood, and reach the maximum value (about 91%) when the number of points in the neighborhood equals to 5. So in the following experiments, we choose five as neighborhood size when determining an IP's physical location.

D. Knowledge Center

Knowledge Center serves as the hub for matching CDN operator's preferences with learned facts to achieve cost-effective CDN provision. Specifically it accepts CDN operator's preferences, cost model and their users' IP addresses. The cost to deploy a node at a certain location may differ among CDN operators due to their business relationship with IaaS (Infrastructure as a Service) Providers and IaaS Providers' existing infrastructure. For example, if a node is determined as a node in Amazon's CloudFront, the unlinear On-Demand Pricing cost model of the CloudFront nodes[8] will be used. Therefore, a CDN operator can compose its own cost model. The Knowledge Center then takes advantage of the three engines to infer the tradeoff among various factors (the average latency, the deployment cost and inter-domain traffic) and supports the CDN operator's decision making.

[8][Online]. Available: https://aws.amazon.com/cloudfront/pricing/

Algorithm 4: pre-process(U, S)

```
// U is the set of users
// S is the set of pairs of old nodes and their capacity,
   S = {(s_1, n_1), (s_2, n_2), ..., (s_m, n_m)}, each s_i is the i-th
   old node , n_i is the capacity of s_i
U' ← U ;
for each s_i in S do
    for j → 1 to n_i do
        h ← the closest host to s_i in U' ;
        U' ← U' − {h} ;
return U' ;
```

The Knowledge Center is flexible enough to accommodate a variety of usage scenarios by pre-processing the input data. We give a few example use cases as follows:

Expanding an existing CDN: CDN operators may expand their systems to meet growing demands and require incremental deployment. DISC can pre-process the input data to support the system's expansion without moving the existing nodes. It does so by preassigning a number of users to the existing nodes and then deploys new nodes for the remaining users. The preassignment of users can be based on the distance to the existing nodes, as well as the capacity of existing nodes. DISC can also plan ahead the future placement with its expected user growth speed by tuning the number of users removed for existing nodes. The process can be stated as Algorithm 4.

Reducing Inter-domain Traffic: For content-provider-operated CDNs, especially distributing video contents, inter-domain traffic is a critical factor to consider. Under this circumstance, Knowledge Center can simply solve it by adding

Algorithm 5: post-process(U, S)

// U is the set of users
// S is the set of nodes determined based on the operator's latency requirements
// R is the set of $(S, S^{'})$ pairs, where S is the set of nodes and $S^{'}$ is the subset of nodes which are multi-homing
$C \leftarrow \{c_1, ..., c_k\}$, where $c_i = \{\}(1 \leq i \leq k)$
for *each* u *in* U **do**
    s_i is closest to u in S;
    $c_i \leftarrow c_i \cup \{u\}$;
$L \leftarrow [1, ..., k]$;
for *each* c_i *in* C **do**
    $t \leftarrow 0$;
    for *each* h *in* c_i **do**
        if $ISP(h) \neq ISP(s_i)$ **then**
            $t \leftarrow t + 1$;
    $L[i] \leftarrow t$;
Sort each elements in sequence L, n_{p_i} is the i-th largest number in L ;
$R \leftarrow \{\}$;
$S^{'} \leftarrow \{\}$;
for $i \rightarrow 1$ *to* k **do**
    $S^{'} \leftarrow S^{'} \cup s_{p_i}$;
    $R \leftarrow R \cup (S, S^{'})$;
return R ;

an inter-domain traffic constraint to Clustering Engine when merging two clusters. For example, based on hosts' ISP information in Algorithm 1 of the *Clustering Engine*, we can determine whether merging the two closest clusters would increase inter-domain traffic. If the inter-domain traffic exceeds the given expectation after merging the closest two clusters, we set the distance between the two clusters as $+\infty$ and never merge them afterwards.

Meeting Performance Requirements: For some commercial CDNs, providers would prioritize reducing the average latency over minimizing the inter-domain traffic and the deployment cost. In this case, the *Knowledge Center* can take advantage of the three engines to determine the nodes based on providers' latency requirement, and then post-process the nodes by making some to be multi-homed using Algorithm 5 to trade off the inter-domain traffic with the cost. In Algorithm 5, the choice of multi-homed nodes can follow a greedy principle, that is, giving preference to the nodes that reduce more of the total inter-domain traffic if building as multi-homing nodes.

VI. Evaluation

We implement, apply and evaluate DISC in mainland China, the country with largest population in the world. To use DISC, a CDN provider needs to have knowledge of the IP addresses of users and the system's requirements on latency and cost, and provides them to DISC as input parameters. In this section, we present a number of usage scenarios illustrating the effectiveness of DISC. In particular, we give examples of how to obtain raw distances of nodes in coordinate space, how to tune the clustering algorithm to solve deployment problem with different constraints.

TABLE II
Description of the Measurement Work

Probes	35 hosts distributed across the Mainland China
Targets	290,000 ".1" addresses grouped by IPs extracted from the one month logs of a commercial CDN
Objective	Latencies among probes and latencies from probes to targets
Period	Every two hours interval from August 22, 2013 to September18, 2013

A. Experiment Setup

It is important that DISC relies on Internet's static performance indicators. In this subsection, We describe the overall experiment setup for the following cases.

Scenario: Given the user set, the objective is to construct a decision support system to help CDN operators to systematically understand the tradeoffs among deployment cost, user-perceived latency, and inter-domain traffic. In this evaluation, the user set is extracted from the one-month service log files of a commercial CDN system (not identified due to privacy) in the country. Moreover, the requested contents are videos and we assume that each user consumes the same amount of bandwidth when requesting the content.

Data Collection: As the probes of the network coordinate system published at[5] are out of our control and not suitable for Mainland China, we specifically deploy 35 *probes* and construct a network coordinate space for Mainland China. These probes are located in more than 30 cities across the country and at the backbone of the major Chinese ISPs. On the other hand, the IP of *targets* in China are obtained from the one-month service log files of a commercial CDN system (not identified due to privacy) in the country. Through the log files, 40 million different IP addresses are extracted, among which 23 million are locate in mainland China, covering all of the provinces and ISPs. In order to reduce the measurement overhead while keeping the integrity, we take the hosts with the same first 3-byte prefix in their IP addresses as a group, and use the default IP address (".1" address with a /24 network prefix) of each group to represent the hosts in it. In this way, the number of targets was reduced to 290,000.

Next, we measure the latencies among the probes themselves and between the probes and targets across the Chinese Internet. To get the raw distance between two hosts, we send 100 ICMP ping packets at one second apart and take the minimum round-trip value as the distance. The measurement work were repeated every two hours from August 22, 2013 to September 18, 2013. The distances between two hosts at different time are recorded and pre-processed before constructing Coordinate Space. Table II describes the measurement work.

Data Pre-Processing: We sanitize the raw data based on following principles. 1) Remove the hosts which cannot be

accessed. We remove a target if it cannot be accessed by any of the probes during the measurement period. These hosts do not exist any longer, are off-line during the period, or are inaccessible due to their firewall settings. 2) Extract the stable Internet information needed for node deployment. We choose the minimum latency between two hosts during the measurement period and regard it as their raw distance, which can be taken as the latency mainly caused by the Internet topology rather than the congestions in the Internet. 3) Eliminate the outlier data from the measurement. We exclude the ordinary hosts whose latency to probes deviates too much from the empirical relationship between the Internet latency and the geographical distance [34], that is, the latency is expected to be the value when bits travel at $\frac{4}{9}_{th}$ of the speed of light in vacuum. After filtering, there are 142,982 ordinary hosts left and we take them as the user set U of our CDN.

With the 35 probes and 142,982 ordinary hosts and according to the practical situation of mainland China network, we utilize the raw distances among the probes, as well as the raw distances between the probes and targets to construct a nine-dimension network coordinate space [435. Among the probes, 17 probes acted as landmarks of GNP.

Performance Metric: The performance of a CDN can refer to different metrics of the system, such as the average latency users perceived and the overall throughput. In order to improve the throughput, CDN operators always deploy nodes close to users, allocate enough capacities to each node, distribute contents to suitable nodes, and route the users requests to the closest node with needed contents [36]. Among them, the former two are relatively static and can be determined when deploying the nodes, while the latter two focus on handling the network dynamics (such as the variance of network conditions) and should be adjusted in real-time. In the case study, DISC focuses on allocating enough bandwidth to each node according to the users served by the node and minimizing the average user-perceived latency, both of which can help improve the throughput. So we use the average latency as the primary performance metric.

Cost Model: The deployment of a single node incurs one-time construction cost and its long-term operational cost. The annualized deployment cost of a node is its depreciation and operational cost per year where the depreciation equals to the ratio between the one-time construction cost and the node's life time in years. Because both the one-time cost and the long-term operational cost of a node are directly correlated with its location and bandwidth, we use the product of the bandwidth price for the location and its bandwidth requirement to quantify its deployment cost.

The bandwidth price and requirement of a node are calculated as follows. We categorize the locations in mainland China into four types based on the metropolitan regions: first-tier, second-tier, third-tier and others. We scale the dollar price for these four types to be 8, 4, 2, and 1, from most to least expensive in this paper. Meanwhile, we use a non-linear function of the number of users to estimate the bandwidth requirement of a node. Specifically, the bandwidth requirement increases at the beginning and then levels off when the number of users is beyond a threshold, which is similar with the cost model of CloudFront. This is because the sharing of bandwidth among users leads to economical deployment, and the technical capability of a CDN determines how well its users can statistically share the links. While the exact function is to be provided through the *Knowledge Center* by the provider, without loss of generality, the bandwidth calculation in our experiments uses the square root of the normalized number of users.

Inter-Domain Traffic Model: With the dynamic of network applications and user behavior variability, the total volume of inter-domain traffic is difficult to predict. In this paper, the inter-domain traffic is calculated as the percentage of users whose closest node is cross-ISP.

Existing Approaches: We compare DISC with traditional methods such as Marginal-benefit-greedy method and Random-selection method, which need candidate sites beforehand. We give 10 different candidate site lists, and choose the best one when the traditional methods perform with best performance.

1) *Marginal-benefit-greedy method:* given a set of candidate sites and the number of nodes to be deployed, pick the next site that obtains the minimum average latency. The method performs best among the existing feasible methods [15], [17], [18], [37]. Formally, there exist M candidate sites to accommodate n nodes. At each iteration, assume $m(0 \leq m \leq n-1)$ nodes have already been selected, we pick the $(m+1)_{th}$ node which obtains best performance among the remaining $M-m$ candidate sites if combining with the existing m nodes. This method is the most frequently used algorithm and has been proved to achieve near-optimal results for several realistic topologies[15].
2) *Random-selection method:* given a set of candidate sites and the number of nodes to be deployed, pick the nodes from the candidate sites randomly [15], [17], [38]. The method performs worst among the existing feasible methods and can act as the upper bound of the average latency.

B. Case 1: Without Interdomain Traffic Concern

If providers deploy their system without concerns about the inter-domain traffic, especially for ISP-operated CDNs, *Knowledge Center* can directly take advantage of the three engines. Fig. 9 shows the relationship among the average latency, the deployment cost and number of nodes.

Fig. 9(a) shows the relationship between the average latency and the deployment size, which is intuitive that the latency reduces with the number of nodes. Gradually such reduction levels off as the node number increases, which is accordance with the law of diminishing marginal utility in economics. While the Fig. 9(b) shows the general trend that the deployment cost increases with the number of nodes, but there exist cases whre the cost decreases locally. This phenomenon is actually a good example to illustrate the complexity in CDN deployment. The real reason behind the cost drops in Fig. 9(b) is the tiered bandwidth price at different locations, just as shown in Fig. 1. To minimize the latency, some nodes are deployed in major metropolitan areas such as Beijing, Shanghai and Guangzhou, with higher

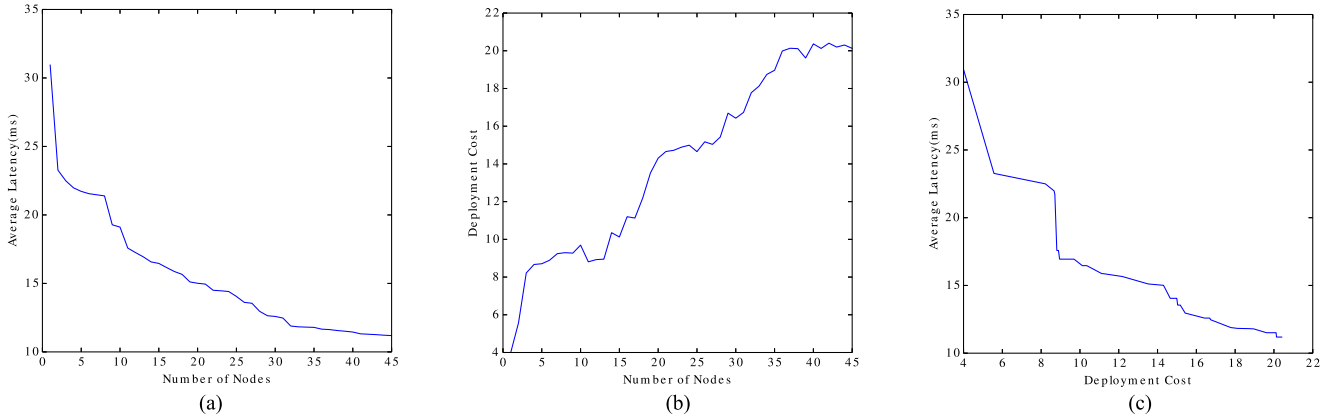

Fig. 9. Case one: not considering inter-domain traffic, relationship between (a) average latency versus node number, (b) deployment cost versus node number, and (c) average latency versus deployment cost. (a) P(N). (b) C(N). (c) P(C).

TABLE III
COMPARISON AMONG DIFFERENT METHODS WITHOUT INTERDOMAIN TRAFFIC CONCERN

	DISC	Margin	Random
Average Latency (ms)	14.0	15.3	16.3
Deployment Cost	14.34	17.69	17.72
Inter-domain Traffic (%)	8.3	12.4	14.9

bandwidth price. As the node number increase, a portion of the nodes are moved to second-tier cities, reducing the overall cost. Fig. 9(c) shows the tradeoff between the average latency and deployment cost without inter-domain traffic constraint, which can help CDN providers make informed decisions.

Table III shows the comparison when deploying 30 nodes without consideration about the inter-domain traffic. As to the average latency users perceived, DISC reduces the average latency by 8.5% than the marginal-benefit-greedy method. This is because DISC can deploy nodes across the Internet while the marginal-benefit-greedy method can only deploy nodes at a limited set of candidate sites. It's likely that the marginal-benefit-greedy method may further reduce latency if given more reasonable candidates. However, that requires a site selection process which itself is an open question. Compared to the random-selection method, DISC reduces the average latency by 14.1%.

DISC reduces the cost by 18.9% compared to the marginal-benefit-greedy method and 19.1% to random-selection. For the marginal-benefit-greedy methods, it depends heavily on the candidate set. Usually, the set is chosen from the large cities with higher cost price, thus incurring higher deployment cost; while the random-selection methods choose nodes randomly, and it's highly likely that majority of users are served by a small set of nodes located in first-tier or second-tier cities, incurring more costs and bandwidth.

For inter-domain traffic, DISC can greatly reduce the percentage of users who access contents cross-ISPs. Compared to marginal-benefit-greedy and random selection, DISC can reduce the percentage of the inter-domain traffic by 33% and 44.3%, respectively.

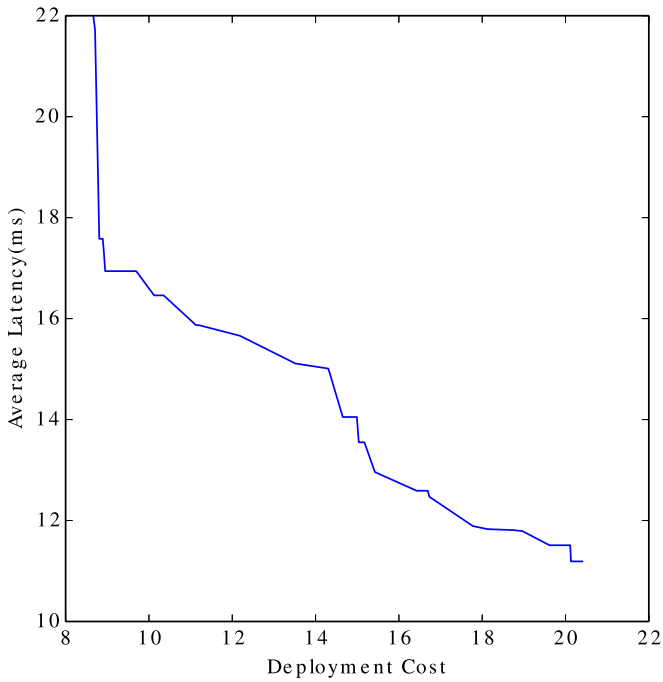

Fig. 10. Tradeoff between average latency and deployment cost with inter-domain traffic concern.

C. Case 2: With Interdomain Traffic Concern

If CDN providers expect to deploy nodes under the constraint of inter-domain traffic, *Knowledge Center* can support their decision-making by obtaining the trade-off between the average latency and the deployment cost with the inter-domain traffic constraints. Fig. 10 shows the tradeoff when limiting the inter-domain traffic to less than 10%, where each point represents the possible minimum latency under the given deployment cost in x-axis. This is intuitive that the more deployment cost, the lower the possible minimum average latency.

Fig. 11 compares DISC with existing approaches when limiting the inter-domain traffic to less than 10%. Given the deployment cost budgets, DISC always achieves the lowest average latency. Especially when the cost budget is low, DISC has a

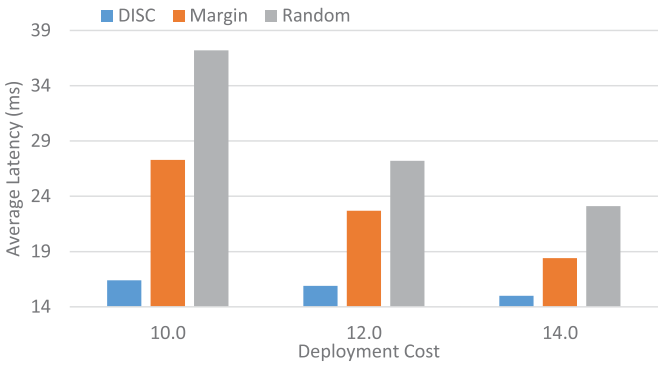

Fig. 11. Comparison among different methods with inter-domain traffic concern.

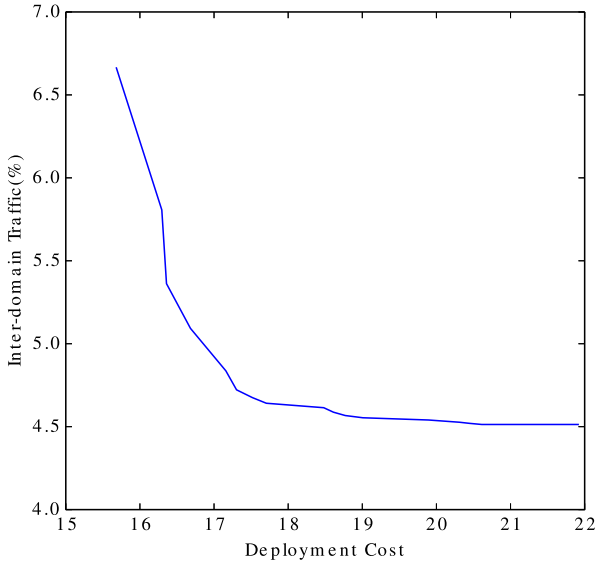

Fig. 12. Tradeoff between inter-domain traffic and deployment cost with average latency constraints.

remarkable advantage. This is mainly because DISC has the ability to discover new server locations with lower prices and good proximity to users.

D. Case 3: With Performance Constraints

If CDN providers have to meet performance constraints due to SLAs while driving down the deployment cost and inter-domain traffic, they can pick out some nodes and build them as multi-homed ones that connect to multiple ISPs. Obviously the deployment cost of these nodes are higher.

Although a city may be covered by several ISPs as shown in Fig. 2, CDN providers would not like to connect their multi-homed nodes to every ISP. This is partly due to the cost and partly due to the tiny market share of some ISPs. Therefore, we assume that if a node is built to be multi-homed, it is only connected to the three major ISPs in that country. Without loss of generality, we set the deployment cost of a multi-homed node to 50% more than a regular node.

Fig. 12 shows the trade-off between inter-domain traffic and the deployment cost under average latency constraints, when limiting the average latency to 15 ms or better. It can be seen

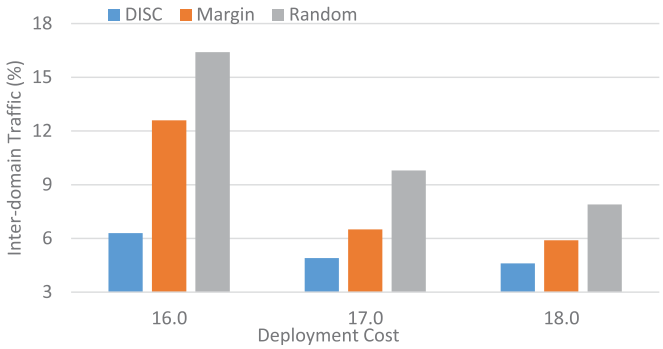

Fig. 13. Comparison among different methods with performance constraints.

that the more multi-homed nodes, the less inter-domain traffic, but higher the deployment cost.

Fig. 13 shows the comparison between DISC and existing approaches when limiting the average latency to 15 ms or better. Compared with the marginal-benefit-greedy methods and random-selections, DISC can reduce the inter-domain traffic by more than 50% and 61.6% respectively when the deployment cost is 16.0.

VII. Conclusion

Traditional methods for infrastructure deployment suffer from many limitations, such as the inability to minimize inter-domain traffic and the difficulty to put forward best suitable candidate sites. This paper presents a novel methodology called DISC, which converts the infrastructure deployment problem in the physical network space to a clustering problem in a network coordinate space and inversely converts the solution back to the physical network space. DISC quantifies the trade-offs of placement strategies that go beyond a fixed set of candidate sites given a priori, and significantly reduces inter-domain traffic, which to the best of our knowledge is considered the first time in deploying network infrastructures. To verify the effectiveness of DISC, we solve a practical node deployment problem in mainland China using DISC. The results show that DISC can reduce inter-domain traffic by up to 33%, the average latency by up to 8.5%, and the deployment cost by up to 18.9% compared with the best of state of art.

Future works include improving the consistency of network coordinate technique, introducing more realistic cost models and taking more practical application scenarios into consideration.

Appendix

Definition A.1: Space: $S = (U, f)$ is a space if and only if S satisfies the following three properties:

1) $\forall u, v \in U,\ f(u, v) \geq 0$
2) $\forall u, v \in U,\ f(u, v) = f(u, v)$
3) $\forall u, v \in U,\ f(u, v) = 0 \Leftrightarrow u = v$.

where U is the element set of S and f is the distance function between two elements.

Definition A.2: Given space $S = (U, f)$:

Cluster: c is called a cluster in S if it is a set of elements in U, that is $c \subset U$;

Clustriod: u_ζ is called the clustriod of c if $u_\zeta \in U$ $s.t.$ $max\ \{f(u_\zeta, u), \forall u \in c\} \leq max\ \{f(v, u), \forall u \in U, v \in c\}$. We denote them as $u_\zeta \odot c$.

Cluster Distance: c_1, c_2 are two clusters in space S, the distance between them is defined as $dist(c_1, c_2) = max\{f(u, v), \forall u \in c_1, v \in c_2\}$;

Cluster Set: C is called a cluster set in S if it is a set of clusters, that is for $\forall c \in C$, $c \subset U$.

Definition A.3: Consistent Spaces: Space $S = (U, f)$ and $S^{'} = (U^{'}, f^{'})$ are two consistent spaces, if and only if there exists a one-to-one mapping function $\varphi : U \rightarrow U^{'}$, $s.t.$ $f(u, v) \leq f(w, r) \Leftrightarrow f^{'}(\varphi(u), \varphi(v)) \leq f^{'}(\varphi(w), \varphi(r))$, for $\forall u, v, w, r \in U$. We denote them as $S \overset{\varphi}{\sim} S^{'}$.

Definition A.4: Given consistent spaces $S = (U, f)$ and $S^{'} = (U^{'}, f^{'})$, $S \overset{\varphi}{\sim} S^{'}$:

Consistent Clusters: cluster c in S and cluster $c^{'}$ in $S^{'}$ are consistent clusters if $c^{'} = \{\varphi(u), \forall u \in c\}$. We denote them as $c^{'} = \varphi(c)$;

Consistent Cluster Sets: cluster set C in S and $C^{'}$ in $S^{'}$ are consistent cluster sets if $C^{'} = \{\varphi(c), \forall c \in C\}$. We denote them as $C^{'} = \varphi(C)$;

Consistent Spaces have the following good properties. Given consistent spaces $S = (U, f)$ and $S^{'} = (U^{'}, f^{'})$, $S \overset{\varphi}{\sim} S^{'}$:

Property A.1: $\varphi(C_1 \cup C_2) = \varphi(C_1) \cup \varphi(C_2)$, where C_1, C_2 are cluster sets in S;

Proof: $\forall c \in \varphi(C_1 \cup C_2)$ we know $\varphi^{-1}(c) \in C_1 \cup C_2$ so, $\varphi^{-1}(c) \in C_1$ or $\varphi^{-1}(c) \in C_2$, we know $c \in \varphi(C_1)$ or $c \in \varphi(C_2)$ which means $c \in \varphi(C_1) \cup \varphi(C_2)$.
In the other hand, $\forall c \in \varphi(C_1) \cup \varphi(C_2)$, similarly, we have $c \in \varphi(C_1 \cup C_2)$. So $\varphi(C_1 \cup C_2 = \varphi(C_1 \cup C_2)$ ■

Property A.2: $\varphi(C_1 - C_2) = \varphi(C_1) - \varphi(C_2)$, where C_1, C_2 are cluster sets in S ;

Proof: Similarly to the proof of property A.1. ■

Property A.3: $dist(c_1, c_2) \leq dist(c_3, c_4) \Leftrightarrow dist(\varphi(c_1), \varphi(c_2)) \leq dist(\varphi(c_3), \varphi(c_4))$, where c_1, c_2, c_3, c_4 are arbitrary four clusters in S.

Proof: 1) $dist(c_1, c_2) \leq dist(c_3, c_4) \Rightarrow dist(\varphi(c_1), \varphi(c_2)) \leq dist(\varphi(c_3), \varphi(c_4))$:

According to Definition A.2, $\exists u_x \in c_1, u_y \in c_2$, $s.t.$ $f(u_x, u_y) = dist(c_1, c_2) \geq f(u_p, u_q)$, where $\forall u_p \in c_1, u_q \in c_2$. Based on Definition A.3, $f^{'}(\varphi(u_x), \varphi(u_y)) \geq f^{'}(\varphi(h_p), \varphi(h_q))$, thus $dist(\varphi(c_1), \varphi(c_2)) = f^{'}(\varphi(u_x), \varphi(u_y))$. Similarly, $\exists u_w \in c_3, u_z \in c_4$, $s.t.$ $dist(c_3, c_4) = f(u_w, u_z)$ and $dist(\varphi(c_3), \varphi(c_4)) = f^{'}(\varphi(u_w), \varphi(u_z))$.

If $dist(c_1, c_2) \geq dist(c_3, c_4)$, then $f(u_x, u_y) \geq f(u_w, u_z)$. Based on Definition A.3, $f^{'}(\varphi(u_x), \varphi(u_y)) \geq f^{'}(\varphi(u_w), \varphi(u_z))$, that is $dist(\varphi(c_1), \varphi(c_2)) \geq dist(\varphi(c_3), \varphi(c_4))$.

2) $dist(c_1, c_2) \leq dist(c_3, c_4) \Leftarrow dist(\varphi(c_1), \varphi(c_2)) \leq dist(\varphi(c_3), \varphi(c_4))$: It can be proved similarly.

Consolidated (1) and (2), the property has been proved. ■

Property A.4: If $h_\zeta \odot c$, where c is a cluster in S, then $\varphi(h_\zeta) \odot \varphi(c)$.

Proof: The property is equivalent to the statement that if $u_\zeta \in U$ $s.t.$ $max\ \{f(u_\zeta, u), \forall u \in c\} \leq max\ \{f(u, v), \forall u \in U, v \in c\}$, then $max\ \{f^{'}(\varphi(u_\zeta), u^{'}), \forall u^{'} \in \varphi(c)\} \leq max\ \{f^{'}(v^{'}, u^{'}), \forall u^{'} \in U^{'}, v^{'} \in \varphi(c)\}$.

It's reasonable to assume that $\exists u_x \in c$, $s.t.$ $f(u_\zeta, u_x) = max\{f(u_\zeta, u), \forall u \in c\}$, thus $f(u_\zeta, u_x) \geq f(u_\zeta, u_w)$, where $\forall u_w \in c$. According to Definition A.3, $f^{'}(\varphi(u_\zeta), \varphi(u_x)) \geq f^{'}(\varphi(u_\zeta), \varphi(u_w))$, thus $f^{'}(\varphi(u_\zeta), \varphi(u_x)) = max\{f^{'}(\varphi(u_\zeta), u^{'}), \forall u^{'} \in \varphi(c)\}$.

Similarly, $\exists u_\xi^{'} \in U^{'}$ $s.t.$ $max\ \{f^{'}(u_\xi^{'}, u^{'}), \forall u^{'} \in \varphi(c)\} \leq max\ \{f^{'}(u^{'}, v^{'}), \forall u^{'} \in U^{'}, v^{'} \in \varphi(c)\}$. Without loss of generality, $f^{'}(u_\xi^{'}, u_y^{'}) = max\{f^{'}(u_\xi^{'}, u^{'}), \forall u^{'} \in \varphi(c)\}$, where $u_y^{'} \in \varphi(c)$. What's more, $f(u_\xi, u_y) = max\{f(u_\xi, u), \forall u \in c\}$, where $u_\xi, u_y \in c$, $\varphi(u_\xi) = u_\xi^{'}$ and $\varphi(u_y) = u_y^{'}$.

Now what should be proved is $u_\xi^{'} = \varphi(u_\zeta)$, that is $u_\xi = u_\zeta$. Assume $u_\xi^{'} \neq \varphi(u_\zeta)$, there would exist $f(u_\zeta, u_x) \leq f(u_\xi, u_y)$ while $f^{'}(u_\xi^{'}, u_y^{'}) < f^{'}(\varphi(u_\zeta), \varphi(u_x))$, which contradicts with Definition A.3. So the assumption is not established, and $max\ \{f^{'}(\varphi(u_\zeta), u^{'}), \forall u^{'} \in \varphi(c)\} \leq max\ \{f^{'}(u^{'}, v^{'}), \forall u^{'} \in U^{'}, v^{'} \in \varphi(c)\}$.

In summary, the property is proved. ■

Based on the definitions and the properties, we can draw the conclusion that if we had obtained the distance between arbitrary two hosts in Network Space (hereinafter called NS), and use *Clustering Engine* to cluster these hosts (replace the parameters in Coordinate Space (hereinafter called CS) with corresponding parameters in NS), the result would be consistent with the result in CS. In detail, it can be stated by the following theorems.

Theorem A.1: CS is a space, and NS is also a space if not considering the asymmetry of the Internet.

Theorem A.2: If the consistent extent of the whole network equal to 1, $NS \overset{\varphi}{\sim} CS$, where φ is the Transform Engine.

The above two theorems can be easily understood according to the Definition A.1 or Definition A.3, thus not giving the proof in detail.

Theorem A.3: If $NS \overset{\varphi}{\sim} CS$, hosts are clustered (Algorithm 1) in both spaces meanwhile, the cluster set C in CS and the cluster set $C^{'}$ in NS are consistent cluster sets.

Proof: The theorem can be demonstrated using mathematical induction method. Assume C_k ($C_k^{'}$) is the cluster set after the k round of Line 4-12 in CS (NS), where $k = 0, 1, ..., N - n_c$, the theorem is equivalent to $C_{N-n_c}^{'} = \varphi(C_{N-n_c})$.

Basis: Show that the statement holds for k = 0.

Based on Line 4, $C_0 = \{\{h_l\} | l = 1, 2, ..., N\}$, and $C_0^{'} = \{\{h_l^{'}\} | l = 1, 2, ..., N\}$, where $h_l = \varphi(h_l)$. Obviously, $C_0^{'}$=$\varphi(C_0)$, so the statement is true for k = 0.

Inductive step: Show that if the statement holds for $k = m$, then also holds for $k = m + 1$. This can be done as follows.

Assume the statement holds for $k = m$, that is $C_m^{'} = \varphi(C_m)$. Suppose in the m loop of Line 7-11, the closest two clusters in cluster set C_m is c_i, c_j, then $dist(c_i, c_j) \leq dist(c_p, c_q)$, $\forall c_p, c_q \in C_k$. According to Property A.3, $dist(c_i^{'}, c_j^{'}) \leq dist(c_p^{'}, c_q^{'})$, where $c_i^{'} = \varphi(c_i), c_j^{'} = \varphi(c_j), c_p^{'} = \varphi(c_p)$, and $c_q^{'} = \varphi(c_q)$, so $c_i^{'}, c_j^{'}$ are the closest two clusters in cluster set $C_m^{'}$.

Then $C_{m+1} = (C_m - (\{c_i\} \cup \{c_j\})) \cup \{c_i \cup c_j\}$, while $C_{m+1}^{'} = (C_m^{'} - (\{c_i^{'}\} \cup \{c_j^{'}\})) \cup \{c_i^{'} \cup c_j^{'}\}$.

According to Property A.1 and Property A.2, $C_{m+1}^{'} = \varphi(C_{m+1})$, thereby showing that the statement holds for $k = m + 1$.

Since both the basis and the inductive step have been performed, by mathematical induction, the statement holds $k = N - n_c$, that is $C'_{N-n_c} = \varphi(C_{N-n_c})$. ■

Theorem A.4: If the cluster set C in CS and the cluster set C' in NS are consistent cluster sets, cluster S (or S') is the set of clustriods from each cluster in C (or C') (Algorithm 2), S and S' are consistent cluster sets.

Proof: According to the Definition A.2, $S = \{p_i | \forall c_i \in C, p_i \odot c_i\}$, and $S' = \{p'_i | \forall c'_i \in C', p'_i \odot c'_i\}$. From Property A.4, $\varphi(S) = \{\varphi(p_i) | \forall c_i \in C, p_i \odot c_i\} = \{p'_i | \forall c'_i \in C', p'_i \odot c'_i\} = S'$, where $p'_i = \varphi(p_i), c'_i = \varphi(c_i)$. So the statement is proved. ■

REFERENCES

[1] J. Song, F. Yang, Y. Zhou, S. Wan, and H. R. Wu, "QoE evaluation of multimedia services based on audiovisual quality and user interest," *IEEE Trans. Multimedia*, vol. 18, no. 3, pp. 444–457, Mar. 2016.

[2] H. Yin *et al.*, "NetClust: A framework for scalable and pareto-optimal media server placement," *IEEE Trans. Multimedia*, vol. 15, no. 8, pp. 2114–2124, Dec. 2013.

[3] "Cisco Visual Networking Index: Forecast and methodology, 2014-2019," Cisco Systems, Inc., San Jose, CA, USA, 2015, Accessed on: May 29, 2016. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.html

[4] M. Calder *et al.*, "Mapping the expansion of googles serving infrastructure," in *Proc. ACM Conf. Internet Meas. Conf.*, 2013, pp. 313–326.

[5] Z. Wang, W. Zhu, M. Chen, L. Sun, and S. Yang, "CPCDN: Content delivery powered by context and user intelligence," *IEEE Trans. Multimedia*, vol. 17, no. 1, pp. 92–103, Jan. 2015.

[6] Z. Li, Q. Wu, K. Salamatian, and G. Xie, "Video delivery performance of a large-scale VoD system and the Implications on content delivery," *IEEE Trans. Multimedia*, vol. 17, no. 6, pp. 880–892, Jun. 2015.

[7] R. V. Brandenburg, L. Peterson, and B. Davie, "Framework for content distribution network interconnection (CDNI)," RFC 7336, Oct. 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7336.txt

[8] C. Guo *et al.*, "BCube: A high performance, server-centric network architecture for modular data centers," in *Proc. ACM SIGCOMM*, 2009, pp. 63–74 .

[9] B. Ager *et al.*, "Anatomy of a large European IXP," in *Proc. SIGCOMM*, pp. 163–174.

[10] V. Valancius, N. Laoutaris, L. Massoulié, C. Diot, and P. Rodriguez, "Greening the Internet with NANO data centers," in *Proc. 5th Int. Conf. Emerg. Netw. Exp. Technol.*, 2009, pp. 37–48. [Online]. Available: http://doi.acm.org/10.1145/1658939.1658944

[11] L. Anegekuh, L. Sun, E. Jammeh, I. H. Mkwawa, and E. Ifeachor, "Content-based video quality prediction for HEVC encoded videos streamed over packet networks," *IEEE Trans. Multimedia*, vol. 17, no. 8, pp. 1323–1334, Aug. 2015.

[12] H. Hu *et al.*, "Social TV analytics: A novel paradigm to transform TV watching experience," in *Proc 5th ACM Multimedia Syst. Conf.*, 2014, pp. 172–175. [Online]. Available: http://doi.acm.org/10.1145/2557642.2579373

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

[14] B. Li, M. Golin, G. Italiano, X. Deng, and K. Sohraby, "On the optimal placement of web proxies in the Internet," in *Proc. IEEE INFOCOM*, Mar. 1999, vol. 3, pp. 1282–1290.

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

[16] P. Radoslavov, R. Govindan, and D. Estrin, "Topology-informed internet replica placement," *Comput. Commun.*, vol. 25, no. 4, pp. 384–392, Mar. 2002. [Online]. Available: http://dx.doi.org/10.1016/S0140-3664(01)00410-8

[17] E. Cronin *et al.*, "Constrained mirror placement on the Internet," *IEEE J. Select. Areas Commun.*, vol. 20, no. 7, pp. 1369–1382, Sep. 2002.

[18] C. Huang, A. Wang, J. Li, and K. W. Ross, "Measuring and evaluating large-scale CDNs," in *Proc. Internet Meas. Conf.*, 2008, pp. 15–29.

[19] 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.*, Dec. 2012, pp. 588–595.

[20] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, "Cutting the electric bill for internet-scale systems," in *Proc. ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 123–134, 2009.

[21] S. Hasan, S. Gorinsky, C. Dovrolis, and R. Sitaraman, "Trade-offs in optimizing the cache deployments of CDNS," in *Proc. IEEE INFOCOM*, Apr. 2014, pp. 460–468.

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

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

[24] 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.

[25] N. Ball and P. Pietzuch, "Distributed content delivery using load-aware network coordinates," in *Proc. ACM CoNEXT Conf.*, 2008, pp. 77:1–77:6.

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

[27] 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, pp. 20:1–20:21, May 2012. [Online]. Available: http://doi.acm.org/10.1145/2168996.2169000

[28] H. Yin, X. Liu, G. Min, and C. Lin, "Content delivery networks: A bridge between emerging applications and future IP networks," *IEEE Netw.*, vol. 24, no. 4, pp. 52–56, Jul./Aug. 2010.

[29] T. Ng and H. Zhang, "Predicting Internet network distance with coordinates-based approaches," in *Proc. IEEE INFOCOM*, Jun. 2002, vol. 1, pp. 170–179.

[30] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," in *Proc. SIGCOMM*, 2004, pp. 15–26.

[31] 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.

[32] H. V. Madhyastha *et al.*, "iPlane: An information plane for distributed services," in *Proc. 7th Symposium Oper. Syst. Des. Implementation*, 2006, pp. 367–380.

[33] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 1, pp. 21–27, Jan. 1967.

[34] E. Katz-Bassett *et al.*, "Towards IP geolocation using delay and topology measurements," in *Proc. Internet Meas. Conf.*, 2006, pp. 71–84.

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

[36] H. Yin, X. Liu, G. Min, and C. Lin, "Content delivery networks: A bridge between emerging applications and future IP networks," *IEEE Netw.*, vol. 24, no. 4, pp. 52–56, Jul. 2010.

[37] Y. A. 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*, Apr. 2011, pp. 2372–2380.

[38] S. Jamin *et al.*, "On the placement of Internet instrumentation," in *Proc. IEEE INFOCOM*, Mar. 2000, vol. 1, pp. 295–304.

Hao Yin received the B.S., M.E., and Ph.D. degrees in electrical engineering from the Huazhong University of Science and Technology, Wuhan, China, in 1996, 1999, and 2002, respectively.

He is currently a Professor with the Research Institute of Information Technology, Tsinghua University, Beijing, China. He is also the Vice-Director of Industry Innovation Center for Future Network, China, and the Secretary-General of Industry Innovation Alliance of Future Internet, China. He was elected as the New Century Excellent Talent of the Chinese Ministry of Education in 2009. His research interests include span broad aspects of multimedia communication and computer networks.

Prof. Yin was the recipient of the Chinese National Science Foundation for Excellent Young Scholars in 2012.

Xu Zhang received the B.S. degree in communication engineering from the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, China, in 2012, and is currently working toward the Ph.D. degree in computer science and technology at Tsinghua University, Beijing, China.

His research interests include content delivery networks, network measurement, and multimedia communications.

Shuoyao Zhao received the B.S. degree in computer science from the Institute for Interdisciplinary Information Science, Tsinghua University, Beijing, China, in 2014.

His research interests include network measurement and network science.

Yan Luo (S'00–A'00–M'05) received the Ph.D. degree in computer science from the University of California Riverside, Riverside, CA, USA, in 2005, and the B.E. and M.E. degrees from Huazhong University of Science and Technology, Wuhan, China.

He is currently a Professor with the Department of Electrical and Computer Engineering, University of Massachusetts Lowell, Lowell, MA, USA. His current projects focus on heterogeneous architecture and systems, software defined networks, and deep learning. His research interests include broadly computer architecture and network systems.

Prof. Luo is a Member of ACM. He has served on the Program Committee of numerous international conferences and as a Guest Editor and referee of premier journals.

Chen Tian received the B.S., M.S., and Ph.D. degrees from the Huazhong University of Science and Technology, Wuhan, China, in 2000, 2003, and 2008, respectively.

He is currently an Associate Professor with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China. He was previously an Associate Professor in the School of Electronics Information and Communications, Huazhong University of Science and Technology. From 2012 to 2013, he was a Postdoctoral Researcher with the Department of Computer Science, Yale University, New Haven, CT, USA. His research interests include data center networks, network function virtualization, distributed systems, Internet streaming, and urban computing.

Vyas Sekar received the B.S. degree in computer science and technology from the Indian Institute of Technology, Madras, India, in 2003, and the Ph.D. degree in computer science from Carnegie Mellon University, Pittsburgh, PA, USA, in 2010.

He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Carnegie Mellon University. His research interest include networking, security, and systems, including the design and management of network appliances or "middleboxes," various aspects of network and systems security, content/video delivery systems, and network monitoring and measurement.

Prof. Sekar was the recipient of the President of India Gold Medal at the Indian Institute of Technology, Madras, India. He is also the recipient of Best Paper Awards at ACM SIGCOMM and ACM Multimedia.