Converged Network-Cloud Service Composition with End-to-End Performance Guarantee

Jun Huang, *Member, IEEE*, Qiang Duan, *Member, IEEE*, Song Guo, *Senior Member, IEEE*, Yuhong Yan, *Member, IEEE*, and Shui Yu, *Senior Member, IEEE*

Abstract—The crucial role of networking in cloud computing calls for federated management of both computing and networking resources for end-to-end service provisioning. Application of the Service-Oriented Architecture (SOA) in both cloud computing and networking enables a convergence of network and cloud service provisioning. One of the key challenges to high performance converged network-cloud service provisioning lies in composition of network and cloud services with end-to-end performance guarantee. In this paper, we propose a QoS-aware service composition approach to tackling this challenging issue. We first present a system model for network-cloud service composition and formulate the service composition problem as a variant of Multi-Constrained Optimal Path (MCOP) problem. We then propose an approximation algorithm to solve the problem and give theoretical analysis on properties of the algorithm to show its effectiveness and efficiency for QoS-aware network-cloud service composition. Performance of the proposed algorithm is evaluated through extensive experiments and the obtained results indicate that the proposed method achieves better performance in service composition than the best current MCOP approaches.

Index Terms—Cloud computing, service-oriented architecture (SOA), network-as-a-service (NaaS), service composition, quality-of-service (QoS)

1 Introduction

Cloud computing has emerged as a novel distributed computing paradigm in which a pool of abstracted, virtualized, dynamically scalable computing functions and services are delivered on demand to external customers over the Internet [1]. In the cloud environment, cloud services normally represent remote delivery of computing resources and users obtain services from a cloud provider via networks (most often the Internet). Services received by end users consist of not only computing functions provided by cloud infrastructure but also communication functions offered by networks [2]. Recent research results have indicated that networking performance has a significant impact on cloud service quality [3], [4], [5]. The end-to-end performance evaluation between network and cloud systems conducted in [6] showed that networks may form a bottleneck that limits cloud service performance. In addition research reported in [7] found that networking system is an important element for improving cloud service reliability.

With increased differentiation in cloud providers' offerings, allocation of application components on multiple public clouds become an attractive option. Research reported in [8] shows that such allocation provides significant cost savings in various cloud use scenarios and also indicates that the delay and cost introduced by inter-cloud communications have a strong impact on application performance. Recent development in cloud federation requires high-performance inter-cloud communications [9]. Hybrid clouds that compose resources in both private and public cloud infrastructures are also widely adopted for cloud service provisioning. In such a scenario, communication demands and available bandwidth on the network links between private and public cloud infrastructures must be considered for scheduling components of workflows onto available resources [10].

Therefore, networking plays a crucial role in cloud computing and becomes an indispensable ingredient for single, hybrid, and inter-cloud service provisioning [11], [12], [13], [14]. The important role that networking plays in cloud computing calls for federated control and management of both computing and networking resources in order to achieve high performance end-to-end service provisioning. To illustrate the impact of networking on cloud service performance, let us consider an example scenario in which a research lab creates 100 GB of raw data that will be processed in Amazon EC2 cloud [15]. Suppose that the lab obtains 10 EC2 virtual machine instances and each instance can process 20 GB data per hour. Then the total processing time of the cloud service is only 30 minutes. However, if the lab uses a network service that offers 200 Mb/s throughput for data transmission to the EC2 server, then just the single-

- *J. Huang is with the Institute of Electronic Information and Networking, Chongqing University of Posts and Telecommunications, Chongqing 400065, China. E-mail: xiaoniuadmin@gmail.com.*
- *Q. Duan is with the Information Science and Technology Department, The Pennsylvania State University, Abington, PA 19001. E-mail: qduan@psu.edu.*
- *S. Guo is with the School of Computer Science and Engineering, The University of Aizu, Tsuruga, Ikki-machi Aizu-Wakamatsu City, Fukushima 965-8580, Japan. E-mail: sguo@u-aizu.ac.jp.*
- *Y. Yan is with the Department of Computer Science and Software Engineering, Concordia University, Montreal, QC H3G 1M8, Canada. E-mail: yuhong@encs.concordia.ca.*
- *S. Yu is with the School of Information Technology, Deakin University, Burwood, Vic. 3125, Australia. E-mail: syu@deakin.edu.au.*

Manuscript received 11 May 2015; revised 7 Aug. 2015; accepted 7 Oct. 2015. Date of publication 0 . 0000; date of current version 0 . 0000. Recommended for acceptance by A. Walid. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TCC.2015.2491939

trip delay for data transmission from the lab to EC2 servers will be more than an hour. In this case network delay for round-trip data transmission contributes more than 80 percent of the total service delay. This example shows that end-to-end services provided to cloud users are typically combination of both network services and cloud services; therefore network QoS often contributes a significant fraction of cloud service performance.

Until recent there exists a gap between the demands of cloud computing for data communications and the services that can be offered by traditional networking systems. High-performance cloud computing requires predictability in networking performance, coordination of both computing and networking resources, and application-driven network control and management [16], [17]. However, traditional networks are designed specifically to support a narrow range of precisely defined communication services, which are implemented on fairly rigid infrastructure with minimal capabilities for ad hoc reconfiguration. Recent advances in networking technologies, especially network virtualization, Software Defined Network (SDN), and application of the Service-Oriented Architecture in networking, has significantly enhanced network control and management for supporting cloud service provisioning [18].

Network virtualization, which decouples network service provisioning from data transport infrastructure, is expected to be a key attribute of future networking [19]. The emerging Network Function Virtualization (NFV) technology allows network functions to be virtualized and implemented on general purpose data center hardware [20]. In the NFV architectural framework recently published by ETSI Industry Specification Group (ISG) [21] infrastructure resources of computing, storage, and network are all abstracted through a uniform virtualization layer. The virtual computing, storage, and network resources then are managed by a common orchestrator for end-to-end service provisioning. As a potential enabler of profound changes in both computing and networking, virtualization may bridge the gap between these two fields; thus supporting federated management of computing and networking resources for composite service provisioning.

The SOA, which provides an effective architectural principle for heterogeneous system integration, has been widely adopted in cloud service provisioning via the paradigms of Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS). Applying the SOA principle in network virtualization enables a *Network-as-a-Service* paradigm in which network resources are exposed and utilized as SOA-compliant services. The NaaS paradigm enables networking functionalities to be utilized on-demand by end users through standard service interfaces; much like computing capacities in clouds being accessed as services by users. Therefore, virtualization combined with service-orientation, is making transforming tradition network service provisioning toward a cloud IaaS service model. Such transform enables network services to be composed with computing services in a cloud environment; thus allowing a convergence of network-cloud service provisioning [22].

A key challenge to realizing notion of converged network-cloud service provisioning lies in QoS-aware service composition across the networking and computing domains. The objective of network-cloud service composition is to select an appropriate sequence of network and cloud services to meet end-to-end service performance requirements while optimizing networking and computing resource utilization. For example in the above lab data processing case, if the user requires a maximum service delay (waiting time from sending data out to receiving results back), then the services for storage, computing, and networking must be selected with a holistic vision to guarantee the end-to-end service delay. A composite service for this example is illustrated in Fig. 1. In this figure cloud services 1 and 2 are respectively Amazon S3 for storage and EC2 for computing. Network services 1, 2, and 3 respectively provide data communications from the user to cloud service 1, between cloud services 1 and 2, and from cloud service 2 back to the user. Selection of these cloud and network services must consider their delay performance in order to guarantee that the end-to-end data transmission, storage, and processing delay meets user's requirement.

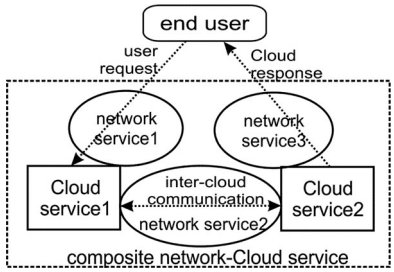


Fig. 1. An example of composite network-cloud service provisioning.

In a general sense, service composition can be formulated as the Multi-Constrained Optimal Path (MCOP) problem, which is known to be NP-hard. Although service composition has been extensively studied, most of current approaches were mainly focused on web services or cloud services instead of composite network-cloud services; therefore may not achieve end-to-end QoS guarantee across networking and computing domains. Due to the real-time requirements of various network protocols, network-cloud services typically need much faster composition algorithms to achieve shorter response time. In contrast, web/cloud service composition, though might have QoS requirements on composite services, typically allows best-effort response time of composition algorithms. Therefore, service composition across network and cloud domains particularly requires more efficient and effective MCOP algorithms that can not only meet diverse QoS constraints but also achieve much shorter response time.

Toward addressing this challenging issue, we investigate QoS-aware service composition in the converged network-cloud context from a general service provisioning perspective. The main contributions made in this paper are summarized as follows.

- We present a model for QoS-aware service composition for converged network-cloud service provisioning. This model allows multiple QoS metrics to be considered in network-cloud services composition with guaranteed end-to-end service performance.

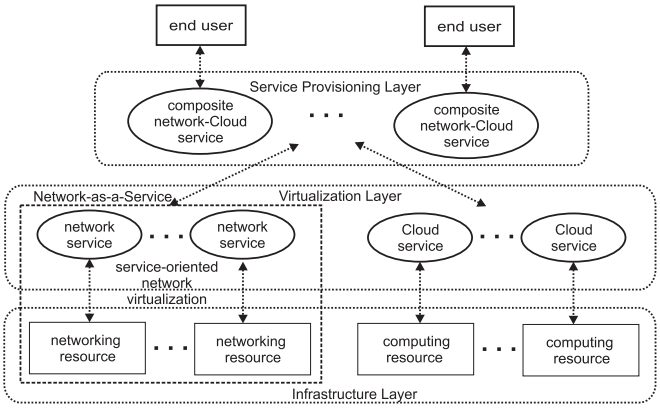


Fig. 2. The layered structure of a service-oriented framework for network-cloud service convergence.

- We formulate the problem of QoS-aware network-cloud service composition as a variant of MCOP problem and develop an approximation algorithm to solve the problem.
- We analyze the theoretical properties of the proposed algorithm. Analytical results show that our algorithm is effective and resilient in selecting services for network-cloud service composition and guarantees end-to-end performance for composite service provisioning.
- We conduct thorough experiments to evaluate performance of the proposed algorithm and compare the algorithm against variations of current best-known MCOP algorithms. Obtained numerical results indicate that the proposed algorithm is efficient and accurate for service composition.

The rest of this paper is organized as follows. In Section 2 we first introduce a service-oriented framework for network-cloud service convergence, and then present a model for network-cloud service composition. We also formulate the QoS-aware service composition problem in this section. Section 3 presents an approximation algorithm to solve the service composition problem and gives theoretical analysis on the proposed algorithm. In Section 4 we evaluate performance of the proposed algorithm and compare it with currently available algorithms through numerical experiments. Section 5 briefly reviews the related research progress on service composition and compares our research with related work. Section 6 draws conclusions.

2 Modeling QoS-Aware Network-Cloud Service Composition

2.1 A Framework for Network-Cloud Service Convergence

Network Function virtualization leverages some key technologies developed for cloud computing to deliver network services upon data transportation infrastructure. Such key technologies include virtualization, which abstracts and partitions computing as well as networking infrastructure by means of hypervisors, and orchestration mechanism that coordinates virtualized resources for service provisioning to meet application requirements. SOA has been widely adopted as the main model for cloud service provisioning, in which various virtualized computing resources and software applications can be delivered as services to users. Applying the SOA principle in networking allows virtualization of network resources and functionalities in the form of SOA-compliant services that can be composed with computing services in a cloud environment.

A service-oriented framework for network-cloud service convergence is illustrated in Fig. 2. At the bottom of this framework is a *resource layer* consisting of physical infrastructure for both networking and computing. Above the resource layer is the *virtualization layer* in which both networking and computing resources are virtualized and abstracted as SOA-complaint services. The virtualization layer offers an abstraction of the underlying networking and computing resources through a standard service interface, which allows cloud service provisioning to be realized across heterogeneous infrastructure domains including both networking and computing systems. Such an abstraction also decouples underlying infrastructure from upper layer service provisioning functions; thus enabling more flexible service development and provisioning. The *service provisioning layer* above the virtualization layer discovers and selects both network and cloud services and orchestrates them into composite network-cloud services that meet the requirements of end users. In this framework the SOA-based NaaS paradigm enables networking resources to be virtualized and exposed as services and composed with cloud computing resources into composite network-cloud services. This framework offers a uniform mechanism to access both networking and computing resources; thus significantly facilitating federated management across network and cloud domains for end-to-end service provisioning.

Again take the lab data processing scenario as an example, this convergence framework enables networking systems of various providers, such as Verizon, AT&T, and Comcast, to be virtualized and offered as SOA-compliant network services through the NaaS paradigm, just like computing and storage resources in cloud to be offered through the IaaS paradigm (e.g. Amazon EC2 and S3 services). Then appropriate network services as well as cloud services are selected and composed for meeting the performance requirement for lab data processing.

The above SOA-based framework of network-cloud service composition enables a much wider range of communication services with more attributes than what can be offered by traditional networking technologies. Networking resources, virtualized and encapsulated in SOA-compliant services, may be combined in almost limitless ways with other service components that abstract both computational and networking resources; thus greatly expanding the services that can be offered by co mposite network-cloud systems. NaaS also offers the ability to match cloud requirements to communication functions through discovering and selecting the appropriate network services and composing them with cloud services.

2.2 System Model

A typical end-to-end provisioning system for composite network-cloud services consists of both computing infrastructure that offers cloud services and network infrastructure that provides services for cloud access and inter-cloud

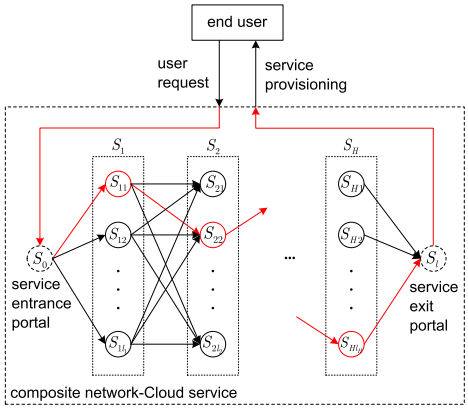


Fig. 3. Modeling for network-cloud service composition.

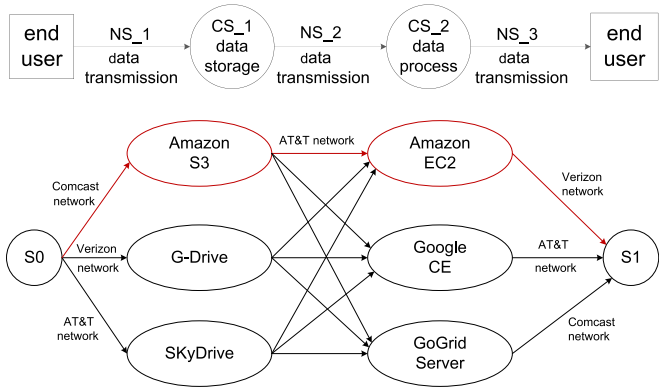


Fig. 4. An example of service composition for network-cloud service provisioning.

communications. In order to achieve service performance guarantee to an end user, the end-to-end service system must expect a certain level QoS from each cloud service and network service involved in service provisioning. In general, such QoS expectation can be specified in form of Service Level Agreements (SLAs) between the end-to-end service provisioning system and the providers of cloud and network services.

When a user submits a service request to the service provisioning system, it will specify information of the following two aspects: a) functional requirements including required functionalities and the inter-relationship between the functionalities; b) performance requirements for example the minimum service capacity and the maximum service delay. Functional requirement can be specified in form of a *workflow* that gives a set of service component (each has certain functionality) and the interconnections between the service components. In a converged networking and cloud computing environment, the functionality of each service component may be realized by multiple available services, referred to as candidate services for the service component in this paper, which may offer different level of performance (e.g. latency and capacity). Therefore, for each service request, a service broker will invoke a composition process to select a series of available service candidates to form an end-to-end composite service that can meet the performance requirements specified by the user.

Fig. 3 shows a model for network-cloud service composition. The end-to-end service has H service components, S_i, $i = 1, 2, \ldots, H$, where the number and sequence of service components are decided by the user's request. Each service component S_i has l_i candidate services S_{ij}, $j = 1, 2, \ldots, l_i$; that is, the functions of component S_i may be realized by any candidate service S_{ij}. Available candidate services for a service component are pre-selected based on the functional requirements for the component specified in the user request and service descriptions published by available network and cloud services. A service component in the model could either perform computing functions for data process and/or storage or networking functions for data communications. As illustrated in Fig. 3, the candidate services of a service component are connected to the candidate services for the next-stage service component through links, which represent data transmissions provided by network services. Then all the candidate services together with the data transmissions between them form a graph. Each path traversing through this graph from S_0 to S_l represents one possible composition of candidate services. In this paper, we refer such a path a *service path*, which represents a particular way for realizing the workflow specified by the end user.

This model allows network-cloud service composition to take QoS requirements into account naturally. Performance metrics can be assigned to each service candidate and to the links connecting service candidates as well. Then the metric of a path traversing a set of candidate services represents the QoS attributes of a composite service comprising the set of candidate services. For example in the model shown in Fig. 3 multiple paths exist from S_0 to S_l, which represents different realization of the composite end-to-end service consisting components $S_1, S_2, \ldots, S_H$. If we assign delay as the metric to each candidate service, then the path with the minimum total metric value, say the path $S_0 \rightarrow S_{11} \rightarrow S_{22} \rightarrow \cdots \rightarrow S_{Hl_H} \rightarrow S_l$, will give a composite service that achieves the optimal delay performance. Therefore, selecting an optimal service path from a pool of network and cloud candidate services while satisfying various QoS requirements is the main optimization focus for converged network-cloud service provisioning.

In this model we assume that service providers always collaborate with each other; that is, there is no contention among all network and cloud services. We also assume a standard interface between the service broker and the available atomic services for service orchestration. Such an interface offers service abstraction that makes implementation details of these services transparent to the service broker and service consumers; thus overcoming "lock-in" issue of cloud computing.

As an illustrative example we still consider the lab data processing case. Suppose the end-to-end service requested by the user include cloud data storage, cloud data process, as well as data transmission from user to storage, from storage to the process server, and from the server back to the user. The path of the end-to-end composite service is shown in the upper part of Fig. 4. This path consists of five service components. Two cloud service CS_1 and CS_2 for data storage and data processing respectively. Network service NS_1 transmits raw data to cloud storage, NS_2 transports data from storage to cloud server for processing, then NS_3 transmits process results back to the user.

There are multiple candidate services that may be used to realize each service component in the path, as illustrated in

the lower part of Fig. 4. For example in this case Amazon S3, Google Drive, and Microsoft Skydrive are all available for cloud storage service. Similarly Amazon EC2, GoGrid Server, and Google Compute Engine are candidate services for cloud data process. Available service candidates for realizing network service components ($NS_i, i = 1, 2, 3$) are represented by the connection links in the lower part of Fig. 4. In this example network services provided by AT&T network, Verizon network, and Comcast network are available candidate services that can be selected to realize $NS_i, i = 1, 2, 3$.

Each path from S_0 to S_l in the graph shown in the lower part of Fig. 4 gives one possible realization; i.e., a workflow, for the composite service. In order to achieve end-to-end QoS provisioning, a service path that meets the QoS requirements specified by the user must be chosen. If we assign metrics to each node (e.g. capacity and latency for cloud storage and process) and each link (e.g., bandwidth and delay for data transmission) in the graph, then a composition algorithm may select an optimal path for meeting the requirements. For example the user requires a maximum service delay for getting data process results back and then the highlighted path is selected for meeting such a delay requirement. The selected service path uses Comcast network for transmitting data to Amazon S3, using AT&T network for communications between Amazon S3 and Amazon EC2, then using Verizon network for transmitting process results back to the user. The total delay on the service path, including latency introduced by Amazon S3 and EC2 and transmission delay in Comcast, AT&T, and Verizon networks, is less than the required end-to-end service delay. Therefore, the QoS-aware composition mechanism should support federated service composition across network and cloud domains in order to achieve optimal end-to-end service performance.

2.3 Problem Formulation

Given the converged service provisioning model, we can formulate the problem of QoS-aware network-cloud service composition as follows

A service graph formed by interconnecting candidate services of a series of service components can be modeled as a directed graph $G(V, E)$ with n vertices and m edges. Each vertex represents a cloud service that is associated with two QoS metrics: capacity c and processing latency d. Each edge represent a network link for data transmission, which is associated with K QoS parameters $w = (w_1, w_2, \ldots, w_K)$. Note that QoS parameters can be either *positive* (quality increases as parameter value increases, e.g., reliability) or *negative* (quality decreases as parameter value increases, e.g., delay), the latter type are considered in the model because positive ones can be transformed to be negative by taking reciprocal. In addition, QoS parameters on each link are supposed to be *additive* since any *concave* metric (e.g. bandwidth) can be pre-processed via eliminating the corresponding vertices and edges that do not satisfy the QoS constraints [23]. That is, we focus on the negative and additive QoS parameters in the model.

Let $R = (C, D, W_1, W_2, \ldots, W_K)$ denote the performance requirement specified in a user's request, where C is the service capacity constraint, D is the processing latency constraint, and $(W_1, \ldots, W_K)$ are K end-to-end QoS constraints.

TABLE 1
Notations and Symbols

n	number of nodes in $G(V, E)$
m	number of links in $G(V, E)$
c	capacity of node
d	processing latency of node
K	number of QoS constraints on each link
H	number of services
ϵ	approximation ratio of the algorithm
$w = (w_1, w_2, \cdots, w_K)$	QoS parameters on each link
$R = (C, D, W_1, W_2, \cdots, W_K)$	user's request
C	capacity constraint
D	processing latency constraint
$W_1, W_2, \cdots, W_K$	K QoS constraints

A series of selected services can be represented as a path p in the service graph.

Definition 1. *Feasible Service Composition. A composed service, i.e., a path p from service entrance portal to service exit portal in the service graph, is said to be feasible if $\forall v \in p$, $\frac{1}{c(v)} \geq \frac{1}{C}, \sum_{v \in p} d(v) \leq D, w_k(p) \leq W_k$ for all $1 \leq k \leq K$.*

Denote $\{p^f\}$ as all feasible service compositions in $G(V, E)$. For each $p_i^f \in \{p^f\}$, there exists a smallest value $\eta_i \in (0, 1]$ such that $w_k(p_i^f) \leq \eta_i \cdot W_k, 1 \leq k \leq K$.

By the above definition, we now formulate the QoS-aware service composition problem.

Problem 1. *QoS-aware Service Composition (QSC). QSC is to find an optimal path p^{opt} among feasible service compositions in $G(V, E)$ and the corresponding smallest value η^{opt} among all η_i such that $w_k(p^{opt}) \leq \eta^{opt} \cdot W_k, k \in [1, K]$ where $K \geq 2$.*

It has been proved that problem QSC is NP-hard [24] as the solution of QSC in the absence of vertex constraints, namely, capacity and processing latency, maps directly that of MCOP, which is a well-known NP-hard problem. To attack it, approximation algorithm has been realized as a powerful tool. Formally, an algorithm is a β-approximation algorithm (or simply, an approximation algorithm) for QSC if the algorithm generates a service path p such that $w_k(p) \leq \beta \cdot \eta^{opt} \cdot W_k$, and the running time of the algorithm is bounded by a polynomial in the input size of the instance as well as in $1/\beta$.

Table 1 lists the frequently used notations in the rest of this paper.

Please note that although the concept of a service path p used in *Definition 1* and *Problem 1* can represent not only a series of candidate services invoked in a sequential order, but also atomic services executed with other flow structures including parallel, case, condition, and loop as shown in Fig. 5. From a perspective of QoS-aware service composition, a service path with these flow structures may be converted to a sequential path based on the following lemma.

Lemma 1. *Computation of the service capacity, processing latency, and end-to-end QoS constraints for service paths in parallel, case, condition, and loop flow structures can be transformed to computation of the corresponding constraint parameters for a service path in the sequential structure.*

Proof. Assume $\{S_1, S_2, \ldots, S_h\}$ is a set of candidate services, they are executed in parallel, case, condition, and loop

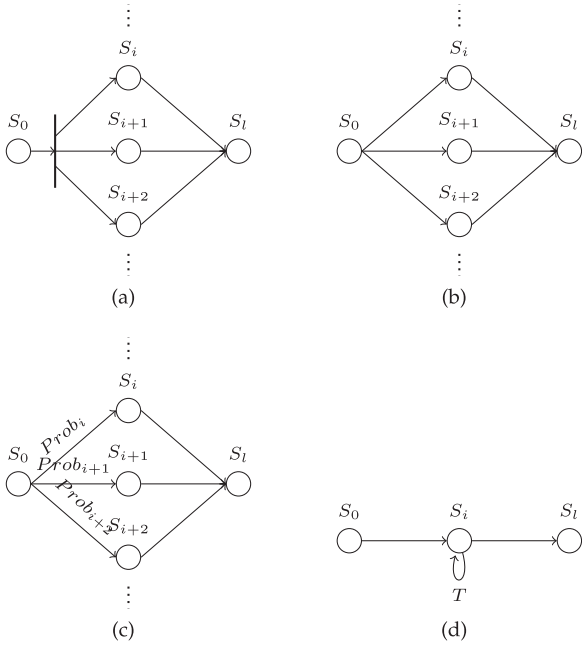


Fig. 5. Parallel, case, condition, loop execution structures of atomic candidate services. S_0 and S_l are service entrance portal and service exit portal, respectively.

flow structures, which are shown in Figs. 5a, 5b, 5c, and 5d, respectively. Without loss of generality, consider each node is associated with a capacity (denoted by c) and a processing latency (denoted by d), each edge is associated with two weights (more weights could be added), i.e., cost (denoted by w_1) and delay (denoted by w_2).

If the services execute in the parallel structure, all paths between S_0 and S_l would be executed simultaneously. The capacity, processing latency, cost, and delay from S_0 to S_l (denoted by c', d', w'_1, and w'_2 respectively) can be expressed as

$$\begin{aligned} c' &= \sum_{i=1}^{h} c(S_i), \\ d' &= \max_{i=1}^{h} \{d(S_i)\}, \\ w'_1 &= \sum_{i=1}^{h} w_1(p_i), \\ w'_2 &= \max_{i=1}^{h} \{w_2(p_i)\}, \end{aligned} \tag{1}$$

where p_i represents the path $S_0 \rightarrow S_i \rightarrow S_l$.

If the services execute in the case structure, the process would be terminated when one of the paths is executed, and thus we have

$$\begin{aligned} c' &= c(S_{i^*}), \\ d' &= d(S_{i^*}), \\ w'_1 &= w_1(p_{i^*}), \\ w'_2 &= w_2(p_{i^*}), \end{aligned} \tag{2}$$

where $i^* = \arg\min_{i=1}^{h} \{d(S_i) + w_2(p_i)\}$.

For condition structure, we have

$$\begin{aligned} c' &= \sum_{i=1}^{h} \{\mathrm{Prob}_i \cdot c(S_i)\}, \\ d' &= \sum_{i=1}^{h} \{\mathrm{Prob}_i \cdot d(S_i)\}, \\ w'_1 &= \sum_{i=1}^{h} \{\mathrm{Prob}_i \cdot w_1(p_i)\}, \\ w'_2 &= \sum_{i=1}^{h} \{\mathrm{Prob}_i \cdot w_2(p_i)\}, \end{aligned} \tag{3}$$

where Prob_i is the execution probability of S_i such that $\sum_{i=1}^{h} \mathrm{Prob}_i = 1$.

For loop structure,

$$\begin{aligned} c' &= T \cdot c(S_i), \\ d' &= T \cdot d(S_i), \\ w'_1 &= T \cdot w_1(p_i), \\ w'_2 &= T \cdot w_2(p_i), \end{aligned} \tag{4}$$

where T is the loop count. □

Lemma 1 indicates that service selection in a service graph with general flow structures can be converted to selection of a service path with a sequential order. Therefore, for a service graph with general flow structures in it, we can first apply *Lemma 1* to transform paths in parallel, case, condition, and loop structures to the sequential structure, then focus on selecting tandem candidate services to form a sequential service path that meets end-to-end QoS requirements.

For example in the illustrative case for lab data processing (shown in Fig. 4), if parallel network paths exist between Amazon S3 storage and Amazon EC2 server and the network management policy of the service provider (e.g. AT&T) allows parallel data transmission, then we can transform the set of parallel network links from S3 storage to EC2 server to one atomic network service, whose bandwidth is the total bandwidth of all links and delay is the maximum delay among all links.

We assume that the path specified by a user request is in a sequential structure. In this work we mainly focus on addressing the challenges brought in by high-level service composition across the networking and cloud computing domains instead of low-level service composition within cloud systems. (The latter has been studied in various papers as discussed in Related Work section.) Sequential paths are typical, although not the only case, for network-cloud service composition. For example user data are first transmitted via a network service to a cloud server, then processed by the cloud service, and then results are delivered by a network service back to the user, as illustrated in the lab data processing case. Please note that non-sequential paths are possible within either a cloud (or network) service component, for example parallel or loop data process inside the Amazon EC2 service, but are encapsulated by a single cloud (or network) service component.

3 An Approximation Algorithm for Network and Cloud Service Composition

3.1 Algorithm

To resolve the QSC problem, we in this section propose an approximation algorithm for network-cloud service composition. The algorithm addresses both node and edge constraints in a service graph to achieve composition across network and cloud services with end-to-end performance guarantee. Specifically, this algorithm is developed based upon a preprocessing procedure, named PrePro, shown in Algorithm 1. The PrePro first constructs a new graph by simply duplicating the original service graph, and then applies *Lemma 1* to convert parallel, case, condition, loop structures to the sequential ones in this new graph. Next, PrePro removes the node capacity constraint and migrates the processing latency constraints onto the edge constraints. As a consequence, QSC problem is formulated as an MCOP.

Algorithm 1. PrePro

Input: Service Graph $G(V, E, c, d, w)$, R.
Output: $G'(V', E', w')$.
1 Construct a graph $G'(V, E, c, d, w)$ by duplicating $G(V, E, c, d, w)$;
2 Apply *Lemma 1* to transform parallel, case, condition, loop structures to the sequential ones;
3 **while** *to each* $u \in V'$ **do**
4 **if** $\frac{1}{c(u)} > \frac{1}{C}$ **then**
5 Update V' and E' by removing u from V and its connected edges from E;
6 **Continue**;
7 **end**
8 To each u's outgoing edges $(u, v_1), (u, v_2), \ldots$, $w'_1(u, v_i) \leftarrow w_1(u, v_i), \cdots,$
9 $w'_\kappa(u, v_i) \leftarrow w_\kappa(u, v_i) + d(u)$,
10 $w'_{\kappa+1}(u, v_i) \leftarrow w_{\kappa+1}(u, v_i), \cdots,$
11 $w'_K(u, v_i) \leftarrow w_K(u, v_i)$, where w_κ is assumed to be the delay over the link.
12 **end**
13 **return** $G'(V', E', w')$;

As shown in Algorithm 1, PrePro trims the service graph topology according to the capacity constraints. If capacity of a node in the service graph is smaller than the required constraint, the node and its associated links will be eliminated from the graph (lines 4 to 7 in Algorithm 1). Therefore, the remaining service graph satisfies all the capacity constraints. While for the latency constraint, PrePro migrates the processing latency from each node to its outgoing links (lines 8 to 11 in Algorithm 1). For example, consider a service path from the entrance portal S_0 to a service node u and then to the exit portal S_l that the delay of link $S_0 \rightarrow u$ is 50 ms, u has a processing latency 100 ms, and the delay of link $u \rightarrow S_l$ is 20 ms. PrePro migrates the 100 ms latency from node u onto the link $u \rightarrow S_l$; thus getting the following equivalent metric assignment on this path: delay of link $S_0 \rightarrow u$ is 50 ms, processing latency of u is zero, and delay of link $u \rightarrow S_l$ is 120 ms. Such transform performed by PrePro enable the QSC problem to be formulated as a MCOP.

It is straightforward to derive that the time complexity of PrePro is $O(m + n)$. But one should be noticed that in the PrePro, transferring processing latency to link delay will not violate the QoS of parallel tasks because this step is executed in the new graph obtained by the structure conversion process (lines 1 and 2 in Algorithm 1). Based on PrePro algorithm, we propose an approximation algorithm named SCA (shown in Algorithm 2) for network-cloud service composition.

Algorithm 2. SCA

Input: Graph $G(V, E, c, d, w)$, R, H, ϵ.
Output: Path p.
1 Call PrePro to obtain $G'(V', E', w')$, set $W'_k \leftarrow W_k$, $2 \le k \le K$, and $W'_\kappa \leftarrow W_\kappa + D$;
2 To each $e \in E'$ in $G'(V', E')$, compute the new weights $w_k^N(e) = \left\lfloor \frac{w'_k(e)}{W'_k} \cdot \frac{H-1}{\epsilon} \right\rfloor$, $2 \le k \le K$, and set $W_1^N = \cdots = W_K^N = \Delta = \lfloor \frac{H+1}{\epsilon} \rfloor$;
3 **for** $\delta_k = 0$ to Δ, $2 \le k \le K$ **do**
4 $d^v[\delta_2, \cdots, \delta_K] \leftarrow \infty, \forall v \in V$;
 $p^v[\delta_2, \cdots, \delta_K] \leftarrow$ NULL; $d^s[\delta_2, \cdots, \delta_K] \leftarrow 0$;
5 **end**
6 **for** *all* $(\delta_2, \cdots, \delta_K) \in \{0, 1, \cdots, \Delta\}^{K-1}$ *in increasing lexicographic order* **do**
7 **for** each $(u, v) \in E$ s.t. $\delta_k = \lambda_k + w_k(u, v) \ge 0$ **do**
8 **if** $d^v[\delta_2, \cdots, \delta_K] > d^u[\lambda_2, \cdots, \lambda_K] + w_1(u, v)$ **then**
9 $d^v[\delta_2, \cdots, \delta_K] \leftarrow d^u[\lambda_2, \cdots, \lambda_K] + w_1(u, v)$;
 $p^v[\delta_2, \cdots, \delta_K] \leftarrow u$;
10 **end**
11 **if** $d^v[\delta_2, \cdots, \delta_K] > d^v[\delta_2, \cdots, \delta_j - 1, \cdots, \delta_K]$ **then**
12 $d^v[\delta_2, \cdots, \delta_K] \leftarrow d^v[\delta_2, \cdots, \delta_j - 1, \cdots, \delta_K]$;
 $p^v[\delta_2, \cdots, \delta_K] \leftarrow p^v[\delta_2, \cdots, \delta_j - 1, \cdots, \delta_K]$, $2 \le j \le K$;
13 **end**
14 **end**
15 **end**
16 **if** $d^t[\delta_2, \cdots, \delta_K] \le \Delta$ **then**
17 Find a source-destination path p in $G(V, E)$ s.t. $w_1(p) \le \Delta$ and $w_k(p) \le \delta_k$, $2 \le k \le K$;
18 **end**
19 **if** $w_k(p) \le W_k$, $2 \le k \le K$ **then**
20 **return** path p;
21 **else**
22 **return** No feasible path, EXIT;
23 **end**

SCA comprises four key steps. The first step (line 1) applies PrePro to transform the QSC into a MCOP. As a result, the capacity and processing latency on each node are removed. In the second step (line 2), the service graph is simplified via a ceiling operation on both weights and constraints. This ceiling operation follows the rounding method in [25] that dramatically reduces the search space, thus facilitating the problem to be solved.

In the third step (lines 3 to 15) we assume that nodes s and t denote the service entrance portal and the service exit portal, respectively. The notation $d^v[\delta_2, \cdots, \delta_K]$ represents the least first weight, namely, w_1 among path p from s to v such that $w_k(p) \le \delta_k, 2 \le k \le K$. Notation $p^v[\delta_2, \cdots, \delta_K]$ is used to record the predecessor of v on the path p such that $w_1(p) = d^v[\delta_2, \cdots, \delta_K]$ and $w_k(p) \le \delta_k, 2 \le k \le K$. Note that

this step is a dynamic programming process for tackling three or more constraints, which follows the same philosophy of *relaxing* operation in Dijkstra.

The fourth step examines the feasibility of the path generated by the second step. If the path satisfies $w_k(p) \leq W_k$, $2 \leq k \leq K$, the algorithm completes with this path as an output; otherwise the algorithm terminates with no feasible path.

We want to point out that the proposed model and algorithm for network-cloud service composition are applicable to multi-tenant service provisioning environments, where resource contention may exist in the underlying network and cloud infrastructures. In our model, the QoS metrics associated with the service graph reflect the currently available amount of resources. We assume that the infrastructure controller regularly collects such information and make it available to the service composition module, where the proposed algorithm is performed. Whenever a newly created composite service is deployed, the required amount of resources will be allocated in network and cloud infrastructures. Then the infrastructure controller will update the resource availability information, based on which the service composition module will update the QoS metrics on the service graph. When searching a service path for a new service, the PrePro step of the proposed algorithm removes nodes and links that have insufficient capacities from the service graph; thus avoiding using such links/nodes for the new service. Some available network and cloud management technologies may be applied to support collecting and updating the availability information of infrastructure resources, for example OpenDaylight platform for network control and OpenStack API for cloud management.

Now we analyze theoretical properties for SCA.

3.2 Analysis

Theorem 1. *The worst-case time complexity of Algorithm SCA is* $O(m(\frac{H}{\epsilon})^{K-1})$ *time.*

Proof. The first and second steps of SCA for processing the topology spends $O(n+m)$ time. The third step takes $O(m(\frac{H+1}{\epsilon})^{K-1})$ time in the worst-case to calculate the path. Checking the feasibility of the obtained path in the fourth step costs $O(K)$ time. Therefore the worst-case time complexity of SCA is $O(m+n+m(\frac{H+1}{\epsilon})^{K-1}+K) = O(m(\frac{H}{\epsilon})^{K-1})$. □

Theorem 1 indicates that the proposed SCA is faster than the best currently avaiable MCOP algorithm, i.e., FPTAS [26]. This is due to the fact that the number of hops of the final path is known a priori in the context of service composition, which guides the SCA to find the path in a smaller searching space. In addition, the space overhead of SCA is also much smaller than that of FPTAS. Since the FPTAS would extend the original graph and the scale of the extended graph increases exponentially with the number of constraints, FPTAS needs more space to store the extended graph. Leveraging algebraical form of graph extending, SCA does not need much storage space for the extended graph. Hence, SCA is more efficient than FPTAS to address the QSC problem.

Next, we show approximation property of service path p obtained by SCA.

Theorem 2. *The path* p *obtained by SCA satisfies* $w_k(p) \leq (1+\rho) \cdot \eta^{opt} \cdot W_k$, $2 \leq k \leq K$, *where* $\rho = \frac{\epsilon}{\eta^{opt}}$ *is an approximation ratio of SCA.*

Proof. For the optimal path p^{opt}, $w_k(p^{opt}) \leq \eta^{opt} \cdot W_k$, $2 \leq k \leq K$, that is

$$\sum\nolimits_{e \in p^{opt}} w_k(e) \leq \eta^{opt} \cdot W_k. \tag{5}$$

Since $w_k^N(e) = \left\lfloor \frac{w_k(e)}{W_k} \cdot \frac{H-1}{\epsilon} \right\rfloor$, and $\left\lfloor \frac{w_k(e)}{W_k} \cdot \frac{H-1}{\epsilon} \right\rfloor \leq \frac{w_k(e)}{W_k} \cdot \frac{H-1}{\epsilon}$ then

$$\sum\nolimits_{e \in p^{opt}} \left\lfloor \frac{w_k(e)}{W_k} \cdot \frac{H-1}{\epsilon} \right\rfloor \leq \eta^{opt} \cdot \frac{H-1}{\epsilon}. \tag{6}$$

According to the definition of $d^v[\delta_2, \cdots, \delta_K]$ as well as the dynamic programming process, we have

$$\max_{2 \leq k \leq K} \{w_k^N(p)\} \leq \max_{2 \leq k \leq K} \{w_k^N(p^{opt})\}. \tag{7}$$

It implies

$$w_k^N(p) \leq \eta^{opt} \cdot \frac{H-1}{\epsilon}. \tag{8}$$

Since $w_k^N(e) + 1 = \left\lfloor \frac{w_k(e)}{W_k} \cdot \frac{H-1}{\epsilon} \right\rfloor + 1 \geq \frac{w_k(e)}{W_k} \cdot \frac{H-1}{\epsilon}$, then

$$\sum\nolimits_{e \in p} \frac{w_k(e)}{W_k} \cdot \frac{H-1}{\epsilon} \leq \eta^{opt} \cdot \frac{H-1}{\epsilon} + \sum\nolimits_{e \in p} 1. \tag{9}$$

Path p has $H-1$ hops, therefore

$$\sum\nolimits_{e \in p} w_k(e) \leq (\eta^{opt} + \epsilon) \cdot W_k, \tag{10}$$

i.e.,

$$w_k(p) \leq (1+\rho) \cdot \eta \cdot W_k, \tag{11}$$

where $\rho = \frac{\epsilon}{\eta^{opt}}$. This completes the proof. □

Theorem 2 implies that the later $(K-1)$ weights of p is able to asymptotically approximate the optimal with approximation ratio $\rho = \frac{\epsilon}{\eta^{opt}}$ while the first weight of p achieves the smallest. This shows that the proposed SCA algorithm provides a provably performance guarantee; thus is effective and efficient for solving the QSC problem.

4 Performance Evaluation

4.1 Evaluation Settings

We implemented a simulator to evaluate the performance of the proposed algorithm. The simulator is publicly available at [27]. we compared the performance of SCA against that of a variant of ADAPT [28] as well as FPTAS [26] through numerical experiments. To the best of our knowledge, ADAPT and FPTAS [26] are two currently best-known algorithms for MCOP when $K = 2$ and $K > 2$, respectively. Moreover, QSC maps to a special case of MCOP directly when the topology is pruned in advance to satisfy the nodes' constraints. Therefore, ADAPT and FPTAS with minor modification can be applied for resolving QSC.

We are aware of some available data sets of web service QoS parameters, for example QWS (http://www.uoguelph.ca/ qmahmoud/qws/) and WS-Dream (http://www.wsdream.net/dataset.html), which could be used for

performance evaluation of service composition schemes. However, these data sets currently contain QoS metrics mainly for computing type of services (web services for data process and storage) and lack sufficient QoS data for networking services. Though adding networking QoS data into the existing web service dataset may work for the simulation purpose, the simulation results generated by such artificially synthesized dataset are not convincible as the simulation input is controlled. Therefore, in this paper we conduct simulation experiments using randomly generated network/cloud services and QoS parameters in order to evaluate performance of the proposed algorithm for service composition across network and cloud domains.

In the first set of evaluations, we generated a set of random service graphs with node numbers ranged from 100 to 1,000. Each link in the generated networks has two QoS parameters and they are uniformly distributed. As for the second set of evaluations, we generated three networks with 60, 80, 100 nodes respectively since a larger network scale will be overflowed in FPTAS. Each link in these three networks has three QoS parameters and they are also uniformly distributed. Note that the uniform distribution used here is only for illustration, other distributions also work in a similar way. We assume that the service components are spread in five service categories, i.e., $H = 5$, which we believe is reasonable for typical cloud service scenarios. In order to evaluate the performance under different parameter configurations, we set $\epsilon \in [0.1, 0.3]$ for both SCA and ADAPT. Also, we assume that the requests are always feasible; that is, $W_1 = 50$, $W_2 \in [2.5 \times 10^{-5}, 5 \times 10^{-5}]$. The reported results were obtained by running both algorithms 100 times independently, and all of the experiments were conducted on IBM P4 2.6 GHz with 2G memory space and Linux system.

4.2 Performance Metric

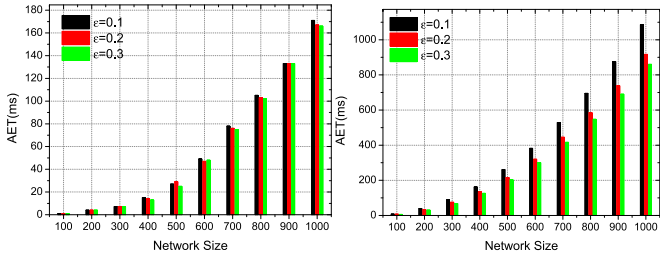
The performance metrics that we use in this section for evaluating algorithms are defined as follows.

Definition 2. *Average Execution Time (AET). AET is the average execution time of an algorithm calculated from 100 independent runs. This metric is used for evaluating the time cost performance of an algorithm, i.e., the time that a user has to wait for getting a service composition result.*

Definition 3. *Path Weights Distance (PWD). For a path p returned by an algorithm,*

$$PWD(p) = \sqrt{\sum_{k=1}^{K} \left(1 - \frac{w_k(p)}{W_k}\right)^2}.$$

PWD denotes the distance between the weights of a found service path and the QoS constraints specified in the service request. This metric reflects the level of QoS guarantee that can be achieved by a service composition algorithm to end users.



(a) SCA (b) ADAPT

Fig. 6. AET comparisons for SCA and ADAPT with different network sizes and ϵ values.

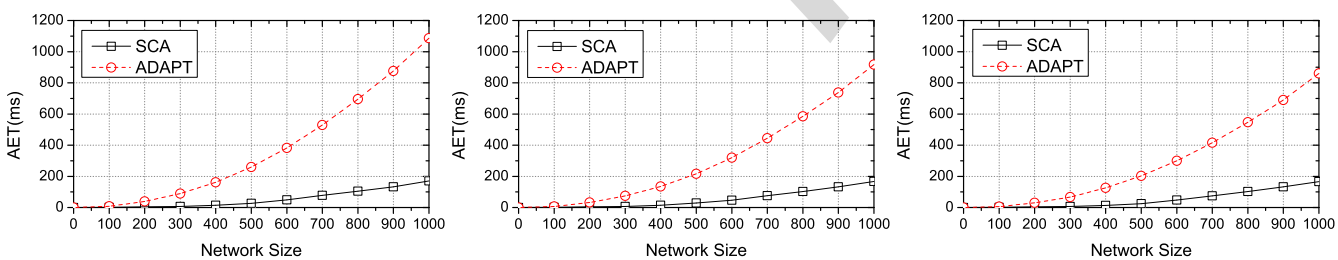
4.3 Evaluation Results

Our experimental results quantify performance of evaluated algorithms in two ways. First, we test the impact of parameter ϵ variation on algorithms' performance by setting constant $W_2 = 3.0 \times 10^{-5}$. Second, we examine the performance of algorithms with constant $\epsilon = 0.1$ for different W_2 settings.

Fig. 6 gives AET for SCA (shown in Fig. 6a) and ADAPT (shown in Fig. 6b) with different ϵ settings. The graphs in this figure show that AET increases with the network size, which is intuitive because both algorithms consume more time for larger scale networks. In addition, we observed that the AET values of both algorithms increase when the parameter ϵ decreases. This is because a smaller ϵ gives a larger path searching space that slows down both algorithms.

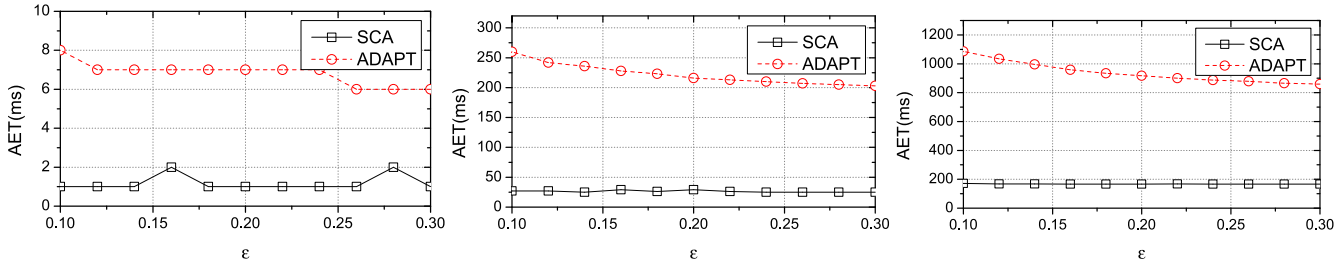
Fig. 7 compares the AET of SCA against that of ADAPT with different ϵ values, where Fig. 7a, Fig. 7b, and Fig. 7c plot the case when $\epsilon = 0.1$, $\epsilon = 0.2$, and $\epsilon = 0.3$, respectively. The curves show that for a certain ϵ value, SCA has smaller AET than ADAPT does, especially in large scale networks; that is, SCA runs much faster than ADAPT. This is because the search space of SCA relies on the number of hops in path, which is very small and can be known in advance. ADAPT employs an approximation test procedure to generate a set of upper and lower bounds for searching the optimal. The search space for ADAPT is determined by the distance between the upper and lower bounds, which is larger compared to SCA. Therefore, SCA has better AET performance than ADAPT.

The AETs (in millisecond) of SCA and ADAPT with different ϵ values for three network sizes, namely, 100, 500, 1,000, are shown in Fig. 8. From this figure we can see that



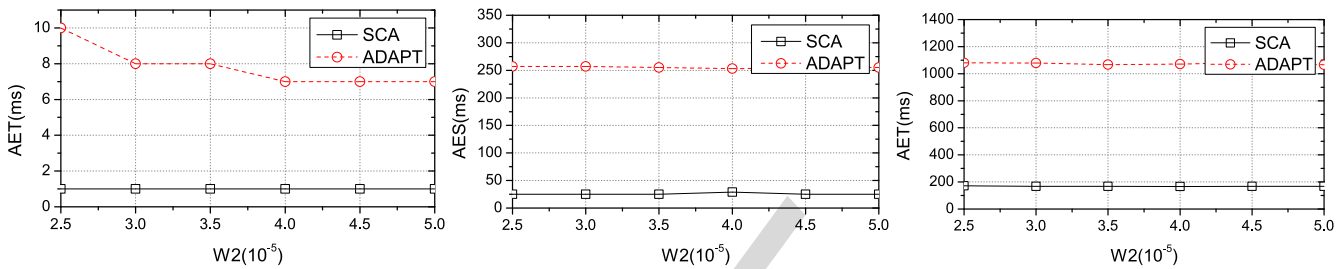
(a) $\epsilon = 0.1$ (b) $\epsilon = 0.2$ (c) $\epsilon = 0.3$

Fig. 7. AET values of SCA and ADAPT for various network scales with different ϵ.



(a) $Network\ \ size = 100$ (b) $Network\ \ size = 500$ (c) $Network\ \ size = 1000$

Fig. 8. AETs of SCA and ADAPT for various ϵ values in different network sizes.



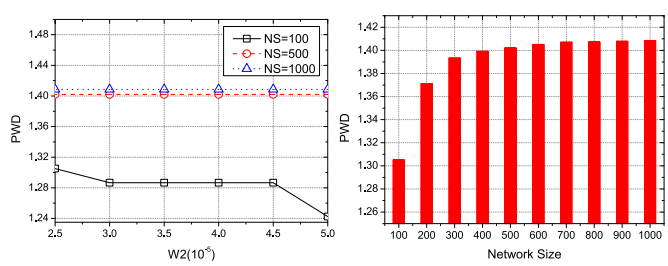
(a) $Network\ \ size = 100$ (b) $Network\ \ size = 500$ (c) $Network\ \ size = 1000$

Fig. 9. AETs of SCA and ADAPT for various constraint values in different network sizes.

regardless of network scale, SCA runs faster than ADAPT, which confirms our observation from Fig. 8. We also observed that SCA has stable AET values for different ϵ while the AET values of ADAPT drop when ϵ increases. That means that SCA is insensitive to ϵ variation but ADAPT is not. Essentially the parameter ϵ plays a vital role in ADAPT for narrowing down the gap between upper and lower bounds, which makes the algorithm sensitive to ϵ variation.

Fig. 9 shows the AET of SCA and ADAPT under various W_2 for three specific network sizes. The results show that the AET of ADAPT is impacted by not only ϵ but also the constraint. It can be seen from Fig. 9a that the AET of ADAPT becomes low when W_2 is loose. This is due to the fact that a looser W_2 makes the search space of ADAPT small; thus reducing its AET. On the other hand, we observe that AET of SCA stays in a relatively constant level, which matches our theoretical analysis result that indicates the running time is not affected by any constraint.

One should be noticed that in the above AET comparisons the paths found by SCA and ADAPT are the same for each specific settings of parameters; therefore the PWD of both algorithms are identical. In other words, we compare the AET under the condition that both of algorithms find the same path. Next, we examine the PWD of our proposed algorithm with different constraints and network sizes.



(a) (b)

Fig. 10. PWDs for SCA with different constraint values and network sizes.

Fig. 10 presents PWD comparisons for SCA with different constraint values and network sizes. Fig. 10a gives the PWD results for different W_2 for three network sizes (NS=100, 500, 1,000). Fig. 10b shows the PWD for different network scales. These two figures indicate that SCA would find a path with possibly greater PWD for large scale networks. Moreover, it can be seen that the PWD of SCA changes little in the condition of loosing W_2. The reason for this is that SCA searches the path primarily based on the path hops. If the path in the right hops that satisfies the constraints, the SCA terminates immediately. Therefore, variations in W_2 have little impact on SCA performance.

Similar to the first set of evaluations, in the second evaluation we compare the performance between SCA and FPTAS in the case of three constraints on each generated topology. Tables 2 and 3 show the AET comparisons for SCA and FPTAS in three networks with different ϵ and

TABLE 2
AET Comparisons for Three Networks with Different ϵ

Nodes	$\epsilon = 0.1$		$\epsilon = 0.2$	
	SCA	FPTAS	SCA	FPTAS
60	38	1,216	17	919
80	58	3,300	25	2,488
100	82	6,727	49	5,295

TABLE 3
AET Comparisons for Three Networks with Different W_2

$W_2(\times 10^{-5})$	$\epsilon = 0.1$		$\epsilon = 0.2$	
	SCA	FPTAS	SCA	FPTAS
2	82	6,727	40	5,295
3	75	6,365	33	4,962
4	74	6,032	33	4,643

constraint. Likewise, the same insight can be obtained; that is, for the same ϵ and network size, SCA executes faster than FPTAS while finding the same path. The loosing of constraints and variation in ϵ almost makes no significant contribution to the AET of SCA. In contrast, variation of constraint and ϵ would affect the performance of FPTAS. Therefore, SCA outperforms FPTAS in terms of AET in the condition of obtaining the same PWD.

5 Related Work

Service composition has been extensively studied in the fields of web services and cloud computing. Various technologies have been developed to achieve service composition that meets functional and/or performance requirements. Most of these technologies are based on either workflow management or AI-planning approach. Surveys of recent research on web service composition can be found in [29] and [30]. Optimization of multiple QoS criteria in service composition has been modeled as a multi-dimension multi-choice 0-1 knapsack problem and solved by integer programming [31] or by heuristic search methods [32]. Genetic algorithms offer another approach to addressing this problem with the advantage of being able to handle nonlinear QoS constraints [33]. In [34] cloud services are modeled as finite state machines and a simple additive weighting technique is used to select an optimal service composition. More recently, Dastjerdi and Buyya presented an ontology-based approach to describing services and their QoS properties and developed a technique to optimize service composition based on user preferences [35]. In [36] the authors studied strategy-proof pricing of cloud service composition and proposed a dynamic programming based approach for service selection with quasi-polynomial computational complexity. More research on QoS-aware web service composition can be found in the survey presented in [37].

The aforementioned research results focus on web/cloud service composition instead of composition between network and cloud services. The modeling and optimization approaches proposed in this paper overcome this limit by employing QoS metrics of both network and cloud services through NaaS paradigm in the composition algorithm; thus finding a path that achieves optimal end-to-end QoS performance. Network-aware service composition is studied in [38], [39]. In [38] the authors first build a network model for estimating the network latency between arbitrary services and potential users, and then develop a selection algorithm leveraging this model to find service compositions that will result in a low latency. Geolocation information is employed in [39] to predict the network delay between candidate services, which is then used in a proposed geolocation-aware service selection algorithm for service composition. The aforementioned work handles network performance and cloud service QoS constraints separately and only considers a single network QoS metric, i.e., the estimated network delay between candidates services, in composition of computing services. In contrast, we address the service composition problem with a holistic vision that abstracts both computing and networking resources as services. The proposed algorithm in this paper integrates multiple network performance metrics and cloud QoS constraints to achieve optimal end-to-end performance for composite network-cloud service provisioning.

The problem of combined service composition and network routing is studied in [40] with a goal to find the optimal service composition in a network that leads to the minimum routing cost. A decision making system is developed in the paper to solve this problem with AI-planning techniques. This work is similar to our research in that service composition and network routing are combined into one problem and solved with a unified mechanism. However, only a single QoS metric (delay) for communications and service processing is considered in [40]. In our work, by formulating network-cloud service composition as a multi-constraint optimal path problem we develop an algorithm that can handle multiple QoS constraints.

Some recent research efforts try to apply techniques for QoS routing, which is typically formulated as Multi-Constraint Optimal Path problem, to tackle the service composition problem. Up to date, much progress has been made toward designing efficient MCOP algorithms for QoS routing. Ergun et al. [28] give an algorithm that adopts an approximation test procedure to generate a set of upper and lower bounds for searching the optimal. This algorithm is known to be one of the fastest algorithms for Delay Constrained Least Cost (DCLC). Xue et al. [41] define variations of MCOP and present a set of approximation algorithms for each problem. Other progress toward QoS routing algorithms can be found from the surveys given in [42], [43], [44].

The aforementioned MCOP techniques were initially proposed for network QoS routing. Further development of these algorithms to enable their applications in network-cloud service composition forms an interesting research topic. The Fully Polynomial Time Approximation Scheme (FPTAS) proposed in [26] has been proved to be currently one of the fastest algorithms for MCOP. However, FPTAS cannot be used directly in the network-cloud service composition context because the vertex in the service graph has QoS metrics and topology of service graph is usually a Directed Acyclic Graph (DAG). In our previous work [45] we proposed an approximation algorithm to tackle the problem of QoS-aware service composition by following the design principle of FPTAS. However, the model proposed in [45] only considers QoS metrics on the edges in a service graph, which limits the flexibility of the model for reflecting different types of services in network and cloud domains. In this paper, we model a service graph with QoS metrics for both nodes and edges entering the nodes, which can be used to respectively represent the service capacities of cloud computing services and QoS parameters of networking services.

6 Conclusions

QoS-aware composition across network and cloud services is a key technology for realizing converged network-cloud service provisioning. In this paper, we have addressed this challenging and important topic by formulating QoS-aware network-cloud service composition as a variant of Multi-Constrained Optimal Path problem and developed an approximation algorithm, termed SCA, for solving the problem. We have given theoretical analysis on properties of SCA and also conducted thorough numerical experiments

for performance evaluation. Both analytical and experimental results show that SCA is effective for meeting multiple QoS constraints and efficient to achieve much shorter response time for network-cloud service composition.

As SCA is designed for sequential task, we plan to extend it to deal with more general tasks such as parallel ones in the future work.

Acknowledgments

This work is supported by NSFC under grant 61309031, Postdoctoral Science Foundation of China under grant 2014M551740, NSF of Chongqing under grant cstc2013jcyjA40026, S&T Research Program of Chongqing Municipal Education Commission under grant KJ130523, and CQUPT Research Fund for Young Scholars under grant A2012-79. This work is also partially supported by Strategic International Collaborative Research Program (SICORP) Japanese (JST) - US (NSF) Joint Research "Big Data and Disaster Research (BDD).

References

[1] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and Grid computing 360-degree compared," in *Proc. Grid Comput. Environ. Workshop*, Nov. 2008, pp. 1–10.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[3] K.-T. Chen, Y.-C. Chang, H.-J. Hsu, D.-Y. Chen, C.-Y. Huang, and C.-H. Hsu, "On the quality of service of cloud gaming systems," *IEEE Trans. Multimedia*, vol. 16, no. 2, pp. 480–495, Feb. 2014.

[4] W.-H. Hsu and C.-H. Lo, "QoS/QoE mapping and adjustment model in the cloud-based multimedia infrastructure," *IEEE Syst. J.*, vol. 8, no. 1, pp. 247–255, Mar. 2014.

[5] K. R. Jackson, K. Muriki, S. Canon, S. Cholia, and J. Shalf, "Performance analysis of high performance computing applications on the amazon web services cloud," in *Proc. IEEE Int. Conf. Cloud Comput. Technol. Sci.*, Nov. 2010, pp. 159–168.

[6] Q. Duan, "Modeling and delay analysis for converged network-cloud service provisioning systems," in *Proc. Int. Conf. Comput., Netw. Commun.*, 2013, pp. 66–70.

[7] Y.-S. Dai, B. Yang, J. Dongarra, and G. Zhang, "Cloud service reliability: Modeling and analysis," in *Proc. 15th IEEE Pacific Rim Int. Symp. Dependable Comput.*, 2009, pp. 1–17.

[8] S. S. Woo and J. Mirkovic, "Optimal application allocation on multiple public clouds," *Comput. Netw. J.*, vol. 68, pp. 138–148, 2014.

[9] T. Kurze, M. Klems, D. Bermbach, A. Lenk, S. Tai, and M. Kunze, "Cloud federation," in *Proc. 2nd Int. Conf. Cloud Comput., Grids, Virtualization*, Sep. 2011, pp. 32–38.

[10] L. F. Bittencourt, E. R. Madeira, and N. L. da Fonseca, "Impact of communication uncertainties on workflow scheduling in hybrid clouds," in *Proc. IEEE Global Commun. Conf.*, 2012, pp. 1623–1628.

[11] M. Fiorani, S. Aleksic, P. Monti, J. Chen, M. Casoni, and L. Wosinska, "Energy efficiency of an integrated intra-data-center and core network with edge caching," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 6, no. 4, pp. 421–432, Apr. 2014.

[12] C. Assi, S. Ayoubi, S. Sebbah, and K. Shaban, "Towards scalable traffic management in cloud data centers," *IEEE Trans. Commun.*, vol. 62, no. 3, pp. 1033–1045, Mar. 2014.

[13] C. Shan, C. Heng, and Z. Xianjun, "Inter-cloud operations via NGSON," *IEEE Commun. Mag.*, vol. 50, no. 1, pp. 82–89, Jan. 2012.

[14] Y. Xia, M. Zhou, X. Luo, Q. Zhu, J. Li, and Y. Huang, "Stochastic modeling and quality evaluation of infrastructure-as-a-service clouds," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 1, pp. 162–170, Jan. 2015.

[15] (2015). Amazon EC2 Cloud. [Online]. Available: http://aws.amazon.com/ec2/

[16] E. Zohar, I. Cidon, and O. Mokryn, "Pack: Prediction-based cloud bandwidth and cost reduction system," *IEEE/ACM Trans. Netw.*, vol. 22, no. 1, pp. 39–51, Feb. 2014.

[17] S. Baset, L. Wang, B. Tak, C. Pham, and C. Tang, "Toward achieving operational excellence in a cloud," *IBM J. Res. Develop.*, vol. 58, no. 2/3, pp. 4:1–4:11, Mar. 2014.

[18] Q. Duan, Y. Yan, and A. V. Vasilakos, "A survey on service-oriented network virtualization toward convergence of networking and cloud computing," *IEEE Trans. Netw. Serv. Manage.*, vol. 9, no. 4, pp. 373–392, Dec. 2012.

[19] N. M. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Comput. Netw.: Int. J. Comput. Telecommun. Netw.*, vol. 54, no. 5, pp. 862–876, 2010.

[20] ETSI-NFV-ISG, "Network function virtualization (NFV)—network operator perspectives on industry progress," Oct. 2014.

[21] ETSI-NFV-ISG, "NFV specification NFV 002: Network function Virtualization (NFV) - architectural framework," Dec. 2014.

[22] Q. Duan, "Modeling and performance analysis on network virtualization for composite network-cloud service provisioning," in *Proc. IEEE World Congress Serv.*, Aug. 2011, pp. 548–555.

[23] P. Van Mieghem and F. Kuipers, "Concepts of exact QoS routing algorithms," *IEEE/ACM Trans. Netw.*, vol. 12, no. 5, pp. 851–864, Oct. 2004.

[24] J. Huang, Y. Liu, and Q. Duan, "Service provisioning in virtualization-based cloud computing: Modeling and optimization," in *Proc. IEEE Global Commun. Conf.*, Dec. 2012, pp. 1728–1733.

[25] S. Sahni, "General techniques for combinatorial approximation," *Oper. Res.*, vol. 25, no. 6, pp. 920–936, 1977.

[26] J. Huang and Y. Tanaka, "QoS routing algorithms using fully polynomial time approximation scheme," in *Proc. ACM/IEEE Int. Workshop Quality Serv.*, Jun. 2011, pp. 1–3.

[27] J. Huang. (2013) Service selection simulator [Online]. Available: https://code.google.com/p/simulator2013/downloads/list

[28] F. Ergun, R. K. Sinha, and L. Zhang, "An improved FPTAS for restricted shortest path," *Inf. Process. Lett.*, vol. 83, no. 5, pp. 287–291, 2002.

[29] J. Rao and X. Su, "A survey of automated web service composition methods," in *Proc. 1st Int. Workshop Semantic Web Serv. Web Process Composition*, Jul. 2004, pp. 43–54.

[30] S. Dustdar and W. Schreiner, "A survey on web services composition," *Int. J. Web Grid Serv.*, vol. 1, no. 1, pp. 1–30, 2005.

[31] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-aware middleware for web services composition," *IEEE Trans. Softw. Eng.*, vol. 30, no. 5, pp. 311–327, May 2004.

[32] T. Yu, Y. Zhang, and K. Lin, "Efficient algorithms for web services selection with end-to-end QoS constraints," *ACM Trans. Web*, vol. 1, no. 1, pp. 1–26, 2007.

[33] A. Klein, I. Fuyuki, and S. Honiden, "SanGA: A self-adaptive network-aware approach to service composition," *IEEE Trans. Serv. Comput.*, vol. 7, no. 3, pp. 452–464, Jul.–Sep. 2014.

[34] H. Bao and W. Dou, "A QoS-aware service selection method for cloud service composition," in *Proc. IEEE 26th Int. Parallel Distrib. Process. Symp. Workshops*, May 2012 pp. 2254–2261.

[35] A. V. Dastjerdi and R. Buyya, "Compatibility-aware cloud service composition under fuzzy preferences of users," *IEEE Trans. Cloud Comput.*, vol. 2, no. 1, pp. 1–13, Jan.–Mar. 2014.

[36] M. Tanaka and Y. Murakami, "Strategy-proof pricing for cloud service composition," *IEEE Trans. Cloud Comput.* vol. PP, no. 99, pp. 1–1, Doi: 10.1109/TCC.2014.2338310.

[37] A. Strunk, "QoS-aware service composition: A survey," in *Proc. 8th IEEE Eur. Conf. Web Serv.*, Dec. 2010, pp. 67–74.

[38] A. Klein, F. Ishikawa, and S. Honiden, "SanGA: A self-adaptive network-aware approach to service composition," *IEEE Trans. Serv. Comput.*, vol. 7, no. 3, pp. 452–464, Jul.–Sep. 2014.

[39] X. Wang, Y. Shen, and J. Zhu, "Network-aware QoS prediction for service composition using geolocation," *IEEE Trans. Serv. Comput.*, vol. 8, no. 4, pp. 630–643, Jul.–Aug. 2015.

[40] X. Huang, S. Shanbhag, and T. Wolf, "Automated service composition and routing in networks with data-path services," in *Proc. 19th IEEE Int. Conf. Comput. Commun. Netw.*, Aug. 2010, pp. 1–8.

[41] G. Xue, W. Zhang, J. Tang, and K. Thulasiraman, "Polynomial time approximation algorithms for multi-constrained QoS routing," *IEEE/ACM Trans. Netw.*, vol. 16, no. 3, pp. 656–669, Jun. 2008.

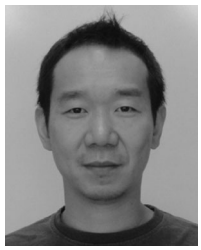
[42] T. Korkmaz, F. A. Kuipers, P. V. Mieghem and M. Krunz, "An overview of constraint-based path selection algorithms for QoS routing," *IEEE Commun. Mag.*, vol. 40, no. 12, pp. 50–55, Dec. 2002.

[43] Z. Tarapata, "Selected multicriteria shortest path problems: An analysis of complexity, models and adaptation of standard algorithms," *Proc. Int. J. Appl. Math. Comput. Sci.*, vol. 17, no. 2, pp. 269–287, Jun. 2007.

[44] S. Giordano, R. G. Garroppo, and L. Tavanti, "A survey on multi-constrained optimal path computation: Exact and approximate algorithms," *Compt. Netw.: Int. J. Comput. Telecommun. Netw.*, vol. 54, no. 17, pp. 3081–3107, Dec. 2010.
[45] J. Huang, G. Liu, Q. Duan, and Y. Yan, "QoS-aware service composition for converged network–cloud service provisioning," in *Proc. IEEE Int. Conf. Serv. Comput.*, Jun. 2014, pp. 67–74.

Jun Huang (M'12) obtained PhD degree in the Institute of Network Technology, Beijing University of Posts and Telecommunications, China, in 2012. He was a visiting researcher at Global Information and Telecommunication Institute, Waseda University, Tokyo, from 2010 to 2011, and a postdoctoral research fellow at Electrical and Computer Engineering, South Dakota School of Mines and Technology, USA, from 2013 to 2014. He was an assistant professor and an associate professor at School of Communications and Information Engineering, Chongqing University of Posts and Telecommunications, where now he is a full professor. His current research interests include Network Optimization and Control, Internet-of-Things, Cloud Computing, and Device-to-Device communications. He has published 70+ refereed papers, several of them are published by prestigious journals and conferences including IEEE TCC, TVT, TBC, TETC, IoTJ, SCC, IWQoS, ACM SAC etc. Dr. Huang is the recipient of Best Paper Runner-up of ACM SAC 2014 and the Best Paper Award of AsiaFI 2011. He has served as a vice Program Chair of ACM RACS 2014, 2015, Co-chair of ACM SAC Software Platform 2015, 2016, Co-chair of EAI MobiMedia 2015 Next Generation Communications and Networking Technology Track, and technical program committee member for numerous conferences such as IEEE GLOBECOM, ICC, ICCCN, CSS, FiCloud.

Qiang Duan received the BS degree in electrical and computer engineering and the MS degree in telecommunications and electronic systems. He received the PhD degree in electrical engineering from the University of Mississippi. He is an associate professor at the Pennsylvania State University Abington College. His general research interest is about data communications and computer networking. His currently active research areas include future internet architecture, network virtualization, network-as-a-service, software defined network, and cloud computing. He has published over 70 journal articles and conference papers and authored five book chapters. He is on the editorial boards for more than 10 international research journals and has served on the technical program committees for numerous international conferences including GLOBECOM, ICC, ICCCN, AINA, WCNC, etc. He is a member of the IEEE.



Song Guo (M'02-SM'11) received the PhD degree in computer science from the University of Ottawa. He is currently a full professor at the School of Computer Science and Engineering, University of Aizu, Japan. His research interests are mainly in the areas of wireless communication and mobile computing, cloud computing, big data, and cyber-physical systems. He has published more than 250 papers in refereed journals and conferences in these areas and received three IEEE/ACM best paper awards. He currently serves as an associate editor of the *IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Emerging Topics in Computing*, and many other major journals. He has also been in organizing and technical committees of numerous international conferences. He is a senior member of the IEEE and the ACM.

Yuhong Yan has been an associate professor in the Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, since June 2008. Before joining the Concordia, she had been a researcher officer in the Institute for Information Technology (IIT) in the Canadian National Research Council's (NRC) since February 2003. Her current research is in service computing. Service computing brings flexibility, interoperability, and composeability to software systems. Service computing combining with mobile computing and cloud computing is going to pave the new landscape of software engineering, pervasive computing and high performance computing. She is the author of over 40 articles and papers. She is one of the organizers of the *IEEE ICWS and SCC* in recent years. Her research is funded by the NSERC Discovery Grant, NSERC Engage Grant, MITACS, Carnari, and multiple industrial companies. She is a member of the IEEE.

Shui Yu (M'05-SM'12) received the BEng and MEng degrees from the University of Electronic Science and Technology of China, Chengdu, P. R. China, in 1993 and 1999, respectively, and the PhD degree from the Deakin University, Victoria, Australia, in 2004. He is currently a senior lecturer with the School of Information Technology, Deakin University, Victoria, Australia. He has published nearly 100 peer review papers, including top journals and top conferences, such as *IEEE TPDS, IEEE TIFS, IEEE TFS, IEEE TMC,* and IEEE INFOCOM. His research interests include networking theory, network security, and mathematical modeling. He actively servers his research communities in various roles, which include the editorial boards of *IEEE Transactions on Parallel and Distributed Systems, IEEE Communications Surveys and Tutorials, and IEEE Access, IEEE INFOCOM TPC* members 2012-2015, symposium co-chairs of IEEE ICC 2014, IEEE ICNC 2013-2015, and many different roles of international conference organizing committees. He is a senior member of the IEEE, and a member of the AAAS.