On QoE-oriented Cloud Service Orchestration for Application Providers

Dmitrii Chemodanov*, Prasad Calyam, Samaikya Valluripally, Huy Trinh, Jon Patman, and Kannappan Palaniappan
University of Missouri-Columbia, USA
*Primary and corresponding author, email: dycbt4@mail.missouri.edu

***Abstract*—New virtualization technologies allow Infrastructure Providers (InPs) to lease their resources to Application Service Providers (ASPs) for highly scalable delivery of cloud services to end-users. However, existing literature lacks knowledge on Quality of Experience (QoE)-oriented cloud service orchestration algorithms that can guide ASPs on how to plan their budget to enhance satisfactory QoE delivery to end-users. In contrast to the InP's cloud service orchestration, the ASP's orchestration should not rely on expensive infrastructure control mechanisms such as Software-Defined Networking (SDN), or require *apriori* knowledge on the number of services to be instantiated and their anticipated placement location within InP's infrastructure. In this paper, we address this issue of delivering satisfactory user QoE by synergistically optimizing both ASP's *management* and *data planes*. The optimization within the ASP *management plane* first maximizes Service Level Objective (SLO) coverage of users when application services are being deployed, and are not yet operational. The optimization of the ASP *data plane* then enhances satisfactory user QoE delivery when applications services are operational with real user access. Our evaluation of QoE-oriented algorithms using realistic numerical simulations, real-world cloud testbed experiments with actual users and ASP case studies show notably improved performance over existing cloud service orchestration solutions.**

***Index Terms*—QoE-oriented cloud service orchestration, multi-constrained path-aware possibilistic C-mean clustering, 3Q interplay model, least cost-disruptive decision tree.**

I. Introduction

The advent of network virtualization has enabled new business models that allow Application Service Providers (ASPs) to run their users' multimedia-rich data-intensive application services upon leased resources from infrastructure providers (InPs) [1], [2]. ASPs have to balance between delivering satisfactory user QoE and renting excessive resources from InPs for running many service instances, which can be very expensive. This in turn can lead to high service costs for users, which may end up impacting customer retention efforts for ASPs. In addition, new business models require ASPs to also address limitations of the best-effort nature of the Internet. Specifically, they need to support users when they access remote services over Internet paths that could degrade the user QoE delivered due to congestion or resource misconfiguration issues [3]. Generally, QoE reflects the user opinion score ranging from bad-1 to excellent-5 quality.

Although many cloud service orchestration approaches for InPs have been proposed, including application-aware networking, cloud service placement, and virtual network embedding, there is still a lack of 'QoE-oriented cloud service orchestration' algorithms for ASPs that can build upon recent advances of network virtualization and related business models. In contrast to InP cloud service orchestration schemes, the ASP service orchestration should not require expensive infrastructure control mechanisms such as SDN-based application aware-networking [4]. Moreover, such orchestration also should not require *a priori* knowledge on the number of service instances that need to be instantiated, and their anticipated placement location within InPs' infrastructure to deliver satisfactory user QoE such as cloud service placement [5] or virtual network embedding [1]. Thus, QoE-oriented cloud service orchestration algorithms are particularly crucial for ASPs to solve their general optimization problem — maximization of the satisfactory QoE service delivery to users that is limited by the ASP's cost budget.

Consequently, ASPs have to plan their budget by making control decisions in their *management plane* such as how many running service instances are needed (i.e., the amount of InPs' resources), where to run them within an infrastructure (preferably close to user locations) and how to assign (or cluster) users to the instantiated services. Doing so, they can overcome inefficient network design (e.g., high latency) that prevents a service delivery with satisfied Service Level Objectives (SLOs) to the end users. Further, ASPs need to continually monitor that their services are delivered to the end users with satisfactory QoE levels within the *data plane* by pertinent tuning of rented resources.

A concrete example scenario where ASPs need to overcome inefficient infrastructure utilization within their management plane and ensure that their user traffic traverses network paths with required SLOs can be seen from an exemplar use case of the "Skytap" ASP [6]. Education organization customers of Skytap request virtual environments for geo-spatially dispersed users at large-scale to deliver fast, easy, and secure training sessions. Hence, the user QoE of such a service delivery is highly sensitive to bandwidth, delay, and losses of the users' network path from their thin-clients to the cloud-hosted Skytap services. Another example scenario where ASPs need to overcome inefficient infrastructure utilization within the data plane by tuning (or adapting) allocated resources can be seen in the LOFT (Likelihood of Features Tracking) application [7] for public safety organization customers. In order to deliver real-time visual situational awareness such as tracking objects of interest from civilian smartphones or surveillance cameras using the LOFT, ASPs should adapt to the changes arising from customer needs involving trade-offs between cost, quality, and real-time context of the service. This tradeoff situation in balancing cost and quality can cause disruptive effects on user QoE delivery due to frequent resource adaptations by the ASPs.

Our Approach. In this paper, we address the lack of 'QoE-oriented cloud service orchestration' algorithms tailored to the ASP business model to maximize satisfactory user QoE delivery within the ASP's budget constraints. *Our approach* is to synergistically optimize the ASP's management and data planes as follow: (i) first maximize the number of users with fully satisfied SLOs within the ASP management plane (upon renting resources leased from InPs); (ii) then maximize the number of users with satisfied QoE within the ASP data plane (during its service delivery to end-users). By solving (i), ASPs first determine the amount of rented resources from InPs, their placement and the user clustering between them which is subject to ASPs' budget, then by solving (ii), ASPs synergistically adapt their rented resources to enhance their service delivery to end-users and reconsider their budget for solving (i) as needed.

Contributions. Our contributions are as follows:

− To solve (i), we first propose the optimal integer linear programming (ILP) approach, which can be the best choice for moderate-scale ASPs. To solve (ii), we then devise a set of predictive models for user QoE and Quality of Application (QoA) estimation, which can be considered as common open-loop (no user feedback required) and closed-loop (requires application feedback) control systems [8], respectively. Opposite to widely used static QoE profiles [9], these models capture dynamics in a Quality of Service (QoS), QoA and user QoE (3Q) interplay. (Section III)

− To overcome NP-hard ILP solution intractabilities for large-scale ASPs, we also propose its polynomial greedy heuristic counterpart that leverages a novel Multi-Constrained Path (MCP) aware Possibilistic C-Means (PCM) approach for infrastructure clustering. Our MCP-PCM ensures the possibility of having satisfactory user SLOs (which caters to user QoE expectations) during cloud service placement. (Section IV)

− To utilize our 3Q model for solving (ii), we propose another polynomial greedy heuristic algorithm that leverages a decision tree scheme which applies adaptations in a least cost-disruptive (LCD) manner. Our LCD decision tree scheme manages the 3Q interplay for handling trade-offs between satisfactory service delivery, cost of adaptations, and user disruption level factors. (Section V)

− Using numerical simulations in the NS-3 simulator [10] with realistic Internet topologies generated with the BRITE tool [11], we evaluate our QoE-oriented cloud service orchestration algorithms within both management and data planes. Particularly, we show how our management plane solution with MCP-PCM clustering scheme is superior than related topological and geographical clustering schemes [12] for the service resource provisioning by allowing more users to experience satisfactory SLOs. Using various workload scenarios (from sequential to full user access rates), we also show how our LCD decision tree is adaptable to these scenarios (by leveraging our 3Q model) and gains of up to 50% in (profiled) user QoE than related solutions. (Section VI-A)

− Using a real-world GENI Cloud testbed [13] with actual users, we show how QoE-oriented service orchestration improves overall system QoS and (actual) user QoE for our Skytap and LOFT application case studies. (Section VI-B)

II. Related Work

The literature of the QoE-oriented application service orchestration is vast, and we only focus here on a few prior works that are related to our work. A complete survey of QoE management challenges for cloud services is discussed in Hoßfeld et al. [3].

QoE Enhancement within the ASP Management Plane. Moving application services to the cloud makes network management a major challenge for delivering satisfactory user QoE [3]. Solutions to this problem can be found in both complementary subareas - application-aware networking [4], [14] and cloud service placement [5]. Application-aware networking aims to enhance user QoE by improving network QoS. It can be used within both management (when services are unavailable) and data (when services are operational) planes. For example, Google [4] fair shares its global data center backbone bandwidth, prioritizing different application traffic, and assigns highest priority to users' data. Georgopoulos et al. [14] in turn fair schedule their network traffic by using video QoE models. However, application-aware networking requires expensive infrastructure control, and thus can be mainly used only by InPs. Although the cloud service placement does not require infrastructure control, it is commonly done by InPs to maximize their revenue within their management plane, e.g., to maximize resource utilization and/or energy efficiency instead of maximizing the satisfactory user QoE delivery of services. Similarly, InPs increase their network scalability and flexibility using a virtual network function placement algorithm [2]. Thus, Calyam et al. [5] minimize financially expensive cloud resources, by over provisioning during virtual desktop service placement, which allows InPs to accept more virtual desktop requests. Another more general management protocol example for InPs is a Virtual Network Embedding (VNE) [1] that is an NP-hard graph matching problem of mapping a constrained virtual network on top of a shared physical infrastructure. However, both cloud service placement and VNE require virtual network or cloud service requests, i.e., requests with specified number of virtual nodes, links and their demands. In contrast to VNE, the general ASP problem assumes no *apriori* knowledge of the required number of application service instances (i.e., virtual nodes), but tries to maximize a satisfactory QoE delivery to users subject to the ASP budget by deciding how many service instances should be rented from the InP, where to place them within its infrastructure and how to dynamically orchestrate them within the ASP data plane.

In this paper, we use the synergy of both subareas to fill the gaps of QoE-oriented cloud service orchestration algorithms within both the ASP management and data planes tailored for an ASP's business model presented in Section I. Thus, we assume we have as input the ASP's budget and have application service demands expressed as SLOs (e.g., network bandwidth, delay, losses and jitter), which can be derived from users' QoE profiles for a particular cloud service.

QoE Enhancement within the ASP Data Plane. To further enhance a satisfactory user QoE delivery within the ASP data plane by dynamically orchestrating its services (running on resources rented from InPs), there are plenty of candidate adaptations proposed in the literature. Since not all adaptation are suitable for ASPs, we describe a few adaptation strategies that are broadly applicable and illustrate using the LOFT application case study, to highlight our contributions to the ASP optimization problem. Examples of unsuitable ASP adaptations include those that require expensive infrastructure control, which can result in excessive or potentially disruptive outcomes, or cannot be used by a particular application. Park et al. [15] proposed a mechanism for adaptations at the client side where they apply certain bit rate adaptation schemes and are mainly focused on improving user QoE. However, the authors show only QoE improvement results and do not show the effect of their adaptations on other factors such as the total cost and the overall disruption impact on other users. Ma et al. [16] used path switching adaptations for the streaming application to enhance user QoE. Similarly, they show improvement in user QoE, but do not evaluate the impact of their approach on the total cost or the overall user disruption. We remark that — provisioning enough infrastructure control needed for path switching can be a very expensive proposition for ASPs. Van Beek et al. [17] proposed adaptations at the server-side, where a bit rate adaptation scheme was applied for all clients attached to a particular server. Although their adaptation strategy shows user QoE benefits with no extra cost, reducing bit rates for all users on an affected server can result in higher overall user disruption. ASPs generally can also benefit from applying adaptations that can scale up or scale down resources rented from InPs to elastically orchestrate them based on their budget. For example, such adaptations can use queuing models to analyze the QoS levels and decide on servers to scale up as shown by Xiong et al. [18]. Another example of such adaptations includes reactive model adaptations (based on the profiled CPU utilization and regression mechanisms), that can dynamically scale up and down a service instance's resources [19].

Although these adaptations are generally applicable for visual-processing applications [20], they are designed for a single service instance and do not take overall ASP budget and user disruption factors into account. To overcome the above limitations tailored to the ASP business model, we propose a least cost-disruptive scheme that manages QoS, QoA and user QoE interplay dynamics and can incorporate all of these specific adaptations to enhance QoE delivery within the ASP data plane while mitigating user disruption level.

III. Application Service Provider Problems

In this section we formally state and discuss two main ASP optimization problems that synergistically maximize the satisfactory QoE service delivery to users under constraints in the ASP's cost budget as illustrated in Figure 1. The two classes of problems are: (i) maximization of the user SLO coverage within the ASP *management plane* when services are unavailable; and (ii) maximization of the user QoE coverage within the ASP *data plane* when services are operational.

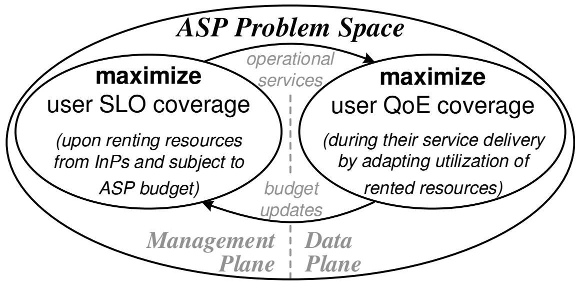

ASP Problem Space
maximize
user SLO coverage
(upon renting resources from InPs and subject to ASP budget)
operational services
maximize
user QoE coverage
(during their service delivery by adapting utilization of rented resources)
budget updates
Management Plane
Data Plane

Fig. 1: Overview of the main ASP optimization problems: (i) ASPs can first maximize the number of users with fully satisfied SLOs (i.e., SLO coverage) by determining the number of rented resources from InPs subject to the ASPs' budget, their placement and the user clustering between them; and (ii) ASPs can then maximize the number of users with satisfied QoE (i.e., QoE coverage) by synergistically adapting their rented resources to enhance the service delivery to end-users and update their budget if needed.

A. ASP optimization problem within management plane.

While deploying service instances within the ASP management plane, we can maximize the user SLO coverage, i.e., the α_{SLO} percent of users with fully-satisfied SLO obtained directly from a corresponding QoE profile of a particular cloud service [9]. To optimally solve the ASP management plane problem, we use a well-known multi-commodity flow problem [21], where each commodity can represent a particular client-server pair. To ensure user SLO coverage, we need to check only if both the assigned service instance and a path to it have required SLO guarantees. There are two common ways of formulating multi-commodity flow problems: (i) arc-based and (ii) path-based. The arc-based formulation can result in an infeasible solution for the ASP, i.e., it can assign users to a path which violates routing policies of InPs. Thus, in this paper we use the path-based multi-commodity flow formulation where available paths are obtained based on the routing policy of InPs. Note that ASPs with full infrastructure control can benefit from changing our path-based formulation to the arc-based. Formally, we have:

Definition 1 (ASP management plane optimization problem): Given an infrastructure topology $\mathcal{T} = \mathcal{T}(V, C, \mathcal{P})$ provided by an InP, where V is a set of physical nodes, $c_v \in C$ is a cost of renting resources at $v \in V$ and $P^k_{ij} \in \mathcal{P}$ is a set of k paths available between source v_i and destination v_j vertices based on InP's routing policy; let x_i be 1 if a service instance is allocated at vertex v_i (or 0 otherwise), let f^k_{ij} be 1 if a flow between an allocated at vertex v_i service and a connected through vertex v_j user is mapped on a k-th path $\in P^k_{ij}$ (or 0 otherwise), and let α_j be 1 if a connected through vertex v_j user is mapped on a path with fully satisfied SLO constraints (or 0 otherwise); finally, let $\bar{l}$, $\bar{p}$ and $\bar{s}$ denote vectors of link, path and server SLO constraints, respectively; where $\bar{l}$ corresponds to min/max path P^k_{ij} weights $w^{\bar{l}}_{ijk}$, $\bar{p}$ corresponds to additive/multiplicative path P^k_{ij} weights $w^{\bar{p}}_{ijk}$, and $\bar{s}$ corresponds to physical node weights $w^{\bar{s}}_i$ (*i.e.*, that $\geq$ or $\leq$ than SLO constraints); the ASP SLO coverage α_{SLO} maximization problem within management plane can be formulated as following:

$$\text{maximize} \quad \alpha_{SLO} = \frac{1}{|V|} \sum_{j=1}^{V} \alpha_j \tag{1}$$

subject to
Budget constraint

$$\sum_{i=1}^{V} c_i x_i \leq Budget, \tag{2}$$

User assignment constraints

$$\sum_{i=1}^{V} \sum_{k \in P_{ij}^{k}} f_{ij}^{k} = 1, \forall j \in V \tag{3}$$

Service placement constraints

$$\sum_{j=1}^{V} \sum_{k \in P_{ij}^{k}} f_{ij}^{k} - |V| x_i \leq 0, \forall i \in V \tag{4}$$

SLO link constraints

$$\sum_{i=1}^{V} \sum_{k \in P_{ij}^{k}} w_{ijk}^{l} f_{ij}^{k} \gtreqless l \mp M(1-\alpha_j), \forall j \in V, l \in \bar{l} \tag{5}$$

SLO path constraints

$$\sum_{i=1}^{V} \sum_{k \in P_{ij}^{k}} w_{ijk}^{p} f_{ij}^{k} \gtreqless p \mp M(1-\alpha_j), \forall j \in V, p \in \bar{p} \tag{6}$$

SLO server constraints

$$w_i^{s} x_i \gtreqless s, \forall i \in V, s \in \bar{s} \tag{7}$$

where symbols and notations of sets, parameters, variables and functions are summarized in Table I. A compact notation is used to cover either upper (i.e., $\leq$) or lower (i.e., $\geq$) bounds in the SLO constraints (see Equations 5, 6 and 7).

Note that to simplify the management plane problem solution, server SLO constraints in Equation 7 can be omitted and all binary x_i variables that do not satisfy them can be set to 0. Note also that if splittable flow is acceptable, $f_{ij}^{k} \in \{0,1\}$ can be relaxed as $f_{ij}^{k} \in [0,1]$ to further simplify this problem solution. Hereon, we assume the unsplittable flow ASP policy, i.e., the end-to-end traffic can traverse only a single path.

When maximizing α_{SLO} (see Equation 1), the management plane ASP problem solution forces more users to choose paths with fully satisfied SLO (see Equations 5 and 6), which in turn can increase number of service instance placements (see Equation 4) which are constrained by the ASP budget (see Equation 2). Finally, all users have to be assigned to exactly one service instance (see Equation 3). We illustrate a management plane problem solution using our Skytap case study example in Section IV-C. Furthermore, as proposed solution uses integer linear programming, it is NP-hard and can be intractable for large-scale ASPs. To cope with these scalability limitations, we propose a polynomial greedy heuristic algorithm based on unsupervised learning clustering techniques in Section IV.

B. ASP optimization problem within data plane.

To further enhance user QoE coverage within data plane (when services are operational), ASPs can periodically optimize their usage of rented resources from InPs by applying various service adaptations including client, network and server side adaptation strategies for unsatisfied users. On the contrary, applying all possible service adaptations for

TABLE I: Symbols and Notations for Optimization Problems

Sets	
V	$\triangleq$ Set of physical nodes within the infrastructure
C	$\triangleq$ Set of corresponding node rental costs
P_{ij}^{k}	$\triangleq$ Set of k paths available between source v_i and destination v_j vertices based on InP's routing policy
R	$\triangleq$ Set of rented resources within the infrastructure
$\mathcal{A}(R)$	$\triangleq$ Set of all possible adaptations for resources R
U^t	$\triangleq$ Set of accessing at time t users
D^t	$\triangleq$ Subset of disrupted users $D^t \subseteq U^t$, *i.e.*, whose QoA (or QoE) was affected by previously made adaptations
Variables	
x_i	$\triangleq$ Binary variable that equals to 1 if a service instance is allocated at vertex v_i
f_{ij}^{k}	$\triangleq$ Binary variable that equals to 1 if a flow between an allocated at vertex v_i service and a connected through vertex v_j user is mapped on a k-th path $\in P_{ij}^{k}$
α_j	$\triangleq$ Binary variable that equals to 1 if a connected through vertex v_j user is mapped on a path with fully satisfied SLO constraints
α_{SLO}	$\triangleq$ Positive variable that equals to the percent of users with fully-satisfied SLO
α_{QoE}^{t}	$\triangleq$ Positive variable that equals to the percent of users with fully-satisfied QoE (or QoA) at time t
Vectors	
$\bar{l}$	$\triangleq$ Vector of link SLO constraints
$\bar{p}$	$\triangleq$ Vector of path SLO constraints
$\bar{s}$	$\triangleq$ Vector of server SLO constraints
Parameters	
w_{ijk}^{l}	$\triangleq$ Parameter that corresponds to the min/max path P_{ij}^{k} weight of the $l \in \bar{l}$ link SLO constraint (*e.g.*, bandwidth)
$w_{ijk}^{\bar{p}}$	$\triangleq$ Parameter that corresponds to the additive/multiplicative path P_{ij}^{k} weight of the $p \in \bar{p}$ path SLO constraint (*e.g.*, latency)
$w_i^{\bar{s}}$	$\triangleq$ Parameter that corresponds to the physical node v_i weight of the $s \in \bar{s}$ server SLO constraint (*e.g.*, CPU)
M	$\triangleq$ Parameter that corresponds to a practically big number used to relax SLO constraints for non-covered users (*i.e.*, the big M method)
$Budget$	$\triangleq$ Parameter that corresponds to ASP budget
λ	$\triangleq$ Positive parameter that corresponds to a user disruption penalty
Functions	
$C(A^t)$	$\triangleq$ Cost function of a subset of adaptations $A^t \subseteq \mathcal{A}(R)$ at time t
$f_{QoE}^{t}(R, A^t, u)$	$\triangleq$ Binary function that equals to 1 if a user $u \in U^t$ has a fully satisfied QoE under a set of rented resources R after ASP management plane optimization with the applied at time t adaptations A^t

unsatisfied users can disrupt other (satisfied) users. Thus, the goal of the ASP optimization data plane problem is to pick a subset of adaptations at some period of time that can best improve users' QoA and QoE coverage for the next period of time with a minimum disruption level. Thus, the ASP data plane optimization problem can be formulated as the NP-hard nonlinear Knapsack problem [22]. Formally, we have:

Definition 2 (ASP data plane optimization problem): Given a subset of rented resources R, a set of all possible adaptations for these resources $\mathcal{A}(R)$, a cost function $C(A^t)$ of a subset of adaptations $A^t \subseteq \mathcal{A}(R)$ at time t, a set of accessing at time t users U^t with a subset of disrupted users $D^t \subseteq U^t$ i.e., whose QoA (or QoE) was affected by previously made adaptations, subject to ASP $Budget$, the ASP optimization problem within the data plane can be formulated as follows:

$$\underset{A^t \subseteq \mathcal{A}(\mathcal{R})}{\text{maximize}} \left[\alpha_{QoE}^{t} = \frac{1}{|U^t|} \sum_{u \in U^t} f_{QoE}^{t}(R, A^t, u) \right] - \lambda \frac{|D^t|}{|U^t|} \tag{8}$$

$$\text{subject to} \quad C(A^t) \leq Budget$$

where symbols and notations of sets, parameters, variables and functions are summarized in Table I. Note that the above optimization problem solution finds an expected optimal (not the actual one) as user QoE is unknown in advance, and function f_{QoE}^{t} can only estimate its expected value.

3Q interplay model for ASP data plane problem. In order to enhance user QoE coverage and minimize user disruptiveness, we aim to develop a model for $f_{QoE}^{t}(R, A^t, u)$ that describes the relationship between the InP's data plane QoS and the ASP's data plane QoA so that we can decide on service adaptation strategies for various scenarios (e.g., congestion in the

InP data plane or a high user access rate in ASP data plane). However, the QoS, QoA and user QoE interplay itself is highly dynamic and depends on many factors including the type of applications, time periods of use (e.g., Friday night vs Saturday morning), etc. To this end, our 3Q interplay model dynamically adapts the 3Q interplay based on the feedback from both user (optional, if available) and application (required), where the former and the latter can be considered as classical open-loop and closed-loop control systems [8], respectively. Note that an application's feedback can be obtained from tools such as VDBench [23] in operational environments. This is how the 3Q model varies from common QoE profiles where this interplay is assumed to be static [9]. Finding the best open-loop and closed-loop control systems for the 3Q model (e.g., using machine learning approaches) is the out of scope for this paper and we leave it as an open area for future work.

Our 3Q model is also different than existing QoE predictive models. Thus, Menkovski et al. [24] proposed a predictive QoE model based on QoS conditions. Our model is in turn prescriptive with regards to how to change QoS and QoA to optimize QoE. For example, in the context of a multimedia application, bandwidth rate of an Internet connection will affect the actual framerate of the application, and the delay of a connection will directly affect the application's responsiveness. To avoid the NP-hard non-linear Knapsack problem solution for the ASP data plane optimization (see Definition 2), we propose another greedy heuristic algorithm in Section V, which is based on the least cost-disruptive decision tree scheme and our 3Q model.

IV. MCP-aware PCM clustering for ASP management plane

A. Background

Maximizing user SLO coverage under constrained ASP's budget for rented resources by solving the ASP *management plane* problem (see *Definition 1*) corresponds to an infrastructure clustering for service placement and consequent user mapping to these service instances with SLO satisfaction.

Existing graph clustering approaches, however, minimize only a single metric as a cluster distortion, and hence can facilitate only a single path SLO satisfaction such as latency, packet loss, etc.; e.g., K-Means method [12] can cluster a set of vertices using a topological order while taking into account an arbitrary number of link SLO constraints such as bandwidth. In contrast to K-Means, we devise a MCP aware clustering scheme by extending a common PCM method [25] to minimize multiple distances related to multiple SLOs (cluster distortion). This in turn allows MCP-PCM to maximize possibility of finding multi-constrained network paths with fully satisfied (both link and path) SLOs between infrastructure nodes and corresponding cluster centers.

B. MCP-PCM Clustering Scheme

Based on prior knowledge of required number of clusters k (amount of resource to be rented by ASPs) and given a cloud infrastructure, we cluster it with a cloud service SLO awareness. To this end, we use a PCM algorithm to estimate possibility U^p_{vk} of each vertex $v \in V$ of having a path which satisfies a particular SLO path constraint $p \in \bar{p}$ to the k cluster center of an underlying infrastructure graph:

$$U^p_{vk} = \frac{1}{1 + (\frac{{d_p}^2(v,k)}{\eta^p_{vk}})^{\frac{1}{m-1}}}, \tag{9}$$

where $m \in (1, \infty)$ is a pre-specified fuzzifier whose increase blurs cluster borders (i.e., degrades users' membership in their most possible clusters), and η^p_{vk} is a scaling factor for a particular path constraint p, vertex v and cluster center k cost. By default, m is set to a common value of 2 [25], which is service/infrastructure specific, and impacts cluster quality. η^p_{vk} is empirically chosen to have 0.5 possibility if k-th cluster center has a unitary cost, and a path distance d_p equals to a half of p SLO constraint: $\eta^p_{vk} = (p/2)^2$. Note how our model assumes having different scaling factors η^p_{vk} for a particular vertex v to cover cases with different application service types and underlying physical resource costs. In addition, to ensure $l \in \bar{l}$ SLO constraints are satisfied, all edges e which do not satisfy l are removed before clustering is performed. In this scheme, we use the Dijkstra algorithm as our distance function $d_p(v,k)$ for additive path metrics[1]. Moreover, if $d_p(v,k) > p$ we set $d_i(v,k) = \infty$. Finally, we estimate overall possibility U_{vk} for all path constraints p to be satisfied for a vertex v using the product rule:

$$U_{vk} = \prod_{p \in \bar{p}} U^p_{vk}. \tag{10}$$

We then cluster an infrastructure by assigning a center K_v that has the highest possibility of all path constraints to be satisfied at each vertex $v \in V$:

$$K_v = arg \max_k U_{vk}. \tag{11}$$

To update centers, we run the Dijkstra algorithm from each vertex $v \in V$ to all other vertices within cluster k and choose v with the minimum normalized distortion σ^k as the new center, where we calculate the distortion as:

$$\sigma^k_v = \sum_{p \in \bar{p}} \sum_{j=1}^{V} \frac{d_p(v,j)}{p}. \tag{12}$$

How complex is our MCP-PCM scheme? The MCP-PCM model runs Dijkstra exactly two times for each vertex $v \in V$ and for a particular $p \in \bar{p}$ constraint: once during possibility estimation and then during center updates. Taking into account that Dijkstra complexity is $O(|V|log|V| + |E|)$, the final complexity of this step is:

$$O(\epsilon \cdot |\bar{p}| \cdot |V| \cdot |V|log|V| + |E|) = O(\epsilon|\bar{p}||V|^2 log|V|), \tag{13}$$

where ϵ is the maximum number of iterations. Based on our empirical observations, we observed that, $\epsilon \leq |V|$, so we use the initialization, $\epsilon = |V|$.

C. Skytap Case Study Example

Considering a Skytap case study of a virtual learning environment as an example, our goal is to place up to 2 Virtual Desktop (VD) servers (based on the ASP budget) by clustering an infrastructure so that all users will have a path to at least one VD server with the following SLOs: bandwidth $\geq$ 15 Mbps (to support high-definition video streaming), delay $\leq$ 50 ms

[1]Note that multiplicative network metrics (e.g., losses) can be converted to additive metrics by composing them with a logarithmic function.

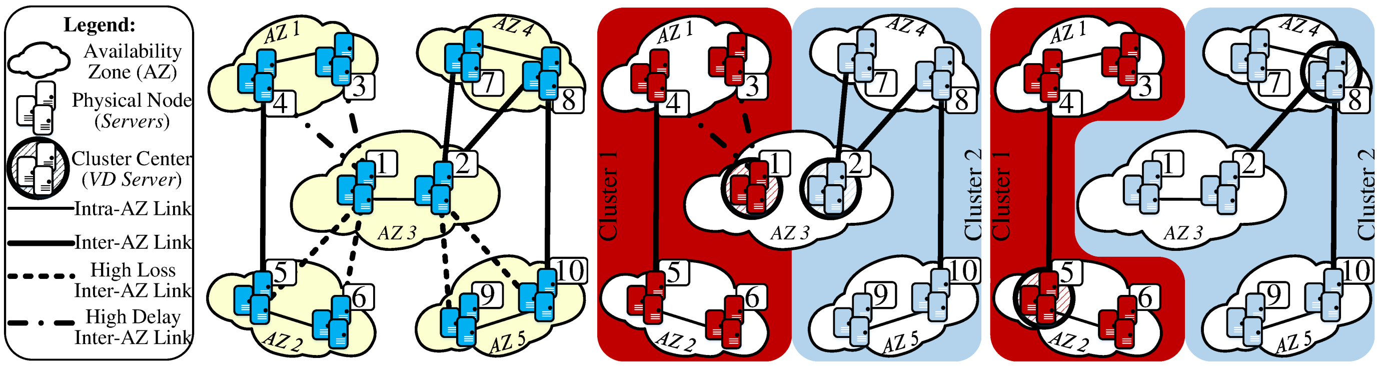


Fig. 2: Illustrative example of an infrastructure clustering (shown in (a)) with K-Means topological proximity (shown in (b)) and Multi-Constrained Path aware PCM-based models (shown in (c)); K-Means approach ignores path SLO constraints during clustering, and as a result, only 2 amongst 8 nodes have paths to the their cluster centers with satisfied SLOs; switching to the proposed MCP-PCM clustering allows all nodes to have network paths with satisfied SLOs (see Table II).

TABLE II: Clustering results

Model	Cluster	Network Path	SLO Violation	SLO Coverage
K-Means	Cluster 1	$3 \rightarrow 1$	**Delay, Losses**	**2/8**
		$4 \rightarrow 1$	**Delay, Losses**	
		$5 \rightarrow 4 \rightarrow 1$	**Delay, Losses**	
		$6 \rightarrow 5 \rightarrow 4 \rightarrow 1$	**Delay, Losses**	
	Cluster 2	$7 \rightarrow 2$	No	
		$8 \rightarrow 2$	No	
		$9 \rightarrow 10 \rightarrow 8 \rightarrow 2$	**Delay, Losses**	
		$10 \rightarrow 8 \rightarrow 2$	**Delay, Losses**	
PCM/ILP	Cluster 1	$3 \rightarrow 4 \rightarrow 5$	No	**8/8**
		$4 \rightarrow 5$	No	
		$6 \rightarrow 5$	No	
	Cluster 2	$1 \rightarrow 2 \rightarrow 8$	No	
		$2 \rightarrow 8$	No	
		$7 \rightarrow 8$	No	
		$9 \rightarrow 10 \rightarrow 8$	No	
		$10 \rightarrow 8$	No	

(for seamless remote control) and packet loss $\leq 0.05\%$ (to have imperceptible service impairments).

Given an infrastructure of 5 availability zones (AZs) and 10 nodes as shown in Figure 2a, we assume that all intra-AZ links have 1 ms one-way delay and 0.01% packet loss, and all inter-AZ links have 40 ms delay and 0.04% losses. Moreover, we assume that inter-AZ links between AZ 1 and AZ 3 have high delay (100 ms) and losses (1%), and inter-AZ links between AZ 2 and AZ 3, and between AZ 5 and AZ 3 have very high packet loss (10%). Finally, all links have more than the 15 Mbps bandwidth required for service provisioning.

The topological proximity graph clustering with K-Means [12] ignores path SLO constraints and places cluster centers to the nodes with the highest node degree (see Figure 2b). As a result, the MCP algorithm [26] finds only 2 paths out of the 8 possible paths which satisfy all SLOs. Our MCP aware PCM clustering (see Figure 2c) places cluster centers in a manner that allows the same MCP algorithm to find all paths with fully satisfied SLOs (see Table II). Note that in this case, PCM service placement also matches with the optimal ILP solution for the ASP management plane (see *Definition 1*).

D. Algorithm for ASP optimization within management plane

To solve the ASP management problem (see *Definition 1*), we propose a scalable greedy heuristic algorithm which utilizes our MCP-aware PCM clustering scheme outlined in Algorithm 1. As SLO coverage improves with the number of increased service instances (that we allocate from InPs), our algorithm starts by estimating the largest number of clusters (service instances) which can fit into the ASP budget (line 3). At each iteration, we first cluster the infrastructure with l and p SLO constraints awareness (line 5) and then check the feasibility of clustering results (line 6), i.e., if mapping of service instances (or cluster centers) K satisfies ASP budget constraint. Once a feasible solution is found, the proposed algorithm finds the best user SLO coverage α_{SLO} (although it is still suboptimal w.r.t. the IP solution in Definition 1). We finally estimate α_{SLO} using $coverage_{SLO}(U, K, V, E, l, p)$ function (line 9) that checks all paths between user-to-service clusters U and their corresponding cluster centers K (service placement) for SLO satisfaction using InPs' routing policies such as OSPF, BGP, etc.

Algorithm 1: ASP management plane optimization

Input: InP topology with V nodes, E links and cost vector C, ASP $Budget$ and SLO l/p constraints.
Output: User-to-service clusters U, their center locations K (service placement), user SLO coverage α_{SLO}.

```
1  begin
2      sort vertices $V$ in ascending order by their cost $C$
3      $k \leftarrow \underset{k_{max}}{\mathrm{argmax}}(\sum_{i=1}^{k_{max}} c_i v_i \leq Budget)$
4      $isBudgetSatisfied \leftarrow 0$
5      while $isBudgetSatisfied == 0$ and $k \geq 1$ do
6          $[U, K] \leftarrow MCP\_PCM(k, V, E, l, p)$
7          if $\sum_{v_i \in K} c_i v_i \leq Budget$ then
8              $isBudgetSatisfied \leftarrow 1$
9              $\alpha_{SLO} = coverage_{SLO}(U, K, V, E, l, p)$
10         end
11         else
12             $k \leftarrow k - 1$
13         end
14     end
15 end
```

V. Least Cost-Disruptive Decision Tree for ASP Data Plane

A. Background

ASPs need to maintain good standing in their client relationships in terms of service reliability, security and adaptability. They also have to service the clients by adapting to different strategies that solve the issues which arise while supporting a particular application at cloud-scale. Thus, ASPs have to balance between enhancing user QoE through various adaptation strategies and mitigating a potential user disruption caused by these adaptations. To preserve this balance, ASPs typically require sophisticated engineering and operations groups, which in turn increases the cost charged to the clients. On the

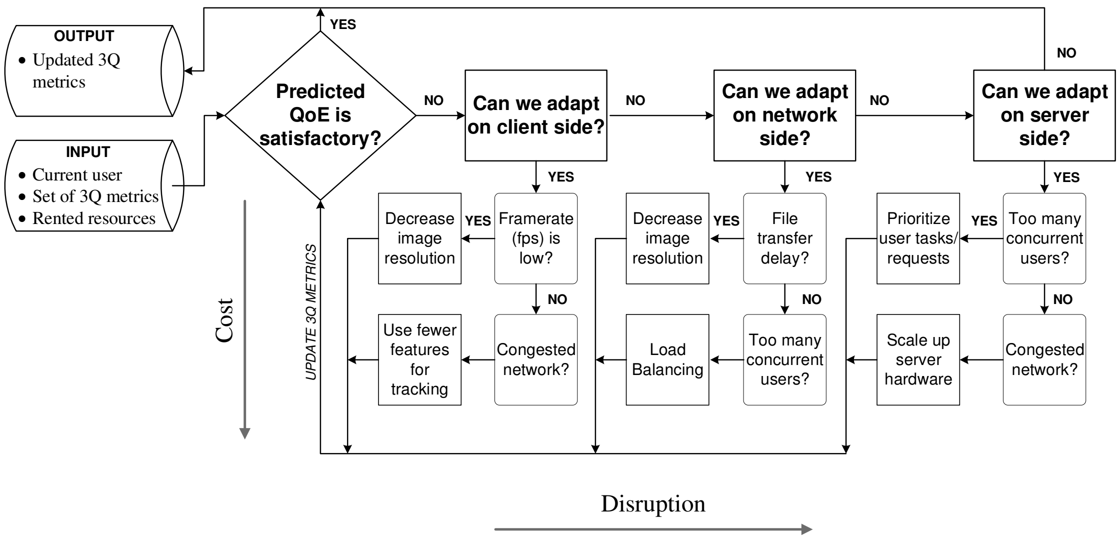

OUTPUT
• Updated 3Q metrics
INPUT
• Current user
• Set of 3Q metrics
• Rented resources
Predicted QoE is satisfactory?
YES
NO
Can we adapt on client side?
Can we adapt on network side?
Can we adapt on server side?
Decrease image resolution
Framerate (fps) is low?
Use fewer features for tracking
Congested network?
File transfer delay?
Load Balancing
Too many concurrent users?
Prioritize user tasks/ requests
Scale up server hardware
UPDATE 3Q METRICS
Cost
Disruption

Fig. 3: Least Cost-Disruptive decision tree scheme example for the LOFT case study: adaptation strategies are applied in a greedy fashion where cost and potential user disruption increase.

contrary, we can minimize the complexity of ASP's engineering operations by using the proposed 3Q interplay model in Section III to predict potential QoE enhancement by applying adaptation strategies that consider user disruption level as well. To this aim, we propose an assessment technique that is based on a LCD decision tree scheme for leveraging adaptation strategies to trade-off between the service quality, cost and user disruption factors.

B. LCD Decision Tree Scheme

While the 3Q model predicts the anticipated service quality changes after an applied adaptation strategy for a particular user, there is still the need to pick a subset of such adaptations to maximize satisfactory QoE delivery to all users of the ASP while tolerating some user disruption level based on the λ penalty coefficient in the ASP *data plane* optimization problem (see *Definition 2*). To address this limitation, we propose a decision tree model that applies various adaptation strategies in a least cost-disruptive manner by leveraging our 3Q model. Particularly, our LCD decision tree classifies all adaptations based on a measurement of user disruptiveness and cost, where adaptations that are made on the server side are commonly more disruptive and potentially more expensive than adaptations made on the client side. Thus, applying the LCD decision tree to our 3Q model solves the ASP *data plane* optimization problem in a greedy manner.

Figure 3 shows our LCD decision tree scheme, where all adaptation strategies are applied based on their level of disruptiveness (e.g., from the client to the server side adaptations) as well as on their cost. Thus, the LCD scheme tries to first apply the cheapest client side adaptation, whereas the most expensive and disruptive server side adaptations are likely going to be applied in the last iteration. We remark that decisions of whether or not it is worth applying some adaptations are based on the 3Q interplay model. We refer to all adaptation strategies that involve the renting of additional resources from InPs as 'promotion' adaptation strategies, and we refer to all adaptation strategies that reduce the amount of rented resources from InPs as 'demotion' adaptation strategies.

How complex is our LCD scheme? The above described LCD decision tree scheme executes the 3Q interplay model at most once for each possible adaptation $a \in \mathcal{A}(R)$ applicable to a set of rented resources R to estimate adaptation impact on each user $u \in U^t$ QoE at time t. Assuming the control system asymptotic complexity $O(\xi)$ (used in the 3Q model), the final complexity of LCD is:

$$O(2 \cdot \xi \cdot |\mathcal{A}(R)| \cdot |U^t|) = O(\xi \cdot |\mathcal{A}(R)| \cdot |U^t|). \tag{14}$$

Note that the $O(\xi)$ asymptotic complexity is a control system-specific [8] and can vary for different control system types that can be utilized in our 3Q model.

C. LOFT Case Study Example

Considering the LOFT object tracking application [7] as an example, our goal is to adapt allocated InPs resources within the ASP data plane in a least cost-disruptive manner to further enhance user QoE. To further enhance LOFT QoA delivery within the data plane right after the initial service placement is done, LCD proceeds with the three adaptation scenarios at client, network and server sides. Strategies such as bit rate reduction (by decreasing image quality) at the client-side, traffic steering to mitigate congestion at the network-side, promotion of hardware resources to speedup image tracking at the server-side are considered for LOFT. Figure 4 shows an example workflow of the LOFT cloud service setup, where a client sends unprocessed images to a server running LOFT for object tracking and receives processed images back for further analysis.

Table III describes potential issues that can arise during LOFT service delivery, and how our decision tree resolves these issues in the least cost-disruptive manner. To assess QoA for our case study, we use the tracking speed (framerate) of the LOFT application measured in frames per second (fps). In this example, we also assume the network bandwidth is 2 Mbps for all physical links. To represent QoA adaptation, we have created two identical video datasets at two frame resolutions. Data type 1 uses images at a frame resolution of 240x180 pixels, and data 2 type has images with a resolution

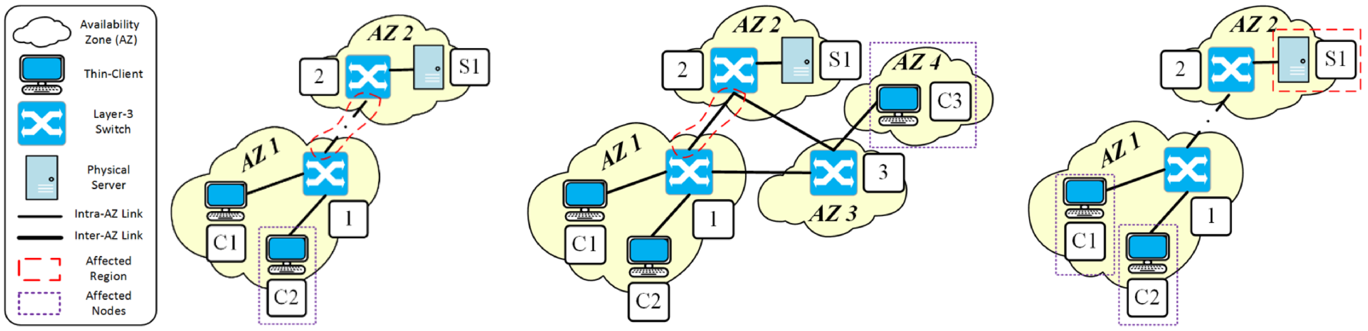


Fig. 4: Illustrative example of LOFT cloud service adaptations at the client (left), network (center) and server (right) sides: all adaptations can improve service quality at the expense of cost and/or other users' QoE disruption (see Table III).

TABLE III: LCD Adaptation results

Adaptation	Status	Data Type	Network Path	Server Cores	QoA Tracking Speed (fps)	SLO Violation	Users Disrupted
Client	Before	1	$\{C1, 1, 2, S1\}$	2	≈ 3	**Delay**	None
		2	$\{C2, 1, 2, S1\}$	2	≈ 3	None	None
	After	**2**	$\{C1, 1, 2, S1\}$	2	$\approx \mathbf{5}$	None	**C2**
		2	$\{C2, 1, 2, S1\}$	2	≈ 3	None	None
Network	Before	1	$\{C1, 1, 2, S1\}$	2	≈ 1	**Delay**	None
		2	$\{C2, 1, 2, S1\}$	2	≈ 3	None	None
	After	1	$\{\mathbf{C1, 1, 3, 2, S1}\}$	2	$\approx \mathbf{3}$	None	**C3**
		2	$\{C2, 1, 2, S1\}$	2	≈ 3	None	None
Server	Before	1	$\{C1, 1, 2, S1\}$	1	≈ 1	**CPU**	None
		1	$\{C2, 1, 2, S1\}$	1	≈ 1	**CPU**	None
	After	1	$\{C1, 1, 2, S1\}$	**4**	$\approx \mathbf{3}$	None	**C1**
		1	$\{C2, 1, 2, S1\}$	**4**	$\approx \mathbf{3}$	None	**C2**

of 120x90 pixels. We also assume that data types 1 and 2 consume 2 and 1 Mbps of bandwidth, respectively.

In the client side adaptation scenario, two clients that are concurrently accessing LOFT services running on a single physical server are considered (see Figure 4-left). Moreover, client $C1$ uses data type 1, and client $C2$ uses data type 2. As bandwidth is fair shared between $C1$ and $C2$, $C1$ tracking speed is affected due to network congestion between switches 1 and 2, $C1$ has 1 Mbps of bandwidth while the data type 1 requires 2 Mbps. Based on our decision tree, we first reduce the image resolution at the $C1$ (client) side to be exchanged between the client and the server. After adaptation, $C1$ tracking speed increases at the expense of image quality and tracking accuracy. In the network side adaptation scenario (see Figure 4-center), two clients requesting LOFT services are used to run on a single physical server, but now $C1$ is restricted to use data type 1. In this case, a client side adaptation cannot be applied, and hence, LCD decision tree proceeds with a traffic steering adaptation to reroute $C1$ data through a different path. This adaptation strategy improves $C2$ tracking without affecting the image quality, but can disrupt the $C3$ user who was unaffected before the adaptation took place. The server side adaptation scenario (see Figure 4-right) involves a situation of under-provisioned hardware. In this example, $C1$ and $C2$ concurrently request LOFT services using data type 1 from the same server. However, we assume server hardware resources are not sufficient for the given workload, and hence, service quality (tracking speed) of both $C1$ and $C2$ is not satisfied. Thus, our decision tree applies the only possible server side adaptation to promote server hardware resources to accommodate for computation loads. This adaptation significantly improves both $C1$ and $C2$ QoA, but is the the most expensive and disruptive option for the users (as service becomes temporarily unavailable for all users during resource promotion).

D. Algorithm for ASP optimization within the data plane

To avoid the NP-hard non-linear integer programming solution for the ASP data plane optimization problem (see *Definition 2*) for a period of time t, we propose a scalable greedy heuristic algorithm that utilizes our 3Q-based LCD decision tree scheme outlined in Algorithm 2. As users with higher QoA (or QoE) have higher chances of being fully satisfied after applied adaptations, our algorithm starts by sorting them in descending order by their QoA (or QoE) (line 2). For each user $u \in U^t$, we first check its satisfaction (line 6), and if u is satisfied we try to apply resource R demotion adaptations without however affecting u satisfaction to procure more available budget B_{avail} (line 8) for future R promotion of unsatisfied users (line 13). Moreover, for each unsatisfied user u, we apply our LCD decision tree scheme that applies adaptations in a least-cost disruptive manner (by leveraging 3Q model) and prescribes $3Q_u^{t+1}$ metrics for the next period of time (line 11). Finally, proposed algorithm estimates the user QoA (or QoE) coverage $\alpha^t_{QoA|QoE}$ at time t.

Algorithm 2: ASP data plane optimization

```
Input: Set of accessing users $U^t$, rented from InP resources $R$ with cost $C$, set
       of $3Q_u^t$ metrics (QoS/QoA/QoE) at time $t$ for each user $u$ and ASP
       $Budget$ constraint.
Output: Prescribed $3Q_u^{t+1}$ metrics for a period of time $t+1$, user QoA/QoE
        coverage $\alpha^t_{QoA|QoE}$.
1  begin
2      sort users $U^t$ in descending order by their QoA (or QoE if available)
3      $satisfiedUsers \leftarrow 0$
4      $B_{avail} \leftarrow Budget - C(R)$
5      for $u \in U^t$ do
6          if QoA (or QoE) of $u$ is satisfied then
7              $satisfiedUsers \leftarrow satisfiedUsers + 1$
8              $B_{avail} \leftarrow B_{avail} + demotion(u, 3Q_u^t, R)$
9          end
10         else
11             $3Q_u^{t+1} \leftarrow LCD(u, 3Q_u^t, R)$
12             if $B_{avail} > 0$ then
13                 $B_{avail} \leftarrow B_{avail} - promotion(u, 3Q_u^t, R, B_{avail})$
14             end
15         end
16         $\alpha^t_{QoA|QoE} \leftarrow |satisfiedUsers|/|U|$
17     end
18 end
```

VI. Performance evaluation

In this section, we establish the effectiveness of our cloud service orchestration algorithms for ASPs. We evaluate the performance of our PCM-based clustering scheme for the management plane and our LCD decision tree scheme for the data plane using both numerical simulations and event-driven experimental testbed evaluations (based on Skytap and LOFT use cases). Our evaluation results fall under three salient thrusts of findings: (i) *MCP-PCM clustering fits best for large-scale ASPs*; (ii) *LCD decision tree best improves the profiled QoE coverage (gains of up to 50%) with a low disruption and is adaptable to different scenarios due to 3Q model*; and (iii) *both our proposed MCP-PCM clustering and LCD decision tree enhance overall QoS and user QoE in real settings.*

A. Numerical simulations under demanding SLO requirements for service delivery

Simulation Settings. For our simulations, we used the *Matlab 2014r* environment (to cluster infrastructure within management plane) and the NS-3 [10] simulator (to adapt rented resources of a particular service within data plane based on QoE profile).[2] We use the BRITE [11] topology generator to create a realistic infrastructure topology that is rented from an InP. Our results are consistent across infrastructure network topologies that follow both Waxman and Barabasi-Albert models that possess salient properties of realistic Internet topologies [27]. Each graph has 100 vertices and 200 edges (the common edge density for Internet topologies [27]). Further, each edge has randomly distributed bandwidth capacity ranging between 1 and 9 Mbps (as a link SLO), arbitrary cost that is randomly distributed between 1 and 9, and propagation delay proportional to the edge length (as path SLOs).

For management plane performance evaluation, we assume that we have only one type of an application service whose SLOs are as follows: the bandwidth SLO $\geq$ 2.5 Mbps ($\approx 28\%$ of maximum corresponding link metric), the cost SLO $\leq$ 18 (200% of maximum corresponding link metric), and the delay SLO $\leq 75\%$ of maximum corresponding link metric. We intentionally choose demanding SLO requirements to stress test our clustering schemes at high application delivery scales. To stress clustering schemes under different InP routing schemes we use common inter-domain BGP routing, common intra-domain OSPF routing and MCP routing [26] policies to find paths with fully satisfied SLO. For data plane performance evaluation, we assume that we have the LOFT application service whose SLOs constraints are as follows: the bandwidth SLO $\geq$ 2 Mbps (for data type 1) and $\geq$ 1 Mbps (for data type 2), and the delay SLO constraint $\leq 100\%$ of maximum corresponding link metric. We use common OSPF routing as an InP routing scheme. In all simulations, we assume that InPs have a $k = 1$ routing policy, i.e., they use only a single shortest path for each source destination pair. For results clarity, we also assume that all provisioned servers have a unitary cost. All our results show 95% confidence intervals, and our randomness lays in the generated infrastructure topologies, as well as in the initialization setup.

1) MCP-PCM performance within management plane:

Comparison Methods and Metrics. To solve the ASP management optimization problem, we use the Geo-location infrastructure clustering based on geographical proximity; the K-Means infrastructure clustering based on hop count (or topological) proximity; our MCP-PCM infrastructure clustering is based on multidimensional (i.e., SLO) proximity; and finally the optimal integer programming solution (see Definition 1) with CPLEX [28]. We vary the number of rented servers (i.e., the ASP's budget), and while placing services, we calculate the following metric (the higher the better):

$$\text{SLO coverage } (\alpha_{SLO}) = \frac{\text{\# of users with satisfied SLOs}}{\text{total \# of users}} \quad (15)$$

where we use BGP, OSPF, or MCP [26] routing protocols to find paths. These paths are then used to connect users with their corresponding service instances.

(i) MCP-PCM clustering fits best for large-scale ASPs. Figure 5 shows SLO coverage (α_{SLO}) results for Geo-location, K-Means, MCP-PCM clustering schemes and for the optimal integer programming solution that are consistent across infrastructure graphs of both Waxman and Barabasi-Albert models. PCM does not show significant SLO coverage gains in comparison with other clustering schemes in two cases: (i) if an ASPs' budget is restricted for renting very limited amount of resources (e.g., 1 or 2 servers) from InPs, or (ii) the ASP budget is unrestricted for placing service instance replicas at all possible locations with the closest user proximity. In all other cases, PCM shows better SLO coverage performance than the compared clustering solutions to enhance satisfactory user QoE delivery subject to the ASP's budget. This is due to the fact, that our PCM clustering takes into account multiple path distances (that correspond to multiple path SLO constraints) in contrast to the compared graph clustering approaches that use a single distance metric. However, we found that PCM benefits degrade when InPs use less advanced routing protocols e.g., MCP versus OSPF or BGP. Due disregarding network design of InPs, the common Geo-location approach performs the worst in comparison with other clustering schemes.

Although the optimal solution with ILP achieves the best possible SLO coverage under ASP's budget constraint, we can conclude that our suboptimal but polynomial MCP-PCM clustering fits best for large-scale ASPs, as their service placement with NP-hard integer programming can take from several hours up to several days (or can be intractable at all) causing long outages that translates to the ASPs' revenue loss.

2) LCD decision tree performance within data plane:

Comparison Methods and Metrics. To estimate an impact of the online adaptations to be included in LCD decision tree (i.e., within data plane) on the (profiled) QoE coverage (α_{QoE}), disruption and (profiled) user QoE, we compare adaptations suitable for LOFT such as a common bit rate reduction on the client side (BR [15]); client migration to the second best server based on the MCP-PCM clustering results (CM [16]) which is equivalent to use the second best path to server, but does not require an infrastructure control; and a bit rate reduction on the server side (SMP [17]) to reduce network load within an infrastructure cluster (formed within management plane). In addition, we use a straightforward combination of all these adaptations that are applied sequentially (All). Finally, we apply all these adaptations based on our LCD

[2]The source code of both matlab and NS-3 simulations is available under GNU license at a public GitHub repository - https://github.com/duman190/qoe_orchestration_tomm.

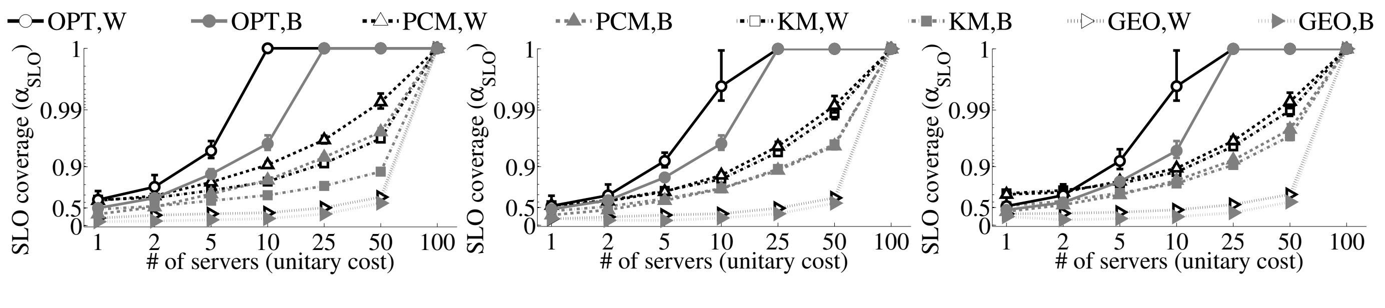


Fig. 5: SLO coverage (α_{SLO}) results of Geo-location (GEO), K-Means (KM), and MCP-PCM (PCM) clustering algorithms versus the optimal ASP management plane solution (OPT) (see Definition 1) using MCP (a), common intra-domain OSPF (b) and common inter-domain BGP (c) routing protocols on both Waxman (W) and Barabasi-Albert (B) infrastructure graphs: unless ASPs' budget is heavily restricted or unrestricted, PCM infrastructure clustering is the second best choice after optimal OPT solution to enhance satisfactory user QoE delivery subject to the ASP's budget. As expected, PCM benefits degrade when InPs use less advanced routing protocols e.g., MCP versus OSPF or BGP.

TABLE IV: LOFT QoE profile

Application Throughput [Mbps]	Estimated QoE	Application Throughput [Mbps]	Estimated QoE
Data Type 1		Data Type 2	
≥ 1.8	5	≥ 0.9	4
$1.8 > \cdots \geq 1.5$	4.5	$0.9 > \cdots \geq 0.75$	3.5
$1.5 > \cdots \geq 1.25$	4	$0.75 > \cdots \geq 0.5$	3
$1.25 > \cdots \geq 1$	3.5	$0.5 > \cdots \geq 0.25$	2.5
$1 > \cdots \geq 0$	2	$0.25 > \cdots \geq 0$	1

decision tree scheme using our greedy heuristic Algorithm 2 for ASPs' data plane (LCD). All adaptations are applied after solving the ASP management plane optimization problem (see Definition 1) with our greedy heuristic Algorithm 1 that uses MCP-PCM clustering. We also show baseline results with no adaptations (N/A).

To represent different operational time periods (e.g., during 24 hours), we use several workload scenarios based on the user access rate — from a sequential (or 1-by-1) to the full access where all clients accessing ASP services concurrently (e.g., during peak hours). Moreover, each server can serve only 25 clients simultaneously without tracking speed (QoA) degradation at the server side, and degrades gracefully after exceeding that limit. We also assume that all servers allow an elasticity policy, i.e., they allow elastic scale up (promote) and scale down (demote) their capacities. For example, promoting some server capacity by one unit (to serve 26 users without performance degradation) requires an additional cost of $1/25$. We then use this policy in our proposed Algorithm 2 to perform promotion/demotion adaptation strategies.

For each data plane service orchestration scheme and each workload scenario, we calculate the following metrics:

$$\text{QoE coverage } (\alpha_{QoE}) = \frac{\text{\# of satisfied users}}{\text{total \# of users}} \tag{16}$$

$$\text{Profiled QoE} = \frac{\sum_{i=1}^{N} \text{QoE profile}(throughput_i)}{\text{total \# of clients}} \tag{17}$$

$$\text{Disruption} = \frac{\text{\# of clients affected by any adaptation}}{(\text{total \# of clients}) \cdot (\text{total \# of adapts})} \tag{18}$$

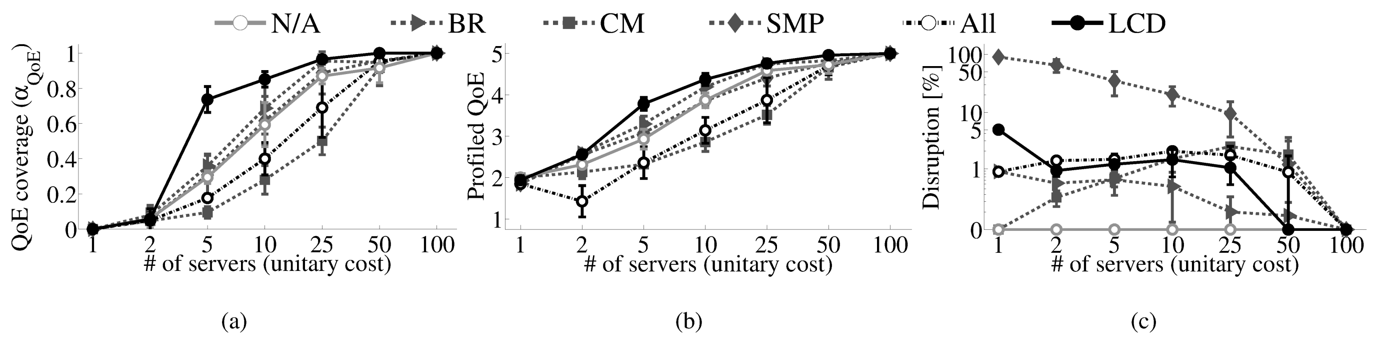
where we use derived from the realistic testbed setup the user QoE profile shown in Table IV of LOFT application (see details in Section VI-B) to determine user QoE. Note that the higher metric values in Equations 16 and 17 the better ASP data plane optimization, whereas the higher metric value in Equation 18 the worse such optimization performance.

(ii) LCD decision tree gains the highest QoE coverage with a low disruption. Figure 6 shows (profiled) QoE coverage (α_{QoE}), (profiled) average QoE and disruption results when ASP's data plane service orchestration has no adaptations *(N/A)*, only client-side bitrate reduction *(BR)*, only client migration *(CM)*, only server-side bitrate reduction *(SMP)*, all aforementioned adaptations used sequentially *(All)* and our LCD decision tree scheme (LCD) applied on Barabasi-Albert infrastructure graphs. Unless ASPs' budget is heavily restricted or unrestricted or the user access rate is low, applying adaptations within ASP's data plane service orchestration with using our LCD decision tree best improves the profiled QoE coverage (e.g., gains of up to 50%) at expense of a low user disruption level. Moreover, our service orchestration scheme is adaptable to the different scenarios and ASPs' budget due use of the 3Q interplay model (see Section III).

On the contrary, applying only a single adaptation (e.g., CM) can improve QoE coverage in one scenario (e.g., for 50% access rate), but can diminish it in another scenario (e.g., for full access rate). Observed poor performance of the network side adaptation with CM is due to superior clustering which places servers in a way to maximize probability of finding paths which fully satisfy SLO constraints. Thus, moving a client to the second best server (or the second best path) has lower probability of satisfying all SLOs and a high chance of other users disruption. The latter can be mitigated by renting larger number of resources or when user access rate is low. Another example is SMP that causes unnecessary user disruptions, e.g., it has the highest user disruption level among all adaptations and can be significantly mitigated when combined with other adaptations (i.e., included in LCD decision tree). We conclude that applying all adaptations sequentially (All) without relying on 3Q model is a risky proposition that can lead to a poor service orchestration or a high user disruption.

B. Skytap and LOFT Case Studies Experimental Evaluation

Experiment Settings. To analyze the impact of SLO violations on QoS/QoA and resulting user QoE within management and data planes, we recreated both our Skytap example of a virtual learning environment shown in Section IV-C and our LOFT example of an on-demand computer vision tracking system shown in Section V-C by allocating corresponding resources in a GENI Cloud (Global Environment for Network Innovations [13]) testbed (see Figures 7a and 7b, respectively).



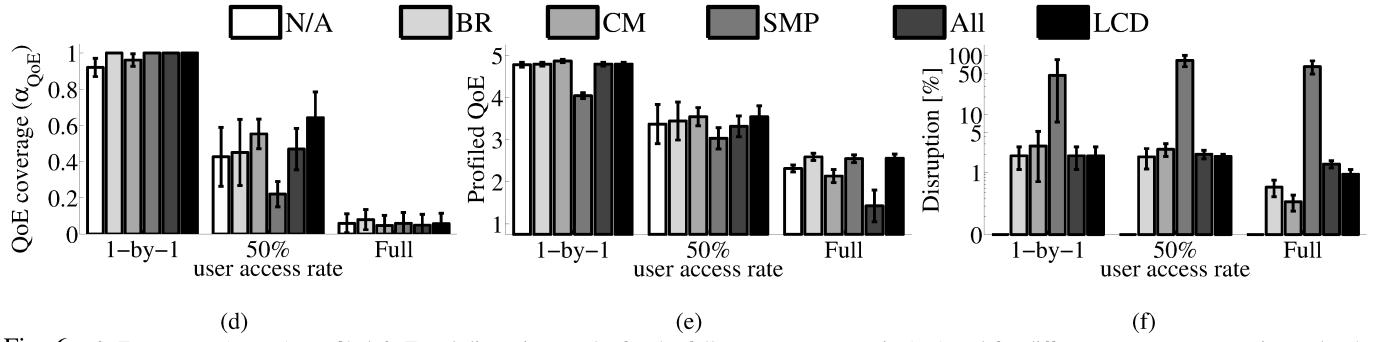


Fig. 6: QoE coverage (α_{QoE}), profiled QoE and disruption results for the full user access scenario (top) and for different access rate scenarios under the fixed ASP's budget = 2 (bottom); when ASP's data plane service orchestration has no adaptations *(N/A)*, only client side bitrate reduction *(BR)*, only client migration *(CM)*, only server side bitrate reduction *(SMP)*, all adaptations applied sequentially *(All)* and all adaptations applied within our LCD decision tree (*LCD*) on Barabasi-Albert infrastructure graphs: unless ASPs' budget is heavily restricted or unrestricted or the user access rate is low, applying adaptations within ASP's data plane service orchestration with using our LCD decision tree best improves the profiled QoE coverage with a low user disruption level.

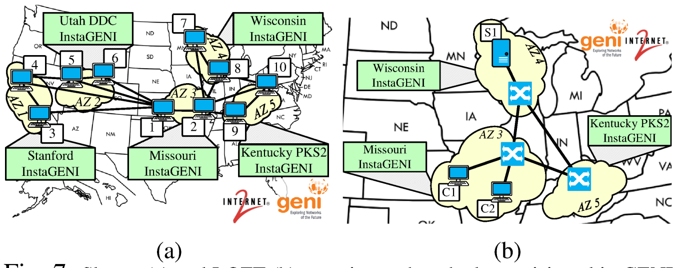


Fig. 7: Skytap (a) and LOFT (b) experimental testbeds provisioned in GENI Cloud: these testbeds matches our examples in Sections IV-C and V-C to estimate impact of SLO violations on overall QoS and user QoE.

We used the Stanford InstaGENI data center site as our availability zone (AZ) 1, the Utah Downtown Data Center InstaGENI site as AZ 2, the University of Missouri InstaGENI as AZ 3, the University of Wisconsin InstaGENI as AZ 4, and the University of Kentucky PKS2 InstaGENI as AZ 5. All link weights (bandwidth, one-way delay and packet loss) were allocated or adjusted (when needed) using a Linux-based *netem* emulator to match with corresponding links in the example.

To recreate Skytap example, all paths were bridged with a Linux *bridge-utils* tool. To recreate LOFT example, we run our server $S1$ in AZ 4, and we connect users via clients $C1$ and $C2$ at AZ 3. All AZs are connected via OpenFlow switches to steer the traffic (e.g., to route traffic in best-effort manner, or to redirect it through an alternative paths during network adaptation scenario). To control OpenFlow switches, we use OpenFlow Floodlight controller [29]. Finally, $S1$ has dual core CPU and 2GB of RAM for client and network side adaptation scenarios. For server side adaptation scenario $S1$ is promoted from single core CPU and 1GB of RAM to quad core CPU and 4 GB of RAM.

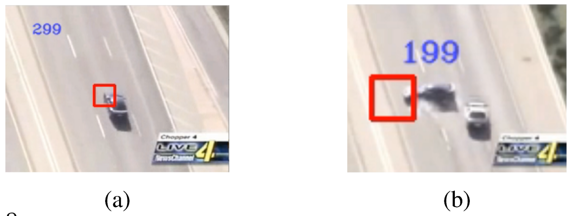


Fig. 8: Tiled images from a tracking dataset [30] were used: (a) the baseline data type 1 includes 240x180 resolution frames and provides best video quality and tracking accuracy (LOFT tracks all 300 frames) at the expense of tracking speed (≈2-3 fps), and (b) the baseline data type 2 includes 120x90 resolution frames providing best tracking speed (≈3-5 fps) at the expense of video quality and tracking accuracy (LOFT tracks only 200 frames) [7].

Comparison Methods and Metrics. To demonstrate the impact of SLOs adherence on the overall QoS and users' QoE within management plane, we use clustering results of *K-Means* and *PCM* schemes from our Skytap example (see Table II). We transfer a 100 MB file between each node and the corresponding cluster center by using the TCP protocol to measure the path Round-Trip Time (the lower the better) and throughput (the higher the better) for QoS estimation. We then stream a full-motion video[3] using UDP protocol to obtain user QoE assessments. To demonstrate the impact of SLOs adherence on the overall QoS and users' QoE within data plane, we use adaptation results from our LOFT example (see Table III). We transfer a 300 frames of either type 1 or type 2 baseline visual data (see Figure 8) by using TCP protocol for both LOFT QoS and user QoE estimation.

We use Linux *scp* tool for file transfer, and to stream the video (in Skytap example) we use the common *vlc* player. We

[3]This video was used previously to demonstrate benefits of Software-Defined Networking paradigm over regular Best-Effort Networking as part of a GENI QoE Lab exercise - https://youtu.be/f3FG_DkskyY.

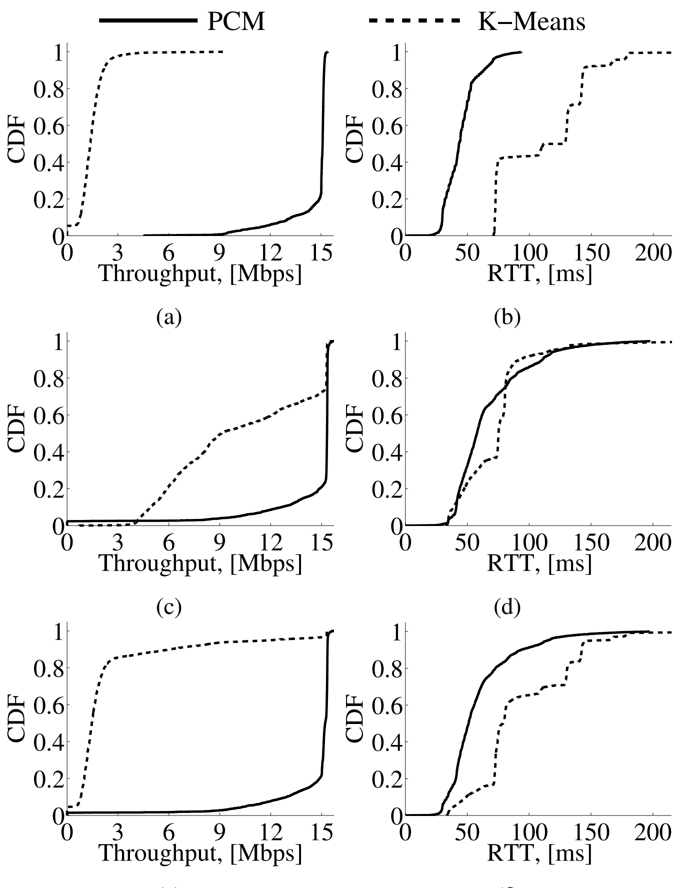


Fig. 9: Cumulative distribution function (CDF) of paths' (a,c,e) throughput and (b,d,f) round-trip time (RTT) within cluster 1 (first row), cluster 2 (second row) and both clusters (third row): in contrast to *K-Means* clustering where 6 of 8 nodes' paths violate SLOs (see Table II), after *PCM* clustering all nodes have a path to the corresponding cluster center with satisfied SLOs. That translates into higher overall QoS.

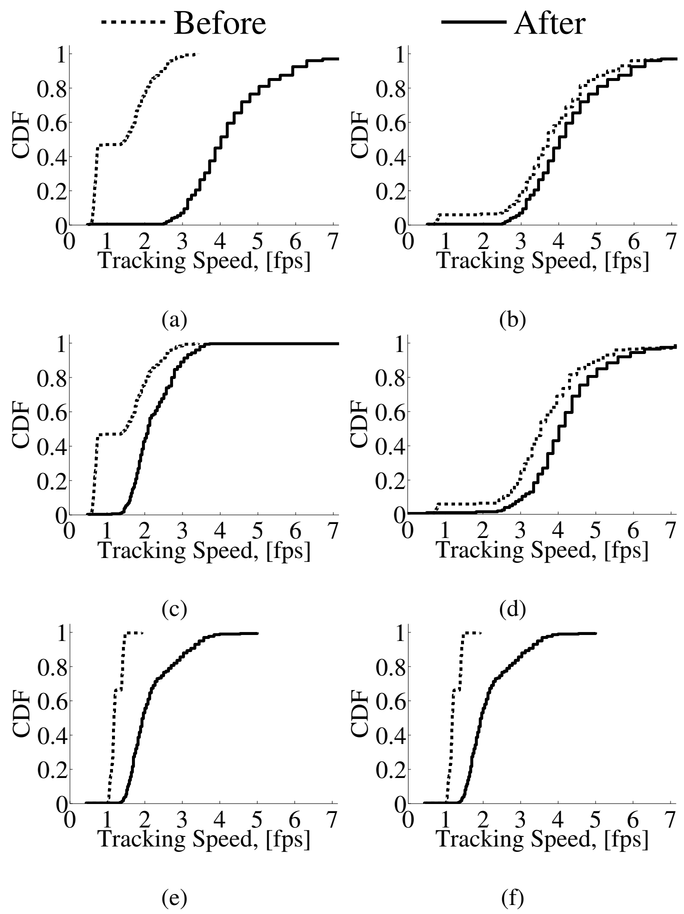


Fig. 10: Cumulative distribution function (CDF) of the LOFT tracking speed (LOFT QoS) for clients C_1 (a,c,e) and C_2 (b,d,f) during client (first row), network (second row) and server (third row) side adaptations: to improve QoS, data plane adaptations in LCD decision tree have to sacrifice the video quality and tracking accuracy, or disrupt other users, or require more resources to be rented from InPs.

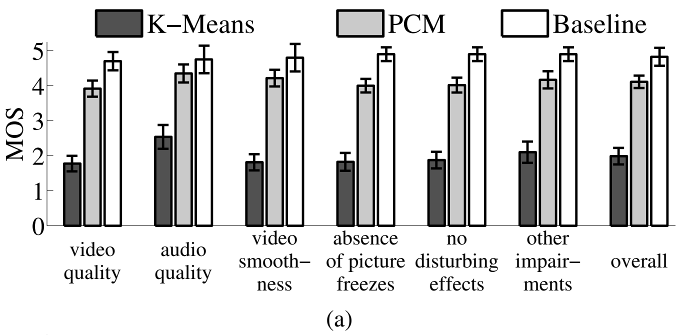
have recruited 20 human subjects[4] by following an approved Institutional Review Board (IRB) protocol that involves asking the users to provide mean opinion score (MOS) rankings on a 1 (Poor) - 5 (Excellent) scale to assess their subjective user QoE (the higher the better). We estimate user QoE results with 95% confidence intervals.

(iii) Proposed ASP service orchestration enhances overall QoS and user QoE. Figure 9 illustrates how *PCM* clustering with MCP awareness improves overall QoS within the management plane in comparison with the *K-Means* scheme. Upon clustering infrastructure to *PCM*, all nodes have a path to the corresponding cluster center which does not violate any SLOs, i.e., the one-way delay SLO constraint $\leq$ 50 ms ($\leq$ 100 ms of TCP RTT) and the bandwidth SLO (TCP throughput) $=$ 15 Mbps. In contrast to *PCM* clustering, *K-Means* clustering results in 6 of 8 nodes' paths violating at least one SLO. This in turn translates into poor overall QoS: the overall one-way delay SLO violation (RTT $>$ 100 ms) and the overall bandwidth SLO violation (80% of time TCP throughput $\leq$ 3 Mbps).

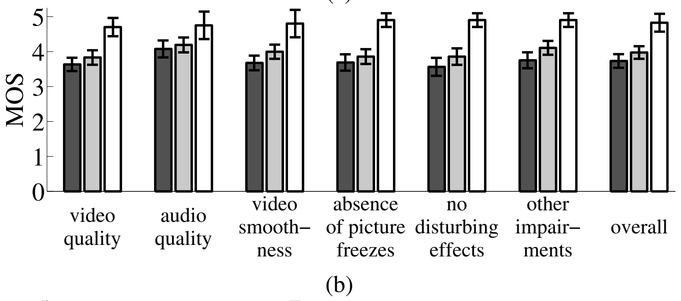
Figure 10 illustrates how the LCD decision tree scheme improves overall QoS within data plane after infrastructure clustering been made within the management plane. From the LOFT case study, a smaller image can result in faster computation time and lower bandwidth consumption. This explains why the tracking speed of $C1$ is increasing after the client-side adaptation. Similarly, finding an alternative (not congested) route allows to use more bandwidth which in turn again enhances LOFT tracking speed of $C1$. During server-side adaptation, we increase tracking speed on the server side which improves QoS of both $C1$ and $C2$. However, to improve QoS, data plane adaptations in LCD decision tree have to sacrifice the video quality and tracking accuracy, or disrupt other users, or require more resources to be rented from InPs. Thus, improvements in overall QoS do not necessarily translate to improvements in overall user QoE.

Figure 11 illustrates how we assessed user QoE of video streams within both clusters: In our example (see Section IV-C), *K-Means* clustering causes significant SLO violations within cluster 1, and insufficient SLO violations within cluster 2 (see Table II). As a result, we can see poor user QoE results within cluster 1 (low MOS for all 6 metrics) and acceptable user QoE results within cluster 2 (slightly lower video quality). In contrast to *K-Means* clustering, the *PCM* clustering results does not cause SLO violations in both clusters, which in turn results in satisfactory user QoE.

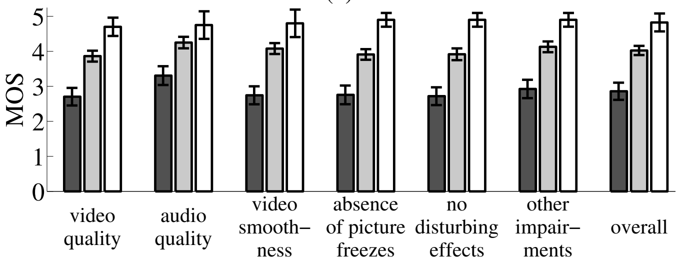
[4]The International Telecommunication Union recommends 4 human subjects for testing as a minimum total for statistical soundness [31].



(a)

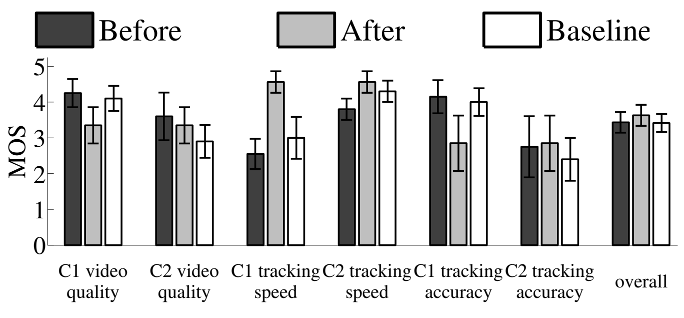


(b)

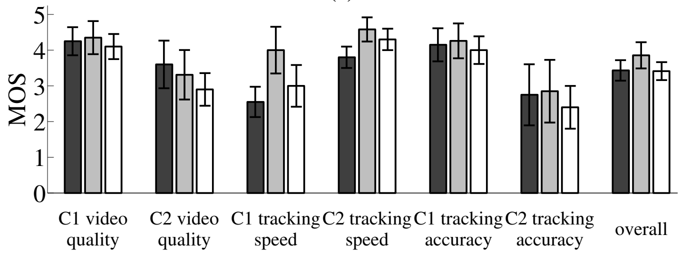


(c)

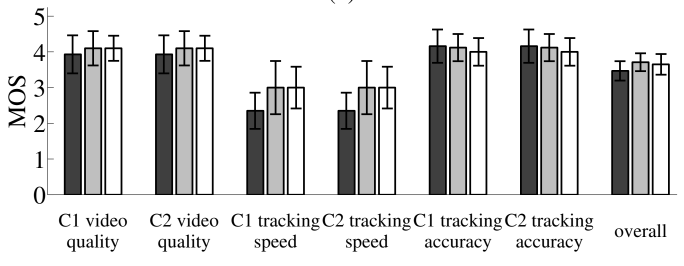
Fig. 11: User MOS assessments of video streams within: (a) cluster 1, (b) cluster 2 and (c) both clusters: *K-Means* clustering causes significant SLO violations within cluster 1 and insufficient SLO violations within cluster 2, whereas after *PCM* clustering there is no SLO violations in both clusters, which in turn results in satisfactory user QoE.



(a)



(b)



(c)

Fig. 12: User MOS assessments of remote LOFT tracking service for (a) client, (b) network and (c) server side adaptations: all online adaptations in LCD decision tree result in either an improved or same user QoE, while other network users also have no perceivable loss and in some cases can actually have improved QoE.

The results in Figure 12 illustrate how in every experiment, our data plane adaptations improve overall QoE. Although, data plane adaptations can disrupt other users (e.g., users connected at AZ 5 after $C1$ network adaptation, or all $S1$ clients during its promotion due to server-side adaptation), one of the most important results from our experiments is that as each of our LCD decision tree adaptations take place, QoE is either improved or remains stable and the other network users also have no perceivable loss and in some cases can actually experience improved QoE.

VII. Conclusions

In this paper, we addressed the lack of QoE-oriented service orchestration algorithms for Application Service Provider (ASP) business models that guide them on how to rent less resources from infrastructure providers (InPs), while delivering satisfactory user QoE. Such algorithms allow ASPs to offer higher QoE in service delivery under larger scale user loads in a cost-effective manner.

Specifically, we used integer linear programming to find an optimal solution to the ASP management plane problem of service placement and maximization of SLO coverage for users. To further avoid ILP NP-hardness, we also proposed a heuristic algorithm which is based on a novel Possibilistic C-Means (PCM) infrastructure clustering algorithm that is polynomial and can be used to optimize resource provisioning within a large-scale ASP's management plane. The novelty of our approach lies in its ability to minimize multiple distances related to service level objectives (SLOs) that demand close proximity of users to the service instances. Such a strategy in turn allows our PCM method to maximize the possibility of finding multi-constrained network paths (which satisfy all SLOs) between infrastructure nodes and corresponding cluster centers in order to deliver satisfactory user QoE. We also proposed a least cost disruptive decision tree (for client, network, or server side adaptations) to optimize the ASP data plane based on the dynamic QoS/QoA/QoE interplay model.

Using extensive numerical simulations under various SLO demands, NS-3 simulator and realistic Internet topologies, we have shown how our proposed ASP service orchestration algorithms outperform related solutions in terms of SLO (within the management plane) and QoA/QoE (within the data plane) coverage of users connected to the corresponding cluster centers (ASP's service instances). This in turn leads to ASPs needing to rent less resources from InPs in order to deliver satisfactory user QoE at a large-scale of application service delivery. Considering case studies featuring an actual Skytap virtual learning environment (e.g., for education organization customers) and a LOFT computer vision object tracking system (e.g., for public safety organization customers), we conducted GENI Cloud testbed experiments that show how our proposed ASP service orchestration algorithms improve overall QoS and enhances user QoE.

Future work will involve applying our ASP service orchestration to other applications such as depth sensor video applications for smart health and patient monitoring. Our goal is to consider other QoE-sensitive ASP use cases that involve large data handling, large-scale video delivery and intelligent resource adaptations to study optimization tradeoffs in the delivery of satisfactory QoE levels. Another interesting avenue to explore is to use machine learning techniques to better model the 3Q interplay during continuous periods of time (e.g., daily or weekly) which in turn can enhance a selection of adaptation strategies in order to improve ASP delivery of satisfactory QoE.

Acknowledgement

This material is based upon work supported by the National Science Foundation (award numbers: CNS-1205658 and CNS-1647182) and VMware. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or VMware.

References

[1] F. Esposito, et al., “Slice Embedding Solutions for Distributed Service Architectures”, *ACM Comp. Surv.*, 2013.
[2] R. Mijumbi, et al., “Network function virtualization: State-of-the-art and research challenges”, *IEEE Comm. Surv. & Tutorials*, 2016.
[3] T. Hoßfeld, et al., “Challenges of QoE management for cloud applications”, *IEEE Comm. Magazine*, 2012.
[4] S. Jain, et al., “B4: Experience with a globally-deployed software defined WAN”, *ACM SIGCOMM*, 2013.
[5] P. Calyam, et al., “Resource defragmentation using market-driven allocation in virtual desktop clouds”, *IEEE IC2E*, 2015.
[6] Skytap Virtual Training Labs - https://www.skytap.com/solutions/virtual-training-labs; Last accessed Aug. 2018.
[7] R. Pelapur, et al.. “Persistent target tracking using likelihood fusion in wide-area and full motion video sequences”, *IEEE FUSION*, 2012.
[8] J. Zabczyk, “Mathematical control theory: an introduction”, *Springer Science & Business Media*, 2009.
[9] O. Dobrijevic, et al., “Ant colony optimization for QoE-centric flow routing in software-defined networks”, *CNSM*, 2015.
[10] T. R. Henderson, et al., ”Network simulations with the NS-3 simulator”, *ACM SIGCOMM Demo*, 2008.
[11] A. Medina, et al., “BRITE: An Approach to Universal Topology Generation”, *IEEE MASCOTS*, 2001.
[12] S.E. Schaeffer, “Graph clustering”, *Elsevier CSR*, 2007.
[13] M. Berman, et al., “GENI: A Federated Testbed for Innovative Network Experiments”, *Elsevier COMNET*, 2014.
[14] P. Georgopoulos, et al., “Towards network-wide qoe fairness using openflow-assisted adaptive video streaming”, *ACM FhMN*, 2013.
[15] J. Park, K. Chung, “Client-side Rate Adaptation Scheme for HTTP Adaptive Streaming Based on Playout Buffer Model”, *IEEE ICOIN*, 2016.
[16] Zheng Ma, et al., “A New Multi-path Selection Scheme for Video Streaming on Overlay Networks”, *IEEE Comm. Soc.*, 2004.
[17] P. Van Beek, et al., “Server-Side Playout Delay Management for Video Streaming”, *IEEE ICIP*, 2006.
[18] K. Xiong, H. Perros, “Service performance and analysis in cloud computing”, *IEEE SERVICES*, 2009.
[19] W. Iqbal, et al., “Adaptive resource provisioning for read intensive multi-tier applications in the cloud”, *Elsevier FGCS*, 2011.
[20] P. Calyam, et al., “Incident-Supporting Visual Cloud Computing Utilizing Software-Defined Networking”, *IEEE TCSVT*, 2016.
[21] N. Garg, J. Koenemann, “Faster and simpler algorithms for multicommodity flow and other fractional packing problems”, *SICOMP)*, 2007.
[22] K. M. Bretthauer, B. Shetty, “The nonlinear knapsack problem - algorithms and applications”, *Elsevier EJOR*, 2002.
[23] T. Nguyen, et al., ”Benchmarking in virtual desktops for end-to-end performance traceability”, *IEEE QCMan*, 2015.
[24] V. Menkovski, et al., ”Predicting Quality of Experience in Multimedia Streaming”, *ACM MoMM*, 2009.
[25] R. Krishnapuram, J. M. Keller, “The Possibilistic C-Means Algorithm: Insights and Recommendations”, *IEEE TFS*, 1996.
[26] P. Khadivi, et al., “Multi-constraint QoS routing using a new single mixed metrics”, *Elsevier JNCA*, 2008.
[27] M. Faloutsos, et al., “On Power-Law Relationships of the Internet Topology”, *ACM SIGCOMM*, 1999.
[28] IBM CPLEX - http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/index.html; Last accessed Aug. 2018.
[29] Floodlight SDN controller - http://www.projectfloodlight.org; Last accessed Aug. 2018.
[30] K. Zhang, et al., “Real-time Compressive Tracking”. *ECCV*, 2012.
[31] S. Mohamed, G. Rubino, “A Study of Real-Time Packet Video Quality using Random Neural Networks”, *IEEE TCSVT*, 2002.

Dmitrii Chemodanov received his MS degree from the Department of Computer Science at Samara State Aerospace University, Russia in 2014. He is currently a PhD student in the Department of Computer Science at University of Missouri-Columbia. His current research interests include distributed and cloud computing, network and service management, and peer-to-peer networks.

Prasad Calyam received his MS and PhD degrees from the Department of Electrical and Computer Engineering at The Ohio State University in 2002 and 2007, respectively. He is currently an Assistant Professor in the Department of Computer Science at University of Missouri-Columbia. His current research interests include distributed and cloud computing, computer networking, and cyber security. He is a Senior Member of IEEE.

Samaikya Valluripally received her M.S. degree in Computer Science from University of Missouri-Columbia in 2016, Bachelor of Technology degree in Computer Science from Jawaharlal Nehru Technological University, India in 2014. She is currently pursuing her Ph.D. degree in Computer Science at University of Missouri-Columbia. Her current research interests include Cloud Computing, Big Data Analytics, Cloud Security.

Huy Trinh received the B.S. degree in Computer Science at University of Missouri, Columbia, USA, in 2015. He is currently a Graduate Research Assistant working toward the M.S degree in the Department of Computer Science, University of Missouri, Columbia, USA. His research interests include image processing, cloud computing and networks.

Jon Patman received his dual-BS degree in Electronics Engineering and Computer Science from Eastern New Mexico University in 2016. He is currently working towards his M.S. as a Graduate Research Assistant for the Electrical Engineering and Computer Science Department at the University of Missouri. His current research focus involves designing intelligent computing networks by synthesizing machine learning, artificial intelligence, cloud computing, and computer vision.

Kannappan Palaniappan received his PhD from the University of Illinois at Urbana-Champaign, and MS and BS degrees in Systems Design Engineering from the University of Waterloo, Canada. He is a faculty member in Computer Science at the University of Missouri where he directs the Computational Imaging and VisAnalysis Lab and helped establish the NASA Center of Excellence in Remote Sensing. At NASA Goddard Space Flight Center he co-invented the Interactive Image SpreadSheet for visualizing large multispectral imagery and deformable cloud motion analysis. His research is at the synergistic intersection of image and video big data, computer vision, high performance computing and artificial intelligence to understand, quantify and model physical processes with applications to biomedical, space and defense imaging. Recent multidisciplinary contributions range across orders of scale from sub-cellular microscopy at the molecular level to aerial and satellite remote sensing imaging at the macro level.