# Software-Defined Networking for RSU Clouds in Support of the Internet of Vehicles

Mohammad Ali Salahuddin, *Member, IEEE*, Ala Al-Fuqaha, *Senior Member, IEEE*, and Mohsen Guizani, *Fellow, IEEE*

***Abstract*—We propose a novel roadside unit (RSU) cloud, a vehicular cloud, as the operational backbone of the vehicle grid in the Internet of Vehicles (IoV). The architecture of the proposed RSU cloud consists of traditional and specialized RSUs employing software-defined networking (SDN) to dynamically instantiate, replicate, and/or migrate services. We leverage the deep programmability of SDN to dynamically reconfigure the services hosted in the network and their data forwarding information to efficiently serve the underlying demand from the vehicle grid. We then present a detailed reconfiguration overhead analysis to reduce reconfigurations, which are costly for service providers. We use the reconfiguration cost analysis to design and formulate an integer linear programming (ILP) problem to model our novel RSU cloud resource management (CRM). We begin by solving for the Pareto optimal frontier (POF) of nondominated solutions, such that each solution is a configuration that minimizes either the number of service instances or the RSU cloud infrastructure delay, for a given average demand. Then, we design an efficient heuristic to minimize the reconfiguration costs. A fundamental contribution of our heuristic approach is the use of reinforcement learning to select configurations that minimize reconfiguration costs in the network over the long term. We perform reconfiguration cost analysis and compare the results of our CRM formulation and heuristic. We also show the reduction in reconfiguration costs when using reinforcement learning in comparison to a myopic approach. We show significant improvement in the reconfigurations costs and infrastructure delay when compared to purist service installations.**

***Index Terms*—Cloud resource management (CRM), intelligent transportation systems (ITS), Internet of Vehicles (IoV), roadside unit (RSU) cloud, software-defined networking (SDN), vehicular *ad hoc* networks (VANETs).**

## I. Introduction

ADVANCES in microelectromechanical systems (MEMS) technology have enabled the development of smart sensors to be dispersed in our environment. These can vary in form factor, but essentially sense data, process, and communicate it to other sensors, base stations, and/or Internet gateways. They are deployed for various applications, ranging from tracking and environmental monitoring in military applications, to control and actuation in manufacturing plants, to mundane consumer products such as wrist watches. The large-scale production of sensors makes it cost-effective to feature in home appliances, such as smart refrigerators and stoves, and use in automobiles for collision avoidance and climate control. Inter-networking of these sensors, base stations, and Internet gateways makes them an ideal candidate for the Internet of Things (IoT) [1] in general. In this paper, we focus on the Internet of Vehicles (IoV) [2].

In the IoV, the integration of sensors and microcontrollers in the vehicles and fixed roadside infrastructure form an intelligent vehicle grid [3] to cooperatively increase traffic flow and road safety. The vehicular cloud [3]–[5] is the fundamental environment that provides the communication protocols, computational infrastructure, and services and applications for the efficiency of the vehicle grid [3]. The vehicular cloud resides on top of the vehicle grid and is the backbone for its operations. The vehicle grid is essentially a vehicular *ad hoc* network (VANET) of on-board units (OBUs) in vehicles and roadside units (RSUs) in the fixed road infrastructure. Vehicle OBUs comprise localization systems (e.g., global positioning system and inertial measurement unit), processing units, sensors, and radio transceivers mounted in and around the vehicle. RSUs are sensors and microcontrollers installed alongside and in the road, e.g., cameras in traffic lights and road signs and pressure sensors and traffic light actuators on the road. The IEEE standards and protocols for wireless access in vehicular environments (WAVE) define the inter-networking for vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications.

The vehicular cloud can contribute in many different ways to the vehicle grid. Some aim to interconnect vehicle resources into a cloud for cooperative sensory, storage, and computing tasks [5]; whereas, Mershad and Artail [6] propose a cloud of OBUs and still some propose RSUs acting as gateways to traditional clouds. It is important to note that all these variants of vehicular clouds can be subsets or generalizations of each other [4]. However, vehicular clouds will be primarily host safety- and nonsafety-related services and applications for the vehicle grid. User interest in infotainment and convenience applications and services, such as video on-demand, online multiplayer gaming, on-the-go Internet, voice over IP, remote vehicle diagnostic, and road traffic management notifications, will be the driving force for IoV market penetration and mass deployment of infrastructure [7]. However, commercialization of IoV is dependent upon its profitability to service providers who will

Manuscript received June 27, 2014; revised October 29, 2014; accepted October 29, 2014. Date of publication November 06, 2014; date of current version March 13, 2015.

M. A. Salahuddin and A. Al-Fuqaha are with the Department of Computer Science, Western Michigan University, Kalamazoo, MI 49009 USA (e-mail: mohammad.salahuddin@ieee.org; ala@ieee.org).

M. Guizani is with Computer Science and Engineering, Qatar University, Doha, Qatar (e-mail: mguizani@ieee.org).

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

Digital Object Identifier 10.1109/JIOT.2014.2368356

rent cloud resources to host services and the quality of service (QoS) of the applications and services for users in the vehicle grid who will subscribe to these services.

In this paper, we propose a novel vehicular cloud architecture called RSU cloud, which consists of traditional RSUs and specialized microscale datacenters. The RSU cloud hosts services to meet frequently changing demands from the vehicle grid. The novelty of our RSU cloud architecture lies in benefiting from the flexibility and deep programmability offered in software-defined networking (SDN). In SDN, there are two communication planes, the physical data plane and an abstracted control plane. This decoupling of control and data planes enables the flexibility and programmability of the SDN. In the RSU clouds, virtualization via virtual machines (VMs) and SDN is employed to dynamically instantiate, migrate, and/or replicate services and dynamically reconfigure data forwarding rules in the network to meet the frequently changing service demands.

Despite the underlying benefits of the programmability of RSU clouds, service providers will incur costs pertaining to service instantiation, migrations, replications, and network reconfiguration [8], [9], in light of changing demands. Network reconfiguration induces network traffic and could potentially lead to congestion in the data and control planes. We design a novel RSU cloud resource manager (CRM) that contributes to the vehicular cloud by increasing its profitability for service providers and QoS for users in the vehicle grid. Specifically, the CRM aims to achieve minimization in three aspects: 1) minimize the RSU cloud infrastructure delay; 2) minimize operational costs by minimizing the number of service instances (replications); and 3) minimize network reconfigurations, which consume limited bandwidth resources and could potentially lead to deteriorating network performance and QoS.

The scope of this paper and our contributions are as follows:

1) provide the architecture for the RSU cloud and its microdatacenters;
2) present the network reconfiguration overhead analysis by emulating an OpenFlow [10]-enabled SDN in Mininet [11];
3) define the RSU cloud resource management (CRM) multiobjective Integer Linear Programming (ILP) model and systematically solve it to achieve a Pareto optimal frontier (POF) of solutions;
4) design an efficient heuristic for RSU CRM;
5) use the Markov decision process (MDP) and reinforcement learning to select a Pareto optimal solution, which minimizes the costly service migrations over the long term.

This paper is organized as follows. In Section II, we delineate the architecture of RSU clouds and RSU microdatacenter and emulate SDN in Mininet and analyze its inherent reconfiguration overhead, respectively. We discuss work related to RSU clouds and RSU CRM in Section III. In Section IV, we present a mathematical model for the RSU CRM problem. In Section V, we design an efficient heuristic for the RSU CRM problem and use reinforcement learning with the MDP to select a configuration that reduces the VM migration overhead over the long term. In Section VI, we present results and compare our joint optimization and heuristic with purist resource management and show significant improvement in the number of VM migrations, while achieving orders of magnitude improvement in the infrastructure delay and minimizing the number of service hosts. We conclude with a summary of the RSU cloud, our novel approach to solve the inherent RSU CRM problem, and give future work and directions in Section VII.

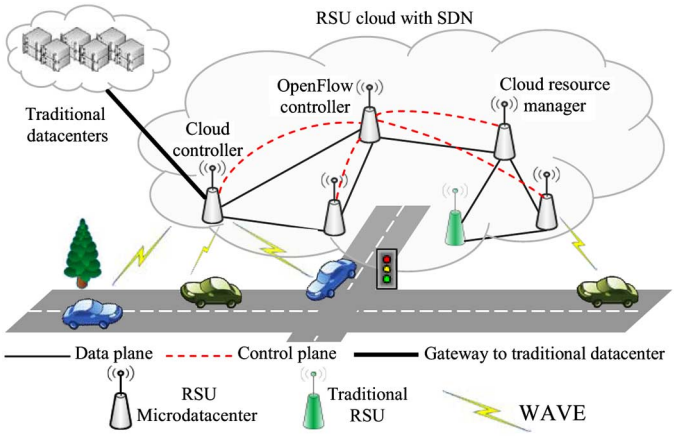



Fig. 1. RSU cloud architecture.

## II. RSU Cloud Architecture

In this section, we discuss the architecture of our RSU cloud, implemented with the SDN. In SDN, there are two communication planes, the physical data plane and an abstracted control plane. This decoupling of control and forwarding planes enables the deep programmability of SDN and allows it to be dynamically reconfigured [8]. The *de facto* communication protocol for SDN is OpenFlow [10]. SDN consists of OpenFlow-enabled switches and controllers, where a switch contains data forwarding rules and the controller has dynamic global network interconnection knowledge. Each switch maintains flows that pertain to data forwarding. Switches receive flow rules, proactively or reactively, from controllers, via the control plane.

Recall that in IoV, users can subscribe for services such as traffic congestion avoidance, remote vehicle diagnostics, on-the-go Internet, online gaming, multimedia streaming, and voice over IP to increase in-vehicle productivity. As illustrated in Fig. 1, RSU clouds include traditional RSUs and microdatacenters that host the services to meet the demand from the underlying OBUs in the mobile vehicles. Traditional RSUs are fixed roadside infrastructure that can perform V2I) and V2V communication using WAVE. A fundamental component of the RSU clouds is the RSU microdatacenter.

An RSU microdatacenter, illustrated in Fig. 2, is a traditional RSU with additional hardware and software components that can offer virtualization and communication capabilities using SDN. The microdatacenter hardware consists of a small form factor computing device and an OpenFlow switch. The software components on the computing device include the host

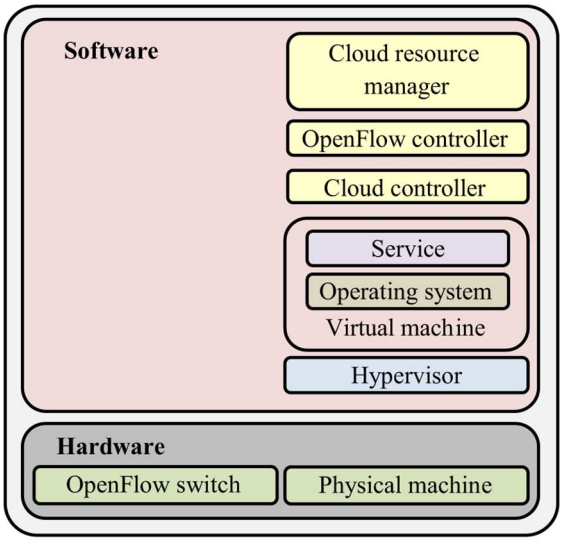



Fig. 2. RSU microdatacenter architecture.

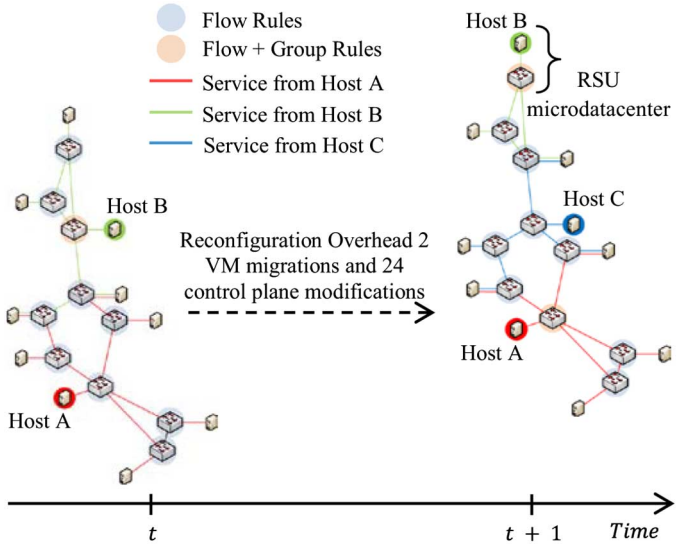



Fig. 3. Reconfiguration overhead in Mininet for our RSU cloud topology.

operating system and a hypervisor. A hypervisor is a low-level middleware that enables virtualization [9] of the physical resources. This allows abstraction of various VMs on a single device. It is a technique widely employed in traditional datacenters that improves resource utilization, portability, and fault tolerance [9]. In this manner, VMs can host services by efficiently sharing resources. VMs also enable service migrations and replications onto other VMs on disparate physical devices. Optionally, one or more of the microdatacenters will have additional software components, namely, OpenFlow controller(s), cloud controller(s) and RSU CRM(s). Our novel RSU CRM will communicate with OpenFlow and cloud controllers, via the data plane, to disseminate information regarding service hosting, service migration, and/or data flow changes, as illustrated in Fig. 1. In the data plane, cloud controllers will govern service migration and hypervisors to instantiate new VMs hosting services. Consequentially, OpenFlow controllers will update their network knowledge and simultaneously update switch flow rules via the control plane.

The dynamic service demands from the vehicle grid may require increasing or decreasing the number of microdatacenters hosting the services or physically migrating the VMs hosting the services from one microdatacenter to another via the data plane. Without loss of generality, we interchangeably use VM migration and service migration. Though, we can reprogram the RSU clouds to dynamically update service hosting and data forwarding information, it is costly and could potentially deteriorate the network performance [8], [9]. VM migration in the data plane, constrains the limited bandwidth in the RSU cloud, and increases network link latency; whereas, updating data forwarding information increases the control plane overhead [12]. Moreover, service providers incur the cost of service migration, and/or replications, and service users' experience deterioration in QoS. Naïve approaches to hosting services across all microdatacenters are too costly, since service providers rent cloud resources from cloud infrastructure providers. Our major contribution is the novel offline CRM, which is responsible for making the decisions regarding the service location, service replications, and data forwarding in the RSU cloud.

### A. *Reconfiguration Overhead Analysis in Mininet*

In this section, we discuss how we emulate an RSU cloud with SDN in Mininet [11], to perform real-world network reconfiguration overhead analysis. SDN switches maintain data forwarding rules in flow tables. A flow table rule is a two tuple with a prefix and an action. A prefix contains, among other things, an ingress port, packet source, and destination information. A typical action specifies the egress port for the incoming packet at the switch. When a packet arrives at a switch, it searches the prefixes in the flow tables and performs the action associated with the first matched prefix. To implement an RSU cloud with SDN in Mininet, we designed and implemented RSU clouds with an RSU microdatacenter as a VM host connected to an OpenFlow switch, with a zero delay, as illustrated in [13]. Our RSU cloud topology is inspired by Florida Department of Transportation (FDOT) deployment of RSUs [14].

It is evident from our topology, illustrated in Fig. 3, that to enable *real-world* reconfiguration overhead analysis, we will have to support multipath. Therefore, we implemented stochastic switching for multipath in an OpenFlow-enabled SDN. OpenFlow offers multipath through group tables, which enable data to be forwarded across multiple egress ports. In this case, a flow table rule *action* points to a group table identification number. A group table rule contains a list of buckets for egress ports for the same ingress port and source-destination pair. We control the data forwarding through these buckets by defining the group type. We use the *select* group type to stochastically select the buckets. The bucket weights are specified so that the traffic between the buckets is split according to the load on multiple paths between the source and the destination.

We used Open vSwitch (OvS) 2.1 [15] that supports group tables and more specifically the optional *select* group type.

However, OvS 2.1 implements the *select* group type by randomly selecting a group bucket; whereas, we want to select buckets stochastically. Therefore, we updated *xlate_select_group* and *group_best_live_bucket* functions on OvS 2.1 to support stochastic switching in accordance with group bucket weights. The Python scripts used for implementing the topology in Mininet and establishing the data forwarding rules, with detailed instructions, are made available online for public access [13].

To perform reconfiguration overhead analysis, we implemented an SDN where each host has demands for a single service and there are a fixed number of service hosts. Fig. 3 depicts two SDNs over the same topology but employing different configurations. We define a configuration as a snapshot of hosts of services and the data forwarding rules in effect. In our analysis, the SDNs are catering to an average demand of 70 and 90 Mb/s, at time $t$ and $t+1$, respectively. We designed a joint optimization model to find the optimal configuration that can meet the demand, while minimizing the number of service hosts and cloud infrastructure delay. Fig. 3 illustrates the configurations highlighting the switches with flow table rules and flow and group table rules for multipath and service hosts. Our reconfiguration overhead heuristic counts the changes in the data forwarding rules and the number of service migrations. In our reconfiguration overhead analysis, removing a service from a host does not induce network traffic and therefore does not count as overhead.

It is important to note that VM migrations induce network traffic in the data plane; whereas, flow and group table modifications make up the control plane overhead. In the control plane, there is a cost associated with adding and deleting flow or group table rules. A flow table and group table rule modification is counted doubly, a deletion of the old rule, followed by an addition of a new rule.

Based on our reconfiguration overhead analysis, we can formally define the configuration and reconfiguration overheads. A configuration is a network snapshot that records the service hosts and the data forwarding rules in the network. Data forwarding rules consist of flow rules and group rules. A configuration is defined as a three tuple $\langle X, Y, Z\rangle$, where $X=\{x_1, x_2, \ldots, x_{|X|}\}$ is the set of service hosts, $Y = \{y_1, y_2, \ldots, y_{|Y|}\}$ is the set of flow rules, and $Z =\{z_1, z_2, \ldots, z_{|Z|}\}$ is the set of group rules.

Reconfiguration overhead consists of two components, number of VM migrations and control plane modifications. For simplicity, we assume that all services are hosted on VMs of the same size and count the VM migrations. This can be extended to different size VMs. Furthermore, tearing down a VM does not induce network traffic and does not add to the reconfiguration overhead. Therefore, given sets of service hosts $X$ and $X'$ for time $t$ and $t+1$, respectively, the VM migrations are equivalent to $|X - X'|$. On the other hand, all control plane modifications add to the reconfiguration overhead. Therefore, given sets of flow rules $Y$ and $Y'$ for time $t$ and $t+1$, respectively, and group rules $Z$ and $Z'$ for time $t$ and $t+1$, respectively, the control plane overhead is calculated as in (1). Intuitively, the control plane overhead is the sum of flow rules to be deleted and added, group rules to be deleted and added, flow rules changed to group rules, and group rules changed to flow rules

$$\begin{aligned}\text{Control plane overhead} = |Y - Y'| + |Y' - Y| + |Z - Z'| \\ + |Z' - Z| + |Y \cap Z'| + |Z \cap Y'|.\end{aligned} \tag{1}$$

## III. Background

Preceding IoV, intelligent transportation systems (ITSs) encompassed the communication and services in VANETs. Taxonomy and classification of ITS services and their requirements have been presented in [4] and [5], and it is evident from these classifications that ITS safety applications are dependent on direct V2V communication; whereas, ITS nonsafety applications reply on resource-constrained RSUs. Therefore, strengthening the RSUs with microdatacenters enable them to provide nonsafety services. We propose an RSU cloud to capitalize on the proximity of RSUs to the end-users and the reliability of fixed infrastructure to deploy a deeply programmable network that can cater to dynamic service demands. Proponents of cloudlets instigate the benefits of moving services to the edges of the network to enhance users' experience.

Strengthening the vehicles in the VANET with a cloud access not only enables myriad computational capabilities that are underutilized by safety applications alone [4], [16], but also overcomes the unreliable V2V communication [17]. Some aim to interconnect OBU and RSU resources into a cloud for cooperative sensory, storage, and computing tasks [5], while others [4], [6] propose that RSUs act as gateways to traditional clouds or design a cloud of OBUs. Vehicular cloud networking (VCN) [5] is being proposed as a revolution to modernize the traditional VANET, which integrates information centric networking and cloud computing with VANETs. In VCN, vehicles and resource-constrained RSUs share their resources in one virtual platform. This is in contrast to our proposed RSU cloud, which only includes RSUs. Among the RSUs in the RSU cloud, some are specialized RSUs that contain microdatacenters with SDN capabilities to dynamically host services and reconfigure data flows. Our RSU cloud can easily coexist with earlier VANET clouds or within the revolutionizing VCN as a vehicular cloud for the vehicle grid in the IoV.

Current techniques to ensure efficient and effective realization of RSU cloud services include rich connectivity at the edge of the network and dynamic routing protocols to balance traffic load [18] and reduce routing delay. Presently, capacity planning tools such as VMWare Capacity Planner, IBM Websphere Cloudburst, and Novell PlateSpin Recon decide the VM placement locations [18]. However, they lack in load balancing at the VM level and result in highly imbalanced traffic distribution [18]. Various researchers [19]–[23] have proposed solutions for low latency cloud service deployment, either independent of cost of service location or jointly. However, like Wu *et al.* [19], we will also jointly minimize cloud service deployment cost and routing delay for the Pareto frontier. In contrast to this work, we leverage the list of Pareto optimal solutions for selecting the one deployment and network configuration, with the least effect on existing service and routing configurations, while [19], using Nash bargaining

techniques, balances optimality with fairness. Our significant contribution lays in the wake of the fact that VM migrations are costly [9].

## IV. RSU CRM

Our RSU cloud consists of RSU microdatacenters that can host services to meet the demands from the vehicles. A configuration takes a snapshot of service hosts and data forwarding rules. Over time, the change in demands requires costly reconfigurations to service hosting, service migration, and/or replications and data forwarding rules. These reconfigurations increase service latency and deteriorate users' experience [8], [9]. We can identify patterns of average demand in the network and classify them according to the time of the day. Therefore, our scheme can be run offline utilizing these traffic patterns. Thus, it is beyond the scope of this paper to analyze the overall complexity of the scheme.

The CRM main concern is the selection of a configuration that minimizes reconfiguration overhead. In this section, we present the problem statement and the CRM model.

### A. Problem Statement

Assume that we have a network graph $G = (V, E)$, a set of services $S$, and a set of average demands $D = \{d_{t_1}, d_{t_2}, \ldots, d_{t_{|T|}}\}$ over time period $T = \{t_1, t_2, \ldots, t_{|T|}\}$ with an initial configuration $\psi^{t_1} = \langle X^{t_1}, Y^{t_1}, Z^{t_1}\rangle$ for demand $d_{t_1}$ at time $t_1$. The set $V$ represents the RSU microdatacenters interconnected by the edges in $E$, each with bandwidth capacity $C_e \; \forall e \in E$. At time $t_i \in T$, there is a demand $b_{n,k}$ for service $k$, $k \in S$ at node $n$, $n \in V$ and the average demand in the network is $d_{t_i}$. Find a Pareto optimal configuration from a set of POF of configurations $\Psi^{t_i}$ to minimize the number of VM migrations $\langle \psi_j^{t_i} | \min(X_j^{t_i} - X^{t_{i-1}}), \; \forall \; \psi_j^{t_i} =< X_j^{t_i}, Y_j^{t_i}, Z_j^{t_i} >, \psi_j^{t_i} \in \Psi^{t_i}\rangle$. Each $\psi_j^{t_i} \in \Psi^{t_i}$ optimally hosts services within a threshold $\phi_k \; \forall k \in S$ to meet service demands at $t_i$, while achieving a load-balanced network, and minimizing infrastructure delay with QoS threshold $\omega_k \; \forall k \in S$.

### B. Delay Model

In this section, we discuss how we compute the cloud infrastructure delay. The delay is based on a lookup table (LUT) with interval $\varphi$, which controls the granularity. The granularity of the LUT is a tradeoff to performance. We compute the delay on a path as a summation of the delays on the edges in the path. The delay on an edge is the summation of processing ($T_p$), queuing ($T_q$), transmission ($T_r$), and propagation ($T_g$) delays. Without loss of generality and similar to [24] and [25], we currently use a G/G/1 queuing system to model the delay on an edge, making it feasible for single-hop and multihop transmissions.

In practice, the LUT table will be built over time, from experimental data. For modeling this, we assume a Poisson process for packet interarrival times $\lambda$ and processing times $\mu$, with mean and standard deviation $t_a, \sigma_a$ and $t_s, \sigma_s$, respectively. This assumption generally enables the problem formulation to be generic, suitable, and adaptable to various different types of scenarios, including multimedia and network studies [24], [25]. The coefficients of variation in interarrival times and packet processing times are $c_a = \sigma_a / t_a$ and $c_s = \sigma_s / t_s$, respectively. These are used in Kingsman formula [26] to approximate the queuing delay in (2). In our LUT, $c_a = 0.7$ and $c_s = 0.7$,

$$T_q = \frac{c_a^2 + c_s^2}{2} \cdot \frac{\lambda/\mu}{\mu - \lambda}. \tag{2}$$

The transmission delay on the edge is based on the distance between the RSUs as in (3). The length of the edge is based on the distance between the RSU nodes. Recall, our topology is inspired by FDOT RSU deployment [14]. We use iTouchMaps [27] to estimate the latitude and longitude of the FDOT RSU locations and coordinate distance calculator [28], to compute the distances between the RSU nodes. We use a processing delay of $T_p = 10\,\mu\text{s}$ and the propagation delay in (4), for a packet size of 800 bytes,

$$T_r = \left( \frac{\text{length}}{\frac{2}{3} \times \text{speed of light}} \right) \tag{3}$$

$$T_g = \left( \frac{\text{packet size}}{C_e} \right). \tag{4}$$

### C. Multiobjective ILP Formulation

We model the RSU CRM problem as a multiobjective ILP problem and systematically solve it to obtain a Pareto frontier. Below, we define and describe the input and output of the formulation.

***Input:***

| | |
|---|---|
| $S$ | Number of services |
| $N$ | $\lvert V\rvert$, the number of RSUs |
| $b_{n,k}$ | Demand at RSU $n$ for service $k$, $\forall\, 1 \le n \le N,\; 1 \le k \le S$ |
| $t_{n,k}$ | $\begin{cases} 1, & \text{if there is a demand for service } k \text{ on RSU } n \\ & \forall 1 \le n \le N, 1 \le k \le S \\ 0, & \text{otherwise} \end{cases}$ |
| $k_{m,n}$ | Number of paths from RSU $m$ to RSU $n$, $\forall 1 \le m, n \le N$ |
| $p_e^{m,n,x}$ | $\begin{cases} 1, & \text{if RSUs } m, n \text{ use path } x \text{ with edge } e \\ & \forall 1 \le m, n \le N, 1 \le x \le k_{m,n}, \\ & 1 \le e \le \lvert E\rvert \\ 0, & \text{otherwise} \end{cases}$ |
| $C_e$ | Bandwidth capacity of edge $e$ |
| $\phi_k$ | Threshold on number of service hosts for service $k$, $1 \le k \le S$ |
| $\omega_k$ | Threshold on infrastructure delay for service $k$ $1 \le k \le S$ |
| $\varphi$ | $1 \le \varphi < \min\{C_e \; \forall 1 \le e \le \lvert E\rvert\}$, used to control the granularity of the LUT for edge $e$ |
| $q_{i,e}$ | Delay for load $i$ on edge $e$, $\forall 0 \le i \le C_e, 1 \le e \le \lvert E\rvert$ |
| $z_{m,k}$ | $\begin{cases} 1 & \text{if service } k \text{ was previously hosted on RSU } m \\ & \forall 1 \le m \le N, 1 \le k \le S \\ 0, & \text{otherwise} \end{cases}$ |

$y_k^{m,n,x}$ $\begin{cases} 1, & \text{if there was a control plane rule for path } x \text{ between RSUs } m,n \text{ for service } k \\ & \forall 1 \le m,\ n \le N, 1 \le x \le k_{m,n}, \\ & 1 \le k \le S \\ 0, & \text{otherwise} \end{cases}$

$B$ A large constant

***Output:***

$h_{m,k}$ $\begin{cases} 1, & \text{if service } k \text{ is hosted on RSU } m, \\ & \forall 1 \le m \le N, 1 \le k \le S \\ 0, & \text{otherwise} \end{cases}$

$r_k^{m,n,x}$ Load carried on path $x$ between RSUs $m$, $n$ for service $k, \forall 1 \le m, n \le N, 1 \le x \le k_{m,n}$, $1 \le k \le S$, $0 \le r_k^{m,n,x} \le \min\{C_e \forall 1 \le e \le |E|, \exists p_e^{m,n,x} = 1\}$

$a_k^{m,n,x}$ $\begin{cases} 1, & \text{if there is a control plane rule for path } x \\ & \text{between RSUs } m,n \text{ for service } k, \\ & \forall 1 \le m,\ n \le N, 1 \le x \le k_{m,n}, 1 \le k \le S \\ 0, & \text{otherwise} \end{cases}$

$l_e$ Load on edge $e$, $1 \le e \le |E|$

$d_e$ Delay on edge $e$, $1 \le e \le |E|$

$v_e$ Maps the load on edge $e$ to a multiple of $\varphi$

$f_{e,i}$ $\begin{cases} 1, & \text{if load on edge } e \text{ is equal to} \\ & i \times \varphi \forall 0 \le i \le C_e, 1 \le e \le |E| \\ 0, & \text{otherwise} \end{cases}$

$g^{m,n,x}$ Delay on path $x$ between RSUs $m$, $n$ $\forall 1 \le m,\ n \le N, 1 \le x \le k_{m,n}$, $0 \le g^{m,n,x} \le j^{m,n,x}$

$w_k^{m,n,x}$ Ancillary variable for nonlinear products of continuous variable $g^{m,n,x}$ with binary variable $h_{m,k}, \forall 1 \le m,\ n \le N, 1 \le x \le k_{m,n}, 1 \le k \le S$

$\alpha_{m,k}$ $\begin{cases} 1, & h_{m,k} - z_{m,k} > 0 \forall 1 \le m \le N, 1 \le k \le S \\ 0, & \text{otherwise} \end{cases}$

$\beta_k^{m,n,x}$ $\begin{cases} 1, & a_k^{m,n,x} - y_k^{m,n,x} > 0 \forall 1 \le m,n \le N, \\ & 1 \le x \le k_{m,n}, 1 \le k \le S \\ 0, & \text{otherwise} \end{cases}$

$\gamma_k^{m,n,x}$ $\begin{cases} 1, & y_k^{m,n,x} - a_k^{m,n,x} > 0 \forall 1 \le m,n \le N, \\ & 1 \le x \le k_{m,n}, 1 \le k \le S \\ 0, & \text{otherwise.} \end{cases}$

Our multiobjective formulation is to minimize the reconfiguration overhead, pertaining to VM migration, control plane modifications, number of service installations, and cloud infrastructure delays. The minimization in the reconfiguration overhead is the sum of VM migrations and control plane modifications as in (5). We use weight $0 \le \rho \le 1$ to control the priority of reconfiguration overheads, such that for $\rho > 0.5$, minimizing VM migration takes priority over minimizing control plane modifications.

In this way, we minimize moving VMs hosting services, i.e., $\min(h_{m,k} - z_{m,k} | 1 \le m \le N, 1 \le k \le S)$, where $z_{m,k}$ is VM $m$ hosting service $k$ from previous configuration and $h_{m,k}$ is VM $m$ hosting service $k$ in current configuration. For example, consider a demand at time $t$ met by $z_{3,1} = 1$, i.e., RSU 3 hosting service 1. Then for a new demand at time $t+1$, the model will benefit by choosing $h_{3,1} = 1$, thus minimizing the VM migration overhead. Control plane overhead entails reducing flow and group rule modifications, i.e., addition and deletion of rules in $\beta_k^{m,n,x}$ and $\gamma_k^{m,n,x}$ $\forall 1 \le m,\ n \le N, 1 \le x \le k_{m,n}, 1 \le k \le S$, respectively,

$$\min \left\{ \begin{array}{c} \sum_{m=1}^{N} \sum_{k=1}^{S} \rho \cdot \alpha_{m,k} + \\ \sum_{m=1}^{N} \sum_{n=1}^{N} \sum_{x=1}^{k_{m,n}} \sum_{k=1}^{S} (1-\rho) \cdot (\beta_k^{m,n,x} + \gamma_k^{m,n,x}) \end{array} \right\}. \quad (5)$$

Simultaneously, we aim to minimize the number of service hosts and achieve a load-balanced network with (6). Mathematically, to model an objective function with disparate units, the delay on an edge $d_e \forall 1 \le e \le |E|$ and the number of service host $h_{m,k} \forall 1 \le m \le N, 1 \le k \le S$, we normalize delay and make the objective unitless [29]. We use weighted sum approach with weight $P$ for this multiobjective optimization. This model pushes the load across multiple paths to minimize the infrastructure delay across all the edges and controls the delay from rising drastically with load

$$\min \left\{ \sum_{m=1}^{N} \sum_{k=1}^{S} P \cdot h_{m,k} + \sum_{e=0}^{|E|} (1-P) \cdot \frac{d_e}{q_{C_e,e}} \right\}. \quad (6)$$

The minimization in infrastructure delay competes with the minimization of service hosts. Trivially, the services could be installed across all RSUs so that every demand is met by services hosted locally. However, this naïve approach would neither be efficient for "resource-constrained" RSU clouds nor cost-effective for service providers. Therefore, there is a direct tradeoff between the number of service hosts and the infrastructure delay. Furthermore, every reconfiguration incurs VM migrations and control plane modifications to the network. Therefore, we have to ensure that we are not superfluous with service hosts. The VM migrations are counted in constraints (7) and (8)

$$B \cdot \alpha_{m,k} \ge h_{m,k} - z_{m,k} \quad \forall 1 \le m \le N, 1 \le k \le S \quad (7)$$

$$h_{m,k} - z_{m,k} + (1 - \alpha_{m,k}) \cdot B \ge 0 \quad \forall 1 \le m \le N, 1 \le k \le S. \quad (8)$$

The control plane overhead is counted as the addition $\beta_k^{m,n,x}$ and deletion $\gamma_k^{m,n,x}$ of control plane rules counted as in constraints (9) through (14)

$$r_k^{m,n,x} \le B \cdot a_k^{m,n,x} \ \forall 1 \le m,n,m \ne n \le N \quad 1 \le x \le k_{m,n}, 1 \le k \le S \quad (9)$$

$$r_k^{m,n,x} \ge a_k^{m,n,x} \ \forall 1 \le m,n,\ m \ne n \le N \quad 1 \le x \le k_{m,n}, 1 \le k \le S \quad (10)$$

$$B \cdot \beta_k^{m,n,x} \ge a_k^{m,n,x} - y_k^{m,n,x} \quad \forall 1 \le m,n,m \ne n \le N, 1 \le x \le k_{m,n}, 1 \le k \le S \quad (11)$$

$$a_k^{m,n,x} - y_k^{m,n,x} + (1 - \beta_k^{m,n,x}) \cdot B \ge 0 \quad \forall 1 \le m,n,m \ne n \le N, 1 \le x \le k_{m,n}, 1 \le k \le S \quad (12)$$

$$B \cdot \gamma_k^{m,n,x} \geq y_k^{m,n,x} - a_k^{m,n,x}$$
$$\forall 1 \leq m,n,m \neq n \leq N, 1 \leq x \leq k_{m,n}, 1 \leq k \leq S \quad (13)$$

$$y_k^{m,n,x} - a_k^{m,n,x} + (1 - \gamma_k^{m,n,x}) \cdot B \geq 0$$
$$\forall 1 \leq m,n,m \neq n \leq N, 1 \leq x \leq k_{m,n}, 1 \leq k \leq S. \quad (14)$$

The minimization objective in (6) is subject to the following constraints. The demand should be met or exceeded by the RSU service providers as in constraints (15). Constraints (16) and (17) ensure that only those paths carry network load that are between RSU service providers and consumers. Note that RSU service providers can meet their demands locally

$$\sum_{m,x} r_k^{m,n,x} \geq b_{n,k} \quad \forall 1 \leq n \leq N, 1 \leq k \leq S \quad (15)$$

$$B.h_{m,k}.t_{n,k} \geq \sum_x r_k^{m,n,x}$$
$$\forall 1 \leq m,n,m \neq n \leq N, 1 \leq k \leq S \quad (16)$$

$$h_{m,k}.t_{n,k} \leq \sum_x r_k^{m,n,x}$$
$$\forall 1 \leq m,n,m \neq n \leq N, 1 \leq k \leq S. \quad (17)$$

The delay on a path is the sum of delays on its edges. The delay on an edge corresponds to the load on the edge and is looked up in the LUT, using indexing. The load and edge must match the enumeration in the LUT. This is in constraints (18)–(22),

$$l_e = \sum_{m,n,x,k} p_e^{m,n,x} \cdot r_k^{m,n,x} \quad \forall 1 \leq e \leq |E| \quad (18)$$

$$l_e = \varphi \cdot v_e \quad \forall 1 \leq e \leq |E| \quad (19)$$

$$v_e = \sum_{i=0}^{C_e/\varphi} i.f_{e,i} \quad \forall 1 \leq e \leq |E| \quad (20)$$

$$\sum_{i=0}^{C_e/\varphi} f_{e,i} = 1 \quad \forall 1 \leq e \leq |E| \quad (21)$$

$$d_e = \sum_{i=0}^{C_e/\varphi} f_{e,i}.q_{i,e} \quad \forall 1 \leq e \leq |E|. \quad (22)$$

The delay on a path $g^{m,n,x}$ is bounded by infrastructure delay threshold $\omega_k$ for service $k$ in constraints (23)–(27),

$$g^{m,n,x} = \sum_{e=0}^{|E|} p_e^{m,n,x} \cdot d_e$$
$$\forall 1 \leq m,n,m \neq n \leq N, 1 \leq x \leq k_{m,n} \quad (23)$$

$$w_k^{m,n,x}.t_{n,k} \leq \omega_k$$
$$\forall 1 \leq m,n,m \neq n \leq N, 1 \leq x \leq k_{m,n}, 1 \leq k \leq S \quad (24)$$

$$w_k^{m,n,x} \leq j^{m,n,x} \cdot h_{m,k}$$
$$\forall 1 \leq m,n,m \neq n \leq N, 1 \leq x \leq k_{m,n}, 1 \leq k \leq S \quad (25)$$

$$w_k^{m,n,x} \leq g^{m,n,x}$$
$$\forall 1 \leq m,n,m \neq n \leq N, 1 \leq x \leq k_{m,n}, 1 \leq k \leq S \quad (26)$$

$$w_k^{m,n,x} \geq g^{m,n,x} - j^{m,n,x}.(1 - h_{m,k})$$
$$\forall 1 \leq m,n,\ m \neq n \leq N, 1 \leq x \leq k_{m,n}, 1 \leq k \leq S. \quad (27)$$

In constraints (28) and (29), we bound the number of service hosts and ensure that every service is hosted

$$\sum_{m=1}^{N} h_{m,k} \geq 1 \quad \forall k \quad (28)$$

$$\sum_{m=1}^{N} h_{m,k} \leq \phi_k \quad \forall k. \quad (29)$$

We assume that all RSUs have demands for all services $k$ and are equally equipped to host any service $k$. Further, we assume that all RSUs have unlimited resources to meet any service demand. Therefore, we assume that the number of service hosts is directly proportional to the financial cost of the service hosting. This removes subjectivity from our results.

Recall, at time $t_i \forall t_i \in T$, the average network demand is $d_{t_i} \in D$, so for every service $k$, we generate demand at every RSU $n$, i.e., normally distributed with mean $d_{t_i}$ and standard deviation $\sigma = 0.05 \times d_{t_i}$ and record it in $b_{n,k}$. As previously described, we used the topology in Fig. 3 and the LUT is generated for the corresponding edges.

There are various techniques for solving multiobjective linear and ILP problems. One common approach is the weighted sum, where weights are used to control the priority of one objective with respect to another. Due to the intrinsically ordered nature of our problem, we are able to solve the multiobjective RSU CRM problem by decomposing it into a smaller dual objective ILP (6). We systematically solve (6) to build a POF of configurations.

In our systematic approach, we first use the lp_solve [30] linear programming engine to solve our objective, by minimizing the number of service hosts and infrastructure delay $\sum_{m=1}^{N} \sum_{k=1}^{S} P \cdot h_{m,k} + \sum_{e=0}^{|E|} (1 - P) \frac{d_e}{q_{C_e,e}}$, subject to constraints (15) through (29), with $P = 0$. For a given number of service host threshold $\phi_k$, the optimization yields an optimal solution in the POF. Next, we populate the POF $\Psi^{t_i}$ by adding an optimal solution $\psi_j^{t_i}$ for each $\phi_k$, such that $1 \leq \phi_k \leq N$ and $\phi_k \in \mathbb{Z}^+$. We start at $\phi_k = N$ and continue until there is no infeasible solution. At the end, we will have $\psi_j^{t_i} \in \Psi^{t_i} \forall j$, $\phi_k$, at time $t_i$ and average network demand is $d_{t_i}$.

Initially, we assume a fresh network; therefore, no previous configurations exist, and thus we select a Pareto optimal with the minimum number of service hosts, as the initial configuration at $t_0$ for demand $d_{t_0}$. However, $\forall\, t_i > t_0$ there will be reconfigurations costs between $\Psi^{t_i}$ and $\Psi^{t_{i-1}}$. We select the Pareto optimal $\left\{\psi_j^{t_i} | \min\left(X_j^{t_i} - X^{t_{i-1}}\right) \forall j,\, \psi_j^{t_i} \in \Psi^{t_i}\right\}$, such that first it minimizes the difference in VM migrations, followed by the control plane overhead, $\left\{\psi_j^{t_i} | \min\left(Y_j^{t_i} - Y^{t_{i-1}}\right) \forall j,\, \psi_j^{t_i} \in \Psi^{t_i}\right\}$, by controlling the weights in (5).

```
FOR i = 1 to ⌈log N⌉
  m = ⌈N/2^i⌉
  FOR j = 1 to n^2
    IF i == 1
      select m random RSU providers and fill in
      configuration set C_j^i as service providers for
      j^th configuration
    ELSE
      select random set of RSU providers with
      C_j^i ⊆ C_j^(i-1) with |C_j^i| = m
    END IF
      identify and accommodate local demands
      FOREACH randomly selected RSU consumer
        WHILE demand is not satisfied
          place a percent p of demand on best path
          between RSU consumer and provider in C_j^i
        END WHILE
      END FOREACH
  END FOR
  select configuration with minimum delay for i^th
  level
END FOR
```

Fig. 4. Pseudocode for CRM heuristic.

## V. RSU CRM Heuristic

We design and implement a novel heuristic for the RSU CRM problem, delineated in Fig. 4. We show that our heuristic efficiently yields near optimal results, by *always* operating on the POF of nondominated solutions. Our heuristic can be decomposed into two components: 1) generate POF and 2) prune POF to find the configuration that minimizes VM migrations, followed by the control plane overhead. It is important to note that for a given average demand $d_{t_i}$, we use the *same* demand ($b_{n,k}$) for each RSU as in the ILP. This ensures accurate comparison of the results from ILP optimization and our heuristic.

To generate the POF, for each service $k \in S$, we begin by randomly selecting $\lceil N/2 \rceil$ nodes to be the RSUs hosting services and meet their demands locally. For all the remaining RSUs, we randomly select an RSU $n \in V$ and satisfy its demand by *iteratively* selecting a service host for its demand, such that a fixed percent of demand receives the current best infrastructure delay. This uniquely enables us to distribute the load across the paths in the networks and achieve a load-balanced network. We repeat this, until all the RSUs demands have been satisfied. At the end, we have a configuration. We repeat this to generate $n^2$ configurations and select the configuration $\psi_j^{t_i} \in \Psi^{t_i}$ that minimizes the infrastructure delay to be included in the POF.

Next, we iteratively select each of the $n^2$ configurations, and randomly select $\lceil |X_j^{t_i}|/2 \rceil$ number of service hosts, and repeat the process of meeting demands in the network until the POF $\Psi^{t_i}$ has been filled with all the valid configurations for the average demand $d_{t_i}$ at $t_i$. Now, we can generate $\Psi^{t_i}$ for $d_{t_i}$ $\forall t_i \in T$ and select the configuration that minimizes the number of VM migrations $\langle \psi_j^{t_i} | \min \left(X_j^{t_i} - X^{t_{i-1}}\right) \forall \psi_j^{t_i} =< X_j^{t_i}, Y_j^{t_i}, Z_j^{t_i} >, \psi_j^{t_i} \in \Psi^{t_i} \rangle$. We break a tie between configurations, by selecting the configuration with minimum VM migrations, followed by control plane modifications.

### A. Analysis

Given a network $G = (V, E)$, where $V$ is a set of vertices and $E$ is the set of edges, with all $|V| = n$ nodes requesting services. Consider $\log n$ levels with a set $C^i = \{C_1^i, C_2^i, \ldots, C_{n^2}^i\}$ of configurations at each level $i$, such that $|C^i| = n^2 \; \forall C_j^i \in C^i$ is a set of providers that meet consumer demands and $|C_j^i| = \lceil \frac{n}{2^i} \rceil \; \forall 1 \leq j \leq n^2$. In level $i+1$, a set $C^{i+1}$ of configuration, such that $|C^{i+1}| = n^2$ holds and $|C_j^{i+1}| = \lceil \frac{n}{2^{i+1}} \rceil \; \forall 1 \leq j \leq n^2$ and $C_j^{i+1} \subseteq C_j^i$. The subset relationship ensures that we get nondominated configurations across levels.

Let $X_j^i$ be a random variable, which is an event of selecting configuration $C_j^i$ and $E\left[X_j^i\right] = p_j^i$, which is the probability of loss in configuration. The loss in configuration is dependent on the selection of this configuration, which eliminates other configurations that may yield better delay. Then $X = \sum_i \sum_j X_j^i$ is the loss in configurations. In the worst case, in level $\lceil \log n \rceil$, the expected loss in configurations $\mu = E[X] = \frac{1}{n^2} \log n$. This can be generalized to other levels. We use Chernoff bounds to compute the deviation of $X$ from $\mu$,

$$\Pr(X < \mu - \lambda) = \Pr\left(X < \left(1 - \frac{\lambda}{\mu}\right)\mu\right)$$

Let $\frac{\lambda}{\mu} = \delta$, then

$$\Pr(X < (1-\delta)\,\mu) \leq e^{\left(-\frac{\delta^2 \mu}{2}\right)} \leq e^{\left(-\frac{\delta^2 \log n}{2n^2}\right)}.$$

*Claim:* With high probability $X \in \mu \pm O\left(\frac{\log n}{n}\right)$.

*Proof:*

Let $\lambda = O\left(\frac{\log n}{n}\right)$, $\delta = \frac{\lambda}{\mu} = \frac{O\left(\frac{\log n}{n}\right)}{\frac{1}{n^2}\log n} = O(n)$, then

$$\Pr(X < (1-\delta)) \leq e^{\left(-\frac{(O(n))^2 \log n}{2n^2}\right)}$$

$$\Pr\left(X < (1-\delta)\,\mu\right) \leq e^{\left(-\frac{cn^2 \log n}{2n^2}\right)} \leq e^{(-c \log n)} \leq -c \log n$$

$$\Pr(X < (1-\delta)\,\mu) \leq \frac{1}{n^c}.$$

Hence, with high probability, $X \in \mu \pm O\left(\frac{\log n}{n}\right)$.

However, there is a short-sightedness in this heuristic. Though we minimize the reconfiguration overhead required to meet the change in demand from $t_{i-1}$ to $t_i$, our heuristic applies a myopic approach to configuration selection. Consequentially, a configuration that minimizes the reconfiguration overhead from $t_{i-1}$ to $t_i$ may not be the best configuration over the long term for the network. To overcome this, we employ reinforcement learning to select configurations that yield the optimal number of reconfigurations over the long term.

### B. MDP

Thus far, the heuristic we used to select a configuration simply minimizes the number of VM migrations. This heuristic is shortsighted and seeks immediate gains, but lacks the long-term knowledge to make an educated decision about the configuration to be selected such that a long-term reduction in the *number* of VM migrations can be achieved. The configuration selection

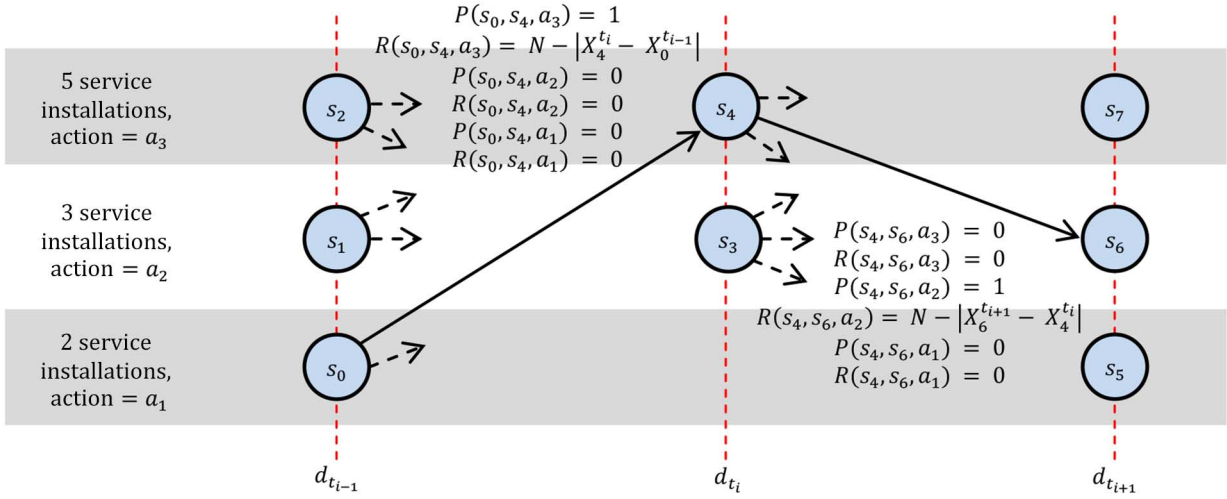



Fig. 5. MDP for minimizing VM migrations.

decision problem lends itself perfectly to the MDP, where the outcome is partially random and controlled by a decision maker.

The MDP is a discrete time stochastic process, defined by a quad-tuple $<S, A, P, R>$, where $S$ is the set of states and $A$ is the set of actions. The transition from state $m$ to state $n$ is based on the action $a$, defined by the probability $P(m, n, a)$, with corresponding reward $R(m, n, a)$. The goal of the MDP is to find a "policy" that dictates the action to take in a state that maximizes the expected reward.

We design the configuration selection process as an MDP in the following manner. First, every configuration from the POF $\psi_j^{t_i} \in \Psi^{t_i}$ from the heuristic, across all $d_{t_i}$ for all $t_i \in T$ is enumerated as a list of states in the MDP. Next, a set of actions are defined. In our scenario, the number of RSUs $N = 10$; therefore the heuristic finds configurations for 5, 3, and 2 service hosts. Then, the actions are defined as $A = \{a_1, a_2, a_3\}$ where the configuration with 2, 3, and 5 installations is selected, respectively. In MDP, we define state transitions using a probability matrix P, where we can only transition to a state, if it is a configuration in the next time instance, i.e., $P(m, n, a)$ $\forall m \in \Psi^{t_{i-1}}$, $n \in \Psi^{t_i}, a \in \mathrm{A}$.

Therefore, the configurations are chronologically ordered. The reward matrix $R$ is populated as the complement of the VM migration costs. Therefore, the reward $R(m, n, a)$ is the difference between $N$, the maximum number of VM migrations, and the number of VM migrations, $(X_j^{t_i} - X_k^{t_{i-1}})$ $\forall \psi_j^{t_i} = <X_j^{t_i}, Y_j^{t_i}, Z_j^{t_i}>, \psi_j^{t_i} \in \Psi^{t_i}, \psi_k^{t_{i-1}} = <X_k^{t_{i-1}}, Y_k^{t_{i-1}}, Z_k^{t_{i-1}}>, \psi_k^{t_{i-1}} \in \Psi^{t_{i-1}}$ and when moving from configuration $m$ to $n$ on $a$ is $R(m, n, a) = N - (X_j^{t_i} - X_k^{t_{i-1}})$, $\forall \psi_j^{t_i} = <X_j^{t_i}, Y_j^{t_i}, Z_j^{t_i}>, \psi_j^{t_i} \in \Psi^{t_i}, \psi_k^{t_{i-1}} = <X_k^{t_{i-1}}, Y_k^{t_{i-1}}, Z_k^{t_{i-1}}>, \psi_k^{t_{i-1}} \in \Psi^{t_{i-1}}$. This is illustrated in Fig. 5.

MDP can be solved using various techniques, such as Q-learning, policy iteration, value iteration, and linear programming. We are interested in policy iteration to get the optimal policy. The optimal policy is one that has not changed in two successive iterations and maximizes the reward. The policy dictates the action to take in a state. For this purpose, we use the MDP toolbox for MATLAB designed and developed by Chadès *et al.* [31]. The toolbox contains various techniques for solving MDP, including the policy iteration. We use this toolbox with its policy iteration technique to generate a policy which maximizes the reward. Recall that our reward is the complement of the number of VM migrations; therefore, a policy that maximizes the reward value will minimize the number of VM migrations in the long term. This MDP-derived policy for configuration selection will ensure that we select configurations that minimize the VM migrations over the long term, even though it may incur a higher VM migration cost at a time $t_i$, over the long run, it will incur the optimally minimal number of VM migrations.

## VI. Results and Discussion

In this section, we present and discuss our results, with services $S = 1$, and assume fast Ethernet connections between the RSUs, such that $C_e = 100\,\mathrm{Mb/s} \quad \forall e \in E$ and $\varphi = 1$, so that we have a fine grain LUT. In practice, the LUT will be built over time from experimental data. We run multiple iterations for the same network, so for demand $d_{t_i} \in D$, every service $k$, and in every iteration, we generate demand at every RSU $n$, such that it is normally distributed with mean $d_{t_i}$ and standard deviation $\sigma = 0.05 \times d_{t_i}$ and record it in $b_{n,k}$. We run five iterations to get the confidence intervals. Recall, similar to [24] and [25], we use a G/G/1 queuing system and assume a Poisson process for packet interarrival times $\lambda$ and processing times $\mu$, with mean and standard deviation, $t_a$, $\sigma_a$, and $t_s, \sigma_s$.

We compare our results with purist approach. In the given problem, there are two possible purist approaches: 1) cost optimization, which optimally hosts services to meet network demands, as illustrated in Fig. 6, irrespective of the infrastructure delay incurred by the services, as illustrated in Fig. 7 and 2) delay optimization, which optimally hosts services to minimize infrastructure delay, irrespective of cost of hosting services. In CRM, we employ joint optimization that minimizes both, the number of service hosts and infrastructure delay. Trivially, delay optimization would maximally deploy hosts, so that services are met locally, incurring no infrastructure delay. Moreover, this would not be viable for RSU cloud service providers. Therefore, we do not compare our joint optimization approach to delay optimization.

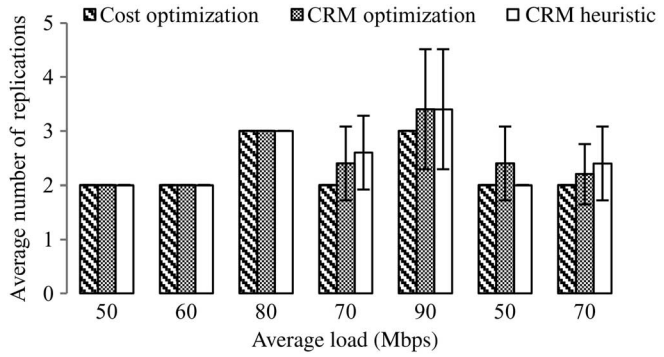



Fig. 6. CRM optimization and heuristic achieve near optimal cost optimization in number of replications.

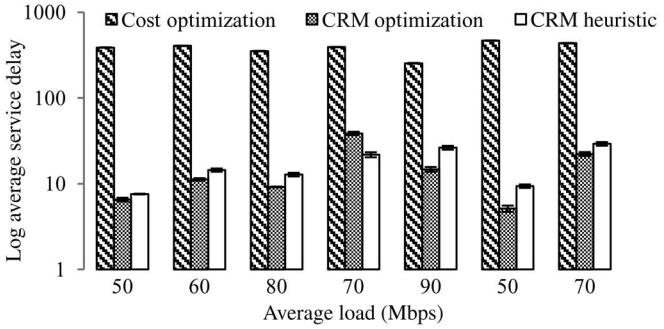



Fig. 7. CRM heuristic yields near optimal infrastructure delay. CRM heuristic outperforms optimization with an increase in number of service replications.

The major benefits of our joint optimization are presented in Fig. 7 with improvements in orders of magnitude in infrastructure delay when compared with cost optimization, with CRM heuristic following optimization results closely. The reduction in cumulative VM migrations in Fig. 8 and control plane overhead (Fig. 9) is attributed to the fact that a purist approach, trying to only minimize the number of service hosts, and is oblivious to the effects of minimization of the number of hosts. Our optimization model minimizes the number of service hosts while meeting QoS bounds, and ensures minimal VM migrations and control plane overhead. For example, in a purist approach, installing in nodes $m$, $n$ is the same as installing in nodes $i$, $j$ without regard to the infrastructure delay and reconfiguration overhead; whereas, in our approach, installing on nodes $m, n$ is not the same as installing on nodes $m, n$, though there is not a change in the number of service hosts, there will be immediate consequences on network reconfiguration and infrastructure delay.

The efficiency of the CRM heuristic, which essentially selects the configuration that minimizes VM migrations, by always operating on the POF, is illustrated in Fig. 10. Therefore, the final configuration selected *is* also an optimal configuration of the number of service hosts and the infrastructure delay, such that each $\psi_j^{t_i} \in \Psi^{t_i}$ is a Pareto optimal configuration that minimizes the infrastructure delay and number of service hosts.

Though Fig. 8 depicts the significant reduction in the number of VM migrations with heuristic and joint optimization, (Fig. 9) illustrates that heuristic performs the worst with respect

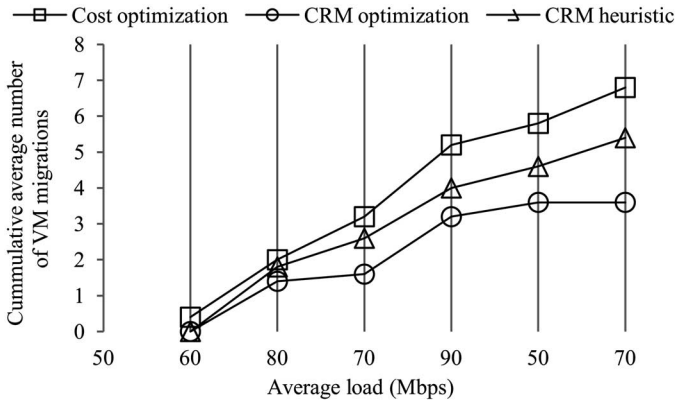



Fig. 8. CRM heuristic and optimization outperform purist cost optimization.

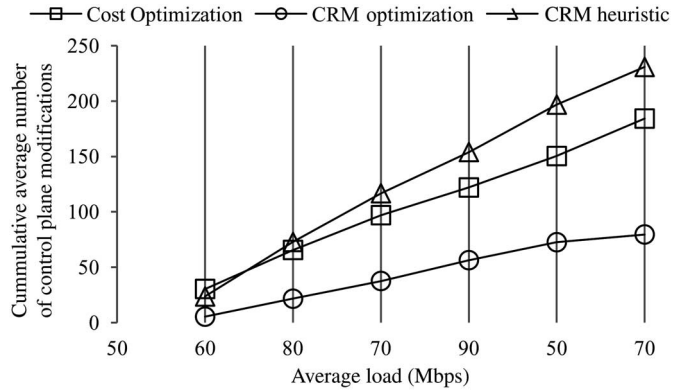



Fig. 9. CRM heuristic incurs highest control plane modifications due to fine grain load balancing.

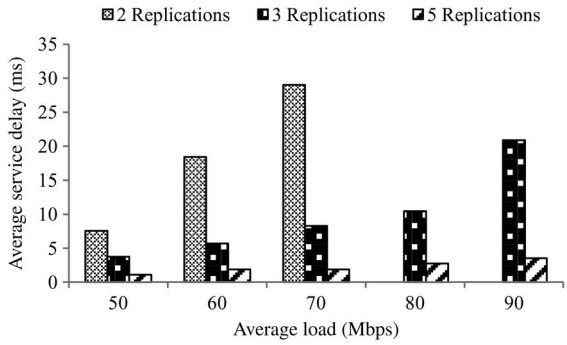



Fig. 10. Every $\psi_j^{t_i} \in \Psi^{t_i}$ is a Pareto optimal configuration w.r.t. number of service hosts and infrastructure delay.

to control plane modifications. This is attributed to our load-balancing technique employed in the heuristic that distributes a fixed percentage of demand across multiple paths until the demands are met. Each path accounts for numerous control plane modifications, as seen, in the high cost of control plane modifications. Our approach to increasing the utilization of the number of paths between a provider and a consumer reduces bottlenecks and potential starvation of other RSUs. This is in contrast to the naïve approach of selecting the shortest path and fully utilizing the capacity of a path(s).

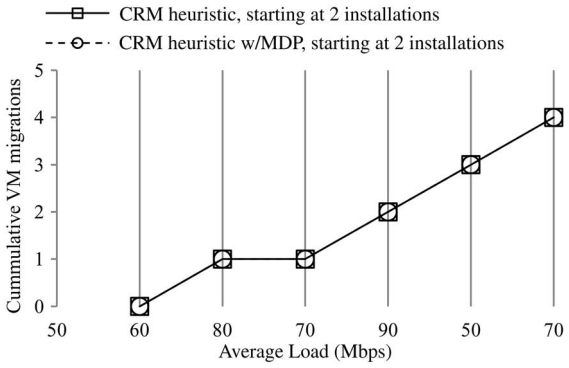



Fig. 11. VM migration comparison of CRM heuristic and CRM heuristic with MDP, when starting with a two installation configuration.

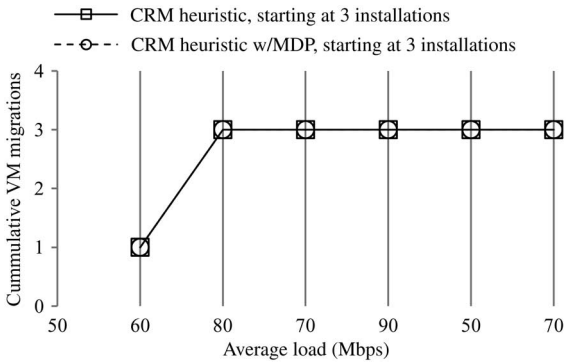



Fig. 12. VM migration comparison of CRM heuristic and CRM heuristic with MDP, when starting with a three installation configuration.

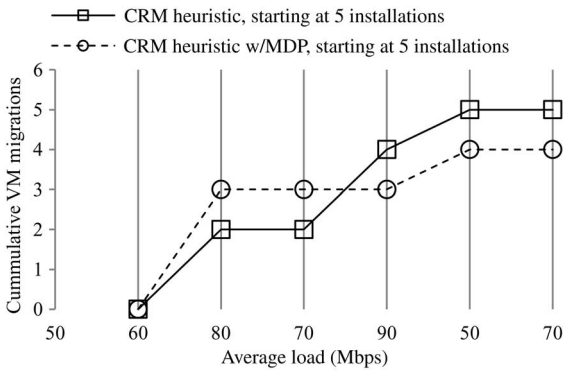



Fig. 13. VM migration comparison of CRM heuristic and CRM heuristic with MDP, when starting with a five installation configuration.

### A. Reinforcement Learning

In different scenarios of the same problem of configuration selection, we see that the heuristic alone performs as well as heuristic with reinforcement learning MDP, as illustrated in Figs. 11 and 12 in selecting configurations that yield minimum VM migrations in the long term. However, in Fig. 13, it is evident that MDP can envision the long-term benefit of choosing a higher VM migration cost at an average load 80 Mb/s and incurs lower cumulative VM migrations over the long run. This novel contribution of our work enables configuration selections that minimize the VM migrations over the long term for service providers hosting services on the RSU cloud. The MDP-enforced heuristic overcomes the myopic shortcomings of the CRM heuristic.

## VII. Conclusion

In this paper, we proposed an RSU cloud as a vehicular cloud for the computational and communication infrastructure supporting vehicle grid in the IoV. They make an integral component for vehicular clouds and its applications. RSU clouds consist of traditional RSUs and specialized RSUs containing microdatacenters. The RSU cloud is implemented using SDN, which can host services to meet OBU demands. In the event of the inherent dynamic demands, RSU clouds can be reconfigured to optimally meet the service demands. We study the effects of reconfiguration in SDN, by designing and implementing RSU cloud in Mininet. For real-world reconfiguration overhead analysis, we implemented stochastic switching for multipath in OpenFlow-enabled SDN. We have made our contribution for implementing stochastic switching in OvS available online [13]. We formally defined reconfiguration overhead, VM migrations, and control plane modifications and instigate the need for RSU CRM.

Our novel contribution is the architecture of RSU CRM and RSU microdatacenter. We model the CRM as a multiobjective optimization problem, for minimizing VM migrations, control plane overhead, number of service hosts, and infrastructure delay. We design a unique approach to solving the multiobjective, so that we are continually operating on the POF. We selected an optimal configuration, such that the VM migrations are minimized over time. We use reinforcement learning to select the configuration that minimizes VM migrations over the long run.

We illustrated how RSU CRM selects configurations that improve the infrastructure delay in orders of magnitude with optimal number of service hosts. Over time and in face of dynamic loads, the configurations are selected as part of a POF, of nondominated solutions. Any Pareto optimal configuration is a candidate that can optimally minimize VM migrations with respect to the infrastructure delay and the number of service hosts. To select the final configuration, we use our heuristic and reinforcement learning, to select configurations that may seem to immediately yield higher VM migrations but over the long term incur lower VM migrations.

Our future work includes minimizing control plane modifications by improving the load-balancing technique. We will extend this work to leverage the resources available in the mobile OBUs. Furthermore, an experimental at-scale analysis of CRM will be conducted on NSF Global Environment for Network Innovations (GENI) [32] testbed.

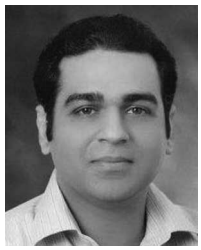
## References

[1] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for smart cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, Feb. 2014.

[2] N. Lu, N. Cheng, N. Zhang, X. Shen, and J. Mark, "Connected vehicles: Solutions and challenges," *IEEE Internet Things J.*, vol. 1, no. 4, pp. 289–299, Aug. 2014.

[3] M. Gerla, E.-K. Lee, G. Pau, and U. Lee, "Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds," in *Proc. IEEE World Forum Internet Things (WF-IoT)*, Seoul, Korea, 2014, pp. 241–246.

[4] R. Hussain, J. Son, H. Eun, S. Kim, and H. Oh, "Rethinking vehicular communications: Merging VANET with cloud computing," in *Proc. IEEE 4th Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, Taipei, China, 2012, pp. 606–609.
[5] E. Lee, E.-K. Lee, and M. Gerla, "Vehicular cloud networking: Architecture and design principles," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 148–155, Feb. 2014.
[6] K. Mershad and H. Artail, "Finding a STAR in a vehicular cloud," *IEEE Intell. Transport. Syst. Mag.*, vol. 5, no. 2, pp. 55–68, Apr. 2013.
[7] M. Amadeo, C. Campolo, and A. Molinaro, "Enhancing IEEE 802.11p/WAVE to provide infotainment applications in VANETs," *Ad Hoc Netw.*, vol. 10, no. 2, pp. 253–269, 2012.
[8] M. Bari *et al.*, "Dynamic controller provisioning in software defined networks," in *Proc. 9th Int. Conf. Netw. Serv. Manage. (CNSM)*, Zurich, Switzerland, 2013, pp. 18–25.
[9] S. Anja and W. Dargie, "Does live migration of virtual machines cost energy?" in *Proc. IEEE 27th Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, Barcelona, Spain, 2013, pp. 514–521.
[10] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.
[11] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proc. 9th ACM Workshop Hot Topics Netw.*, Monterey, CA, USA, 2010, pp. 19:1–19:6.
[12] A. Tootoonchian and Y. Ganjali, "HyperFlow: A distributed control plane for OpenFlow," in *Proc. Internet Netw. Manage. Conf. Res. Enterp. Netw. (INM/WREN'10)*, San Jose, CA, USA, 2010.
[13] M. Salahuddin, *Stochastic Switching Using Open vSwitch in Mininet*, GitHub, 18 Feb. 2014 [Online]. Available: https://github.com/saeenali/openvswitch/wiki/Stochastic-Switching-using-Open-vSwitch-in-Mininet, accessed on Apr. 18, 2014.
[14] Traffic Engineering and Operations Office, "Florida Department of Transportation," Florida Department of Transportation [Online]. Available: http://www.dot.state.fl.us/trafficoperations/its/projects_deploy/cv/connected_vehicles-wc.shtm, accessed on Apr. 18, 2014.
[15] B. Pfaff *et al.*, "Extending networking into the virtualization layer," in *Proc. 8th ACM Workshop Hot Topics Netw.*, New York, NY, USA, 2009.
[16] M. Abuelela and S. Olariu, "Taking VANET to the clouds," in *Proc. 8th Int. Conf. Adv. Mobile Comput. Multimedia (MoMM '10)*, Paris, France, 2010, pp. 6–13.
[17] Y. Qin, D. Huang, and X. Zhang, "VehiCloud: Cloud computing facilitating routing in vehicular networks," in *Proc. IEEE 11th Int. Conf. Trust Security Privacy Comput. Commun. (TrustCom)*, Liverpool, U.K., 2012, pp. 1438–1445.
[18] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proc. IEEE INFOCOM*, San Diego, CA, USA, 2010, pp. 1–9.
[19] D. Wu, J. He, Y. Zeng, X. Hei, and Y. Wen, "Towards optimal deployment of cloud-assisted video distribution services," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 10, pp. 1717–1728, Sep. 2013.
[20] J. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint VM placement and routing for data center traffic engineering," in *Proc. IEEE INFOCOM*, Orlando, FL, USA, 2012, pp. 2876–2880.
[21] F. Chang, R. Viswanathan, and T. Wood, "Placement in clouds for application-level latency requirements," in *Proc. IEEE Int. Conf. Cloud Comput.*, Honolulu, HI, USA, 2012, pp. 327–335.
[22] K. Tran and N. Agoulmine, "Adaptive and cost-effective service placement," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Houston, TX, USA, 2011, pp. 1–6.
[23] A. Sailer, M. Head, A. Kochut, and H. Shaikh, "Graph-based cloud service placement," in *Proc. IEEE Int. Conf. Services Comput.*, Miami, FL, USA, 2010, pp. 89–96.
[24] A. Anttonen and A. Mammela, "Interruption probability of wireless video streaming with limited video lengths," *IEEE Trans. Multimedia*, vol. 16, no. 4, pp. 1176–1180, May 2014.
[25] T. Luan, L. Cai, and X. Shen, "Impact of network dynamics on user's video quality: Analytical framework and QoS provision," *IEEE Trans. Multimedia*, vol. 12, no. 1, pp. 64–78, Jan. 2010.
[26] G. Curry and R. Feldman, *Manufacturing Systems Modeling and Analysis*, 2nd ed. New York, NY, USA: Springer, 2011.
[27] iTouchMap.com. (2014). *Latitude and Longitude of a Point* [Online]. Available: http://itouchmap.com/latlong.html, accessed on Apr. 19, 2014.
[28] Boulter, *Coordinate Distance Calculator* [Online]. Available: http://boulter.com/gps/distance/, accessed on Apr. 18, 2014.
[29] R. Marler and J. Arora, "Survey of multi-objective optimization methods for engineering," *Structural Multidisciplinary Optim.*, vol. 26, no. 6, pp. 369–395, 2004.
[30] M. Berkelaar, K. Eikland, and P. Notebaert, "Open source (mixed-integer) linear programming system (lp_solve)," *GNU LGPL (Lesser General Public License)*, 2004.
[31] I. Chadès, M.-J. Cros, F. Garcia, and R. Sabbadin, *Markov Decision Process (MDP) Toolbox*. INRA, 2009.
[32] Raytheon BBN Technologies and National Science Foundation (NSF). (2014). *GENI (Global Environment for Network Innovations)* [Online]. Available: http://www.geni.net, accessed on Oct. 31, 2014.



**Mohammad Ali Salahuddin** (S'09–M'15) received the B.S. degree in computer science from the University of Karachi, Karachi, Sindh, Pakistan, in 1999, the M.S. degree in computer science from the Shaheed Zulfiqar Ali Bhutto Institute of Science and Technology, Karachi, Pakistan, in 2001, and the M.S. and Ph.D. degrees in computer science from Western Michigan University, Kalamazoo, MI, USA, in 2003 and 2014, respectively.

Currently, he is a Postdoctoral Research Fellow with the Department of Computer Science, University of Quebec, Montreal, QC, Canada. His research interests include wireless sensor networks, indoor and outdoor localization techniques, QoS and QoE in VANETs (WAVE, IEEE 802.11p and IEEE 1609.4) and service placement and routing for cloud resource management.

Dr. Salahuddin serves as a Technical Program Committee member and Reviewer for IEEE publications and conferences.

**Ala Al-Fuqaha** (S'00–M'04–SM'09) received the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Missouri, Columbia, MO, USA, in 1999 and 2004, respectively.

Currently, he is an Associate Professor and Director of the NEST Research Laboratory, Computer Science Department, Western Michigan University, Kalamazoo, MI, USA. His research interests include intelligent network management and planning, QoS routing and performance analysis and evaluation of software-defined networks, and VANETs.

Dr. Al-Fuqaha currently serves on the Editorial Board for *Security and Communication Networks Journal*, *Wireless Communications and Mobile Computing Journal*, *Industrial Networks and Intelligent Systems Journal*, and *International Journal of Computing and Digital Systems*. He has also served as a Technical Program Committee member and Reviewer for many international conferences and journals.

**Mohsen Guizani** (S'85–M'89–SM'99–F'09) received the B.S. degree (with distinction), M.S. degree in electrical engineering, and M.S. and Ph.D. degrees in computer engineering from Syracuse University, Syracuse, NY, USA, in 1984, 1986, 1987, and 1990, respectively.

He is currently a Professor and Associate Vice President of Graduate Studies with Qatar University, Doha, Qatar. Previously, he served as the Chair of the Computer Science Department, Western Michigan University, from 2002 to 2006 and Chair of the Computer Science Department, University of West Florida, from 1999 to 2002. He also served in academic positions with the University of Missouri–Kansas City, University of Colorado–Boulder, Syracuse University, and Kuwait University. He has authored nine books and more than 300 publications in refereed journals and conferences. He has been a Guest Editor for a number of special issues in IEEE journals and magazines. His research interests include wireless communications and mobile computing, computer networks, mobile cloud computing and smart grid.

Dr. Guizani currently serves on the Editorial Boards of many international technical journals. He was the Founder and Editor-in-Chief of *Wireless Communications and Mobile Computing*. He also served as a Member, Chair, and General Chair of a number of conferences. He was the Chair of the IEEE Communications Society Wireless Technical Committee and Chair of the TAOS Technical Committee. He served as the IEEE Computer Society Distinguished Speaker from 2003 to 2005. He is the member of ASEE and Senior Member of ACM. He was the recipient of the Best Teaching Assistant Award for two consecutive years at Syracuse University (1988 and 1989).