Wang *et al.*
EURASIP Journal on Advances in Signal Processing (2022) 2022:32
https://doi.org/10.1186/s13634-022-00862-7

EURASIP Journal on Advances in Signal Processing

RESEARCH

Open Access

Joint optimization of dynamic resource allocation and packet scheduling for virtual switches in cognitive internet of vehicles

Yang Wang[1], Xiong Wang[1*], Zhuobin Huang[1], Wei Li[2] and Shizhong Xu[1]

*Correspondence: wangxiong@uestc.edu.cn
[1] University of Electronic Science and Technology of China, Chengdu, China
Full list of author information is available at the end of the article

Abstract
The rapidly evolving machine learning technologies have reshaped the transportation system and played an essential role in the Cognitive Internet of Vehicles (CIoV). Most of the cognitive services are computation-intensive or storage-intensive, and thus they are usually deployed in edge or cloud data centers. In today's data center networks, the virtual machines hosted in a server are connected to a virtual switch responsible for forwarding all packets for the cognitive services deployed on the virtual machines. Therefore, the virtual switches will become a performance bottleneck for cognitive services without an efficient resource allocation and data scheduling strategy. However, the highly dynamic characteristics of cognitive services make the resource allocation and packet scheduling problem for virtual switches surprisingly challenging. To guarantee the performance of cognitive services, we investigate the joint optimization problem of dynamic resource allocation and packet scheduling for virtual switches. We first model the joint optimization problem of dynamic resource allocation and packet scheduling for virtual switches as a mathematical optimization problem. Then, we analyze the problem with Lyapunov Optimization Framework and derive efficient optimization algorithms with performance tradeoff bounds. At last, we evaluate these algorithms on a testbed and a network-wide simulation platform. Experiment results show that our algorithms outperform other designs and meet the theoretical performance bound.

Keywords: Cognitive Internet of Vehicle, Machine Learning, Resource Allocation, Virtual Switch

1 Introduction

The emerging of Machine Learning has enabled a technical evolution of transportation systems in recent years. Cognitive Internet of Vehicles (CIoV) incorporates rapidly evolving Deep Neural Networks, Deep Reinforcement Learning, and Graph Neural Networks to achieve intelligent cognition, resource optimization, traffic prediction, and decision-making on the traditional Internet of Vehicles (IoV). CIoV integrates machine learning techniques into Cognitive Signal Processing [1–8], Image Processing [9, 10], Autonomous Driving [11, 12], and Traffic Congestion Monitoring [13–15]. These applications offload machine learning workloads to the CIoV framework.

We demonstrate the architecture of CIoV framework in Fig. 1. The CIoV architecture consists of cognitive engines, cloud data centers, edge data centers, and vehicles. The cognitive engine is a set of software systems running in the cloud data center and edge data center, which provides software runtime environment, service management, and task scheduling for the above machine learning tasks. Cloud and edge data centers offer centralized and distributed hardware environments for cognitive engines, respectively. Cloud data centers centrally host high-performance servers and accelerators for machine learning model training and inference. Edge data centers are deployed at the edge of networks and geographically distributed in the transportation system. Servers in cloud data centers are connected through the data center networks, and servers in edge data centers are connected to data centers via high-speed backhaul links. ❶ All data transmissions of cognitive services rely on the CIoV network system. The cognitive service request generated by a vehicle is first received by an edge data center server. ❷ For emerging and latency-sensitive service requests, edge data centers handle them in local virtual machines and forward other requests to cloud data centers. ❸ When a service request demands a larger amount of computation and storage resources, the service request needs to traverse through data center networks and acquire resources on other servers. Then, the request is forwarded to a server through the data center network, which hosts the cognitive service in a virtual machine. After that, the cognitive service replies the request along the reverse path to the vehicle.

All cognitive service requests and responses are transmitted through the edge or cloud data center networks. The edge and cloud data center networks play an irreplaceable role in CIoV. Although high-performance ASIC-based switches and sophisticated network scheduling algorithms are well equipped in these data center networks to handle dynamic and urgent requests. A critical missing component, the software-based virtual switch, becomes a hindrance to system optimization. Servers in edge and cloud data centers typically host all machine learning services in virtual machines through a virtualized environment. In Fig. 2, we illustrate the architecture of a virtualized server in edge and cloud data centers. The virtualized server hosts multiple virtual machines (VMs), a hypervisor, and a vswitch from the application layer to the system layer. Cognitive service providers rent edge and cloud data centers' compute, storage, and bandwidth resources for deploying machine learning services in this architecture. And the cloud manager virtualizes the hardware resources and manages the virtual machines through

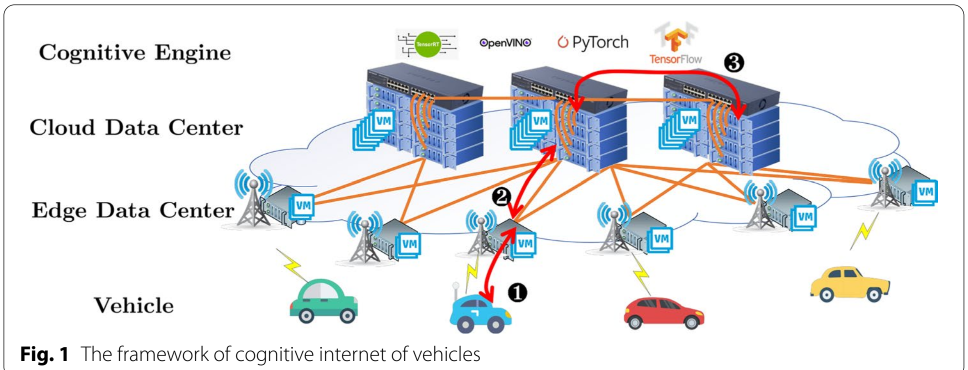


Fig. 1 The framework of cognitive internet of vehicles

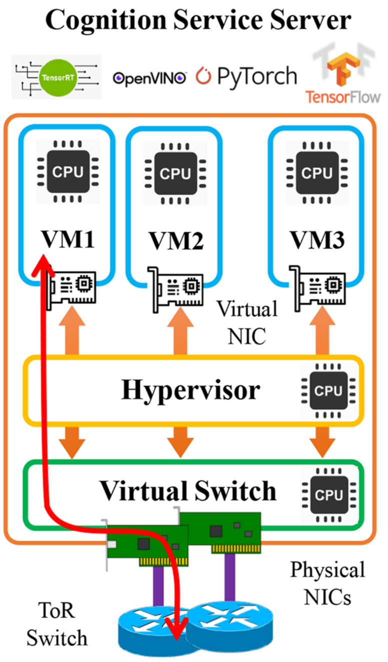


Fig. 2 The architecture of a virtualized server

the hypervisor. Furthermore, the cloud manager usually deploys a vswitch on each virtualized server to provide numerous network services, such as Access-Control List (ACL), Quality of Service (QoS), and Virtual Private Cloud (VPC), etc. The vswitch is a system software that takes over the physical Network Interface Cards (NICs) plugged into the server and virtual (software) NICs created by virtual machines. According to the cloud manager's configuration, the vswitch forwards packets between these NICs by forwarding rules. All cognitive service requests and replies are forwarded by virtual switches, which affects the quality of service of each cognitive service. The virtual machines, the hypervisor, and the vswitch share hardware resources on a server. The virtual machines occupy most of the CPU cores on the server and bring significant revenue for the cloud manager. The hypervisor occasionally takes up a small number of CPU cores to marinate the virtualization system. The vswitch takes up a considerable amount of CPU resource to provide line-rate packet forwarding for the VMs. In addition, with the evolution of data center networks, the number of NICs and NIC bandwidth are growing substantially. Thus, the cloud manager has to allocate more CPU resource to the vswitch. However, allocating more CPU resource to the vswitch reduces the revenue of cloud services and increases operational expenses (OpEx).

To tackle this problem, we need to efficiently allocate resource to each vswitch in edge and cloud data centers. However, the rapid development of edge and data center networks in CIoV also creates enormous challenges in resource allocation for vswitches. On the one hand, the cloud manager can hardly predict the users' requirements accurately in real-time. Virtual machines deployed by cognitive service providers may receive and transmit data in an arbitrary time. For example, cognitive service requests may be generated at any time when vehicles pass through an edge network node, which makes requests unpredictable. Thus, it is difficult for the cloud manager to accurately predict when virtual machines receive requests and reply data. Although it is possible to summarize traffic patterns statistically and allocate resource to vswitches based on prior knowledge, the prediction and resource allocation is coarse-grained and inaccurate, which leads to improper resource allocation in ICoV.

On the other hand, the emerging hardware architecture of virtualized servers in data centers also complicates the dynamic resource allocation problem for vswitches. In recent years, the cloud manager will plug multiple physical NICs on each server to increase physical network bandwidth, achieve load balance, and improve server access reliability. These physical NICs are connected to Top-of-Rack (ToR) switches, and the vswitch will take over the NICs in the same server. However, deploying multiple NICs on each server will further complicate the dynamic resource allocation of vswitches in data centers. First, the packet scheduling decisions of a vswitch should be made based on the amount CPU resource allocated to the vswitch. The vswitch executes packet scheduling on CPU cores in runtime for classifying and forwarding each packet to NICs. The packet scheduling decision determines which NIC should a packet be forwarded to, and it impacts the incoming workload of a NIC and the number of allocated CPU cores on a NIC. For example, the vswitch could schedule more packets to a NIC allocated with more CPU cores to reduce CPU idling. Therefore, to achieve better packet delivery performance, we need to jointly optimize the dynamic resource allocation problem and the packet scheduling problem. Second, to enhance the packet forwarding throughput for cognitive services with high QoS requirements, the resource allocation and packet scheduling strategies should be made in a very short time according to the dynamic traffic pattern, which makes the joint optimization problem more difficult.

To address the above challenges, we jointly model the dynamic vswitch resource allocation problem and the packet scheduling problem in CIoV. The model jointly optimizes the resource allocation strategy for each vswitch NIC and the packet scheduling strategy for each packet in each time slot. To solve the joint optimization problem efficiently, we propose dynamic resource allocation and packet scheduling algorithms for vswitches based on Lyapunov optimization.

The main contributions of this paper can be summarized as follows.

- We model the problem into an optimization problem, which minimizes the time expenditure of resource allocation on vswitches under resource constraints and network constraints.
- We apply the Lyapunov Optimization Framework [16–19] to transform the problem into a discrete-time queueing system, which converts the solution of the optimization problem into a queue stability problem. Lyapunov optimization is an optimization framework for dynamic systems, and it can make decisions using dynamic queue backlogs (lengths) to stabilize the entire system in real-time. It does not require prior knowledge about the input data and only acquires queue backlogs to optimize the system in real-time.
- We derive the performance tradeoff $O(1/V)$ and $O(V)$ between the time-average expectation of resource allocation and queue backlogs and design low-complexity distributed scheduling algorithms to solve the problem. The factor V is a positive penalty factor, which indicates the relation between the system stability and the

object of the problem. Lyapunov optimization framework can also provide a performance bound between the optimization object and queue backlogs, enabling us to balance the stability of the system and the optimization object. We also propose extended algorithms to improve queue backlogs without breaking the performance tradeoff.

- We evaluate our packet scheduling and resource allocation algorithms on virtualized servers and network-wide simulations. Experiment results show that our algorithms outperform other methods in real-world scenarios and workloads.

2 Related works

Cognition IoV empowers Artificial Intelligence with IoV through integrating data mining, reinforcement learning, and deep learning in its architecture. CIoV frameworks usually provide cognitive services by deploying services in edge and cloud data centers. Cloud data centers provide high-powered computation and deploy more accurate machine learning models. It also allows global resource optimization to provide more efficient and dynamic cognition services. On the other hand, edge data centers handle a part of lightweight computation and storage tasks, which reduce transmission latencies and network congestion. Chen et al. [20] first introduce a Cognition Layer to enhance intelligence in IoV architecture. Cognition Layer handles vehicle tasks, including traffic condition quality analysis, driver behavior modeling, etc. The Cognition Layer is deployed in edge and cloud data centers. The cognition layer processes latency-sensitive tasks in edge data centers. Otherwise, it forwards nonlatency-sensitive tasks or computation-intensive tasks to cloud data centers. Qian et al. [21] present a three-layer CIoV design that deploys cognitive services in a Cognitive Engine Layer. The Cognitive Engine Layer handles cognition requests in data centers, which are generated from Road Side Units (RSU) and vehicles. Hasan et al. [22] propose a five-layer CIoV model, which includes an Edge Computing and Data Pre-Processing Layer and a Cognition and Control Layer. The Edge Computing and Data Pre-Processing Layer is deployed in edge data centers and collects data through intra-vehicular and inter-vehicular communications. It forwards the preprocessed data to the Cognition and Control Layer. The Cognition and Control Layer is deployed in cloud data centers and provides data storage, computing, and machine learning-based data processing of cognition services. Lu et al. [11] design a CIoV architecture for Autonomous Driving. It preprocesses real-time cognitive requests in Fog Layer and offloads others requests to Cloud Layers. Therefore, all above CIoV architectures rely on networks between edge and cloud data centers, and the optimization of the network resource allocation is essential in CIoV.

The data networks of edge and cloud carry all cognitive service requests and responses, which plays a critical role in CIoV. However, as a key component in data center networks, software-based vswitches require a significant amount of computing resources on servers to provide high-performance packet processing. Some works have studied the dynamic resource allocation of a single vswitch. Shenango [23] designs a congestion detection algorithm to track the duration time of each packet in the ingress queue of vswitch as an indicator of core allocation. It reduces the granularity of core allocation time intervals to microsecond timescales with a polling kernel module. Arachne [24] provides a dynamic CPU core allocation algorithm for vswitch, which scales up the number of CPU core allocation by CPU load factor and scales down by hysteresis factor. Snap [25] allocates CPU cores to vswitch through a centralized resource controller, which supports allocating CPU cores to reduce the tail latency of packet forwarding for vswitches. In contrast, our network model and algorithms can dynamically allocate CPU cores to virtual switches based on the analysis of Lyapunov optimization framework and provide performance bounds on queue length and resource allocation.

On the other hand, network-wide vswitch resource allocation also reduces the resource overhead of cognitive services and improves the performance of cognitive services from a global perspective. A lot of researches have investigated network-wide vswitch resource optimization on edge and cloud data centers. Yang et al. [26] joint optimize the vswitch deployment and network routing to improve the network load balancing in Software Defined Networks (SDNs). It formulates the optimization problem as an Integer Linear Problem (ILP) and solves the problem with approximation algorithms. However, the algorithm only optimizes resource allocation on virtual switches based on statically, and all cognitive service requests are dynamic and hard to predict and collect in real-time in CIoV networks, which prevents the deployment of these methods in real-world scenarios. Yang et al. [27, 28] optimize the resource allocation of virtual switches and network embedding problems together in an online model and design a heuristic algorithm to solve the relaxed problem. The real-time resource allocation algorithm requires solving the resource allocation problem for each vswitch in a centralized network controller. CIoV networks are large-scale and geographically distributed, which complex the solution of the problems and introduce significant transmission latency. Instead, our virtual switch resource allocation algorithm can dynamically allocate resources to virtual switches without prior knowledge. Moreover, we can decompose the network-wide virtual switch resource allocation into distributed algorithms through Lyapunov Optimization Framework. And each virtual switch only executes the algorithm on its own server to adjust the local resource allocation, which does not require a centralized resource scheduling.

3 Network model and performance analysis

In this section, we propose a mathematical model of the dynamic vswitch resource allocation problem and the packet scheduling problem in CIoV. It optimizes the time expenditure of resource allocation on vswitches and packet scheduling for each network connection. Besides, we apply Lyapunov Optimization Framework [16–18] to transform the joint optimization problem into a discrete-time queuing system. It converts the model constraints to virtual queues and defines the Lyapunov Function to represent the stabilization of the system. To solve the optimization problem and to stabilize the queue system, we minimize the drift of the Lyapunov Function and derive the optimization policies from the function. In addition, we deduce the performance tradeoff of the policies from the theoretical analysis, and it shows that the time-average expectation of vswitch resource allocation and queue backlogs conform to $[O(1/V), O(V)]$ tradeoff.

3.1 Network model

We provide an instance of the dynamic vswitch resource allocation problem and the packet scheduling problem in CIoV in Fig. 3. All cognitive services are deployed in virtual machines with virtualized servers, and service requests and responses can be considered as network connections in this model. The data center operator manages the system in slotted time. ➊ The vswitch in the server s_1 creates a vswitch virtual port u for the virtual machine VM1 and binds NICs as vswitch physical ports. The vswitch in server s_2 also creates a vswitch virtual port v for the virtual machine VM2 and binds NICs as vswitch physical ports. ➋ Each vswitch port p enqueues received packets to a Receive Queue (Rx-Queue), and a packet classifier dequeues packets at the rate of $f(r_p(t))$. The vswitch resource allocation decision variable $r_p(t)$ represents that the cloud operator allocates resource $r_p(t)$ to vswitch port p on slot t. The performance function $f(\cdot)$ describes the relationship between resource allocation and packet processing rate, which can be measured on servers. The cache miss and the data bus bandwidth limit the forwarding performance of vswitch [29–31], which conduct $f(\cdot)$ to a nondecrease concave function [32]. The cloud operator allocates CPU resource to port u by setting the vswitch resource allocation variables $r_u(t)$ to port u for forwarding packets on the server s_1.

In the time slot t, ➍ VM1 creates a connection m to VM2 that requires networks bandwidth $A_m(t)$. The required bandwidth $A_m(t)$ is a stochastic variable and i.i.d over slots. The cloud operator should not only allocate the resource to vswitches but also decide packet scheduling decisions on this slot. For the packet scheduling decisions, the cloud operator needs to choose a forwarding path that connects s_1 to s_2 for the connection m. In this example, the cloud operator selects the forwarding path $y_m^{ij}(t)$, which connects the port i in s_1 to port j in s_2. The connection occupies the bandwidth $A_m(t)$ on the forwarding path (i, j). The capacity of the forwarding path is β_{ij}. ➎ The vswitch in server s_2 handles the connection m on port j and enqueues packets in the queue $Q_j(t)$. ➏ The cloud operator also needs to allocate the resource $r_j(t)$ to port j for forwarding

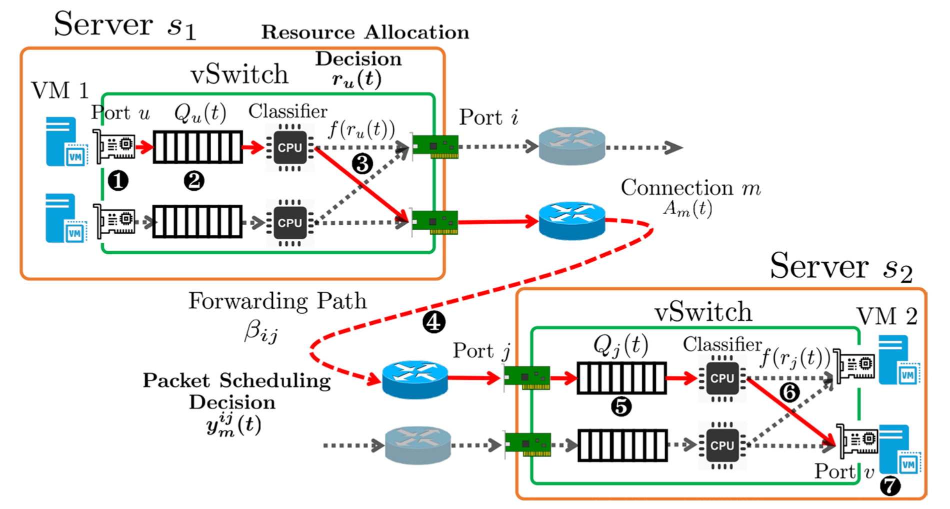


Fig. 3 An example of packet forwarding in edge and cloud data centers

packets. ❼ VM2 receives packets of the connection m from vswitch port v, and handles the cognition service request. The cognition service response also can be presented as a similar process. Without loss of generality, we model all bidirectional service requests and responses as two connections (Table 1).

The packets in connection m transverse through port u and are enqueued in Rx-Queue $Q_u(t)$. The classifier dequeues packets in Rx-Queue $Q_u(t)$ and forwards packets to port i. The packets are forwarded through the path (i,j) in DCN. After being received by port j, the packets are enqueued in Rx-Queue $Q_j(t)$ and forwarded to port v. We define the vswitch virtual port backlog update Eq. (1) and the vswitch physical port backlog update Eq. (2), which describe the backlog of vswitch queues in each slot.

$$Q_u(t+1) = \max\left(Q_u(t) - f(r_u(t)) + \sum_{m\in M} A_m(t) z_m^u(t), 0\right), \forall u \in P_{\text{VM}} \tag{1}$$

$$Q_j(t+1) = \max\left(Q_j(t) - f(r_j(t)) + \sum_{m\in M}\sum_{i\in P_{\text{NIC}}}\sum_{u\in P_{\text{VM}}}\sum_{v\in P_{\text{VM}}} A_m(t) z_m^u(t) z_m^v(t) y_m^{ij}(t), 0\right) \quad \forall j \in P_{\text{NIC}} \tag{2}$$

To optimize the dynamic vswitch resource allocation problem and the packet scheduling problem on time average, we define the optimization problem (3–12). The object of the optimization (3) is to minimize the time-average expectation of resource allocation on vswitches, when the resource capacity constraint (4), the network path capacity constraint (5), the flow conservation constraint (6), the single path constraint (7, 8) and the queue stable constraint (9) hold. The resource allocation constraint (4) restricts that the

Table 1 Key notations

Symbol	Description
M	The set of all connections
$A_m(t)$	A stochastic variable, connection m requires bandwidth $A_m(t)$ on time slot t, which is i.i.d over slots
S	The set of all virtualized servers
P^s_{NIC}	All vswitches physical ports in server $s, s \in S$
P^s_{VM}	All vswitches virtual ports in server $s, s \in S$
P^s	All vswitches ports $P^s_{\text{VM}} \bigcup P^s_{\text{NIC}}$ in server $s \in S$
P_{VM}	All vswitches virtual ports $\bigcup_{s\in S} P^s_{\text{VM}}$
P_{NIC}	All vswitches physical ports $\bigcup_{s\in S} P^s_{\text{NIC}}$
P	All vswitches ports $P_{\text{VM}} \bigcup P_{\text{NIC}}$
$Q_u(t), Q_j(t)$	Queue backlogs of vswitch port u and port j
$z^u_m(t), z^v_m(t)$	Connection m transmits through vswitch port $u, v \in P$ on time slot t
β_{ij}	The bandwidth of network path $(i,j), i,j \in P_{\text{NIC}}$
$r_u(t), r_j(t)$	The vswitch resource allocation decision variables, which denotes the resource allocated to vswitch port $u \in P^s_{\text{NIC}}, j \in P^s_{\text{VM}}$ on time slot $t, s \in S$
$y^{ij}_m(t)$	The packet scheduling decision variable, which denotes the connection m transverses through the network path (i,j) on time slot t
$f(\cdot)$	A nondecrease, concave function $f(\cdot)$ describes the relationship between the resource allocation on a vswitch port and the forwarding rate of the port
V	The tradeoff factor indicates the relation between the system stability and the object of the problem, $V > 0$

resource allocation for all ports of a vswitch is not greater than the capacity of the server. The forwarding path capacity constraint (5) guarantees that the traffic on each forwarding path does not exceed the bandwidth of the path. The flow conservation constraint (6) ensures that the incoming traffic is equal to the outgoing network traffic for each forwarding path. The single path constraint (7–8) indicates that the network does not support multi-path. The queue stable constraint (9) states that the queue backlog will not increase to infinite. Queue mean rate stable can be defined as $\lim_{t\to\infty} \frac{\mathbb{E}\{|Q(t)|\}}{t} = 0$. We assume the requirement of each connection and the resource allocation in each server are bounded (10–11). The binary variables $z_m^u(t), z_m^v(t)$ represent that the connection m traverses through the vswitch virtual ports u and v. The binary decision variable $y_m^{ij}(t)$ indicates that the connection m is scheduled to traverse through the forwarding path (i, j). Define the time average of a stochastic variable as $\overline{x} = \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} x(\tau)$.

$$\min \sum_{p\in P_{\text{NIC}}\cup P_{\text{VM}}} \overline{r_p} \tag{3}$$

$$\text{s.t.} \quad \sum_{u\in P_{\text{VM}}^s} \overline{r_u} + \sum_{j\in P_{\text{NIC}}^s} \overline{r_j} \le R_s, \quad \forall s \in S \tag{4}$$

$$\sum_{m\in M} \sum_{u\in P_{\text{VM}}} \sum_{v\in P_{\text{VM}}} \overline{A_m z_m^u z_m^v y_m^{ij}} \le \beta_{ij}, \quad \forall i,j \in P_{\text{NIC}} \tag{5}$$

$$\sum_{m\in M} \sum_{u\in P_{\text{VM}}} \overline{A_m z_m^u y_m^{ij}} = \sum_{m\in M} \sum_{v\in P_{\text{VM}}} \overline{A_m z_m^v y_m^{ij}}, \quad \forall i,j \in P_{\text{NIC}} \tag{6}$$

$$\sum_{i\in P_{\text{NIC}}} \sum_{j\in P_{\text{NIC}}} \overline{y_m^{ij}} \le 1, \quad \forall m \in M \tag{7}$$

$$\forall s \in S, \quad \text{if} \quad i,j \in P_{\text{NIC}}^s, \quad y_m^{ij}(t) = 0 \tag{8}$$

$$\text{Queue } Q_p(t) \text{ is mean rate stable,} \quad \forall p \in P_{\text{NIC}} \cup P_{\text{VM}} \tag{9}$$

$$0 \le A_m(t) \le A_m^{\max} \tag{10}$$

$$0 \le r_p(t) \le R_s, \quad p \in P_{\text{NIC}}^s \cup P_{\text{VM}}^s \tag{11}$$

$$z_m^u(t), \quad z_m^v(t), \quad y_m^{ij}(t) \in \{0, 1\} \tag{12}$$

3.2 Transformed problem

The constraints of the above optimization can also be transformed as queue backlog update equations, which have similar forms as the queue update equation of vswitch ports. We transform the constraints (4–7) to *virtual queues* (13–16). And if a virtual queue is mean rate stable, it denotes that the corresponding constraint holds in the optimization problem.

$$H_s(t+1) = \max\left(H_s(t) + \sum_{u\in P^s_{\mathrm{VM}}} r_u(t) + \sum_{j\in P^s_{\mathrm{NIC}}} r_j(t) - R_s, 0\right), \quad \forall s \in S \tag{13}$$

$$G_{ij}(t+1) = \max\left(G_{ij}(t) + \sum_{m\in M}\sum_{u\in P_{\mathrm{VM}}}\sum_{v\in P_{\mathrm{VM}}} A_m(t)z^u_m(t)z^v_m(t)y^{ij}_m(t) - \beta_{ij}, 0\right), \quad \forall i,j \in P_{\mathrm{NIC}} \tag{14}$$

$$\begin{aligned} N_{ij}(t+1) = N_{ij}(t) &+ \sum_{m\in M}\sum_{u\in P_M} A_m(t)z^u_m(t)y^{ij}_m(t) \\ &- \sum_{m\in M}\sum_{v\in P_{\mathrm{VM}}} A_m(t)z^v_m(t)y^{ij}_m(t), \quad \forall i,j \in P_{\mathrm{NIC}} \end{aligned} \tag{15}$$

$$K_m(t+1) = \max\left(K_m(t) + \sum_{i\in P_{\mathrm{NIC}}}\sum_{j\in P_{\mathrm{NIC}}} y^{ij}_m(t) - 1, 0\right) \tag{16}$$

We define a vector (17) which includes all vswitches queues and virtual queues. We can define the Lyapunov function (18) and the Lyapunov drift (19) from the vector. The Lyapunov function is the sum of the squares of all vswitches queue backlogs and virtual queue backlogs, which are always nonnegative. If the Lyapunov function is a large number, it denotes that at least one queue in the vector $\mathbf{\Theta}(t)$ is large. It represents that too many resources are allocated to the virtual switches, or some constraints are not satisfied in the optimization problem. We define the Lyapunov drift as the difference of the Lyapunov function with the previous time slot.

$$\mathbf{\Theta}(t) \triangleq [Q_u(t), Q_j(t), H_s(t), G_{ij}(t), N_{ij}(t), K_m(t)] \tag{17}$$

$$\begin{aligned} L(\mathbf{\Theta}(t)) = \frac{1}{2}\Bigg(&\sum_{u\in P_{\mathrm{VM}}} Q_u(t)^2 + \sum_{j\in P_{\mathrm{NIC}}} Q_j(t)^2 + \sum_{s\in S} H_s(t)^2 \\ &+ \sum_{i\in P_{\mathrm{NIC}}}\sum_{j\in P_{\mathrm{NIC}}} G_{ij}(t)^2 + \sum_{i\in P_{\mathrm{NIC}}}\sum_{j\in P_{\mathrm{NIC}}} N_{ij}(t)^2 + \sum_{m\in M} K_m(t)^2\Bigg) \end{aligned} \tag{18}$$

To reduce the expenditure of resource allocation and to stabilize the system, we apply Lyapunov Optimization Framework [16–19] which minimizes the drift-plus-penalty (20) on each slot. The drift-plus-penalty is defined as the weighted sum of the Lyapunov drift and the object of the optimization problem. The factor V is a positive penalty factor, which indicates the relation between the system stability and the object of the problem. Applying the fact that if $a = max(b, 0)$, then $a^2 \leq b^2$ to the definition of queues (1–2, 13–16), we obtain the inequation (21) which is the bound of the drift-plus-penalty.

Because these stochastic variables are bounded (10–11), B is a constant independent from the factor V, which contains the remaining square terms of the inequation (21).

$$\Delta(\boldsymbol{\Theta}(t)) \triangleq \mathbb{E}[L(\boldsymbol{\Theta}(t+1)) - L(\boldsymbol{\Theta}(t))|\boldsymbol{\Theta}(t)] \tag{19}$$

$$\Delta(\boldsymbol{\Theta}(t)) + V \sum_{p\in P_{\mathrm{NIC}}\cup P_{\mathrm{VM}}} r_p(t) \tag{20}$$

To minimize the drift-plus-penalty, we minimize the right side of the inequation (21) on each slot, which could deduce the vswitch resource allocation problem (22–23) and packet scheduling problem (24).

According to the above analysis, we can design distributed resource allocation policies for the dynamic vswitch resource allocation problem and the packet scheduling problem. On each slot, the vswitch on each server can solve the optimization problems (22–23) independently to allocate resources for each vswitch virtual port and vswitch physical port. The cloud operator solves the optimization problem (24) to select the physical path for each connection. In addition, the resource allocation and packet scheduling only depend on vswitches queue backlogs and virtual queue backlogs on the current slots that will not require prior information about the distribution of the traffic.

$$\begin{aligned}
\Delta(\Theta(t)) &+ V \sum_{p\in P_{\mathrm{NIC}}\cup P_{\mathrm{VM}}} r_p(t) \le B + V \sum_{u\in P_{\mathrm{NIC}}} r_u(t) + V \sum_{j\in P_{\mathrm{VM}}} r_j(t) \\
&+ \sum_{j\in P_{\mathrm{NIC}}} Q_j(t)\left(\sum_{m\in M}\sum_{i\in P_{\mathrm{NIC}}}\sum_{u\in P_{\mathrm{VM}}}\sum_{v\in P_{\mathrm{VM}}} A_m(t)z_m^u(t)z_m^v(t)y_m^{ij}(t) - f(r_j(t))\right) \\
&- \sum_{u\in P_{\mathrm{VM}}} Q_u(t)\left(f(r_u(t)) + \sum_{s\in S} H_s(t)\left(\sum_{u\in P_{\mathrm{VM}}^s} r_u(t) + \sum_{j\in P_{\mathrm{NIC}}^s} r_j(t)\right)\right. \\
&\left.+ \sum_{i\in P_{\mathrm{NIC}}}\sum_{j\in P_{\mathrm{NIC}}} G_{ij}(t)\left(\sum_{m\in M}\sum_{u,v\in P_{\mathrm{VM}}} A_m(t)z_m^u(t)z_m^v(t)y_m^{ij}(t)\right)\right) \\
&+ \sum_{i\in P_{\mathrm{NIC}}}\sum_{j\in P_{\mathrm{NIC}}} N_{ij}(t)\left(\sum_{m\in M}\sum_{u\in P_{\mathrm{VM}}} A_m(t)z_m^u(t)y_m^{ij}(t)\right. \\
&\left.- \sum_{m\in M}\sum_{v\in P_{\mathrm{VM}}} A_m(t)z_m^v(t)y_m^{ij}(t)\right) + \sum_{m\in M}\sum_{i\in P_{\mathrm{NIC}}}\sum_{j\in P_{\mathrm{NIC}}} K_m(t)y_m^{ij}(t)
\end{aligned} \tag{21}$$

$$\min\ (H_s(t)+V)r_u(t) - Q_u(t)f(r_u(t)), \quad \forall u\in P_{\mathrm{VM}}^s, \quad \forall s\in S \tag{22}$$

$$\min\ (H_s(t)+V)r_j(t) - Q_j(t)f(r_j(t)), \quad \forall j\in P_{\mathrm{NIC}}^s, \quad \forall s\in S \tag{23}$$

$$\min\left((Q_j(t)+G_{ij}(t))\sum_{m\in M}\sum_{u,v\in P_{\mathrm{VM}}}A_m(t)z_m^u(t)z_m^v(t)+N_{ij}(t)\left(\sum_{m\in M}K_m(t)\right.\right.$$
$$\left.\left.+\sum_{m\in M}\sum_{u\in P_{VM}}A_m(t)z_m^u(t)-\sum_{m\in M}\sum_{v\in P_{VM}}A_m(t)z_m^v(t)\right)\right)y_m^{ij}(t) \tag{24}$$

3.3 Performance analysis

Based on the above transformation and optimization policies, we provide a theorem that reveals the tradeoff $[O(1/V), O(V)]$ between the time-average expectation of resource allocation and queue backlogs. We can evaluate the system's stability with the optimization object by tuning the factor V, which balances the vswitch resource allocation and the satisfaction of optimization constraints.

Theorem 1 *Assume the optimization problem (3–8) is feasible, the bounded restraints hold (10–11), the initial queue backlogs are finite ($\mathbb{E}[L(0)]\leq\infty$), and the Slater's condition holds.*

$$\begin{aligned}
&\sum_{p\in P}r_p(t)=\omega(\eta),\quad 0\leq\omega(\eta)\leq\sum_{p\in P}r_p^{\max}\\
&\mathbb{E}[H_s(t)]\leq 0,\quad \forall s\in S\\
&\mathbb{E}[G_{ij}(t)]\leq 0,\quad \forall i,j\in P_{\mathrm{NIC}}\\
&\mathbb{E}[N_{ij}(t)]\leq 0,\quad \forall i,j\in P_{\mathrm{VM}}\\
&\mathbb{E}[K_m(t)]\leq 0,\quad \forall m\in M\\
&\mathbb{E}[Q_p(t)]\leq \eta,\quad \forall p\in P \text{ and } \eta>0
\end{aligned}$$

If a C-additive algorithm applies on every slot to minimize the right side of inequality (21) and gets the optimal solution $\sum_{p\in P}r_p^{\mathrm{opt}}$, then(a) the time-average expectation of resource allocation satisfies (25) and (b) the time-average expectation of queue backlogs meets (26).

$$\limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{p\in P}\mathbb{E}[r_p(\tau)]\leq\sum_{p\in P}r_p^{\mathrm{opt}}+\frac{(B+C)}{V} \tag{25}$$

$$\limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{p\in P}\mathbb{E}[Q_p(\tau)]\leq\frac{1}{\eta}\left(B+C+V\left(\sum_{p\in P}r_p^{\mathrm{opt}}-\omega(\eta)\right)\right) \tag{26}$$

1 ***Proof 1*** We insert the Slater's condition on the right side of (21) and apply the C-additive algorithm on each slot to get the optimal solution, which yields (27). We get (28) from taking expectation of (27) with the law of total expectation, and applying the law of telescoping sum from slot 0 to $t-1$. Then, we rearrange the inequation (28)–(29), divide (29) by Vt and take limits $t\to\infty$ to prove (25). Similarly, we rearrange the inequation (28)–(30), divide ηt into (30) and take limits $t\to\infty$ to prove (26).

$$\Delta(\mathbf{\Theta}(t)) + V\mathbb{E}\left[\sum_{p\in P} r_p(t)|\mathbf{\Theta}(t)\right] \leq B + C + V\sum_{p\in P} r_p^{\text{opt}} - \eta\sum_{p\in P} Q_p(t) \tag{27}$$

$$\begin{aligned}&\mathbb{E}(L(\mathbf{\Theta}(t))) - \mathbb{E}(L(\mathbf{\Theta}(0))) + V\sum_{\tau=0}^{t-1}\mathbb{E}\left[\sum_{p\in P} r_p(\tau)\right]\\ &\quad\leq Bt + Ct + Vt\sum_{p\in P} r_p^{\text{opt}} - \eta\sum_{\tau=0}^{t-1}\sum_{p\in P}\mathbb{E}(Q_p(\tau))\end{aligned} \tag{28}$$

$$\begin{aligned}&V\sum_{\tau=0}^{t-1}\mathbb{E}\left[\sum_{p\in P} r_p(\tau)\right] \leq Bt + Ct + Vt\sum_{p\in P} r_p^{\text{opt}} + \mathbb{E}(L(\mathbf{\Theta}(0)))\\ &\quad - \eta\sum_{\tau=0}^{t-1}\sum_{p\in P}\mathbb{E}(Q_p(\tau)) - \mathbb{E}(L(\mathbf{\Theta}(t)))\end{aligned} \tag{29}$$

$$\begin{aligned}&\eta\sum_{\tau=0}^{t-1}\sum_{p\in P}\mathbb{E}[Q_p(\tau)] \leq Bt + Ct + Vt\sum_{p\in P} r_p^{\text{opt}} + \mathbb{E}(L(\mathbf{\Theta}(0)))\\ &\quad - V\sum_{\tau=0}^{t-1}\mathbb{E}\left[\sum_{p\in P} r_p(\tau)\right] - \mathbb{E}(L(\mathbf{\Theta}(t)))\end{aligned} \tag{30}$$

4 Algorithm design

In this section, we design the dynamic vswitch resource allocation and packet scheduling algorithm (Algorithm 1) to solve the problem (3–12). The algorithm calls the vswitch resource allocation algorithm (Algorithm 2) on each vswitch and the packet scheduling algorithm (Algorithm 3) on the network controller. The dynamic vswitch resource allocation and packet scheduling algorithm follow the performance tradeoff $[O(1/V), O(V)]$.

4.1 Algorithm design

According to the performance analysis in Sect. 3.3, we can derive algorithms to optimize the problems (22–24). We first design Algorithm 1 to solve the dynamic vswitch resource allocation problem and the packet scheduling problem. The cloud operator initializes the vswitch queue backlogs (1–2), virtual queue backlogs (13–16), resource allocation decision variable $r_p(t)$ and packet scheduling decision variables $y_m^{ij}(t)$. On each slot, vswitches execute Algorithm 2 to allocate the resource individually, and the cloud operator executes Algorithm 3 to schedule packets for each connection. Finally, the cloud operator updates vswitch queue backlogs (1–2) and virtual queue backlogs (13–16).

Algorithm 1 Resource Allocation and Packet Scheduling Algorithm

```
1: function ALLOCATE RESOURCE AND SCHEDULE PACKETS
2:     Initialize all queue backlogs and decision variables
3:     loop                                                    ▷ On each time slot
4:         Each vswitch executes Algorithm 2 separately.
5:         The cloud operator schedules packets for each connection by Algorithm 3.
6:         Update vswitch queues (1-2) and the virtual queues (13-16).
7:     end loop
8: end function
```

Then, Algorithm 2 solves the vswitch resource allocation problem (22–23). On slot t, each vswitch $s \in S$ calculates $f'(r_p(t))$ for each vswitch physical port and vswitch virtual port independently. The function $f(\cdot)$ is a nondecrease concave function. Therefore, we can set $r_p(t)$ to minimize the problem (22–23). The time complexity of Algorithm 2 is $O(n)$ for each vswitch, and n is the number of vswitch physical ports and vswitch virtual ports on a vswitch.

Algorithm 2 Vswitch Resource Allocation Algorithm

```
1: function ALLOCATE RESOURCE($s$)
2:     for each port $r_p \in P^s_{\text{NIC}} \bigcup P^s_{\text{VM}}$ in vswitch $s$ do
3:         Update vswitch queue backlogs (1) and (2), and update server capacity virtual queue back-
   logs (13).
4:         Update the factor $(H_s(t) + V)/Q_p(t)$.
5:         Set $r_p$ to minimize $f(r_p(t))$.
6:     end for
7: end function
```

At last, we design Algorithm 3 to schedule packets (24) for each connection. For each connection, the cloud operator updates the factor (31) on each forwarding path. If the factor is positive, the cloud operator will not select the corresponding forwarding path for the connection and vice versa. The algorithm's time complexity for each connection is $O(l)$, and l is the number of possible forwarding paths for a connection.

$$\begin{aligned}&(Q_j(t) + G_{ij}(t))\left(\sum_{u,v \in P_{\text{VM}}} A_m(t) z^u_m(t) z^v_m(t)\right)\\&\quad + N_{ij}(t)\left(\sum_{u \in P_{VM}} A_m(t) z^u_m(t) - \sum_{v \in P_{VM}} A_m(t) z^v_m(t)\right)\end{aligned} \tag{31}$$

Algorithm 3 Packet Scheduling Algorithm

```
1: function SCHEDULE PACKETS
2:     for each connection $m \in M$ do
3:         for each physical path $i, j$ do
4:             Calculate the factor (31)
5:             if the factor (31) is greater than $0$ then
6:                 Set $y_m^{ij} = 0$
7:             else
8:                 Set $y_m^{ij} = 1$
9:             end if
10:        end for
11:    end for
12: end function
```

Besides, our dynamic vswitch resource allocation and packet scheduling algorithm is compatible with data center networks' existing control plane protocol. For example, OpenFlow provides a standard network control plane abstraction. On the one hand, it supports per-flow meters to measure each flow size in a fine-grained granularity on each vswitch. Algorithm 2 can update the vswitch queue backlogs by utilizing the measured flow size. On the other hand, Algorithm 3 can configure the forwarding path of each flow by setting flow forwarding rules in the OpenFlow protocol. Thus, our algorithm is compatible with data center network control plane protocols and can be deployed in data center networks.

4.2 Extended algorithm

In the real-world CIoV network, if the derivative of the optimization problem (22–23) is greater than 0 (32) for the specific vswitch port u, the vswitch will not allocate any CPU resource to the port u according to Algorithm 2. The packets in $Q_u(t)$ could not be dequeued until the vswitch allocates resource to port u, which delays packets forwarding and increases the queue backlog of port u.

$$H_s(t) - Q_u(t)\frac{\mathrm{d}f(r_u(t))}{\mathrm{d}r_u(t)} + V > 0 \tag{32}$$

To decrease the queue backlog of port u, we deceive the vswitch by inserting the queue backlog place-holder [33] (33) to the derivation of the optimization problem (22–23). The queue backlog place-holder $P_u(t)$ is a nonpositive parameter to decrease the derivation of (22–23) until the derivation is not greater than 0.

$$H_s(t) + P_u(t) - Q_u(t)\frac{\mathrm{d}f(r_u(t))}{\mathrm{d}r_u(t)} + V \leq 0 \tag{33}$$

The queue backlog place-holder of port u will not break the performance tradeoff (25–26). A queue backlog place-holder can be considered as an invisible queue backlog (34) which decreases the resource capacity virtual queue (13) for the port u. The cloud operator will allocate more resources to port u. To stabilize the resource capacity virtual queue, Algorithm 2 will decrease the resource allocation for other ports on the vswitch.

$$\hat{H}_s(t) = H_s(t) + P_u(t) \tag{34}$$

The dynamic vswitch resource allocation and packet scheduling algorithm also can use the extended resource allocation algorithm to decrease queue backlogs violating the performance tradeoff.

5 Experiments and evaluation

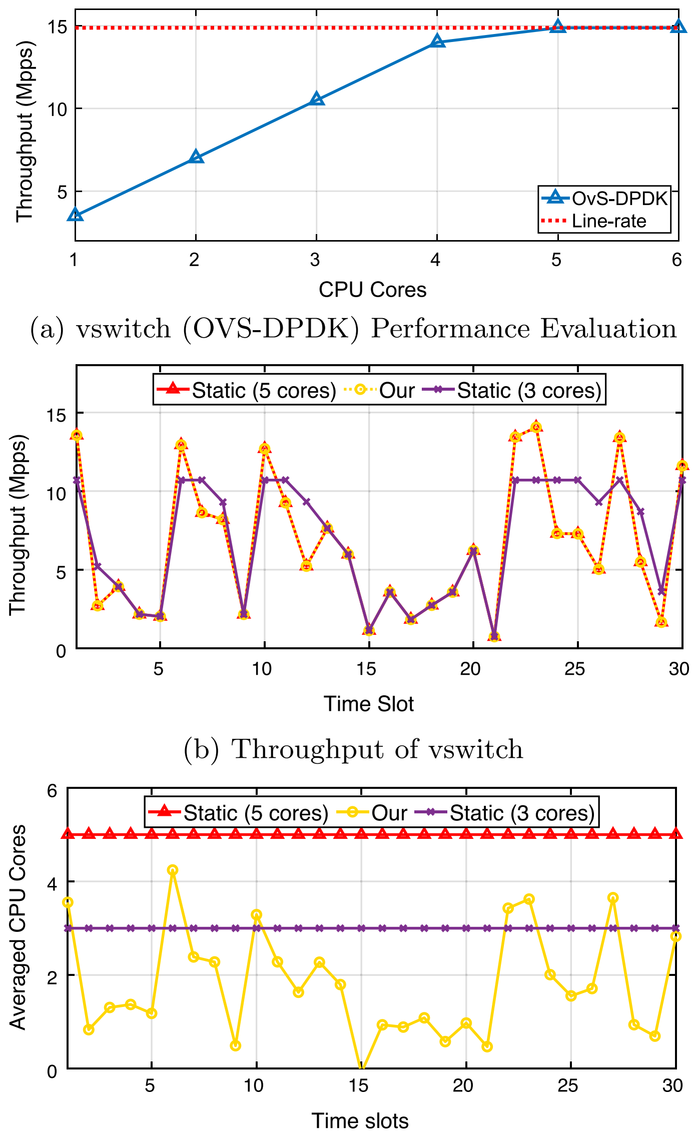
In this section, we first measure the performance of vswitches on a virtualized server and evaluate the vswitch resource allocation algorithm on the testbed. Second, we evaluate the dynamic vswitch resource allocation and the packet scheduling algorithm in the network-wide simulation on a real-world data center network topology. Finally, we verify the algorithm performance under various experiment settings and validate our performance bounds.

5.1 Single-node evaluation

We first evaluate the performance of the virtual switch on a hardware testbed. The hardware testbed consists of a packet generation server and an under-test server. Each server equips two CPUs (E5-2620v2, 12 cores) and connects to each other with 10 Gbps NICs (Intel 82599). We implement the vswitch resource allocation and packet scheduling algorithms on OpenvSwitch (OVS) [29] 2.5.1 and Data Plane Development Kit (DPDK) [34] 2.2.0 on the under-test server. We generate 10 Gbps network traffic at line rate with the traffic generator pktgen-DPDK [35]. The network traffic reaches 14.88 Mpps under 64 bytes packet length, which is the upper bound throughput of 10 Gbps NICs.

We configure the flow table in the vswitch to forward all incoming packets and allocate different numbers of CPU cores to the vswitch for performance evaluation. We show the measurement results in Fig. 4a. First, we find that vswitch can process more network packets as the number of allocated CPUs increases. The allocation of CPU resources and the throughput rate of vswitch is a nondecrease concave function, which is consistent with our assumptions in Sect. 3.1. Then, we find that the virtual switch requires at least five CPU cores to process line-rate network traffic with the current hardware settings. The vswitch has already occupied 41.7% (5/12) of the total CPU cores. Data center providers would have to statically allocate a significant portion of CPU resources to virtual switches to provide line-rate forwarding rates without dynamic resource allocation algorithms.

We also evaluate the vswitch resource allocation algorithm on the testbed for 30 time slots, and the interval of each time slot is 60 s. In each time slot, we generate a random packet flow with an uniform random distribution packet size and collect the averaged queue backlogs and the number of allocated CPU cores on the vswitch in Fig. 4b and c. The current implementation of OVS-DPDK only supports static resource allocation, and we statically allocate three cores and five cores in this experiment. On our testbed, the vswitch achieves line-rate forwarding with five CPU cores. When allocating three CPU cores to the vswitch, it cannot achieve peak performance and delay packet forwarding. Our dynamic allocation algorithm alters CPU allocation to the incoming packets, allowing the vswitch to handle peak and lower network traffic dynamically. Experiments show that our dynamic virtual switch resource allocation algorithm can reduce the CPU



(a) vswitch (OVS-DPDK) Performance Evaluation

(b) Throughput of vswitch

(c) Averaged CPU allocation of vswitch

Fig. 4 The vswitch resource allocation evaluation on the single-node testbed

resource allocation compared to static allocation while providing sufficient forwarding throughput.

5.2 Network-wide evaluation

To evaluate our dynamic vswitch resource allocation and the packet scheduling algorithm, we build up an event-driven simulation environment. The network-wide simulation environment includes an edge data center architecture [36, 37] and the cloud data center architecture [38, 39]. We equip each server in the edge data center with 20 CPU cores and a 25 Gbps NIC, and each server in the cloud data center with 40 CPU cores and a 40 Gbps NIC. We generate cognition service workloads from real-world edge data center network distribution [40] and cloud data center network traces [38]. The tradeoff factor V is set to 1.5×10^7.

We implement three resource allocation algorithms as baselines for the experiment. (1) Static resource allocation. It allocates resources to virtual switches according to the mean value of the input data. (2) Shenango-like [23] resource allocation. It directly allocates resources according to vswitch queue backlogs. When the queue backlogs increase, it increases CPU allocation and vice versa. (3) Snap-like [25] resource allocation. It also allocates resources according to queue backlogs. Besides, it employs a more aggressive allocation strategy that always over-allocates CPUs to reduce tail latency.

We show the performance evaluation of these four algorithms in Fig. 5 for edge and cloud data centers, respectively. We collect the number of allocated CPUs, the average queue length, and the throughput of virtual switches. Since the static resource allocation is based on the prior knowledge of the input data, it can achieve higher throughput and lower queue backlogs with less CPU resources. However, it is difficult to obtain the precise data distribution in real-world CIoV scenarios, making this algorithm hard to apply in practice. Both Shenango-like and Snap-like algorithms allocate resources according to the queue backlogs of virtual switches. Shenango-like algorithm occupies less resources and achieves lower throughput compared with the Snap-like algorithm. Snap-like algorithm achieves the shortest queue backlogs and the highest throughput through aggressive resource allocation. However, it clearly consumes a large number of CPU resources. Compared with these algorithms, our resource allocation algorithm not only utilizes vswitch queue backlogs, but also considers virtual queue backlogs, which are transformed from the optimization constraints by the Lyapunov Optimization Framework. It balances resource allocation and queue backlogs in the network-wide scenario and achieves better performance.

5.3 Theoretical performance tradeoff evaluation

To validate the $[O(1/V), O(V)]$ performance tradeoff, we execute the Algorithm 1 for 10^3 time slots and tunes the tradeoff factor V from 1.2 to 1.3×10^7 under the same experiment configurations in Sect. 5.2. We collect the time-average expectation of queue backlogs and resource allocation for each V in Fig. 6a and b. In Fig. 6a, the time-average expectation of resource allocation reduces and closes to the optimal solution with

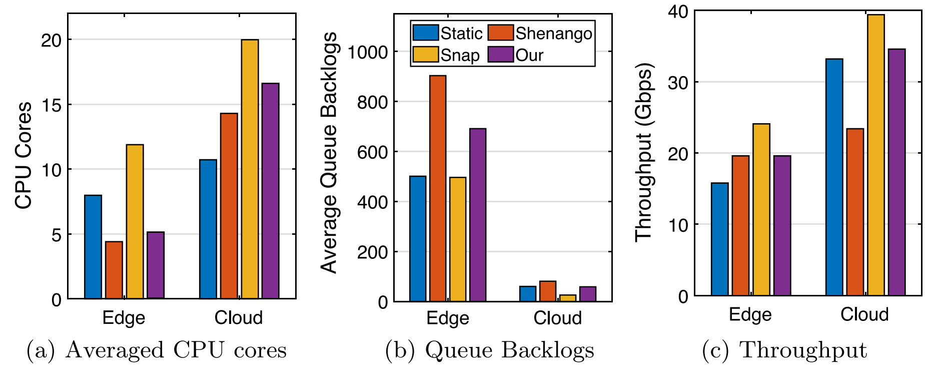


Fig. 5 Network-wide evaluation

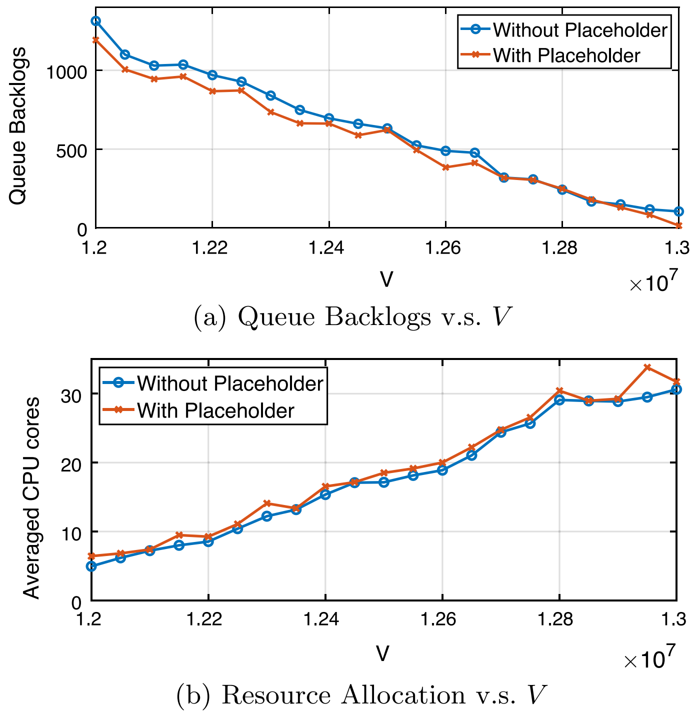


Fig. 6 Theoretical performance tradeoff evaluation

V increasing. And in Fig. 6a, the time-average expectation of queue backlogs increases to infinity as V increasing. These results both confirm that our algorithm follows the $[O(1/V), O(V)]$ performance tradeoff.

We also evaluate the extended Algorithm 1 which inserts queue backlog place-holders to virtual queues. We execute the extended algorithm under the same configuration and collect the time-average expectation of resource allocation and the time-average expectation of queue backlogs in Fig. 6a and b. Figure 6a and b reveals that the extended algorithm reduces queue backlogs and enlarges a little bit of the resource allocation on vswitches due to the additional backlog place-holders. The results validate that the extended algorithm also satisfies the $[O(1/V), O(V)]$ tradeoff.

6 Conclusions

In this paper, we study the dynamic vswitch resource allocation problem and the packet scheduling problem in CIoV. We first formulate the joint optimization problem to optimize resource allocation on vswitches and analysis it through Lyapunov Optimization Framework. Then, we transform the optimization problem into a discrete-time queueing system and decompose the problem into a queue stability problem. Next, we prove the performance tradeoff $[O(1/V), O(V)]$ of the policies between the time-average expectation of resource allocation and queue backlog. We design low-complexity vswitch resource allocation and packet scheduling algorithms from the analysis. Finally, we design experiments to evaluate the performance of our algorithms on a real-world testbed and a network-wide simulation. Results show that our algorithms satisfy the performance tradeoff and outperform other allocation algorithms.

Abbreviations
IoV: : Internet of vehicles;; CIoV: : Cognitive internet of vehicles;; VM: : Virtual machine;; NIC: : Network interface card;; OpEx: : Operational expenses;; ToR: : Top-of-rack;; RSU: : Road side units;; SDN: : Software defined network;; ILP: : Integer linear problem;; DPDK: : Data plane development kit..

Acknowledgements
Not applicable.

Yang Wang
received the BS degree from University of Electronic Science and Technology of China in 2014, and he is a Ph.D student of Electronic Science and Technology of China. His research interest includes network measurement, software defined networking, and computer architecture.

Xiong Wang
is an associate professor in school of information and communication at the University of Electronic and Technology Science of China, Chengdu, China. He received his Ph.D. in communication and information system from the University of Electronic and Science of China, Chengdu, china, in 2008. From 2013 to 2014, he was a visiting scholar in electrical and computer engineering at the University of California, Davis, CA, USA. His research interests include network measurement, modeling and optimization, algorithm analysis and design, and network management in communication networks.

Zhuobin Huang
is a master student in the school of at the University of Electronic Science and Technology of China.

Wei Li
works for The State Radio Monitoring Center, Beijing, China.

Shizhong Xu
received his B.S., M.S., and Ph.D. degrees in electrical engineering in 1994, 1997, and 2000, respectively. He now is a professor in the school of from the University of Electronic Science and Technology of China, Chengdu, China, information and communication at the University of Electronic Science and Technology of China, Chengdu, China. His research interests include Internet of Things, next generation network, and network science.

Authors' contributions
All authors read and approved the final manuscript.

Funding
This work is supported by National Key R&D Program of China (2019YFB1803304), NSFC Fund (62072079, 62001087, U20A20156), and Open Research Projects of Zhejiang Lab (No. 2022QA0AB01).

Availability of data and materials
The datasets and network traces are presented in this paper are all open-source.

Declarations

Competing interests
The authors declare that they have no competing interests.

Author details
[1]University of Electronic Science and Technology of China, Chengdu, China. [2]The State Radio Monitoring Center, Beijing, China.

Received: 17 January 2022 Accepted: 2 March 2022
Published online: 04 April 2022

References
1. F. Li, K.-Y. Lam, Z. Ni, D. Niyato, X. Liu, L. Wang, Cognitive carrier resource optimization for internet-of-vehicles in 5g-enhanced smart cities. IEEE Netw. (2021)
2. J. Mu, Y. Gong, F. Zhang, Y. Cui, F. Zheng, X. Jing, Integrated sensing and communication-enabled predictive beamforming with deep learning in vehicular networks. IEEE Commun. Lett. **25**(10), 3301–3304 (2021)
3. A. Jagannath, J. Jagannath, T. Melodia, Redefining wireless communication for 6g: signal processing meets deep learning with deep unfolding. IEEE Trans. Artif. Intell. (2021)
4. X. Liu, C. Sun, M. Zhou, B. Lin, Y. Lim, Reinforcement learning based dynamic spectrum access in cognitive internet of vehicles. China Commun. **18**(7), 58–68 (2021)
5. H. Zhang, H. Zhang, W. Liu, K. Long, J. Dong, V.C. Leung, Energy efficient user clustering, hybrid precoding and power optimization in terahertz MIMO-NOMA systems. IEEE J. Sel. Areas Commun. **38**(9), 2074–2085 (2020)
6. A. Paul, A. Daniel, A. Ahmad, S. Rho, Cooperative cognitive intelligence for internet of vehicles. IEEE Syst. J. **11**(3), 1249–1258 (2015)

7. X. Liu, X. Zhang, NOMA-based resource allocation for cluster-based cognitive industrial internet of things. IEEE Trans. Industr. Inf. **16**(8), 5379–5388 (2019)
8. X. Liu, C. Sun, M. Zhou, C. Wu, B. Peng, P. Li, Reinforcement learning-based multislot double-threshold spectrum sensing with Bayesian fusion for industrial big spectrum data. IEEE Trans. Industr. Inf. **17**(5), 3391–3400 (2020)
9. B. Wei, L. Zhang, K. Wang, Q. Kong, Z. Wang, Dynamic scene deblurring and image de-raining based on generative adversarial networks and transfer learning for internet of vehicle. EURASIP J. Adv. Signal Process. **2021**(1), 1–19 (2021)
10. T.M. Ghazal, R.A. Said, N. Taleb, Internet of vehicles and autonomous systems with AI for medical things. Soft Comput. 1–13 (2021)
11. H. Lu, Q. Liu, D. Tian, Y. Li, H. Kim, S. Serikawa, The cognitive internet of vehicles for autonomous driving. IEEE Netw. **33**(3), 65–73 (2019)
12. H. Chang, N. Ning, An intelligent multimode clustering mechanism using driving pattern recognition in cognitive internet of vehicles. Sensors **21**(22), 7588 (2021)
13. S.J. Kamble, M.R. Kounte, Machine learning approach on traffic congestion monitoring system in internet of vehicles. Procedia Comput. Sci. **171**, 2235–2241 (2020)
14. C. Wang, L. Zhang, Z. Li, C. Jiang, SDCoR: software defined cognitive routing for internet of vehicles. IEEE Internet Things J. **5**(5), 3513–3520 (2018)
15. X. Liu, X.B. Zhai, W. Lu, C. Wu, QoS-guarantee resource allocation for multibeam satellite industrial internet of things with NOMA. IEEE Trans. Industr. Inf. **17**(3), 2052–2061 (2019)
16. M.J. Neely, E. Modiano, C.-P. Li, Fairness and optimal stochastic control for heterogeneous networks. IEEE/ACM Trans. Netw. (TON) **16**(2), 396–409 (2008)
17. M.J. Neely, Stochastic network optimization with application to communication and queueing systems. Synth. Lect. Commun. Netw. **3**(1), 1–211 (2010)
18. M.J. Neely, Delay-based network utility maximization. IEEE/ACM Trans. Netw. **21**(1), 41–54 (2013)
19. H. Zhang, B. Wang, C. Jiang, K. Long, A. Nallanathan, V.C. Leung, H.V. Poor, Energy efficient dynamic resource optimization in NOMA system. IEEE Trans. Wirel. Commun. **17**(9), 5671–5683 (2018)
20. M. Chen, Y. Tian, G. Fortino, J. Zhang, I. Humar, Cognitive internet of vehicles. Comput. Commun. **120**, 58–70 (2018)
21. Y. Qian, M. Chen, J. Chen, M.S. Hossain, A. Alamri, Secure enforcement in cognitive internet of vehicles. IEEE Internet Things J. **5**(2), 1242–1250 (2018)
22. K.F. Hasan, T. Kaur, M.M. Hasan, Y. Feng, Cognitive internet of vehicles: motivation, layered architecture and security issues. In: 2019 International Conference on Sustainable Technologies for Industry 4.0 (STI), pp. 1–6 (2019). IEEE
23. A. Ousterhout, J. Fried, J. Behrens, A. Belay, H. Balakrishnan, Shenango: achieving high {CPU} efficiency for latency-sensitive datacenter workloads. In: 16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19), pp. 361–378 (2019)
24. H. Qin, Q. Li, J. Speiser, P. Kraft, J. Ousterhout, Arachne: core-aware thread management. In: 13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18), pp. 145–160 (2018)
25. M. Marty, M. de Kruijf, J. Adriaens, Alfeld, C., S. Bauer, C. Contavalli, M. Dalton, N. Dukkipati, W.C. Evans, S. Gribble, *et al.*, Snap: a microkernel approach to host networking. In: Proceedings of the 27th ACM Symposium on Operating Systems Principles, pp. 399–413 (2019)
26. X. Yang, H. Xu, L. Huang, G. Zhao, P. Xi, C. Qiao, Joint virtual switch deployment and routing for load balancing in SDNs. IEEE J. Sel. Areas Commun. **36**(3), 397–410 (2018)
27. Y. Yang, X. Chang, J. Liu, L. Li, Towards robust green virtual cloud data center provisioning. IEEE Trans. Cloud Comput. **5**(2), 168–181 (2015)
28. Y. Yang, S. Guo, G. Liu, L. Yi, Fine granularity resource allocation of virtual data center with consideration of virtual switches. J. Netw. Comput. Appl. **175**, 102916 (2021)
29. B. Pfaff, J. Pettit, T. Koponen, E.J. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, et al., The design and implementation of open vswitch. In: NSDI, pp. 117–130 (2015)
30. Y. Zhao, L. Iannone, M. Riguidel, Software switch performance factors in network virtualization environment. In: 2014 IEEE 22nd International Conference on Network Protocols (ICNP), pp. 468–470 (2014). IEEE
31. P. Emmerich, D. Raumer, F. Wohlfart, G. Carle, Assessing soft-and hardware bottlenecks in PC-based packet forwarding systems. ICN **2015**, 90 (2015)
32. P. Emmerich, D. Raumer, S. Gallenmüller, F. Wohlfart, G. Carle, Throughput and latency of virtual switching with open vswitch: a quantitative analysis. J. Netw. Syst. Manage. **26**(2), 314–338 (2018)
33. L. Huang, M.J. Neely, Delay reduction via lagrange multipliers in stochastic network optimization. In: 2009. 7th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, 2009. WiOPT, pp. 1–10 (2009). IEEE
34. L. Foundation, Data Plane Development Kit (DPDK) (2015). http://www.dpdk.org
35. The pktgen application. https://pktgen-dpdk.readthedocs.io/
36. N. Panwar, S. Sharma, A.K. Singh, A survey on 5g: the next generation of mobile communication. Phys. Commun. **18**, 64–84 (2016)
37. H. Zhang, M. Feng, K. Long, G.K. Karagiannidis, V.C. Leung, H.V. Poor, Energy efficient resource management in SWIPT enabled heterogeneous networks with NOMA. IEEE Trans. Wirel. Commun. **19**(2), 835–845 (2019)
38. A. Roy, H. Zeng, J. Bagga, G. Porter, A.C. Snoeren, Inside the social network's (datacenter) network. In: Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, pp. 123–137 (2015)
39. A. Andreyev, A. Andreyev, Introducing data center fabric, the next-generation Facebook data center network (2018). https://engineering.fb.com/2014/11/14/production-engineering/introducing-data-center-fabric-the-next-generation-facebook-data-center-network/
40. E. Bastug, M. Bennis, M. Debbah, Living on the edge: the role of proactive caching in 5g wireless networks. IEEE Commun. Mag. **52**(8), 82–89 (2014)