Dynamic Service Placement in Geographically Distributed Clouds

Qi Zhang, Quanyan Zhu, Mahamed Faten Zhani, Raouf Boutaba, and Joseph L. Hellerstein

***Abstract*—Large-scale online service providers have been increasingly relying on geographically distributed cloud infrastructures for service hosting and delivery. In this context, a key challenge faced by service providers is to determine the locations where service applications should be placed such that the hosting cost is minimized while key performance requirements (e.g., response time) are ensured. Furthermore, the dynamic nature of both demand pattern and infrastructure cost favors a dynamic solution to this problem. Currently most of the existing solutions for service placement have either ignored dynamics, or provided solutions inadequate to achieve this objective. In this paper, we present a framework for dynamic service placement problems based on control- and game-theoretic models. In particular, we present a solution that optimizes the hosting cost dynamically over time according to both demand and resource price fluctuations. We further consider the case where multiple service providers compete for resources in a dynamic manner. This paper extends our previous work [1] by analyzing the outcome of the competition in terms of both price of stability and price of anarchy. Our analysis suggests that in an uncoordinated scenario where service providers behave in a selfish manner, the resulting Nash equilibrium can be arbitrarily worse than the optimal centralized solution in terms of social welfare. Based on this observation, we present a coordination mechanism that can be employed by the infrastructure provider to maximize the social welfare of the system. Finally, we demonstrate the effectiveness of our solutions using realistic simulations.**

***Index Terms*—Cloud computing, resource management, model predictive control.**

I. INTRODUCTION

CLOUD computing has become a cost-effective model for delivering large-scale services over the Internet in recent years [2]. In a Cloud computing environment, Infrastructure Providers (InPs, also known as cloud providers), build large data centers in geographically distributed locations to achieve reliability while minimizing operational cost. The Service Providers (SPs), on the other hand, leverage geo-diversity of data centers to serve customers from multiple geographical regions. Today, large companies like Google, Yahoo and Microsoft have already adopted this model in their private clouds, offering a broad range of services to millions of users world-wide. As Cloud computing technologies become mature, more companies are expected to adopt this model by moving into clouds.

A key technique adopted by SPs in cloud service management is to distribute servers in multiple data centers in order to meet the performance requirements specified in Service Level Agreements (SLA) contracted to their customers (i.e., end users), while reducing operational costs by optimizing the placement of servers in multiple data centers. This typically involves solving two problems jointly: (1) deciding on the number of servers placed in each data center, and (2) routing each request to appropriate servers to minimize response time. As InPs typically offer on-demand and elastic resource access, it is possible to adjust the number of servers to dynamically match service demand in an agile and responsive manner.

Another factor that needs to be considered in making placement decisions is the fact that, the price of resources offered by InPs are also subject to change. In particular, energy consumption is a major contributor to the operational cost of a data center [3]. In many parts of the U.S., the electricity grid of each region is managed independently by a Regional Transmission Organization (RTO) which operates wholesales electricity markets in order to match supply and demand for electricity, as illustrated in Fig. 1. As a result, electricity prices in each region can vary independently over time. Based on this fact, recently there have been several studies on dynamic server placement [4], [5] and request dispatching [6] in private clouds, taking into account fluctuating energy costs. The same benefit of geographical load balancing can be achieved in public clouds by introducing some degree of dynamic pricing [7], such as the one used by Amazon EC2 [8]. Combining the above observations, a SP is facing the problem of dynamically controlling the number of servers placed in each data center to minimize the total resource cost while satisfying SLA requirements, taking into consideration the fluctuation of both demand and resource price. We call this problem *dynamic service placement problem (DSPP)*. This problem shares many similarities with traditional replica placement problem studied (e.g., [9]), however, the price fluctuation is often neglected in the existing literature. Recently, there are several papers that have studied the problem of price-aware geographical load balancing (e.g., [5]). However, the dynamic aspect of the problem, particularly the reconfiguration cost, is still largely unaddressed. However, in addition to achieve high responsiveness to demand fluctuation, one must also take into account the cost of reconfiguration (i.e., the cost of adding and removing servers). The consideration of reconfiguration cost is important for ensuring system stability and minimum

Manuscript received DATE; revised DATE.

Q. Zhang and M. F. Zhani are with the David R. Cheriton School of Computer Science, University of Waterloo, Canada (e-mail: {q8zhang, mfzhani}@uwaterloo.ca).

R. Boutaba is with the Division of IT Convergence Engineering, Pohang University of Science and Technology, Pohang, Korea (e-mail: rboutaba@uwaterloo.ca).

Q. Zhu is with the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Champaign, IL, 61820, USA (e-mail: zhu31@illinois.edu).

J. L. Hellerstein is with Google Inc., Seattle, WA, 98144, USA (e-mail: jlh@google.com).

Digital Object Identifier 10.1109/JSAC.2013.1308.XX

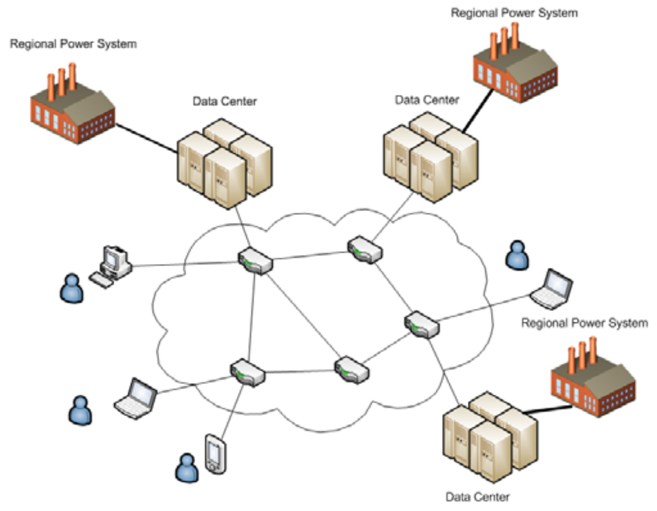

Fig. 1. Service placement in geographically distributed data centers.

management overhead. For instance, adjusting the number of servers typically incur switching costs for setup (e.g., VM image distribution) and tear-down (e.g., saving data and states of the server to storage devices). On the other hand, stability is perhaps a more important objective. During our experiments, we have found a reconfiguration cost-oblivious algorithm can cause frequent oscillation of server assignment as resource price changes overtime. As SPs strive to reduce operational cost while improving the agility of their services, it is a major challenge to dynamically adjust the resource allocations while considering the reconfiguration costs.

In this paper, we study the DSPP problem using both control and game theoretic methods. Specifically, we first propose a control framework based on Model Predictive Control (MPC) approach to provide an online adaptive control mechanism which aims at reducing service provider costs, namely, resource allocation and reconfiguration costs. We further extend this framework to a game-theoretic model to consider the competition among multiple SPs, taking into account the capacity constraint of each data center. This model is realistic for several reasons: (1) on-demand resource allocation mechanisms can often lead to situations where resource demand exceeds the capacity available in a data center (e.g., during holiday seasons). (2) Recently, there are numerous proposals that advocate for small-scale data centers (e.g., [10]). In both cases, limited data center capacity can result in some SPs not getting the resources they desire. As a result, we analyze the outcome of resource competition. As an extension of our previous work [1], we show that while there exists an optimal Nash Equilibrium (NE) [11] that maximizes the social welfare in the MPC setting (i.e., every SP forecasts the future demand and uses this information for dynamic control of service placement), the worst NE can be far from optimal. To address this issue, we present an algorithm that can be implemented by the InP to attain the optimal NE. Finally, we evaluate the performance of our solutions using realistic simulations.

The rest of the paper is organized as follows: Section II surveys the related work. Section III presents the DSPP framework for a single SP. Section IV-A describes the problem formulation of DSPP for the case of a single SP. The design of our controller for DSPP is provided in Section V. Section VI extends the framework to a multi-provider scenario and analyze the outcome of the resource competition game. We present our experiment results in Section VII, and conclude the paper in Section VIII.

II. Related Work

Service placement in large-scale shared service hosting infrastructures has been studied in many contexts in the past. Early works on this problem primarily focused on placing content replicas in the context of Content Delivery Networks (CDNs) (e.g. [12]). However, most of the work in that context have addressed the centralized cases where the demand profile is static or time invariant. More recent studies have also investigated dynamic cases [13] where iterative improvement algorithms are proposed. Laoutaris et. al. formulated the service placement problem as an Uncapacitated Facility Location Problem (UFLP) [14], and presented a local search heuristic algorithm for optimizing the placement of services over time. However, the objective of these studies is to ensure the algorithm converges to a near optimal solution for a static topology in a finite number of iterations, instead of optimizing the overall system performance in the presence of demand and resource dynamics. Furthermore, the cost of dynamic reconfiguration is not considered in these studies.

With the growth of large-scale data center infrastructures, energy consumption has recently become an acute problem. Not only does energy consumption account for a significant fraction of data center operating cost, it also raises concerns regarding environmental impact and sustainability of these infrastructures. Driven by the fact that electricity grids are independently managed in different geographical regions, several studies have also exploited geo-diversity to achieve energy cost reduction. For instance, Qureshi et. al. [6] provided a detailed analysis of regional electricity markets and have shown that energy-aware request routing can achieve significant cost savings for large CDNs. Following this line of research, Rao et. al. [4] studied the problem of server placement in a multi-electricity market environment with the goal of minimizing electricity cost. Liu et. al. [5] presented a distributed solution for the same problem, taking into consideration both request response time and energy cost. However, these studies have only considered static cases. Lastly, the application of control theory to capacity provisioning in data centers has been studied recently, primarily in the context of autonomic computing. Kusic et. al. [15] presented a control-theoretic framework for reducing energy consumption while satisfying SLA constraints. However, their work only applies to intra-data center environments (i.e., inside a data center), while the impact of geographical location was not considered.

III. System Architecture and Design

We consider a multi-regional cloud environment that consists of multiple data centers situated at different geographical locations. Our system architecture consists of 4 components as depicted in Fig. 2: (1) request routers, (2) monitoring

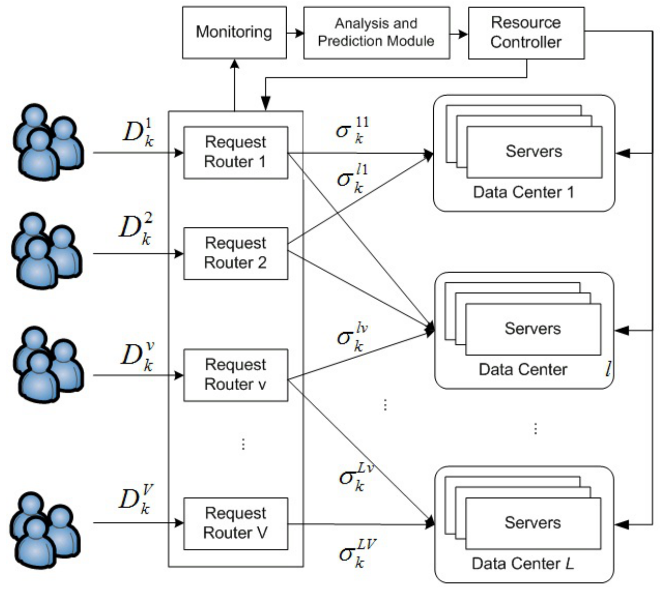

Fig. 2. System architecture for a single service provider.

module, (3) analysis and prediction module, and (4) resource controller. Both the request router and the monitoring module can be directly owned by the SP, or leased from other service providers who offer them as services. In particular, the service provider controls *request routers* (a.k.a. redirectors) which are responsible for redirecting the requests to appropriate servers [16]. In practice, request redirection can play a key role in improving server accessibility through load balancing, latency minimization and content replication. For instance, Amazon EC2 Elastic Load Balancing service [17] is an example of a simplified request router. More sophisticated designs (e.g., [16]) have also been studied in the literature. The *monitoring module* is responsible for collecting statistics, including the number of requests received (i.e., the demand) at the different request routers and the prices offered by each data center. The *analysis and prediction module* models the dynamics of demand and price fluctuations, and forecasts their future values. In practice, it has been shown that both demand and price in production data centers generally show daily fluctuation patterns [5], [6]. It is worth noting that our framework is not restricted to any particular prediction technique. In practice, the prediction technique should depend on the application characteristics. For example, recent work shows that the Auto-Regressive Integrated Moving Average (ARIMA) model [18] is sufficient to achieve high prediction accuracy for realistic HTTP workloads [19]. Finally, the *resource controller* is responsible for solving DSPP and making online control decisions at run time. It dynamically adjusts the number of servers leased in each data center in order to satisfy the SLA requirements (in terms of response time), while minimizing the resource rental cost. Furthermore, it informs the request routers about the number of servers allocated in each data center. The request routers must then find appropriate assignment of demand to the allocated servers. In our system architecture, request router adopts a simple strategy which is to split demand proportionally among the servers that satisfy the SLA requirements. We will formally define the demand assignment model in Section IV-C. In the next section, we will first provide a mathematical model for DSPP, and then describe the request assignment policy employed by each request router.

IV. Problem Formulation

We model the network as a bipartite graph $G = (L \cup V, E)$, where L denotes the set of data centers, V denotes the location of customers. For instance, V can be the set of access networks to which customers are connected. Denote by $E \subseteq L \times V$ the communication paths between customers and data centers. We also assign constant weights d_{lv} to denote the network latency between a data center $l \in L$ and a client location $v \in V$.

In our framework, we consider a discrete-time system model where time is divided into multiple time periods called *reconfiguration periods* corresponding to the timescale at which server placement and routing decisions are made. We assume that there is an interval of interest $\mathcal{K} = \{0, 1, 2, ..., K\}$ that consists of $K+1$ periods. Let $N = \{1, 2, ..., n\}$ denote the set of SPs. We assume that at time $k \in \mathcal{K}$, each customer location $v \in V$ has demand D_k^v in terms of average arrival rate of requests from location v at time k. For simplicity, we assume that all the servers leased by each SP have identical size and functionality. For instance, a server can be a virtual machine (VM) that runs a specific application image. We define the state variable $x_k^l \in \mathbb{R}_+$ as the number of servers owned by the SP at location $l \in L$ at time k. To simplify the model, we assume that x_k^l can take continuous values rather than discrete values. This assumption is reasonable for large-scale services that require tens or hundreds of servers, where the weight of each individual server in the overall solution is small. In this case, we can always obtain a feasible solution by rounding up the continuous values to the nearest integer values. Based on this assumption, we can further decouple x_k^l by defining $x_k^{lv} \in \mathbb{R}_+$ as the number of servers at location l serving demand from $v \in V$:

$$x_k^l = \sum_{v \in V} x_k^{lv}, \quad \forall l \in L, 0 \leq k \leq K. \tag{1}$$

Let $u_k^{lv} \in \mathbb{R}$ denote the change in the number of servers in x_k^{lv} at time k, we then have:

$$x_{k+1}^{lv} = x_k^{lv} + u_k^{lv}, \quad \forall l \in L, v \in V, 0 \leq k \leq K. \tag{2}$$

A. Modeling the Server Allocation and Reconfiguration Cost

To model the cost of server allocation, we assume that there is a price p_k^l for running a server at data center $l \in L$ at time k. The total resource cost H_k for service hosting at time k is

$$H_k = \sum_{l \in L} x_k^l p_k^l = \sum_{l \in L} \sum_{v \in V} x_k^{lv} p_k^l, \qquad \forall 0 \leq k \leq K \tag{3}$$

We also assume that there is a convex function $S : \mathbb{R} \to \mathbb{R}_+$ that computes the cost of reconfiguration. For instance, we can define $S(\cdot)$ as a quadratic function $S(u_k^{lv}) = c^l (u_k^{lv})^2$. Quadratic penalty functions are widely used in control theory literature, as they penalize rapid reconfiguration of system states. The actual value of the constant c^l is usually determined experimentally by finding a fair tradeoff between convergence

TABLE I
TABLE OF NOTATIONS

Symbol	Meaning
x_k^l	Num. of servers at DC l at time k
x_k^{lv}	Num. of servers at l serving demand from v at time k
D_k^v	Avg. demand arrival rate originated from v
σ_k^{lv}	Avg. arrival rate of demand from v to DC l at time k
u_k^{lv}	Change in the number of servers at DC l at time k
λ_k^{lv}	Avg. arrival rate to each server from v to l at time k
d_{lv}	Network latency between location v and data center l
μ	Request process rate of a single server
p^l	Price of each server at DC l
C^l	Capacity of DC l
H_k	Resource allocation cost at time k
G_k	Reconfiguration cost at time k
J	Total operational cost

rate and reconfiguration cost [20]. In this case, the total reconfiguration cost is:

$$G_k = \sum_{l\in L}\sum_{v\in V} S(u_k^{lv}) = \sum_{l\in L}\sum_{v\in V} c^l (u_k^{lv})^2, 0 \le k \le K. \quad (4)$$

However our system can also be extended to adopt any other convex reconfiguration cost functions.

B. Modeling the Constraints

While minimizing the total operational cost, the allocation of servers and demand assignment must satisfy a set of constraints, including (1) demand constraint and (2) SLA performance constraint. Define σ_k^{lv} as the demand arrival rate from v assigned to data center l at time k, the demand constraint ensures that all demands are satisfied:

$$\sum_{l\in L} \sigma_k^{lv} = D_k^v, \quad \forall v \in V,\ l \in L,\ 0 \le k \le K. \quad (5)$$

In addition, there is a SLA performance constraint that specifies a maximum delay $\bar{d}_{lv}$ that the SP tries to achieve between a location v and a data center l. We focus on modeling this constraint in the rest of this subsection. For data center $l \in L$, we assume that the demand σ_k^{lv} arriving from location v is equally split among the local servers x_k^{lv}. Let $\lambda = \frac{\sigma_k^{lv}}{x_k^{lv}}$ denote the arrival rate of requests for each server. To simplify the discussion, we model the behavior of a single server using the standard $M/M/1$ queueing model. However, we believe it is straightforward to adapt our framework to other queueing models as well. Thus, the queueing delay between location $v \in V$ to a server at $l \in L$ can be computed as:

$$q(x_k^{lv}, \sigma_k^{lv}) = \frac{1}{\mu - \lambda} = \frac{1}{\mu - \frac{\sigma_k^{lv}}{x_k^{lv}}}, \quad (6)$$

where μ is the service rate of each server. We aim to ensure that for any $(v,l) \in E$ with $\sigma_k^{lv} > 0$, the average delay (i.e., the sum of propagation and queuing delay) is less than $\bar{d}_{lv}$[1]:

$$d_{lv} + q(x_k^{lv}, \sigma_k^{lv}) \le \bar{d}_{lv}, \quad \forall v \in V, l \in L, 0 \le k \le K. \quad (7)$$

By defining the constant

$$a^{lv} = \begin{cases} \frac{1}{\mu - (\bar{d}_{lv} - d_{lv})^{-1}}, & \text{if } \bar{d}_{lv} - d_{lv} > 0, \\ \infty, & \text{otherwise ,} \end{cases} \quad (8)$$

we can rewrite the constraint (7) as:

$$x_k^{lv} \ge a^{lv} \sigma_k^{lv}, \quad \forall v \in V, l \in L. \quad (9)$$

We can combine constraints (5) and (9) to eliminate σ_k^{lv}:

$$\sum_{l\in L} \frac{x_k^{lv}}{a^{lv}} \ge D_k^v, \quad \forall v \in V, 0 \le k \le K. \quad (10)$$

C. Modeling the Demand Assignment

We can define the demand assignment policy for each request router as:

$$\sigma_k^{lv} = D_k^v \cdot \frac{\frac{x_k^{lv}}{a^{lv}}}{\sum_{l\in L} \frac{x_k^{lv}}{a^{lv}}}. \quad (11)$$

Imposing constraint (11) implies that SLA requirement is met by all request routers. In practice, each request router for location $v \in V$ can implement the policy by splitting the demand D_k^v proportionally according to equation (11) using any standard load balancing technique.

D. DSPP Formulation

Given the system model described above, the goal of DSPP is to minimize the total cost of server allocation and reconfiguration cost. Based on (3) and (4), the goal of DSPP is to satisfy constraints (10) and (2) while minimizing the following objective function:

$$J := \sum_{k=0}^{K} H_k + G_k = \sum_{k=0}^{K}\sum_{v\in V}\sum_{l\in L} x_k^{lv} p_k^{lv} + c^l (u_k^{lv})^2.$$

Define $\mathbf{x}_k = [x_k^{11}, ...x^{L1}, ..., x_k^{lv}, ..., x_k^{LV}]^\top \in \mathbb{R}_+^{LV}$, $\mathbf{p}_k^{'} = [p_k^1, p_k^2, ..., p_k^L] \in \mathbb{R}_+^L$, $\mathbf{p}_k = [\mathbf{p}_k^{'}, \mathbf{p}_k^{'}, ...\mathbf{p}_k^{'}]^\top \in \mathbb{R}_+^{LV}$, $\mathbf{u}_k = [u_k^{11}, ...u^{L1}, ..., u_k^{lv}, ..., u_k^{LV}]^\top \in \mathbb{R}^{LV}$, $\mathbf{a}_k^v = [\frac{1}{a_k^{1v}}, \frac{1}{a_k^{2v}}, ..., \frac{1}{a_k^{Lv}}]^\top$, $\mathbf{a}_k = \text{diag}^{-1}\{\mathbf{a}_k^1, ..., \mathbf{a}_k^V\} \in \mathbb{R}_+^{LV\times V}$, $\mathbf{R}^{'} = [c^1, c^2, ..., c^L] \in \mathbb{R}_+^L$, $\mathbf{R} = \text{diag}\{[\mathbf{R}^{'}, \mathbf{R}^{'}, ..., \mathbf{R}^{'}]\} \in \mathbb{R}_+^{LV\times LV}$, $\mathbf{D}_k = [D_k^1, ..., D_k^V]^\top$, $\mathbf{C} = [C^1, C^2, ...C^L]^\top \in \mathbb{R}_+^L$,

[1]It should be pointed out that even though our model focuses on guaranteeing the average delay, it is straightforward to extend it to handle more general cases, such as ϕ-percentile delay (where ϕ is typically 95%). According to queueing theory, the ϕ-percentile delay of an $M/M/1$ queue can be computed as $q = \frac{\ln(\frac{1}{1-\phi})}{\mu-\lambda}$, assuming $\mu > \lambda$. Thus the only change that needs to be made is to multiply $q(x_k^{lv}, \sigma_k^{lv})$ by a constant factor $\ln\left(\frac{1}{1-\phi}\right)$.

Algorithm 1 MPC Algorithm for DSPP

```
Provide initial state $\mathbf{x}_0$, $k \leftarrow 0$
loop
  At beginning of control period $k$:
    Predict $\mathbf{D}^l_{k+t|k}$ for horizons $t = 1, \cdots, W$ using a demand prediction model
    Solve DSPP to obtain $\mathbf{u}_{k+t|k}$ for $t = 0, \cdots, W-1$
    Change the resource allocation according to $\mathbf{u}_{k|k}$
    Update demand assignment policy of request routers according to equation (11)
    $k \leftarrow k+1$
end loop
```

$\mathbf{s} = [\mathbf{I}^{L\times L}, ..\mathbf{I}^{L\times L}]^\top \in \mathbb{R}_+^{LV\times L}$, we can rewrite DSPP as:

$$\begin{aligned}
\min_{\{\mathbf{u}_0,..,\mathbf{u}_{K-1}\}} \quad & J = \sum_{k=1}^{K} \mathbf{p}_k^\top \mathbf{x}_k + \sum_{k=0}^{K-1} \mathbf{u}_k^\top \mathbf{R} \mathbf{u}_k \\
\text{s.t.} \quad & \mathbf{a}_k^\top \mathbf{x}_k \succeq \mathbf{D}_k, \qquad \forall 0 \le k \le K, \\
& \mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{u}_k, \qquad \forall 0 \le k \le K-1, \\
& \mathbf{x}_k \in \mathbb{R}_+^{LV}, \mathbf{u}_k \in \mathbb{R}^{LV}, \quad \forall 0 \le k \le K-1.
\end{aligned}$$

V. Controller Design for DSPP

The DSPP formulation described in the previous section is a linear-quadratic program that can be solved using standard methods [21]. Even though DSPP can be solved optimally, in practice, the resource controller must solve this problem in an online setting where future demand is unknown. In this case, we use the Model Predictive Control (MPC) framework which is widely used for solving online control problems. Algorithm 1 is our MPC algorithm used by the resource controller for solving DSPP online. It can be described as follows. At time k, the resource controller predicts the future demand $\mathbf{D}^v_k$ for multiple periods $[k+1, ..., k+W]$, where W is the prediction horizon. Denote by $\mathbf{D}^v_{k+t|k}$ the demand predicted for time $k+t$ at time k. The controller then solves the optimization problem for the horizon $[k, ..., k+W]$, starting with the initial state $\mathbf{x}^{lv}_{k|k} = \mathbf{x}^{lv}_k$. Even though the solution of the optimization problem will contain a set of values $\mathbf{u}^{lv}_{k|k}, ..., \mathbf{u}^{lv}_{k+W-1|k}$, the controller will only execute the first step in sequence $\mathbf{u}^{lv}_{k|k}$. When the next period $k+1$ starts, the same procedure is repeated by the controller. Using this algorithm, the controller can effectively adjust the number of servers in each data center.

However, simply applying MPC framework is not sufficient to guarantee the satisfaction of SLA, as an underestimation of future demand (e.g., unexpected demand spikes) can lead to under-provisioning of server resources. There are several possible ways to deal with this limitation. The simplest solution is to provision additional server capacities (i.e., resource padding) to account for demand uncertainty. However, if the demand spike does not occur, these additional allocated capacity will be wasted. Another possibility is to increase the frequency at which the MPC algorithm runs. Doing so will reduce the duration in which servers are under-provisioned at the cost of high computational overhead. However, in our experiment we have found the computational overhead of the controller is almost negligible. Thus we believe it is reasonable to run the controller frequently, as long as reconfiguration cost is properly chosen in the computation.

As for the running time of Algorithm 1, computing the predicted values using ARIMA model takes $O(|V| \cdot |L| \cdot l)$ time, where l is the number of lags used in the ARIMA model. Solving DSPP can be done in polynomial time, although techniques such as interior point methods usually run reasonably fast. Section VII reports the running time of Algorithm 1.

VI. Competition among Multiple Providers

In this section, we extend our previous model and consider the case where multiple SPs share the cloud platform in terms of resources in data centers. The goal of each SP is to minimize its operational costs while respecting the SLA performance requirements and the data center capacity constraints. In our model, we assume that the placement configuration of each SP is kept private from other SPs. In this scenario, strategic interactions may arise as each SP makes decisions independently. Therefore, we can model the system as a multi-person non-cooperative game. Our objective is to analyze the equilibrium outcome, and design appropriate algorithms if the resulting competition leads to sub-optimal outcome in terms of social welfare. Generally speaking, the social welfare is defined as the sum of the service revenue minus the resource rental cost of all the SPs. In our case, the service revenue of each SP is fixed assuming that it can provision resources to satisfy its demand. As a result, the social welfare is maximized when the sum of the total resource rental cost of all SPs is minimized.

A. Problem Formulation

We formally define the resource competition game in this section. For each $v \in V$, define $\mathcal{N} = \{1, 2, ..., N\}$ as the set of SPs, and let $i \in \mathcal{N}$ represent the index of each SP. Let $\mathcal{K} = \{0, 1, 2, ...K\}$ denote the set of stages (i.e., time indices) of the game. At time k, $0 \le k \le K$, each SP i has a state $\mathbf{x}^{iv}_k = [x^{i1v}_k, ..., x^{iLv}_k]^\top \in \mathbb{R}^L_+$ that describes the number of servers allocated to demand from $v \in V$ in data center $l \in L$. Each SP i also makes a control decision $\mathbf{u}^{iv}_k = [u^{i1v}_k, ..., u^{iLv}_k]^\top \in \mathbb{R}^L$ at time $k \in \mathcal{K}$, where u^{ilv}_k denotes the change in the number of servers serving $v \in V$ at data center $l \in L$ at time k. Given an initial system state $\mathbf{x}^i_0$, the system dynamics are captured by the following state equation:

$$\mathbf{x}^{iv}_{k+1} = \mathbf{x}^{iv}_k + \mathbf{u}^{iv}_k \forall i \in \mathcal{N}, v \in V, 0 \le k \le K-1, \qquad (12)$$

At time $k \in \mathcal{K}$, we assume each SP i has a demand $\mathbf{D}^i_k = [D^{i1}_k, D^{i2}_k, ..., D^{iv}_k]^\top$, where D^{iv}_k represents the demand for SP i originated from $v \in V$ at time $k \in \mathcal{K}$. Similar to the single SP scenario, the total resource allocation for each SP should be sufficient to handle all demands without violating the SLA:

$$\mathbf{a}^{i\top}_k \mathbf{x}^{iv}_k \succeq D^{iv}_k, \quad \forall i \in \mathcal{N},\ v \in V,\ 0 \le k \le K, \qquad (13)$$

where $\mathbf{a}^i_k$ is defined as in Section IV-D for each SP i. We also require the total allocated resources to satisfy the data center capacity constraint. If there is insufficient capacity for hosting a server, the server request is rejected. In this case, the SP will try to find another data center to host the server. Given a set of resource types (e.g. CPU, memory and disk)

R, define $\mathbf{s}^i_r = s^i_r \cdot \mathbf{I}^{L\times L} \in \mathbb{R}^{L\times L}_+$ as a vector that represents the "size" of a server owned by SP i for resource type $r \in R$. Denote by $\mathbf{s}^i = [\mathbf{s}^i_1, ..., \mathbf{s}^i_R]^\top$, $\mathbf{C_r} = [C^1_r, C^2_r, ...C^L_r] \in \mathbb{R}^L_+$ and $\mathbf{C} = [\mathbf{C}_1, ..., \mathbf{C}_R]^\top$, The capacity constraint can be written as:

$$\sum_{i\in\mathcal{N}}\sum_{v\in V} \mathbf{s}^i \mathbf{x}^{iv}_k \preceq \mathbf{C}, \quad 0 \le k \le K. \tag{14}$$

Lastly, we define $\mathbf{u}^{iv} = \{\mathbf{u}^{iv}_0, \mathbf{u}^{iv}_1, ..., \mathbf{u}^{iv}_{K-1}\}$ $\mathbf{x}^{iv} = \{\mathbf{x}^{iv}_0, \mathbf{x}^{iv}_1, ..., \mathbf{x}^{iv}_{K-1}\}$, $\mathbf{u}^i = \{\mathbf{u}^{i1}, \mathbf{u}^{i2}, ..., \mathbf{u}^{iV}\}$. Furthermore, let $\mathbf{u}^{-i} = \{\mathbf{u}^1, ..., \mathbf{u}^{(i-1)}, \mathbf{u}^{(i+1)}, ..., \mathbf{u}^N\}$ represent the control decisions of the other SPs $N\backslash\{i\}$, the objective of SP i is to minimize its cost function:

$$J^i(\mathbf{u}^i, \mathbf{u}^{-i}) = \sum_{k=0}^{K}\sum_{v\in V} \mathbf{p}_k \mathbf{x}^{iv}_k + \mathbf{u}^{iv\top}_k \mathbf{R}^i \mathbf{u}^{iv}_k$$

subject to equation (12), (13) and (14). Here, $\mathbf{R}^i = [c^{i1}, c^{i2}, ..., c^{iL}] \in \mathbb{R}^L_+$ which captures the weight factor for reconfiguration cost for SP i in every data center $l \in L$.

We now characterize the Nash equilibrium (NE) of the resource competition game. The NE refers to the stable outcome of the competition, where no SP can improve its cost by unilaterally changing its server allocation over time. Formally, the resource competition game can be represented as a N-player dynamic non-cooperative game Ξ. Notice that as our controller relies on the MPC framework for dynamic resource allocation, we need to introduce a new version of NE for control strategies using the MPC framework. We first start with the following general definitions:

Definition 1 (η-Nash Equilibrium [11]). *Let $\mathcal{I}^i_k$ be the information set of a SP i at time k under a given information structure η^i, and Γ^i is the set of all admissible policies of SP i under η^i. The policy $\{\gamma^{i*}, i \in \mathcal{N}\}$ is an $\eta-$Nash equilibrium of the game Ξ, where $\mathbf{u}^i = \gamma^{i*}(\mathcal{I}^i_k)$ and $\eta = \{\eta^i, i \in \mathcal{N}\}$ if $J^i(\gamma^{i*}, \gamma^{-i*}) \le J^i(\gamma^i, \gamma^{-i*})$, for all admissible policies $\gamma^i \in \Gamma^i$ and for all $i \in \mathcal{N}$, where $\gamma^{-i*} = \{\gamma^j, j \ne i, j \in \mathcal{N}\}$.*

Definition 1 provides a general description of NE under a given information structure (IS) η^i. The dynamic game Ξ can admit different NEs under different information structures η. Typical information structures are, for example, open-loop IS, where the policy is only dependent on the initial conditions, and the perfect-state feedback IS, where the policy depends on the perfect measurement of the system state. The IS under MPC algorithms in Algorithm 1 can be deemed a special mixture between open-loop IS and feedback IS since at each stage each SP computes within a window in an open-loop manner but the initial condition of the computation is the current state known to SPs. With this special IS, we can define NE under MPC-type computations for our resource competition game.

Definition 2 ($\mathbf{W}$-MPC Nash Equilibrium). *Let W^i be the prediction window of SP i and every SP adopts MPC as outlined in Algorithm 1. The dynamic non-cooperative game Ξ admits $\mathbf{W}-$MPC Nash Equilibrium, $\mathbf{W} = \{W^i, i \in \mathcal{N}\}$, if the sequences $\mathbf{u}^{iv*} := \{\mathbf{u}^{iv*}_k, 0 \le k \le K\}$ obtained under MPC algorithms satisfy $J^i(\mathbf{u}^{i*}, \mathbf{u}^{-i*}) \le J^i(\mathbf{u}^i, \mathbf{u}^{-i*})$, for all admissible sequences $\mathbf{u}^i \in \mathcal{U}^i$ and for all $i \in \mathcal{N}$, where $\mathcal{U}^i$ is the set of admissible control sequences under MPC algorithms, and $\mathbf{u}^{-i*} = \{\mathbf{u}^j, j \ne i, j \in \mathcal{N}\}$.*

Note that NE solutions may not be unique, and hence we let $\mathcal{U}^*$ to denote the set of NE solutions $\mathbf{u}^* := \{\mathbf{u}^i, \mathbf{u}^{-i}\}$ that satisfy Definition 2. The $\mathbf{W}-$MPC Nash equilibrium $\{\mathbf{u}^{i*}, i \in \mathcal{N}\}$ can be used to compare with the optimal MPC solution $\{\mathbf{u}^{i\circ}, i \in \mathcal{N}\}$ to the following Social Welfare Problem (SWP):

$$\min_{\{\mathbf{u}^1,...,\mathbf{u}^N\}} \quad \sum_{i\in\mathcal{N}} J^i(\mathbf{u}^1, ..., \mathbf{u}^N)$$

subject to equation (12), (13) and (14). The NE is defined as:

$$J^i(\mathbf{u}^*) = \min_{\mathbf{u}^i\in\mathbb{R}^{LV}} J^i(\mathbf{u}^i, \mathbf{u}^{-i*}) \qquad \forall i \in \mathcal{N}$$

The price of anarchy (PoA) ρ_{MPC} and the price of stability (PoS) ξ_{MPC} of the dynamic non-cooperative game Ξ under centralized MPC Algorithm 1 are defined by

$$\begin{aligned}\rho_{\text{MPC}} &= \inf_{\mathbf{u}^*\in\mathcal{U}^*} \frac{\sum_{i\in\mathcal{N}}\sum_{v\in V} J^i_v(\mathbf{u}^{i\circ})}{\sum_{i\in\mathcal{N}}\sum_{v\in V} J^i_v(\mathbf{u}^{i*})} \\ \xi_{\text{MPC}} &= \sup_{\mathbf{u}^*\in\mathcal{U}^*} \frac{\sum_{i\in\mathcal{N}}\sum_{v\in V} J^i_v(\mathbf{u}^{i\circ})}{\sum_{i\in\mathcal{N}}\sum_{v\in V} J^i_v(\mathbf{u}^{i*})},\end{aligned}$$

where $\{\mathbf{u}^{i\circ}, i \in \mathcal{N}\}$ is the optimal solution to (SWP) obtained by MPC algorithm 1, and $\{\mathbf{u}^{i*}, i \in \mathcal{N}\}$ is the $\mathbf{W}-$MPC Nash equilibrium of the game Ξ. The metrics ξ_{MPC} and ρ_{MPC} are measures of the best-case and worse-case efficiency loss of the game, respectively. It is easy to observe that both ρ_{MPC} and ξ_{MPC} are always greater or equal to 1.

Theorem 1. *Assume that the prediction horizon of each SP $i, i \in \mathcal{N}$, is the same, i.e., $W^i = \bar{W}$ and $\bar{W}$ is also the prediction window used for (SWP). Then, the price of stability ξ_{MPC} of the game Ξ is always equal to 1, i.e., there exists a NE solution that yields no efficiency loss under the common knowledge of the capacity constraint.*

Theorem 2. *The price of anarchy ρ_{MPC} of the game Ξ is unbounded.*

The proofs of Theorem 1 and 2 are given in the Appendix.

B. Mechanism Design for a Single Infrastructure Provider

The result provided in the previous section is rather discouraging: If each SP behave selfishly in an uncoordinated manner, then the outcome can be severely unfair: certain SPs will experience much higher cost than others due to insufficient resource capacities in their preferred locations. This also hurts the efficiency of the NE. To address this issue, in this section we analyze the outcome with the participation of the InP.

Similar to existing work on Cloud resource pricing (e.g. [22]), we consider a market where there is a single InP who wishes to maximize the social welfare of all the participants, including both the SPs and the InP itself. For the InP, the utility is determined by (1) the revenue from selling resources and (2) service quality determined by number of rejected resource requests. In a highly dynamic scenario where it is not always possible to satisfy all resource demand at all times, rejecting a resource request may cause an under-provisioning of the SP's service infrastructure, resulting in a loss in SP's revenue. This, in turn, will result in poor robustness and hurt the customer satisfaction of the InP's service. Therefore, we assume there is a convex penalty function $\pi(\mathbf{s}^i\mathbf{x}_k^{iv}-\mathbf{C}) \in \mathbb{R}_+$ that captures the penalty of demand rejection. The utility of the InP becomes:

$$\begin{aligned} J^{InP}(\mathbf{p},\mathbf{u}^1,..\mathbf{u}^N) &= \sum_{k=0}^{K}\sum_{i\in\mathcal{N}}\sum_{v\in V}\mathbf{p}_k^\top\mathbf{x}_k^{iv}-\mathbf{e}_k^\top\mathbf{s}^i\mathbf{x}_k^{iv} \\ &\quad -\sum_{k=1}^{K}\pi(\sum_{i\in\mathcal{N}}\sum_{v\in V}\mathbf{s}^i\mathbf{x}_k^{iv}-\mathbf{C}) \end{aligned}$$

where $\mathbf{e}_k$ represents the production cost of resources. Typically, $\mathbf{e}_k$ includes the cost of electricity, server and land cost amortized over time. Notice that we assume the data center is energy proportional[2]. However, our model can be adapted to the cases where energy consumption is a convex function of resource utilization [15]. On the other hand, the social welfare of SP is determined by $J^i(\mathbf{p},\mathbf{u}^i,\mathbf{u}^{-i}) = -\sum_{k=0}^{K}\sum_{v\in V}\mathbf{p}_k^\top\mathbf{x}_k^{iv}+\mathbf{u}_k^{iv\top}\mathbf{R}^i\mathbf{u}_k^{iv}$, thus the social welfare can be computed as $J^{SW}(\mathbf{p},\mathbf{u}^1,..\mathbf{u}^N) = J^{InP}(\mathbf{p},\mathbf{u}^1,..\mathbf{u}^N)+\sum_{i\in\mathcal{N}}J^i(\mathbf{p},\mathbf{u}^i,\mathbf{u}^{-i})$, which becomes

$$\begin{aligned} \min \quad & \sum_{k=0}^{K}\sum_{i\in\mathcal{N}}\sum_{v\in V}\mathbf{e}_k^\top\mathbf{s}^i\mathbf{x}_k^{iv}+\mathbf{u}_k^{iv\top}\mathbf{R}^i\mathbf{u}_k^{iv} \\ & +\sum_{k=1}^{K}\pi(\sum_{i\in\mathcal{N}}\sum_{v\in V}\mathbf{s}^i\mathbf{x}_k^{iv}-\mathbf{C}) \end{aligned}$$

subject to constraints (12), (13). In this case, it is straightforward to design a mechanism for this problem using dual decomposition technique. To facilitate the decomposition, we can introduce an ancillary variable $\mathbf{v}_k = \sum_{i\in\mathcal{N}}\sum_{v\in V}\mathbf{s}^i\mathbf{x}_k^{iv}-\mathbf{C}$. The problem can be rewritten as

$$\min\sum_{k=0}^{K}\sum_{i\in\mathcal{N}}\sum_{v\in V}\mathbf{e}_k^\top\mathbf{s}^i\mathbf{x}_k^{iv}+\mathbf{u}_k^{iv\top}\mathbf{R}^i\mathbf{u}_k^{iv}+\sum_{k=1}^{K}\pi(\mathbf{v}_k)$$

subject to the constraint that $\mathbf{v}_k \succeq \sum_{i\in\mathcal{N}}\sum_{v\in V}\mathbf{s}^i\mathbf{x}_k^{iv}-\mathbf{C}$, $\mathbf{v}_k \succeq 0$ as well as constraints (12), (13). The lagrangian dual problem can be stated as:

$$\begin{aligned} &\max_{\lambda_k}(\inf_{\mathbf{x}_k^{iv},\mathbf{v}_k}\textstyle\sum_{k=0}^{K}\sum_{i\in\mathcal{N}}\sum_{v\in V}(\mathbf{e}_k^\top\mathbf{s}^i+\lambda_k\mathbf{s}^i)\mathbf{x}_k^{iv} \\ &\quad +\mathbf{u}_k^{iv\top}\mathbf{R}^i\mathbf{u}_k^{iv}+\textstyle\sum_{k=1}^{K}\pi(\mathbf{v}_k)-\lambda_k(\mathbf{C}+\mathbf{v}_k)) \end{aligned}$$

which is separable. Therefore, our dual decomposition mechanism is described by Algorithm 2. At time k, the InP announces the future resource prices for a window $1 \le t \le W$, and each provider $i \in \mathcal{N}$ submits the demand $\mathbf{x}_{k+t|k}^{iv}\forall v \in V$. At each step, the SP solves the problem

$$\min_{\mathbf{u}_{k+t|k}^{iv}}\sum_{t=0}^{W}\sum_{v\in V}(\mathbf{e}_{k+t|k}^\top\mathbf{s}^i+\lambda_{k+t|k}\mathbf{s}^i)\mathbf{x}_{k+t|k}^{iv}+\mathbf{u}_{k+t|k}^{iv\top}\mathbf{R}^i\mathbf{u}_{k+t|k}^{iv}$$

subjects to constraints (12), (13). The InP will solve the following problem:

$$\min_{\mathbf{v}_{k+t|k}\in\mathbb{R}^V}\quad\sum_{t=0}^{W}\pi(\mathbf{v}_{k+t|k})-\lambda_{k+t|k}(\mathbf{C}+\mathbf{v}_{k+t|k})$$

[2]Even though achieving perfect energy proportionality is difficult in practice, recent work has shown that the most energy efficient data centers today can achieve a power usage efficiency (PUE) less than 1.1 [23].

Algorithm 2 Iterative Algorithm for Achieving the best NE

```
1:  At beginning of time $k$, provide initial state $\mathbf{x}_0$, $k \leftarrow 0$,
    Initialize $\mathbf{C}^i \in \mathbb{R}_+^L$, $\bar{\mathbf{x}}_k^i \leftarrow 0$, $\forall k \in \mathcal{K}$ $\bar{J}(\mathbf{u}^1,...,\mathbf{u}^N) \leftarrow$
    $\infty$, $converged \leftarrow$ **false**
2:  $\mathbf{p}_{k+t|k} \leftarrow \mathbf{s}^i\mathbf{e}_k \forall 0 \le t \le W-1$
3:  **repeat**
4:      **for** $i = 1 \to N$ **do**
5:          $\mathbf{u}^i \leftarrow$ solution of DSPP$^i$ with price $\mathbf{p}_{k+t|k}, \forall 0 \le t \le$
            $W-1$
6:      **end for**
7:      $J(\mathbf{u}^1,...,\mathbf{u}^N) = \sum_{i\in\mathcal{N}} J^i(\mathbf{u}^1,...,\mathbf{u}^N)$
8:      **if** $|J(\mathbf{u}^1,...,\mathbf{u}^N) - \bar{J}(\mathbf{u}^1,...,\mathbf{u}^N)| \le 0.01 \times$
        $\bar{J}(\mathbf{u}^1,...,\mathbf{u}^N)$ **then**
9:          $converged \leftarrow$ **true**
10:     **end if**
11:     **if** $converged \ne$ **true then**
12:         $\lambda_{k+t|k} := (\lambda_{k+t|k} + \alpha(\sum_{i\in\mathcal{N}}\sum_{v\in V}\mathbf{s}^i\mathbf{x}_{k+t|k}^{iv} - \mathbf{C} -$
            $\mathbf{v}_{k+t|k}))_+$
13:         $\mathbf{p}_{k+t|k} = \mathbf{e}_k^\top\mathbf{s}^i + \lambda_k\mathbf{s}^i$
14:     **end if**
15:     $\bar{J}(\mathbf{u}^1,...,\mathbf{u}^N) \leftarrow J(\mathbf{u}^1,...,\mathbf{u}^N)$
16: **until** $converged =$ **true**
```

and update the $\lambda_{k+t|k}$ according to the following equation, where $\alpha \in \mathbb{R}_+$ is the step size:

$$\lambda_{k+t|k} := (\lambda_{k+t|k}+\alpha(\sum_{i\in\mathcal{N}}\sum_{v\in V}\mathbf{s}^i\mathbf{x}_{k+t|k}^{iv}-\mathbf{C}-\mathbf{v}_{k+t|k}))_+$$

and announce a new price $\mathbf{p}_{k+t|k} = \mathbf{e}_k^\top\mathbf{s}^i+\lambda_k\mathbf{s}^i$. This process repeats until the solution converges to a local optimal solution. Finally, even though convergence rate can be a practical concern for gradient-based algorithms. However, our mechanism can simultaneously adjust prices many steps into the future, which gives more time for prices to converge. The convergence rate of Algorithm 2 is analyzed in Section VII.

Remark 1. *In this work we assume a monopolized market where the goal of the InP is to maximize social welfare rather than purely optimizing revenue. This is a reasonable model as the social welfare, in some sense, is a measure of service quality. An InP that maximizes social welfare is likely to attract more businesses in the future. Secondly, to prevent the InP from setting unfair prices, governments are likely to impose a fair return price [24]. One limitation of setting fair return price is the possibility of low income of the InP. However, it is straightforward to consider the case where the fair return price considers the revenue gain of the InP.*

In summary, the lesson we learned in the theoretical analysis is that simply rejecting requests when capacity is reached can lead to inefficient outcomes where certain SPs may be treated unfairly. A dynamic congestion-pricing mechanism can be helpful for mitigating this problem.

VII. Simulations

We have implemented our solutions and conducted several simulation studies. In our simulations, we have used a real Internet topology graph from the *Rocketfuel project* [25],

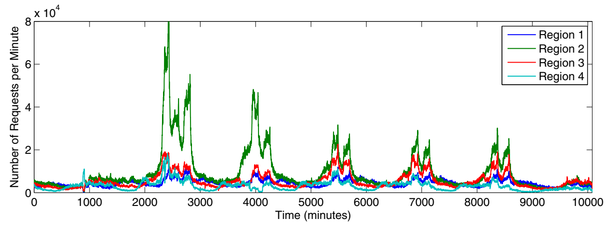

Fig. 3. HTTP requests in the Worldcup 98 dataset.

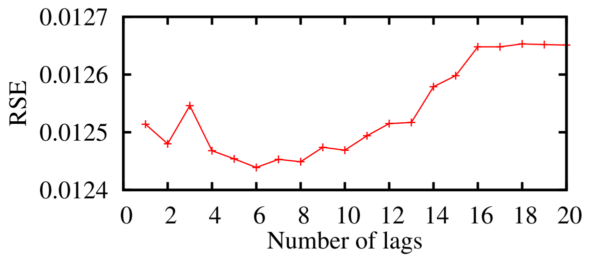

Fig. 4. Prediction accuracy vs. number of lags used.

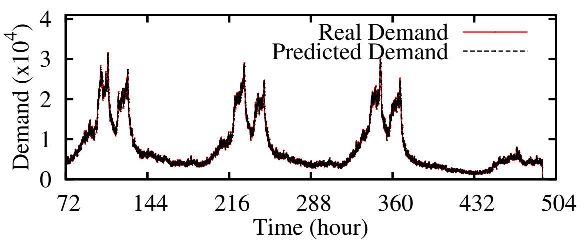

Fig. 5. Actual demand vs. Predicted demand for region 2.

which contains link latency information. However, as the data set only contains topologies for several tier-1 Internet Service Providers (ISPs), we have augmented the topology graph by introducing regional and local ISPs, similar to the procedure for generating transit-stub networks in the GT-ITM network topology generator [26]. We specify the communication latency at intra-transit, stub-transit and intra-stub domain links to be $20ms$, $5ms$ and $2ms$, respectively [27]. Based on our experience with Google's data centers, in our experiment, we have created 3 large data centers located in Mountain View, CA, Houston, and TX, Atlanta, GA. To generate realistic service requests, we have decided to use the Worldcup 98 dataset [28] which contains HTTP requests for a total duration of 92 days. This dataset contains several occurrences of demand spikes, which is useful for demonstrating the performance of our algorithm. However, as the dataset does not contain location information of access networks, we use the request *regions* provided in the dataset to approximate the source of requests. In our simulation, we assume there is one access network responsible for generating requests from a single region. Fig. 3 illustrates the service demand for week between June 13 - June 20 for 4 different regions. In our experiment, the price of resources in each data center is set to the electricity price per VM according to the VM size. Fig. 9 shows the electricity price during different times of the day. For comparison purpose, we assume the price in Atlanta is constant (i.e., not market-driven).

A. The Case for a Single Service Provider

In this subsection, we report our experiment results for the single service provider case. We first evaluate the quality of demand prediction model (ARIMA in our case), which plays an important role in determining the quality of the solution. In our evaluation, we divided the workload into two data sets. The first set is used to estimate the parameters of the ARIMA model. The second set is used to assess the accuracy of the prediction. The prediction error is measured by the Relative Squared Error (RSE). Typically, the smaller is the RSE, the better is the prediction. Fig. 4 shows the prediction error as a function of the number of lags (previous samples) used as inputs for the ARIMA model. It can be seen that the prediction using 6 lags achieves the lowest RSE. Fig. 5 shows the predicted demand compared to the actual demand for region 2, which has the highest demand fluctuation among all the regions. It is clear that ARIMA model can achieve an accurate prediction of service demand (RSE≈ 0.01245).

To demonstrate how our controller adjusts resource allocation to handle demand fluctuation, we consider the case where there is a single data center responsible for requests from two regions. Fig. 6 shows that the controller always tries to adjust the resource allocation to match the demand. We also analyzed the effect of the prediction horizon W on the outcome of dynamic resource allocation. Fig. 7 shows that the change in the number of servers decreases as W increases. The controller with a long window size is less aggressive than a controller that only looks few steps into the future. But at the same time, it can cause higher cost due to poor prediction of future demand (Fig. 8). To demonstrate controller's reaction to price change, we have simulated a scenario where 2 data centers (Mountain View and Atlanta) are used to serve demand from region 2. Our experiment result is shown in Fig. 10. Accordingly, Fig. 10 shows that our controller allocates fewer servers in the Mountain View DC in the afternoon, since price in Atlanta is cheaper. This confirms our algorithm can balance load according to price change. Finally, we demonstrate the importance of reconfiguration cost. We have implemented a greedy algorithm that ignores reconfiguration cost. The output of the algorithm is shown in Fig. 11. It can be seen that, since the price in Atlanta is cheaper in the afternoon the greedy algorithm performs a massive migration of servers in the afternoon. Similarly, once the price in Mountain View becomes cheaper, the greedy algorithm performs another massive migration to send them back to Mountain View. It is evident that reconfiguration cost plays a crucial role in avoiding massive migrations in this scenario. We also found the prediction window produces a similar effect as reconfiguration cost (Fig. 7 and Fig. 12). However, since the prediction window is used to capture trend in system inputs, and its effect is highly dependent on the prediction accuracy, it should not be used to control the aggressiveness of dynamic adaptation. Finally, we have also evaluated the running time using all 4 DC locations and 24 access networks with synthetic workloads, and found Algorithm 1 usually runs in less than 1 second.

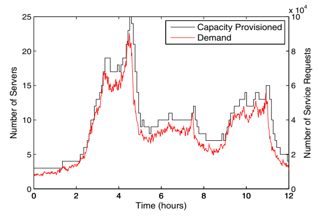
Fig. 6. Response to demand fluctuation.

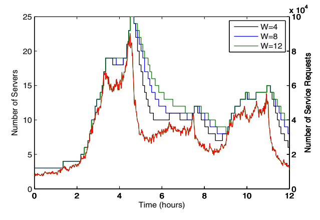
Fig. 7. Effect of prediction window size on the number of servers.

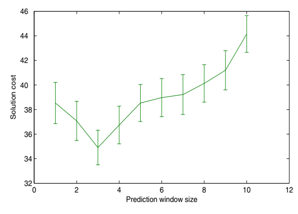
Fig. 8. Effect of prediction horizon on the solution cost.

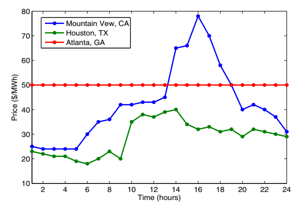
Fig. 9. Prices of electricity used in experiments.

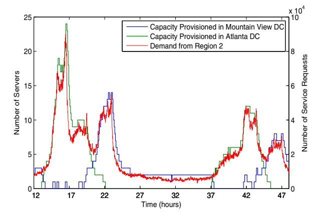
Fig. 10. Impact of price on solution quality.

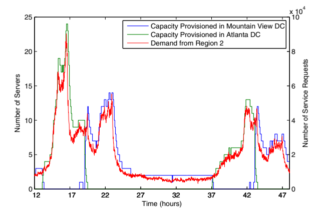
Fig. 11. Output of the greedy algorithm.

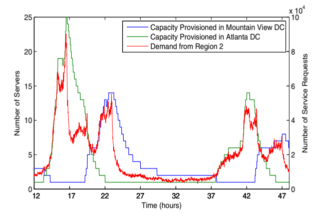
Fig. 12. Output with no reconfiguration cost and long window size.

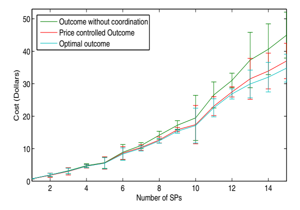
Fig. 13. Comparing the cost of NEs versus the number of players.

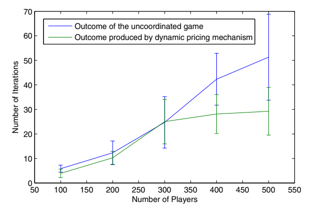
Fig. 14. Number of players vs. convergence rate.

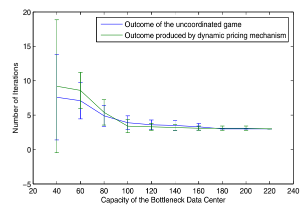
Fig. 15. Capacity of the Bottleneck DC vs. convergence rate.

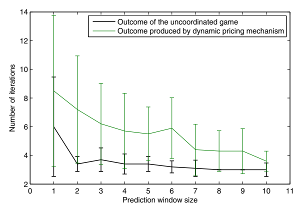
Fig. 16. Prediction horizon length vs. convergence rate.

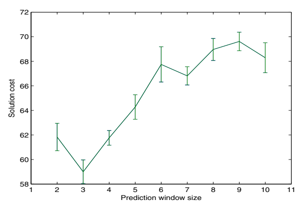
Fig. 17. Impact of prediction horizon length on the cost.

B. The Case for Multiple Service Providers

To analyze the outcome of the resource competition game, we generate the input parameters (μ^i, $\mathbf{D}^i_k$, s^i, c^{il},$\bar{d}^i$) for each SP $i \in \mathcal{N}$ randomly. We first simulated the standard game by allowing every SP to move in a sequential order, until no SP can further reduce its cost without violating the capacity constraint. We also implemented our coordination algorithm described by Algorithm 2. We set the step size $\alpha = 0.5$, and define $\pi(\mathbf{v}) = \mathbf{v}^\top \mathbf{I}\mathbf{v} \cdot P$, where $P = 10$. We plotted the average cost of uncoordinated NEs and the cost produced by Algorithm 2 for the duration of 24 hours in Fig. 13. For comparison purpose, we also plotted the optimal offline solution in Fig. 13. Clearly, Algorithm 2 improves the social welfare by $10-20\%$ compared to uncoordinated NEs.

We then analyze the scalability of the algorithm in terms of convergence rate. To produce a competition scenario, we set the number of servers in the data center with the cheapest cost (i.e., Data center in Houston, TX) to 500 respectively, and record the number of iterations required to produce an approximately stable outcome. In our experiment, we call an outcome approximately stable if $|\bar{J}(\mathbf{u}^1,...,\mathbf{u}^N) - J(\mathbf{u}^1,...,\mathbf{u}^N)| \le 0.01 \cdot \bar{J}(\mathbf{u}^1,...,\mathbf{u}^N))$, where $\bar{J}(\mathbf{u}^1,...,\mathbf{u}^N)$ is the cost of the solution in the previous iteration. Fig. 14 shows the number of iterations to obtain a stable outcome grows with the number of players. However, as mentioned before, the convergence process can start $W-1$ steps ahead, thus the convergence rate is still acceptable. Finally, even though dynamic conditions such as machine failures, SPs joining and leaving the system can hurt the convergence rate, by correctly modeling the penalty caused by dynamic conditions (e.g. using penalty function $\pi(\cdot)$) and using short time intervals, it is possible to bring the impact of dynamic conditions to a minimum. Finally, we also conducted experiments to examine the impact of prediction horizon W on the solution optimality and convergence rate. Fig. 16 suggests that longer prediction horizon can improve convergence rate. However, the selecting the right window size should also consider the solution quality. Similar to Fig. 7, we found setting window size to $W = 3$ archives the best outcome. Finally, Fig. 18 shows the number of rejected request in a scenario with 2 DCs and 20 SPs having identical demand (same as Region 2 in Fig. 5). The number of requests rejected is proportional to the capacity of the bottleneck DC, who has lower cost. We found the percentage of rejected request is small even when demand sometimes can be $20\times$ more than the capacity of the data center.

VIII. CONCLUSION

In this paper, we presented a framework for the dynamic service placement problem based on control- and game-theoretic models. In particular, we provided a solution that optimizes the hosting cost dynamically over time according to both demand and resource price fluctuations. We also considered the case where multiple SPs compete for revenue dynamically. Our analysis showed that in an uncoordinated scenario where

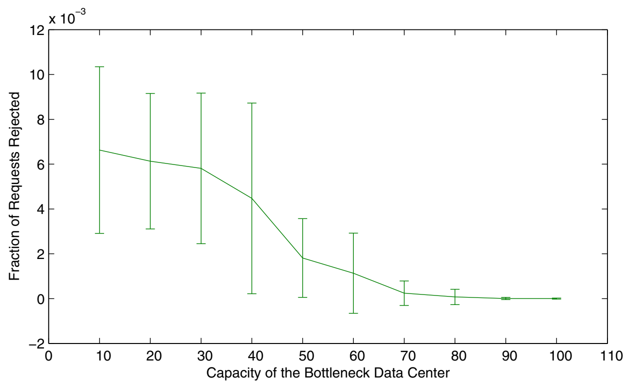

Fig. 18. Req. rejection rate vs. the capacity of the bottleneck DC.

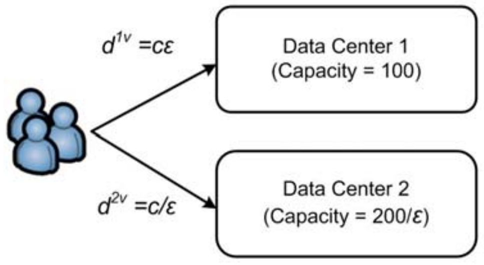
$d^{1v} = c\varepsilon$
Data Center 1 (Capacity = 100)
$d^{2v} = c/\varepsilon$
Data Center 2 (Capacity = 200/ε)

Fig. 19. Example illustrating the PoA of the game.

service providers behave in a selfish manner, the resulting NE can be significantly worse than the optimal NE in terms of social welfare. Based on this observation, we proposed a mechanism that can be adopted by the InP to maximize the social welfare of the system. Our simulations not only confirm the theoretical findings, but also demonstrate the benefits of the proposed approach. Finally, we believe there are many interesting directions for future exploration, such as analyzing the competition among multiple InPs, and studying the impact of pricing models on the outcome of the competition.

Acknowledgment

This work was supported in part by the Natural Science and Engineering Council of Canada (NSERC) under the Smart Applications on Virtual Infrastructure (SAVI) Research Network, in part by a Google Faculty Research Award, and in part by the World Class University (WCU) Program under the Korea Science and Engineering Foundation funded by the Ministry of Education, Science and Technology (Project No. R31-2008-000-10100-0).

Appendix

Proof of Theorem 1: Define $\mathbf{u}_k^{iv} = [u_k^{i1v}, u_k^{i2v}, ...u_k^{iLv}]^\top$, $\mathbf{z}_{v,k}^i = [(\mathbf{u}_1^{iv})^\top, ..., (\mathbf{u}_k^{iv})^\top]^\top$, $\mathbf{R}^i = [c^{i1}, c^{i2}, ..., c^{iL}]$ and $\mathbf{E}_{v,k}^i = \text{diag}\{\text{diag}^{-1}\{\mathbf{R}^i\}, ..., \text{diag}^{-1}\{\mathbf{R}^i\}\} \in \mathbb{R}_+^{Lk\times Lk}$, $\mathbf{F}_{v,k}^i = [k\mathbf{p}_1^\top, (k-1)\mathbf{p}_2^\top, ..., \mathbf{p}_k^\top]^\top \in \mathbb{R}^{Lk}$, $\mathbf{w} = \mathbf{I}^{L\times L}$, $\mathbf{M}_{k,1}^{iv} = \begin{bmatrix} \mathbf{a}_1^{iv\top} & \dots & 0 \\ \vdots & \ddots & \vdots \\ \mathbf{a}_k^{iv\top} & \dots & \mathbf{a}_k^{iv\top} \end{bmatrix} \in \mathbb{R}^{k\times Lk}$, $\mathbf{M}_{k,3}^{iv} = \begin{bmatrix} \mathbf{w^i} & \dots & 0 \\ \vdots & \ddots & \vdots \\ \mathbf{w^i} & \dots & \mathbf{w^i} \end{bmatrix} \in \mathbb{R}^{Lk\times Lk}$, $\mathbf{M}_{v,k}^i = [\mathbf{M}_{k1}^{iv\top}, \mathbf{M}_{k2}^{iv\top}, \mathbf{M}_{k3}^{iv\top}]^\top$, $\gamma_{k,1}^{iv} = [D_1^{iv} - \mathbf{a}_1^{iv}\mathbf{x}_0^{iv}, D_2^{iv} - \mathbf{a}_2^{iv}\mathbf{x}_0^{iv}, ..., D_k^{iv} - \mathbf{a}_K^{iv}\mathbf{x}_0^{iv}]^\top$, $\gamma_{v,k}^i = [\gamma_{k,1}^{iv\top}, \gamma_{k,1}^{iv\top}, \gamma_{k,1}^{iv\top}]^\top$ where $\gamma_{k,2}^{iv}[\mathbf{C_1}^\top, ..., \mathbf{C}_L^\top]^\top \in \mathbb{R}_+^{Lk}$, $\gamma_{k,3}^{iv} = \left[-\mathbf{x}_0^{iv\top}, -\mathbf{x}_0^{iv\top}, ..., -\mathbf{x}_0^{iv\top}\right]^\top \in \mathbb{R}_+^{Lk}$, $\mathbf{S}_{v,k}^i = [s^i, s^i, ..., s^i]^\top \in \mathbb{R}_+^{Lk}$. At time k each SP i solves the following problem:

$$\min_{\mathbf{z}_{W,k}^i} \quad \bar{J}^i := \textstyle\sum_{v\in V}(\mathbf{z}_{v,k}^i)^T\mathbf{F}_{v,k}^i + (\mathbf{z}_{v,k}^i)^T\mathbf{E}_{v,k}^i(\mathbf{z}_{v,k}^i) \quad (15)$$

$$\mathbf{M}_{v,k}^i\mathbf{z}_{v,k}^i \geq \gamma_{v,k}^i, \forall v \in V \quad (16)$$

$$\textstyle\sum_{i\in\mathcal{N}}(\mathbf{S}_{v,k}^i)^T\mathbf{z}_{v,k}^i \preceq \mathbf{C}. \quad (17)$$

Each SP faces two constraints in the above problem. One is an internal constraint given by (16) and the other is the coupled constraint on all the players, which is given by (17). We can associate each internal constraint (16) with Lagrange multipliers $\mu_v^i, i \in \mathcal{N}, v \in V$ and the coupled constraint with ν. The Lagrangian of SP i is given by

$$\mathcal{L}^i = \bar{J}^i + \sum_{v\in V}\left(\mu_v^i(\mathbf{M}_{v,k}^i - \gamma^i)\right) + \nu\left(\sum_{i\in\mathcal{N}}(\mathbf{S}_{v,k}^i)^T\mathbf{z}_{v,k}^i - \mathbf{C}\right). \quad (18)$$

On the other hand, for (SWP), we face the following problem at every time k:

$$\min_{\{\mathbf{z}_{K,k}^i, i\in\mathcal{N}\}} \quad \textstyle\sum_{i\in\mathcal{N}}\bar{J}^i \quad (19)$$

$$\mathbf{M}_{v,k}^i\mathbf{z}_{v,k}^i \geq \gamma_{v,k}^i, \forall v \in V, \forall i \in \mathcal{N}. \quad (20)$$

$$\textstyle\sum_{i\in\mathcal{N}}(\mathbf{S}_{v,k}^i)^T\mathbf{z}_{v,k}^i \preceq \mathbf{C}. \quad (21)$$

By associating Lagrangian multipliers $\tilde{\mu}_v^i, i \in \mathcal{N}, v \in V$ with constraints (20) and ν with (21), we have the Lagrange of the social welfare problem

$$\mathcal{L} = \sum_{i\in\mathcal{N}}\bar{J}^i + \sum_{i\in\mathcal{N}, v\in V}\tilde{\mu}_v^i(\mathbf{M}_{v,k}^i - \gamma^i) + \tilde{\nu}\left(\sum_{i\in\mathcal{N}}(\mathbf{S}_{v,k}^i)^T\mathbf{z}_{v,k}^i - \mathbf{C}\right). \quad (22)$$

By letting $\mu_v^i = \tilde{\mu}_v^i$, and $\nu = \tilde{\nu}/N$, we can further decompose $\mathcal{L}$ into $\mathcal{L} = \sum_{i\in\mathcal{N}}\mathcal{L}_i$. Since it is strictly convex and separable in i, the $K-$horizon social welfare problem admits a unique solution, which also corresponds to the solution of each convex subproblem associated with $\mathcal{L}_i$. Hence, the social optimal solution is a NE at every k and the result follows. ■

Proof of Theorem 2: We provide an example to illustrate that the price of anarchy is unbounded even when demands of SPs are static. Consider the scenario illustrated by Fig. 19: there are two data centers serving demand from a single location v. The data centers have capacities $C^1 = 100$ and $C^2 = \frac{200}{\epsilon}$ respectively. The distance to each data center are $d^{1v} = \epsilon c$, $d^{2v} = \frac{c}{\epsilon}$ respectively, where c and ϵ are constants. Both data centers lease resources at the same unit price p, i.e., $p_k^1 = p_k^2 = p \forall k \geq 0$. Furthermore, There are two SPs in the game. Their SLAs are $\bar{d}^1 = (1 + \epsilon + \frac{1}{\epsilon})c$ and $\bar{d}^2 = (K + 1 + \epsilon + \frac{1}{\epsilon})c$, respectively, where $K \geq 1$ is a constant. For both SPs, a single server can process requests at rate $\mu = \frac{1}{c}$. The demand from location v for both SPs are $D_1 = \frac{100}{c(1+\epsilon)}$, $D_2 = \frac{100}{c(1+\frac{1}{K\epsilon+1})}$. Now, consider the following allocation for both SPs: SP 2 serves all its demands using capacities in DC 1, and SP 1 serve all its demands from DC2. It is easy to see this is a NE, as there is no free capacity in DC 1 for SP 1. The total cost of this NE is $J_{NE1} = \sum_{v \in V} \sum_{i \in \mathcal{N}} J_v^i = (1 + \frac{1}{\epsilon})100p$. Now consider another NE, where SP 1 uses all the capacities in DC 1, and SP 2 serve all its demands from DC 2. The total cost of this NE is $J_{NE2} = 100(1 + \frac{1+\frac{1}{K+\epsilon}}{1+\frac{1}{K\epsilon+1}})p$. As $\epsilon \to 0$, we have $\rho_{\text{MPC}} \geq \lim_{\epsilon \to 0} \frac{J_{NE1}}{J_{NE2}} = \lim_{\epsilon \to 0} \frac{1+\frac{1}{\epsilon}}{1+\frac{1+\frac{1}{K}}{2}} = \infty$. ■

References

[1] Q. Zhang, Q. Zhu, M. F. Zhani, and R. Boutaba, "Dynamic service placement in geographically distributed clouds," in *Proc. Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2012.

[2] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: State-of-the-art and research challenges," *J. Internet Services Applications (JISA)*, 2010.

[3] Q. Zhang, M. F. Zhani, Q. Zhu, S. Zhang, R. Boutaba, and J. Hellerstein, "Dynamic energy-aware capacity provisioning for cloud computing environments," in *Proc. IEEE/ACM Int. Conf. Autonomic Comput. (ICAC)*, 2012.

[4] L. Rao, X. Liu, and W. Liu, "Minimizing electricity cost: Optimization of distributed Internet data centers in a multi-electricity-market environment," in *Proc. IEEE INFOCOM*, 2010.

[5] Z. Liu, M. Lin, A. Wierman, S. Low, and L. Andrew, "Greening geographical load balancing," in *Proc. SIGMETRICS*, 2011.

[6] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, "Cutting the electric bill for Internet-scale systems," in *Proc. ACM SIGCOMM*, 2009.

[7] Q. Zhang, Q. Zhu, and R. Boutaba, "Dynamic resource allocation for spot markets in cloud computing environments," in *Proc. IEEE Int. Conf. Utility Cloud Comput. (UCC)*, 2011.

[8] "Amazon ec2 spot instances," [Online]. Available: http://aws.amazon.com/ec2/spot-instances/

[9] L. Qiu, V. Padmandabhan, and V. Geoffrey, "On the placement of web server replicas," in *Proc. IEEE INFOCOM*, 2001.

[10] S. Islam and J. Grégoire, "Network edge intelligence for the emerging next-generation Internet," *Future Internet*, 2010.

[11] T. Başar and G. Olsder, *Dynamic Noncooperative Game Theory*. Society for Industrial Mathematics, 1999.

[12] Y. Chen, R. Katz, and J. Kubiatowicz, "Dynamic replica placement for scalable content delivery," in *Proc. Int. Workshop Peer-To-Peer Syst. (IPTPS)*, 2002.

[13] C. Vicari, C. Petrioli, and F. Presti, "Dynamic replica placement and traffic redirection in content delivery networks," in *Proc. IEEE MASCOTS*, 2007.

[14] N. Laoutaris, G. Smaragdakis, K. Oikonomou, I. Stavrakakis, and A. Bestavros, "Distributed placement of service facilities in large-scale networks," in *Proc. IEEE INFOCOM*, 2007.

[15] D. Kusic, J. Kephart, J. Hanson, N. Kandasamy, and G. Jiang, "Power and performance management of virtualized computing environments via lookahead control," in *IEEE Int. Conf. Autonomic Comput.*, 2009.

[16] P. Wendell, J. Jiang, M. Freedman, and J. Rexford, "Donar: Decentralized server selection for cloud services," in *Proc. ACM SIGCOMM*, 2010.

[17] "Amazon elastic computing cloud (amazon ec2)," [Online]. Available: http://aws.amazon.com/ec2/

[18] J. Hamilton, *Time Series Analysis*, vol. 2. Cambridge Univ Press, 1994.

[19] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *IEEE Int. Conf. Cloud Comput. (CLOUD)*, 2011.

[20] E. Camacho and C. Bordons, *Model Predictive Control*. Springer, 2004.

[21] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[22] D. Niu, C. Feng, and B. Li, "Pricing cloud bandwidth reservations under demand uncertainty," in *Proc. SIGMETRICS*, 2012.

[23] "Designing a very efficient data center," [Online]. Available: https://www.facebook.com/note.php?note_id=10150148003778920

[24] C. McDonnell, *Economics-Princples, Problems, and Policies*, 1984.

[25] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring ISP topologies with rocketfuel," *IEEE/ACM Trans. Netw.*, 2009.

[26] "GTITM homepage," [Online]. Available: www.cc.gatech.edu/projects/gtitm/

[27] S. Ratnasamy, M. Handley, R. Karp, and S. Scott, "Topologically-aware overlay construction and server selection," in *Proc. IEEE INFOCOM*, 2002.

[28] "World cup 1998 web site requests," [Online]. Available: http://ita.ee.lbl.gov/html/contrib/WorldCup.html

Qi Zhang received his B.A.Sc. and M.Sc. from the University of Ottawa (Canada) and Queen's University (Canada), respectively. He is currently pursuing a Ph.D. degree in computer science from the University of Waterloo. His current research focuses on resource management for cloud computing systems. He is also interested in related areas including network virtualization and management.

Quanyan Zhu (S'04-M'13) is a Ph.D. candidate at the Department of Electrical and Computer Engineering and the Coordinated Science Laboratory (CSL) at the University of Illinois at Urbana-Champaign (UIUC). He has received his master of applied science in electrical engineering from the University of Toronto, and bachelor of engineering in honors electrical engineering from McGill University. He has been a visiting researcher at the University of Waterloo, University of Avignon, University of Houston, INRIA-Sophia Antipolis, Idaho National Laboratory, SUPELEC, and the University of Washington and Chinese Academy of Mathematics and System Science. He is a recipient of the NSERC Canada Graduate Scholarship, University of Toronto Fellowship, Ernest A. Reid Fellowship, and Mavis Future Faculty Fellowships. He is a recipient of the best track paper award at the 4th International Symposium on Resilient Control Systems (ISRCS).

Mohamed Faten Zhani received engineering and M.S. degrees from the National School of Computer Science, Tunisia, in 2003 and 2005, respectively. He received his Ph.D. in computer science from the University of Quebec in Montreal, Canada, in 2011. Since then, he has been a postdoctoral research fellow at the University of Waterloo. His research interests include virtualization, resource management in the cloud computing environment, and network performance evaluation.

Raouf Boutaba received the M.Sc. and Ph.D. degrees in computer science from the University Pierre & Marie Curie, Paris, in 1990 and 1994, respectively. He is currently a professor of computer science at the University of Waterloo and a distinguished visiting professor at the division of IT convergence engineering at POSTECH. His research interests include network, resource, and service management in wired and wireless networks. He is the founding editor in chief of the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT (2007–2010) and on the editorial boards of other journals. He has received several best paper awards and other recognition such as the Premiers Research Excellence Award, the IEEE Hal Sobol Award in 2007, the Fred W. Ellersick Prize in 2008, and the Joe LociCero and Dan Stokesbury awards in 2009. He is a fellow of the IEEE.

Joseph L. Hellerstein manages the Computational Discovery Department at Google Inc. in Seattle, WA. Dr. Hellerstein received the Ph.D. in computer science from the University of California at Los Angeles. He was a Principal Architect at Microsoft Corp. in Redmond, WA (USA) from 2006 to 2008, and a researcher and senior manager at the IBM Thomas J. Watson Research Center in Hawthorne, NY, from 1984 to 2006. He has published over 100 peer-reviewed papers and two books, and has taught at Columbia University and the University of Washington. Dr. Hellerstein is a Fellow of the IEEE.