Adaptive and Robust Routing With Lyapunov-Based Deep RL in MEC Networks Enabled by Blockchains

Zirui Zhuang, *Member, IEEE*, Jingyu Wang, *Member, IEEE*, Qi Qi, *Member, IEEE*, Jianxin Liao, *Member, IEEE*, and Zhu Han, *Fellow, IEEE*

***Abstract*—The most recent development of the Internet of Things brings massive timely sensitive and bursty data flows. Also, joint optimization on storage, computation, and communication is in need for multiaccess edge computing frameworks. The adaptive network control has been explored using deep reinforcement learning (RL), but it is not sufficient for bursty network traffic flows, especially when the network traffic pattern may change over time. We formulate the routing control in an environment with time-variant link delays as a Lyapunov optimization problem. We identify that there is a tradeoff between optimization performance and modeling accuracy when the propagation delays are included. We propose a novel deep RL (DRL)-based adaptive network routing method to tackle the issues mentioned above. A Lyapunov optimization technique is used to reduce the upper bound of the Lyapunov drift, improving queuing stability in networked systems. By modeling the network traffic pattern using the Markovian arrival process, we show that network routing problems can be modeled as Markov decision processes and value-iteration-based RL methods can be used to solve them. We design a blockchain-based protocol using proof of elapsed time consensus mechanism to ensure a trustworthy network statistics information exchange for the routing framework. Experiment results show that the proposed method can learn a routing policy and adapt to the changing environment. The proposed method outperforms the baseline backpressure method in multiple settings and converges faster than existing methods. Moreover, the DRL module can effectively learn a better estimation of the long-term Lyapunov drift and penalty functions, providing superior results in terms of the backlog size, end-to-end latency, age of information, and throughput. Furthermore, the blockchain-based network statistics exchange can provide the routing framework against malicious nodes. In addition, the proposed model performs well under various topologies, and thus can be used in general cases.**

***Index Terms*—Adaptive control, Lyapunov optimization, network routing, reinforcement learning (RL).**

Manuscript received April 30, 2020; revised July 22, 2020 and September 24, 2020; accepted October 21, 2020. Date of publication October 29, 2020; date of current version February 4, 2021. This work was supported in part by the National Key Research and Development Program of China under Grant 2020YFB1807803 and Grant 2020YFB1807800; in part by the National Natural Science Foundation of China under Grant 62071067, Grant 62001054, and Grant 61771068; in part by the Beijing Municipal Natural Science Foundation under Grant 4182041; in part by the Ministry of Education and China Mobile Joint Fund under Grant MCM20180101; in part by the Ph.D. Student Short-Term Exchange Program of Beijing University of Posts and Telecommunications; and in part by NSF under Grant EARS-1839818, Grant CNS1717454, Grant CNS-1731424, and Grant CNS-1702850. *(Corresponding author: Jingyu Wang; Qi Qi.)*

Zirui Zhuang, Jingyu Wang, Qi Qi, and Jianxin Liao are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: zhuangzirui@bupt.edu.cn; wangjingyu@bupt.edu.cn; qiqi8266@bupt.edu.cn; liaojx@bupt.edu.cn).

Zhu Han is with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77004 USA, and also with the Department of Computer Science and Engineering, Kyung Hee University, Seoul 446-701, South Korea (e-mail: zhan2@uh.edu).

Digital Object Identifier 10.1109/JIOT.2020.3034601

I. Introduction

THE EXPLOSION of the Internet of Things (IoT) generates massive sensory and time-series data with burstiness. Ten years ago, the death of Michael Jackson shocked people all around the world, and it shocked us even more as it brought the Internet down at the same time. End users experienced difficulties in accessing services from almost all major information technology providers. Today, the Internet, carriers, and service providers are facing an even more challenging environment. For instance, the COVID-19 pandemic forces people to work from home, highly relying on teleconference, video conference and online collaboration, which creates bursty, long-range peer-to-peer and high-volume network traffic. As ten years have gone by, the dramatically increased volume of online social networks, the upgraded mobile networks, and the growth of ubiquitous IoT devices, provide a rapid channel for information diffusion upon triggering events. It means that there will be more and more information explosion, as well as burstiness in the communication networks. Sometimes, the burst of network traffic may even come from security vulnerabilities. Attackers may exploit IoT terminals and use it to perform (distributed-) denial-of-service attacks [1], [2].

As a result, these massive data are required to be transmitted, delivered, and processed in a timely manner [3]. To achieve lower latency, clouds and servers are deployed increasingly closer to the end users, which is called multiaccess edge computing (MEC) [4]. With the arising deployment of MEC, it is vital for the routing methods to take into account the edge service utilities [5] and to jointly optimize storage, computation, and communication [6], [7]. There is existing work from the perspective of storage and computation, such as edge caching [8], [9] and computation offloading [10]–[12]. However, the perspective of communication still needs to be explored.

Recent studies have summarized the timeliness of the networked system using the age of packet received, defining a new metric called the Age of Information (AoI) [13], [14], which can cover all the latency components of storage, computation, and communication. The information generated from hotspot events should be updated and delivered as soon as possible. The demand for timeliness also applies to sensory data generated from the IoT devices, monitors, and roadside infrastructures [15]. Also, the network should be able to stabilize itself against the burstiness caused by hotspot events. Although important, few studies emphasize the impact of burstiness on emerging network traffics [16]. This requirement pushes network management into a whole new level. There is a vision that future communication networks require a convergence of communication, computing, and control [17], [18]. The communication networks may achieve better performance and higher efficiency with the help of artificial intelligence computing techniques, and it may become more reliable and robust with the integrated co-design of control systems. The highly dynamic and unpredictable traffic pattern shifts the demand for the ability to adaptively adjust control policy in network management. Also, the control policy should be sufficiently stable against random burstiness in traffic flows. It is our vision that the combination of deep reinforcement learning (RL) and Lyapunov optimization is the very first step toward more intelligent, autonomous, and reliable communication networks in future the Internet-of-Everything environment.

On the one hand, with the deep RL (DRL) methods as powerful tools generating adaptive network control policies, it becomes an increasingly trending concept [3] that these problems are better solved with the cross-layer optimization involving access control, resource allocation, and routing. On the other hand, the Lyapunov optimization uses a quadratic function of queue backlog sizes as the Lyapunov function, and a control policy is used to reduce the upper bound of the Lyapunov function drift. In this way, the queue backlog sizes will be sufficiently stable as long as the arrival rates are interior to the capacity region [19], [20].

Combining the DRL methods with the Lyapunov optimization generates more power than simply pooling their efforts together. Not only the Lyapunov optimization part gives better stability guarantee and more robust decision making, but also the RL part achieving a better cumulative t-step control result as a whole than the t-slot time average in standard Lyapunov optimizations. Moreover, the short-term and long-term constraints of heterogeneous IoT Quality of Service (QoS) demands stated by Chen *et al.* [14] can be met over time as bonded by the Lyapunov methods.

But there is a major concern about the environment's stationarity when we try to apply DRL onto a networking system whose state will be affected by input from the outside of the system. We show that the Markovian arrival process (MAP) can be used to model the traffic's arrival, and on top of that, the environment will be stationary for a RL agent to learn from. The MAP can be used to accurately model any arbitrary random arrival process if the state space is sufficiently large and the state transition matrix is sufficiently dense [21], [22].

As the routing decisions are made using information about the queue backlog size in each node, it is vital to ensure the backlog size acquired is accurate and trustworthy. This can be a challenge because the nodes may advertise wrong information either actively or passively. In the active scenario, the medium of a forwarding node, such as a roadside unit or a drone, can be breached and taken control of by malicious attackers. In the passive scenario, the hardware or software in a forwarding node may by overloaded and facing performance degradation, and thus provided inaccurate queue backlog size information to other nodes.

Therefore, we propose an adaptive network routing control framework utilizing the strengths of both DRL and the Lyapunov optimization. The blockchain-based network statistics exchange protocol as an add-on enables a trusted and reliable environment for the routing framework. The proposed framework unravels the issues mentioned above.

In summary, the main contributions in this article are listed as follows.

1) To our best knowledge, we are the first to model the queuing dynamics in the network routing environment with time-variant propagation delay properties. This leads to a more accurate model and contributes to better control policies.
2) We formulate the control objective using Lyapunov optimization techniques. An objective function is derived to achieve any given optimization goal with the ability to stabilize the system in the sense of Lyapunov. We also identify that there is a tradeoff between optimization performance and modeling accuracy by adjusting the time-slot interval.
3) We analyze why the (deep-) RL methods can be used to solve the network routing problem where there is traffic randomly injected into the system. We model the traffic as a general random arrival process using the MAP and show that the system environment is stationary so that the RL-based methods can be applied.
4) We design a RL algorithm using deep neural networks to adaptively search for the optimal control policy as the network channel characteristics change.
5) We apply a blockchain-based protocol to provide a distributed and trusted routing information exchange. The security is improved against actively malicious nodes and passively malfunctioning nodes.

The remainder of this article is organized as follows. Section II reviews related work. Section III reveals the system models. Section IV explains the methodology. Section V gives the experiment results and analysis. Section VI concludes this article.

II. Related Work

A. Adaptive Control and Deep Reinforcement Learning in Communication Networks

The adaptive network control is previously exploited to improve the Quality of Experience (QoE) [23], [24], the connectivity in IoT networks [25], the reliability in vehicular ad hoc network (VANET) [26], and the maximum sending rate known as traffic engineering [27]. As adaptive control policies are adjusted online and they usually can be modeled as Markov decision processes (MDPs), RL techniques are used extensively, and many of them [23], [24], [26] use DRL methods. This is because that DRL techniques are capable of extracting highly sophisticated system dynamics and make policy adaptation according to the guidance of given rewards.

Kim *et al.* [28] used logistic regression to classify congestion status and build an adaptive rate control on top of this congestion classifier. Zhou *et al.* [29] combined traffic forecasting and DRL to choose frequency scaling in network function virtualization-based network operations dynamically. Fu *et al.* [30] used DRL to efficiently solve service function chain embedding problem in network function virtualization-enabled IoT systems. Wang *et al.* [31] used federated DRL to improve QoS in edge cache services for IoT devices.

Guan *et al.* [32] proposed a cooperative topology control scheme improving the network capacity by jointly consider upper layer network capacity and physical layer cooperative communications. Guan *et al.* [33] jointly consider authentication and topology control and formulate a discrete stochastic optimization problem to adaptively achieve security according to the available resource for mobile ad hoc networks with cooperative communication.

There are recent works trying to use DRL methods to solve network routing problems [34]–[36]. Stampa *et al.* [34] used DRL to adaptively adjust link weight upon changes in the traffic demand matrix, and the authors use these link weights to generate routing decisions. Suarez-Varela *et al.* [35] used DRL to schedule flows among a set of previously generated paths. Xu *et al.* [36] used DRL to find the optimal traffic split ratios in traffic engineering, where the state space vectors represent the throughput and delay in each session. Note that the new traffic arrivals in these works are all modeled as Poisson processes, which is a strong and simplified assumption. Due to the fact that RL techniques require stationary environments to learn from, these attempts all make abstraction of network dynamics, either in the state space [34]–[36] or the action space [34], [35]. These abstractions help DRL algorithms converge despite that the communication networks are highly dynamic and transient events are seen from time to time. However, the abstractions also cost the networked system the ability to make the optimal control decision since all the burstiness is smoothed out. Upon changes in traffic patterns, although the DRL algorithms are adaptive, they have to converge again to the new (sub-)optimal solutions, and there are concerns whether the algorithms can converge faster than network traffic pattern shifts.

B. Modeling Methods of Arrival Processes

Klemm *et al.* [37] used the Batch MAP (BMAP) to model the Internet protocol (IP) traffic and develop an expectation-maximization (EM) algorithm to estimate the BMAP's parameters. Okamura *et al.* [21] used EM algorithms to fit network traffic to MAP and Markovian modulated arrival process (MMAP) models with group data. Rezaei *et al.* [22] used MAP to model the arriving flows and give delay analysis in cache-enabled networks. Neely *et al.* [38] used the Markovian arrival model to analyze the robustness of the Lyapunov Optimization when the latter is used in network routing. Kadota and Modiano [39] used the Bernoulli arrival process to model the stochastic arrival of packets and minimizes the AoI under this setting. Markovian models are also used in the power system's generation dynamics [40] to help electricity pricing.

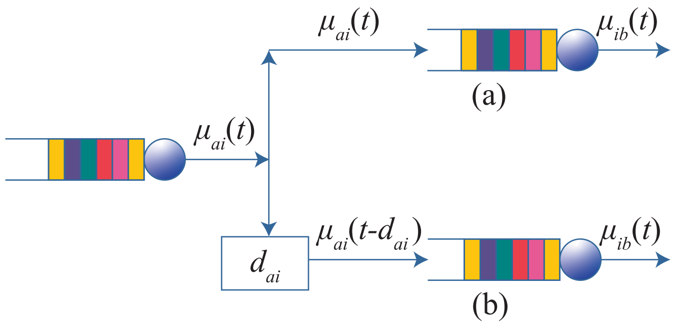


Fig. 1. Queuing dynamics in networked queuing systems: (a) case in which the propagation delay between two queues is sufficiently small to be ignored or does not exist and (b) case in which there is a d_{ai} delay between two queues.

C. Lyapunov Optimization Methods

The Lyapunov second method states that if a system at its stable state is given a trivial interference, it will always return to its stable state as long as the drift of the Lyapunov function is smaller than or equal to zero [41]. Lyapunov optimization has been widely used as a tool to solve stochastic network optimization problems, as in energy harvesting [42], smart grid [43], cache placement [8], and network routing [44], [45].

Network routing with queuing stability objective is known as backpressure routing [46] which uses differential queue backlog sizes between neighboring nodes as an intermediate state to minimize the system's Lyapunov drift [19]. However, backpressure routing often comes with long convergence time, and thus cannot provide sufficient QoS guarantees. Kabou *et al.* [47] used distributed and prioritized control policy to improve QoS in terms of end-to-end delay. To improve the system stability in communication networks with constraints, Neely [44], [48] and Neely and Urgaonkar used the Lyapunov optimization and drift-plus-penalty algorithm as tools to perform stochastic network optimization in queuing systems. Ying *et al.* [49] used virtual queues to add hop constraints to Lyapunov-based backpressure routing, achieving queuing stability with the smallest hop counts possible. Stefanovic and Pavel [50] consider optical networks with constant link delay and optimize the Lyapunov bound under the worst case scenarios. Neely [51] also designs a scheduling policy that gives bounded worst case delay in multihop networks. Núñez-Martínez and Mangues-Bafalluy [52] designed a distributed drift-plus-penalty algorithm with adaptive weighting between queuing stability and the distance to the destination node.

The queuing dynamics are often modeled in a time-slotted manner, where the Lyapunov drift is computed between two consecutive time slots and a control policy is used to minimize the drift's upper bound to stabilize the queuing system. A more responsive control policy requires a more fine-grained time-slot interval. Fig. 1(a) shows how the queuing dynamics work

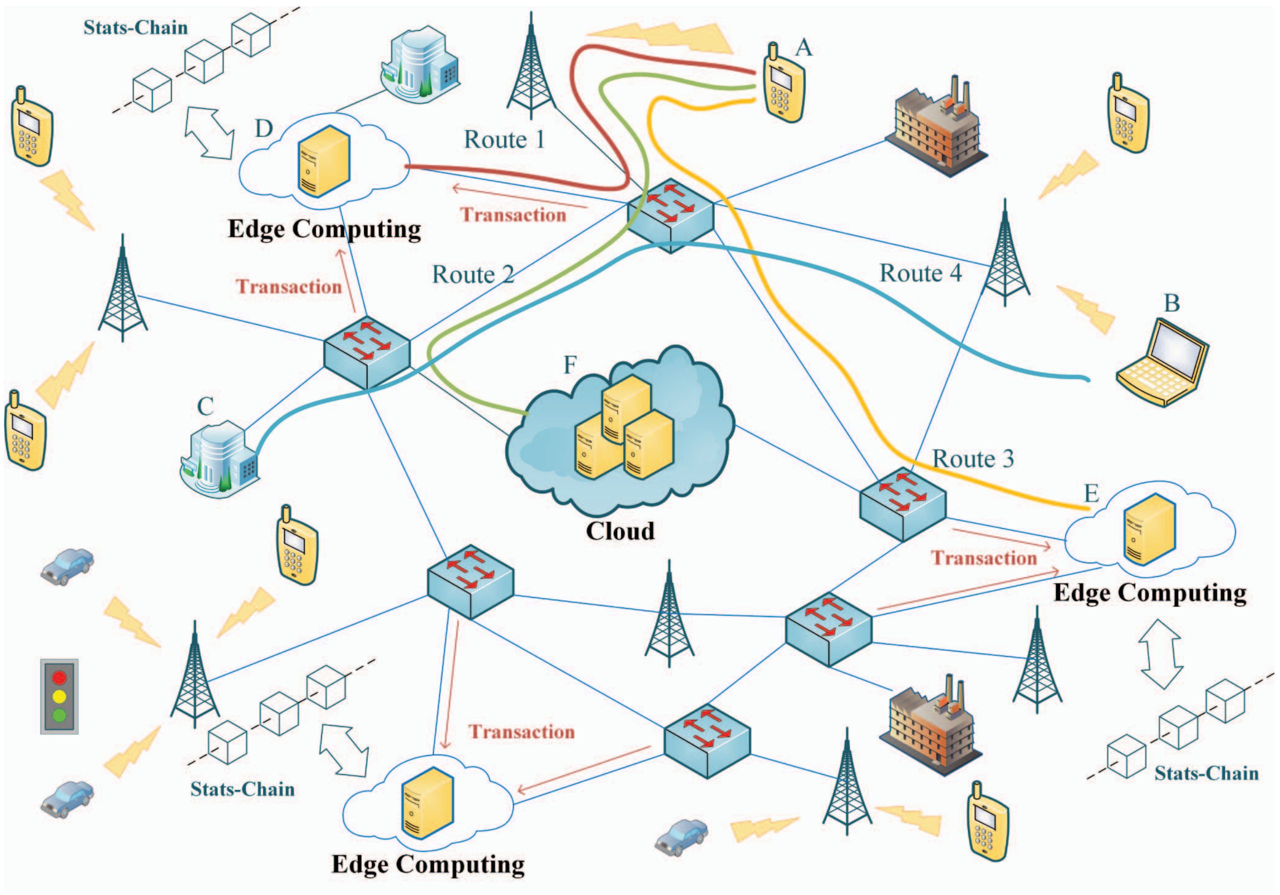


Fig. 2. Structure of mobile edge computing networks. Example routes are shown from endpoint A to computing services D, E, and F, as well as route from endpoint B to office building C.

when there is no delay between the two queues. However, this can be a problem when the system introduces delays moving the content among queues, such as the propagation delays when packets travel from one node to another, as shown in Fig. 1(b). If the time-slot interval is sufficiently small, the Lyapunov drift may depend on the control decisions made many time slots ago, which means the control policy has to make decisions minimizing the future expected Lyapunov drift. In traditional Lyapunov optimizations, the smaller the time-slot interval is, the more inaccurate the queuing dynamics modeling will be, and hence it will be more difficult to stabilize the system.

When the queuing dynamics are challenging to formulate mathematically, predictive or approximation methods are often used. de la Peña and Christofides [53] used a predictive model to resists inaccurate network state sensing which comes from the time-varying data loss. Huang *et al.* [54] used a predictive service model to predict packet arrival in future time slots. Yu and Neely [45] used the last recorded injection rate to help approximate queuing dynamics.

D. MEC and Blockchain in IoT

With the development of cloud computing, network function virtualization, software-defined networking and 5G radio access, the MEC provides computation offloading, close proximity services and the potential of better service QoE to IoT devices [5]. In particular, MEC provides computation and storage capacity to resource-limited mobile IoT devices and enables blockchain usage in these devices [55]. The usage of blockchain can establish a distributed peer-to-peer communication pattern and add security to microtransactions in IoT services [56]. Liu *et al.* [57] used blockchains to distributively deliver video streams in MEC networks. Liu *et al.* [58] used DRL to promote data exchange in the industrial IoT settings in a secure and efficient way. Yang *et al.* [59] surveyed the integration of blockchain and edge computing systems, pointing out that blockchain brings security improvement and trust to edge computing, and edge computing brings more computation and storage resources to the blockchain network.

In terms of network routing, Yang *et al.* [60] used Proof-of-Authority (PoA) consensus mechanism to build a trusted routing framework in wireless sensor networks leveraging PoA's fast transactional speed, but the validation process is limited to a small group of trusted nodes. Saad *et al.* [61] used Clique to reach a consensus of routes over border gateway protocol (BGP) nodes, which is also a PoA algorithm. Among the existing consensus mechanisms, the Proof-of-Elapsed-Time (PoET) is known to provide randomly distributed validation selecting the node with the shortest wait time as the validator [62]. Sharding can improve the scalability of a blockchain-based system [63] and it can be used in together with PoET [64].

III. System Model

A. Network Model

Fig. 2 shows the communication network structure in an MEC setting. There is heterogeneous traffic in the network. While some computation related traffic can be offloaded from the central cloud to the edge computing clusters, other traditional traffic also exists and produces nonoffloadable end-to-end traffic volumes. For example, the endpoint A may request a computation service which can be offloaded either to edge computing clusters D or E, or no offloading at all. For endpoint A, the traffic can be routed via one of routes 1, 2, and 3. Also, nonoffloadable traffic from endpoint B to office building C can be routed via a possible route 4. As the edge computing clusters are relatively immobile, it is important for the network to route the requested tasks according to the current network and computation resource status.

Let us consider a communication network represented in graph $\mathscr{G}(\mathscr{V}, \mathscr{E})$, where $\mathscr{V} = \{v_1, v_2, \dots, v_n\}$ is the set of nodes and $\mathscr{E} = \{e_1, e_2, \dots, e_m\}$ is the set of links. $\mathbf{D}$ is the link propagation delay matrix. μ is the forwarding policy tensor. At each node v_i, it keeps a queue for each destination node v_j, and the queue backlog size at time t is noted as $Y_{i,j}(t)$. For the packets whose destination is a normal endpoint, the queue backlog size is always set to zero once the packets hit their destination. For the packets whose destination is an edge computing server, the backlog size is reduced if and only if the corresponding task is finished. The system has a transmission capacity for each directed link $e = (v_i, v_j)$, and the capacity is noted as $C_{i,j}$. T denotes the time-slot interval. For each source–destination pair, the end-to-end latency is denoted by $l_{i,j}$, the throughput is $f_{i,j}$, and the delivery ratio is $u_{i,j}$. The packets' arrival is modeled by a random process, and $I_{i,j}(t)$ denotes the number of arrival packets from outside sources at time t. A list of symbols used in this article and the corresponding descriptions can be found in Table I.

TABLE I
Symbols and Definitions

Symbol	Definition
$\mathscr{G}$	The graph representing a communication network
$\mathscr{V}$	The set representing all nodes in the network
$\mathscr{E}$	The set representing all links in the network
v_i	A node in $\mathscr{V}$
e_i	A link in $\mathscr{E}$
n	The number of nodes
m	The number of links
$\mathfrak{P}^{\mathrm{M}}$	The underlying continuous-time Markov chain of a given M-state MAP or BMAP
$\mathfrak{Q}$	The infinitesimal generator of a given Markov process
$\mathfrak{D}_b$	The state transition matrix in a Markov process, where b stands for the number of packets arrived as the transition happens
$Y_{i,j}(t)$	The queue backlog size at node v_i for destination v_j at time t
$L(t)$	Lyapunov function value at time T
ΔL	Lyapunov drift
B	An upper bound for Lyapunov drift
f	Optimization objective function
V	Tradeoff preference parameter
$\mathbb{S}$	The state-space in the Markov decision process
$\mathbb{A}$	The action-space in the Markov decision process
$\mathbb{P}(s, a; s')$	The state transition probability in the Markov decision process from state s under action a to a new state s'
$\mathbb{P}^{\mathrm{c}}(s_{\mathrm{c}}, a; s'_{\mathrm{c}})$	The state transition probability in the Markov decision process caused by the control policy from state s_{c} under action a to a new state s'_{c}
$\mathbb{P}^{\mathrm{i}}(s_{\mathrm{i}}; s'_{\mathrm{i}})$	The state transition probability in the Markov decision process caused by the underlying CTMC from state s_{i} to a new state s'_{i}
$\mathscr{p}(b)$	The matrix of *hidden state* transition probability density function for a batch arrival of b packets
γ	Reward discount factor used in reinforcement learning
r	The immediate reward for the deep-Q-network
$Q(s, a)$	The expected cumulative reward for a given pair of state and action
θ	The neural network parameters used in deep-Q-networks

B. Traffic Model

Let us define the arrival of injected network traffic as a BMAP, whose underlying Markov process is $\{N(t), J(t)\}$, where $N(t)$ is the number of arrived packets at time t, and $J(t)$ is the system's state at time t. For a M-state BMAP noted by BMAP(M), $1 \le J(t) \le M$. The underlying continuous-time Markov chain (CTMC) can be written as follows:

$$\mathfrak{P}^{\mathrm{M}} = \{N(t), J(t)\}, \quad N(t) \ge 0, \quad 1 \le J(t) \le M. \tag{1}$$

In the setting of BMAP, the infinitesimal generator for process $\mathfrak{P}^{\mathrm{M}}$ can be written as follows:

$$\mathfrak{Q} = \begin{bmatrix} \mathfrak{D}_0 & \mathfrak{D}_1 & \dots & \mathfrak{D}_b & \mathfrak{D}_{b+1} & \mathfrak{D}_{b+2} & \mathfrak{D}_{b+3} & \dots \\ 0 & \mathfrak{D}_0 & \mathfrak{D}_1 & \dots & \mathfrak{D}_b & \mathfrak{D}_{b+1} & \mathfrak{D}_{b+2} & \dots \\ 0 & 0 & \mathfrak{D}_0 & \mathfrak{D}_1 & \dots & \mathfrak{D}_b & \mathfrak{D}_{b+1} & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{2}$$

Each element $\mathfrak{D}_b$ in $\mathfrak{Q}$ is an $M \times M$ matrix, giving the transition matrix between the states, and b denotes the number of packets arrived when a state transition happens. The elements in $\mathfrak{D}_b$ states the state transition rate, i.e.,

$$\mathfrak{D}_b = \begin{bmatrix} \lambda_{1,1}^b & \lambda_{1,2}^b & \cdots & \lambda_{1,M}^b \\ \lambda_{2,1}^b & \lambda_{2,2}^b & \cdots & \lambda_{2,M}^b \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{M,1}^b & \lambda_{M,2}^b & \cdots & \lambda_{M,M}^b \end{bmatrix}. \tag{3}$$

An example of the state transition in a BMAP(2) is shown in Fig. 3. Each arrowed line represents a transition between the two given states. The dashed lines show the transitions where no arrivals occur, and the solid lines show those with arrivals. The transition rate also represents the arrival rate where there are arrivals. Rate $\lambda_{i,j}^b$ stands for a transition with a batch arrival of b packets when $b > 1$.

It has been shown that MAP(M) can be analytically constructed when $M \le 3$, and a few known arrival processes, such as the Poisson process, the Erlang renewal process, and the Markov modulated Poisson process, can be represented in the form of MAP [65]. However, there is a tradeoff between

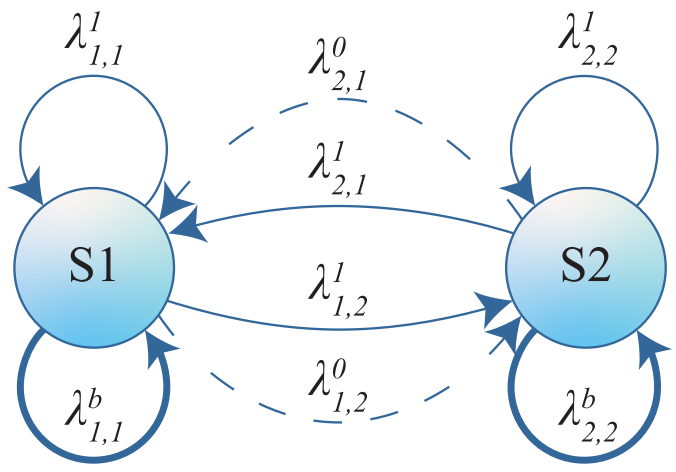


Fig. 3. State transition diagram of a 2-state BMAP. The dashed lines represent transitions between states where no arrival occurs, and the solid lines represent transitions where there are b arrivals at rate $\lambda^b_{i,j}$, $b \geq 1$.

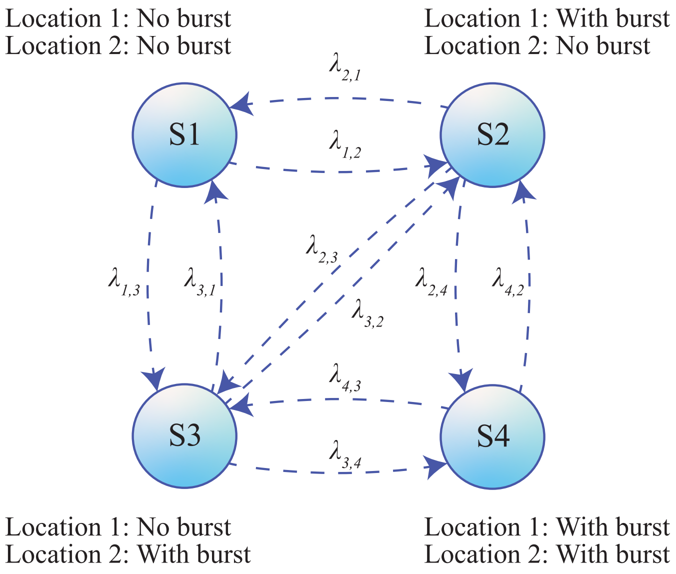


Fig. 4. State transition diagram of a 4-state BMAP. Each state describes whether there is a burst of traffic at each individual location. For simplicity, the batched arrivals are not shown here.

the tractability and the accuracy of the fitted (B)MAP models. For better accuracy, it will require a more general (B)MAP(M) where $M \gg 3$, say $M = 64$ in real-world practices [66].

Modeling the arrival of traffic in a network as a whole can be more sophisticated when considering the correlation between extra factors such as the geographical locations and application services. Fig. 4 shows an example of possible state transitions between two correlated locations when a burst of traffic occurs. Let us assume there is a social event that happens at *Location 1* with a rate $\lambda_{1,2}$ that causes a burst of traffic from *Location 1*. For a rate $\lambda_{2,4}$, some other users at *Location 2* might start generating traffic as well due to the fact that they share the same interests with the users at *Location 1* or simply because they are geographically close to each other. The heat might fade at *Location 2*, and then falls back to state $S2$ with a rate $\lambda_{4,2}$. Depending on the type of event, the transition between states will possibly go along with a different path, say $S1 \rightarrow S2 \rightarrow S3 \rightarrow S1$. Note that in this example, the states are designed to capture the correlation of interarrival times among different locations, not knowing there can be different services that the users are with, different events that trigger the burst of traffic, different groups of users, and many other dimensions that can be involved. For a complete model, the number of states needs to be the multiplication of the cardinals of every dimension, which will dramatically increase the number of states.

Because the number of states can be very large, it raises concerns for people when applying BMAP into real-world scenarios. If the number of states is too small, the fitted model will not be sufficiently accurate to guide the optimization of the system's operation. Moreover, if the number of states is too large, it will be less likely that the fitted model will be analytically tractable. Let alone the fitting process of BMAP is time consuming, knowing it requires minutes of CPU time when using EM algorithms [37].

Note that to distinguish between the states of underlying CTMC and the states of MDP in the following sections, we will call the states of underlying CTMC as the *hidden states* in the following context.

C. Queuing Dynamics Modeling

Let us first consider a queuing dynamic where the propagation delay in each link is ignored. For a forwarding queue at node i with packets for destination j, the queuing dynamics can be easily written as follows:

$$\mu^a_{i,j}(t) = \sum_{a \in \mathscr{V}, a \neq i} \mu_{a,i,j}(t) \tag{4}$$

$$\mu^d_{i,j}(t) = \sum_{b \in \mathscr{V}, b \neq i} \mu_{i,b,j}(t) \tag{5}$$

$$Y_{i,j}(t+1) = Y_{i,j}(t) + I_{i,j}(t+1) + \mu^a_{i,j}(t) - \mu^d_{i,j}(t) \tag{6}$$

where (4) is the sum of ingress packets, (5) is the sum of egress packets, and (6) gives the whole queuing dynamics. Later, the optimization framework will try to find an optimal routing policy μ using this dynamic. However, when the propagation delay of links cannot be ignored, the queuing dynamics should be updated accordingly to present a more accurate model. In this case, the ingress packets are determined by a previous routing policy multiple time slots before. Equation (4) should be modified as follows:

$$\mu^{a*}_{i,j}(t) = \sum_{a \in \mathscr{V}, a \neq i} \mu_{a,i,j}\big(t - D_{a,i}/T\big) \tag{7}$$

$$Y_{i,j}(t+1) = Y_{i,j}(t) + I_{i,j}(t+1) + \mu^{a*}_{i,j}(t) - \mu^d_{i,j}(t). \tag{8}$$

Note that only the arrival rates are updated and the departure rates remain the same in the queuing dynamics when comparing (6) and (8). This is because at any given time slot, the departure of packets from a node can always be precisely controlled while the arrival of packets is affected by the delay of a link, which is possibly unreliable and varies over time.

IV. Lyapunov Optimization-Based Deep Reinforcement Learning

A. *Lyapunov Optimization*

We use a quadratic function of the queue backlog sizes as a scalar measurement for the level of the system's instability. It can be used as a Lyapunov candidate function as

$$L(t) = \frac{1}{2}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} Y_{i,j}(t)^2. \tag{9}$$

We use the quadratics of queue backlog size because it is a widely use candidate to model and improve network stability [45], [54], [67]. Now that we have the definition of the Lyapunov function, the Lyapunov drift can be derived as follows:

$$\begin{aligned}\Delta L &= L(t+1) - L(t) &(10)\\ &= \frac{1}{2}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\Big(Y_{i,j}(t) + I_{i,j}(t) + \mu_{i,j}^{a*}(t) - \mu_{i,j}^{d}(t)\Big)^2 \\ &\quad - \frac{1}{2}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} Y_{i,j}(t)^2 &(11)\\ &\le B + \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} Y_{i,j}(t)\mathbb{E}\Big[I_{i,j}(t) + \mu_{i,j}^{a*}(t) - \mu_{i,j}^{d}(t)|\mathbf{Y}(t)\Big] &(12)\end{aligned}$$

where the upper bound is a constant defined as follows:

$$B = \frac{1}{2n}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\Big(I_{i,j} + \max_a\big(\mu_{a,i,j}\big)\Big)^2 + \Big(\max_b\big(\mu_{i,b,j}\big)\Big)^2. \tag{13}$$

If the network traffic is within the capacity region, the expectation in the right part of (12) will be negative. Let it be a number $\epsilon > 0$, i.e.,

$$\mathbb{E}\Big[I_{i,j}(t) + \mu_{i,j}^{a*}(t) - \mu_{i,j}^{d}(t)|\mathbf{Y}(t)\Big] < -\epsilon. \tag{14}$$

It means that as long as the system's queue backlog sizes are sufficiently large, say $\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} Y_{i,j}(t) \ge (B/\epsilon)$, the Lyapunov drift ΔL will always be smaller than or equal to zero, and thus pushing the system back to a stable status. In other words, the system's queue backlog size is bounded by (B/ϵ) in the long term, as in

$$\mathbb{E}\left[\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} Y_{i,j}(t)\right] < \frac{B}{\epsilon}. \tag{15}$$

To find the optimal routing control policy, we leverage the drift-plus-penalty algorithm. The objective function becomes

$$f = \Delta L + V \cdot P \tag{16}$$

where P is the penalty we want to minimize, and V is a parameter controlling the preference between system stability and penalty reduction.

However, there is a dilemma. Like (14) states, the expected departure rate is lower than the combined arrival rate. If the time-slot interval is too large, some of the queues may underrun, which means a portion of service rate may be wasted while waiting for new packets' arrival. If the time-slot interval is too small, the current arrival rate estimation would be more inaccurate, as shown in (4) and (7). If the links' propagation delay is time-invariant, it might be possible to record each queue's previous $D_{a,i}/T$-slots service rate and use it in optimization and minimize the Lyapunov bound for the worst case scenarios [50], [51]. However, in networked systems with time-variant propagation delay, this is not an option, because the future propagation delay is not known *a priori* and it needs more sophisticated methods to take future propagation delays into consideration, whether directly or indirectly.

B. *Robust Deep Reinforcement Learning Enabled by Lyapunov Optimization*

We propose a dual DRL-based method to tackle the issues mentioned above.

First, we model the control of network routing as an MDP. The system consists of a state-space $\mathbb{S}$, an action-space $\mathbb{A}$, a state-transition probability

$$\mathbb{P}\big(s, a; s'\big) = \Pr\big(s_{t+1} = s'|s_t = s, a_t = a\big) \tag{17}$$

an immediate reward function $\mathcal{R}(s, a; s')$ indicating the reward received by taking action a at state s and transitioning into new state s', and a decision policy $\mathcal{P}(s) : \mathbb{S} \to \mathbb{A}$. The problem is how to design a policy $\mathcal{P}(s)$ that maximizes the expected cumulative long-term reward

$$\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t; s_{t+1}), \quad a_t = \mathcal{P}(s_t). \tag{18}$$

This cumulative reward can be approximated using a value function $\mathcal{V}$, where the update is made along with the MDP.

$$\mathcal{V}(s) = \sum_{s'} \mathbb{P}\big(s, \mathcal{P}(s); s'\big)\big[\mathcal{R}\big(s, \mathcal{P}(s); s'\big) + \gamma\mathcal{V}\big(s'\big)\big] \tag{19}$$

Also, the policy function is updated by

$$\mathcal{P}(s) = \arg\max_a \left\{\sum_{s'} \mathbb{P}\big(s, a; s'\big)\big[\mathcal{R}\big(s, a; s'\big) + \gamma\mathcal{V}\big(s'\big)\big]\right\}. \tag{20}$$

In practice, we can try to merge the calculation of policy function into the calculation of the value function, as shown by Bellman [68], which is called the value iteration. The new update procedure is as follows:

$$\mathcal{V}_{i+1}(s) = \max_a \left\{\sum_{s'} \mathbb{P}\big(s, a; s'\big)\big[\mathcal{R}\big(s, a; s'\big) + \gamma\mathcal{V}_i\big(s'\big)\big]\right\}. \tag{21}$$

Here, i indicates the number of iterations, and eventually, the value function at both sides of the equation will converge.

However, the number of states can be so enormous as mentioned in Section III-B that it will be impractical to build a value function revealing each and every state. With the help of deep neural networks, we are able to build a model approximating this large state-space and to train a DRL agent on top of it.

Just like in an MDP, the DRL agent induces an action for any given state and adjusting the decision policy according to the rewards received from the environment. The proposed model reacts with the environment as shown in Fig. 5. To

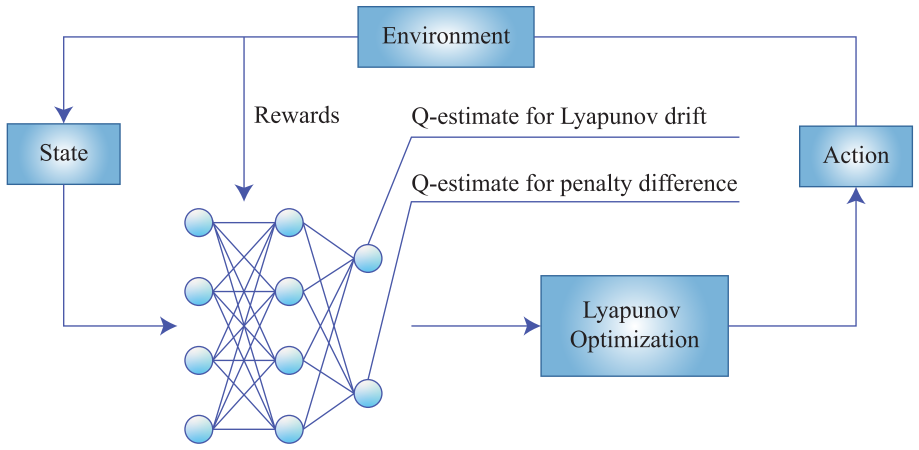

Environment
Rewards
Q-estimate for Lyapunov drift
State
Action
Q-estimate for penalty difference
Lyapunov Optimization

Fig. 5. Proposed Lyapunov-based DRL loop. The system will learn Q-values for expected cumulative Lyapunov drift and penalty, respectively. Later, the Lyapunov optimization is performed to select an action minimizing the estimated drift-plus-penalty objective.

be specific, two deep-Q-networks are used to approximate the long-term expected cumulative reward, or in other words, the value function $\mathcal{V}(s)$. To train the neural networks more efficiently, we make these two neural networks share a few layers which are connected to the input, and these shared layers can be regarded as a low-level feature extractor.

The following paragraphs will explain the state space, action space, and rewards definitions in detail.

State Space: The networked system's queue backlog sizes $\mathbf{Y}(t)$ are used as the state space. For a network with n nodes, the size of state space is $n \times n$. In addition, a list of previous control actions is included to capture delayed queuing behaviors.

Action Space: A set of service rate patterns is preconfigured, which satisfies that each queue has and only has one next-hop routing target. These service rate patterns are mapped into a discrete action space, and each action represents one service rate pattern.

Reward for Lyapunov Deep-Q-Network: The calculated current Lyapunov drift as defined in (11) is used as the immediate reward for the Lyapunov deep-Q-network, i.e.,

$$r^L = \frac{1}{2}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} Y_{i,j}(t)^2 - \frac{1}{2}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} Y_{i,j}(t-1)^2. \quad (22)$$

Penalty Deep-Q-Network: The difference between two consecutive time-slots' penalty is used as the immediate reward for the penalty deep-Q-network as

$$r^P = P(t) - P(t-1). \quad (23)$$

The deep-Q-networks are updated in a value-iteration manner by the Bellman operators, as stated in (21)

$$\Delta Q^{L*}(s,a) = \alpha\left(r^L(s,a) + \gamma \min_{a'} Q^L(s',a') - Q^L(s,a)\right) \quad (24)$$
$$\Delta Q^{P*}(s,a) = \alpha\left(r^P(s,a) + \gamma \min_{a'} Q^P(s',a') - Q^P(s,a)\right) \quad (25)$$

and an ϵ-greedy policy as shown in Algorithm 1 is used to do the exploration.

Algorithm 1: Lyapunov-Based DRL for Online Adaptive Routing Control

```
initialize deep-Q-networks $\theta^L$ and $\theta^P$;
for $t := 1, 2, \ldots$ do  Online decision loop
    Generating the control action
    Get current system status $s$;
    1 Generate random number $\eta$;
    if $\eta < \epsilon$ then
        Choose a random action $a$;
    else
        Greedy action selection
          $a^* := \arg\min_{a'} \widehat{Q^{\mathrm{L}}}(s,a') + V \cdot \widehat{Q^{\mathrm{P}}}(s,a')$;
    end
    Assign rewards
    Calculate Lyapunov function $L(T)$ ;
    Record current penalty $P(t)$;
    assign Lyapunov drift and penalty drift as rewards
    $r^L = \Delta L = L(t) - L(t-1)$;
    $r^P = P(t) - P(t-1)$;
    Update deep-Q-networks
    $\Delta Q^{\mathrm{L}*}(s,a) =$
      $\alpha(r^{\mathrm{L}}(s,a) + \gamma \min_{a'} Q^{\mathrm{L}}(s',a') - Q^{\mathrm{L}}(s,a))$;
    $\Delta Q^{\mathrm{P}*}(s,a) =$
      $\alpha(r^{\mathrm{P}}(s,a) + \gamma \min_{a'} Q^{\mathrm{P}}(s',a') - Q^{\mathrm{P}}(s,a))$;
    $\theta^L = \arg\min_{\theta^L} \sum_{s \in S^{\mathrm{batch}}}([\Delta Q^{\mathrm{L}*}(s,a)]^2)$;
    $\theta^P = \arg\min_{\theta^P} \sum_{s \in S^{\mathrm{batch}}}([\Delta Q^{\mathrm{P}*}(s,a)]^2)$;
end
```

Note that since we want to reduce the Lyapunov drift to improve system stability and to reduce the penalty to achieve smaller end-to-end latency, we use minimizing in (24) and (25) instead of maximizing in (21). When choosing the greedy action

$$a^* := \arg\min_{a'} \widehat{Q^L}(s,a') + V \cdot \widehat{Q^P}(s,a'). \quad (26)$$

Using the queue backlog size as state space gives the benefit that the learned deep-Q-network models can still perform well when facing changes in the traffic pattern. This is because any imbalance in the traffic pattern will show up in the queue backlog sizes eventually. The changes in network's propagation delay can be captured and the optimal routing control policy can be adaptively learned by the deep-Q-network models. The employment of two separated deep-Q-network models leads to the models' faster convergence, and also provides the flexibility for carriers or network managers to adjust the emphasis on system stability and penalty reduction through parameter V without the need to retrain the models from ground zero again.

C. Proof of Stationarity Under Markovian Arrival

RL methods requires the environment to be stationary for a possible (sub-) optimal policy or value function to be learned. In this section, we are going to show that the proposed method has a stationary environment and a value function can be learned if the traffic follows a Markovian arrival, by the construction of state transition probability.

Although the expression of the value iteration as in (21) omits the explicit use of state transition probability $\mathbb{P}(s, a; s')$, a valid state transition probability is still needed so that the value function update (19) and the policy function update (20) can be rewritten in the form of value iteration. This requirement is automatically satisfied if the system is self-contained and all the changes in the states are done by the interactions between the system and the control policy, or in other words, between the environment and the RL agent.

However, this requirement is not directly satisfied in the network routing environment, because the changes in the states can also come from a stochastic arrival of packets that was injected into the system with randomness and burstiness. In this section, we prove that the following lemma stands.

Lemma 1: For a networked queuing system without input, the state transition is only related to the control policy's actions, and the state transition probability $\mathbb{P}^{\mathrm{c}}(s_{\mathrm{c}}, a; s'_{\mathrm{c}})$ is deterministic from any given current state s_{c}, to next state s'_{c} under action a.

Proof: First, let us rewrite the queuing dynamics in (8). For simplicity, we show the queuing dynamics for a given source–destination pair, and thus omit i, j in subscripts. We have

$$I(t) = N(t) - N(t-1) \tag{27}$$

$$C(t) = \mu^{a*}(t) - \mu^{d}(t) \tag{28}$$

where (27) uses the cumulative arrival count $N(t)$ defined in Section III-B to represent the arrival at the $t+1$ time slot $I(t+1)$, and $C(t)$ represents the packet flow caused by the control policy as shown in (28). The queuing dynamics now become

$$Y^{C}(t+1) = Y(t) + C(t) \tag{29}$$

$$Y(t+1) = Y^{C}(t+1) + I(t+1) \tag{30}$$

where (29) reveals the queuing dynamics when only the network routing control policy is involved, and the whole queuing dynamics consist of two decoupled parts $Y^{C}(t+1)$ and $I(t+1)$ as shown in (30).

It is apparent that we can write a state transition probability function $\mathbb{P}^{\mathrm{c}}(s_{\mathrm{c}}, a; s'_{\mathrm{c}})$ for the dynamics in (29). It is because that $C(t)$ creates a deterministic mapping $a \to \mathbb{R}$ between the action and changes in queue backlog size caused by the control policy at current time slot t. Once a network routing control policy $\mathbb{S} \to \mathbb{A}$ is determined, we immediately have a mapping describing the (29), $\mathbb{S} \times \mathbb{A} \to \mathbb{S}$. ■

Lemma 2: For a networked queuing system with the MAP as input and without departure process, the state transition probability $\mathbb{P}^{\mathrm{i}}(s_{\mathrm{i}}, s_{\mathfrak{h}}; s'_{\mathrm{i}})$ from any given current state s_{i}, to next state s'_{i} can be derived, under *hidden state* $s_{\mathfrak{h}}$

$$\mathbb{P}^{\mathrm{i}}\big(s_{\mathrm{i}}, s_{\mathfrak{h}}; s'_{\mathrm{i}}\big) = \sum_{s'_{\mathfrak{h}}} \left[\frac{\mathfrak{D}_b}{\mathfrak{D}_0} \cdot \left(e^{\mathfrak{D}_0} - 1\right)\right]_{s_{\mathfrak{h}}, s'_{\mathfrak{h}}}. \tag{31}$$

Proof: For the second part involving newly arrived network traffic $I(t+1)$, we here show how the state transition probability $\mathbb{P}^{\mathrm{i}}(s_{\mathrm{i}}; s'_{\mathrm{i}})$ is calculated. Note that since $I(t+1)$ only relies on the network traffic pattern as shown in (27) and is independent of the control policy's actions, the notation of action a is removed from the state transition probability. The matrix of the *hidden state* transition probability density function can be written as

$$(b) = e^{\mathfrak{D}_0 t} \cdot \mathfrak{D}_b. \tag{32}$$

The state transition probability can be calculated as

$$\begin{aligned}
\mathbb{P}^{\mathrm{i}}\big(s_{\mathrm{i}}, s_{\mathfrak{h}}; s'_{\mathrm{i}}\big) &= \sum_{s'_{\mathfrak{h}}} \int_0^1 (b)|_{s_{\mathfrak{h}}, s'_{\mathfrak{h}}}\, dt \\
&= \sum_{s'_{\mathfrak{h}}} \int_0^1 \left[e^{\mathfrak{D}_0 t} \cdot \mathfrak{D}_b\right]_{s_{\mathfrak{h}}, s'_{\mathfrak{h}}} dt \\
&= \sum_{s'_{\mathfrak{h}}} \left[\frac{\mathfrak{D}_b}{\mathfrak{D}_0} \cdot e^{t\mathfrak{D}_0}\right]_{s_{\mathfrak{h}}, s'_{\mathfrak{h}}} \Bigg|_0^1 \\
&= \sum_{s'_{\mathfrak{h}}} \left[\frac{\mathfrak{D}_b}{\mathfrak{D}_0} \cdot \left(e^{\mathfrak{D}_0} - 1\right)\right]_{s_{\mathfrak{h}}, s'_{\mathfrak{h}}}.
\end{aligned} \tag{33}$$

In (33), we assume the shifting from current state s_{i} to the next state s'_{i} causes an arrival of b packets, that is to say, $I(t+1) = N(t+1) - N(t) = b$. Besides, it shows the state transition probability can be calculated given sufficient information about the *hidden state* $s_{\mathfrak{h}}$ which is part of the BMAP used to model the traffic pattern. ■

Lemma 3: For a networked queuing system with the MAP as input and without departure process, the state transition probability $\mathbb{P}^{\mathrm{i}}(s_{\mathrm{i}}; s'_{\mathrm{i}})$ from any given current state s_{i}, to next state s'_{i} can be derived, regardless of *hidden state* $s_{\mathfrak{h}}$.

Proof: The state s_{i} is directly observable by the RL agent, and the *hidden state* $s_{\mathfrak{h}}$ is not directly observable, which makes the problem a partially observable Markov decision process (POMDP). Researchers have studied the POMDP problems and there are many techniques to solve them [69]. One of the methods is constructing beliefs. A belief is the probability of being at *hidden state* $s_{\mathfrak{h}}$ after a series of history observations $\mathfrak{O}_t = \{s_{\mathrm{i},t}, s_{\mathrm{i},t-1}, s_{\mathrm{i},t-2}, \ldots\}$

$$\mathscr{P}\big(s_{\mathfrak{h}}, \mathfrak{O}_t\big) = \Pr\big\{s_{\mathfrak{h}} | \mathfrak{O}_t\big\}. \tag{34}$$

Over time, the belief vector $\mathscr{B}$ is updated to become closer to the actual distribution. The update of the probability of $s'_{\mathfrak{h}}$ being the actual *hidden state* at each step is given by

$$\begin{aligned}\mathscr{B}\left(s'_{\mathfrak{h}}\right) &= \Pr\left(s'_{\mathfrak{h}}|\mathscr{B}, \mathfrak{O}_{t+1}\right)\\ &= \frac{\Pr\left\{s'_{\mathfrak{h}}, \mathscr{B}, \mathfrak{O}_{t+1}\right\}}{\Pr\{\mathscr{B}, \mathfrak{O}_{t+1}\}}\\ &= \frac{\Pr\left\{\mathfrak{O}_{t+1}|s'_{\mathfrak{h}}, \mathscr{B}\right\}\Pr\left\{s'_{\mathfrak{h}}|\mathscr{B}\right\}\Pr\{\mathscr{B}\}}{\Pr\{\mathfrak{O}_{t+1}|\mathscr{B}\}\Pr\{\mathscr{B}\}}\\ &= \frac{\Pr\left\{\mathfrak{O}_{t+1}|s'_{\mathfrak{h}}, \mathscr{B}\right\}\Pr\left\{s'_{\mathfrak{h}}|\mathscr{B}\right\}}{\Pr\{\mathfrak{O}_{t+1}|\mathscr{B}\}} \end{aligned} \tag{35}$$

where

$$\begin{aligned}\Pr\left\{\mathfrak{O}_{t+1}|s'_{\mathfrak{h}}, \mathscr{B}\right\} &= \sum_{s_{\mathfrak{h}}}\mathscr{B}\left(s_{\mathfrak{h}}\right)\int_0^1 (b)|_{s_{\mathfrak{h}},s'_{\mathfrak{h}}}\,dt\\ &= \sum_{s_{\mathfrak{h}}}\mathscr{B}\left(s_{\mathfrak{h}}\right)\int_0^1 \left[e^{\mathfrak{D}_0 t}\cdot\mathfrak{D}_b\right]_{s_{\mathfrak{h}},s'_{\mathfrak{h}}}dt\\ &= \sum_{s_{\mathfrak{h}}}\mathscr{B}\left(s_{\mathfrak{h}}\right)\left[\frac{\mathfrak{D}_b}{\mathfrak{D}_0}\cdot e^{t\mathfrak{D}_0}\right]_{s_{\mathfrak{h}},s'_{\mathfrak{h}}}\Bigg|_0^1\\ &= \sum_{s_{\mathfrak{h}}}\mathscr{B}\left(s_{\mathfrak{h}}\right)\left[\frac{\mathfrak{D}_b}{\mathfrak{D}_0}\cdot\left(e^{\mathfrak{D}_0}-1\right)\right]_{s_{\mathfrak{h}},s'_{\mathfrak{h}}}\\ b &= s_{\mathfrak{i},t+1} - s_{\mathfrak{i},t}\end{aligned} \tag{36}$$

and

$$\Pr\left\{s'_{\mathfrak{h}}|\mathscr{B}\right\} = \sum_{s_{\mathfrak{h}}}\mathscr{B}\left(s_{\mathfrak{h}}\right)\mathfrak{Q}_{s_{\mathfrak{h}},s'_{\mathfrak{h}}} \tag{37}$$

$$\begin{aligned}\Pr\{\mathfrak{O}_{t+1}|\mathscr{B}\} &= \sum_{s_{\mathfrak{h}}}\mathscr{B}\left(s_{\mathfrak{h}}\right)\Pr\left(s_{\mathfrak{i},t+1}|s_{\mathfrak{i},t}, s_{\mathfrak{h}}\right)\\ &= \sum_{s_{\mathfrak{h}}}\mathscr{B}\left(s_{\mathfrak{h}}\right)\mathbb{P}^{\mathfrak{i}}\left(s_{\mathfrak{i},t}, s_{\mathfrak{h}}; s_{\mathfrak{i},t+1}\right).\end{aligned} \tag{38}$$

Here, (36), (37), and (38) show that these probabilities are all derived from the *hidden state* transition probability density function in (32) which is fixed for a given traffic pattern and can be learned implicitly by the deep neural networks.

Once the belief converges, we can extract the *hidden state* $s^*_{\mathfrak{h}}$ by

$$s^*_{\mathfrak{h}} = \arg\max_{s_{\mathfrak{h}}}\mathscr{B}\left(s_{\mathfrak{h}}\right). \tag{39}$$

As a consequence of Lemma 2, we can write the state transition probability caused by arrival traffic as

$$\begin{aligned}\mathbb{P}^{\mathfrak{i}}\left(s_{\mathfrak{i}}; s'_{\mathfrak{i}}\right) &= \mathbb{P}^{\mathfrak{i}}\left(s_{\mathfrak{i}}, s^*_{\mathfrak{h}}; s'_{\mathfrak{i}}\right)\\ &= \sum_{s'_{\mathfrak{h}}}\left[\frac{\mathfrak{D}_b}{\mathfrak{D}_0}\cdot\left(e^{\mathfrak{D}_0}-1\right)\right]_{s^*_{\mathfrak{h}},s'_{\mathfrak{h}}}.\end{aligned} \tag{40}$$

■

Theorem 1: Given an MAP as the packet arrival model and rate allocation schemes as the action-space of the control policy, there exists a valid state transition probability $\mathbb{P}(s, a; s')$ from current state s to next state s' under action a.

Proof: Provided with the state transition probability for scenarios without input and without control, given by Lemma 1 and Lemma 3, respectively, the whole state transition probability is derived by

$$\mathbb{P}\left(s, a; s'\right) = \mathbb{P}^{\mathfrak{c}}\left(s_{\mathfrak{c}}, a; s'_{\mathfrak{c}}\right) + \mathbb{P}^{\mathfrak{i}}\left(s_{\mathfrak{i}}; s'_{\mathfrak{i}}\right) \tag{41}$$

where $s = s_{\mathfrak{c}} + s_{\mathfrak{i}}$ and $s' = s'_{\mathfrak{c}} + s'_{\mathfrak{i}}$. ■

Corollary 1: Given an MAP as the packet arrival model and rate allocation schemes as the action-space of the control policy, value-iteration-based RL methods can be used to solve the optimal control problem in network routing.

Now that the state transition probability can be derived, it is safe to say that the network routing problem can be modeled as an MDP and use value-iteration-based RL method to solve it, while the BMAP-based modeling preserves the highly variant, bursty, and stochastic properties of the network traffic arrival.

In addition, since we introduce BMAP to the network traffic modeling, we can derive a more precise upper bound for the LRL-based routing in communication networks. The upper bound for new traffic arrival is

$$\begin{aligned}\overline{I(t)} = \max\ & b\\ \text{s.t.}\ & \mathbb{P}^{\mathfrak{i}}\left(s_{\mathfrak{i}}, s^*_{\mathfrak{h}}; s_{\mathfrak{i}} + b\right) > 0.\end{aligned} \tag{42}$$

Similar to (13), the upper bound for the Lyapunov drift can be derived as

$$\overline{B(t)} = \frac{1}{2n}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\left(\overline{I_{i,j}(t)} + \max_a\left(\mu_{a,i,j}\right)\right)^2 + \left(\max_b\left(\mu_{i,b,j}\right)\right)^2. \tag{43}$$

Because of the most possible *hidden state* $s^*_{\mathfrak{h}}$ is selected rather than all possible *hidden states*, the bound of drift given in (43) is tighter than the one given in (13), and the bound of stability also becomes tighter, as in

$$\mathbb{E}\left[\sum_{i=0}^{n}\sum_{j=0}^{n}Y_{i,j}(t)\right] < \frac{\overline{B}}{\epsilon}. \tag{44}$$

D. Blockchain-Based Implementation With Proof of Elapsed Time Consensus

Since the accuracy and trustworthiness of queue backlog size are critical to the selection of the proposed Lyapunov-based DRL routing method, we further design an information exchange protocol based on blockchains. As shown in Fig. 6, each node has two kinds of peers that it needs to communicate with. The greens ones are the direct neighbors, and the blue ones are the indirect peers. The peers exchange information in twofold: 1) rapidly exchange current queue backlog size with direct peers (green ones) and 2) blockchain-based information exchange with all the peers (green and blue ones).

1) For Direct Peers: The difference in backlog sizes between the direct peers builds up the backpressure to push the flow of packets to the destination, and thus it is important that the backlog sizes are updated and exchanged frequently with each pair of direct peers. To reduce overhead, only backlog sizes are exchanged.

2) For All the Peers: Each peer exchange with each other the number inbound packets, the number of outbound packets, and the current queue backlog size, at a smaller rate. Once the data from all nodes in the network is acquired, it is easy to validate if there is a node sending inconsistent information. Traditional in-band hop-by-hop message delivery may cope with malfunction nodes passively sending wrong statistics due to performance degradation, but it cannot deal with malicious nodes actively sending or alternating the information on the way. As a result, we propose to use blockchain to distribute the statistic information among the peers in the network.

The architecture is in two-folds. First, network nodes actively transmit the inbound and outbound statistics through secure HTTPs channels to the controller in a nearby MEC cluster. The controller records the statistics and prepares them to be sent to the blockchain network, and the statistics is aggregated to queue backlog size and send back to all nodes it controls within the same region. Second, the controllers submit and exchange statistics through the *Stats-Chain*, as shown in Fig. 2. Once new statistics are acquired, the controller aggregates them and updates the local copy of the queue backlog size accordingly. When conducting statistics aggregation, the controller audits the history data to check if there is any anomaly and apply security patches accordingly, whether the data is acquired directly from nodes within the region or through blockchain.

The PoET consensus mechanism is used in our blockchain using Hyperledger Sawtooth [70], which leverages Intel software guard extensions (SGX) technology as the trusted execution environment. Each peer randomly generates a wait time inside SGX enclave, register to the system, and wait for the registered time. The first peer who resumes from waiting gets to be the next leader in the blockchain system. The randomness distribution is enforced inside the SGX environment, so that no attacker can fake a short wait time without breaching the SGX beforehand.

If the trusted execution hardware is not breached, the attacker would have no choice but to follow the protocol [71], and thus each node can acquire a trustworthy copy of current network statistics. However, of course, the attacker can gain control of the blockchain and then manipulating the network statistics if the trusted execution hardware is breached. Although it is not impossible, it is considered to be far more unlikely for the trusted execution hardware to be practically breached compared with upper layer software [72]. Thus, the blockchain-based statistics information exchange provides added security, robustness, and reliability to the network.

V. Performance Evaluation

A. Experimental Setup

We compare the proposed Lyapunov-based DRL routing method against two families of algorithms, the standard backpressure routing [67] and backpressure routing with Lyapunov optimization [52]. All algorithms tested are implemented in distributed forms.

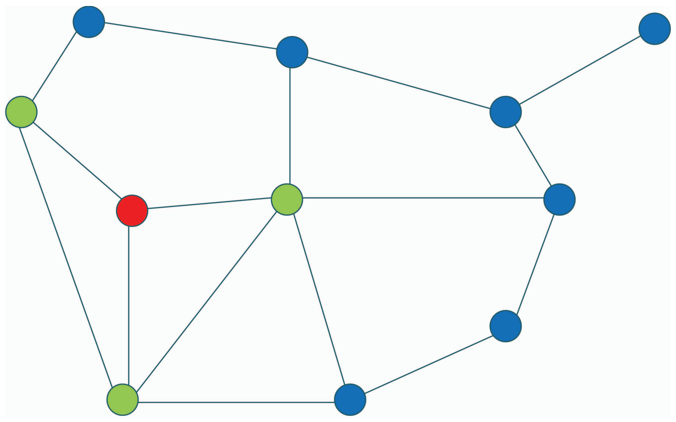


Fig. 6. Illustration of information exchange between the nodes. The source node is colored in red, and its direct neighbors are colored in green. The blue nodes represent all the other peer nodes.

We use 11 topologies to evaluate the performance of the proposed method, as shown in Fig. 7. The major experimental results are achieved using a grid topology Fig. 7(a), and the rest topologies Fig. 7(b)–(k) are used to show the results and conclusions also apply to general cases.

The link latency is 0.01 s with one exception for 0.1 s to create imbalance, and the latency of all the links will shift from -10% to 10% in cycles of 4 s. The link bandwidth is set to 100 Mb/s. The burstiness of traffic is controlled by a parameter p, which is used in an ON-OFF model to produce a burst of packets at the chance of probability p. In the experiments, each node sends traffic designated for each and every other node in the network. The topologies we used have 6 nodes, and that makes $6 \times 5 = 30$ tasks generating traffic in the network. In the setup five selected tasks generate traffic around 100 Mb/s and other tasks generate traffic at a lower rate around 2 Mb/s to serve as the background traffic. The time-slot interval T is set to 0.01 s as well. The end-to-end latency is chosen as the penalty function. And the five selected tasks generate traffic to the same destination node.

In our experiments, the neural networks used consist of five fully connected layers, in which three layers are shared common layers and two dedicated branches are for the Lyapunov part and the penalty part, respectively. In shared layers, there are 32 neurons in each layer. In each branch, there are two layers with sixteen neurons. The number of output neurons depends on how many direct neighbors that a communication node has, and it varies.

B. Evaluation Metrics

Queue Backlog Size: The queue backlog size is used to measure the system's queuing stability. Since the system starts with empty queues, a system with a smaller measured queue backlog size will be the one more stable against the burstiness in the inputs.

End-to-End Latency: Because the end-to-end latency is used as the penalty function, it is also chosen as a metric to reflect how well the system performs. A lower end-to-end latency would show that the system has better performance.

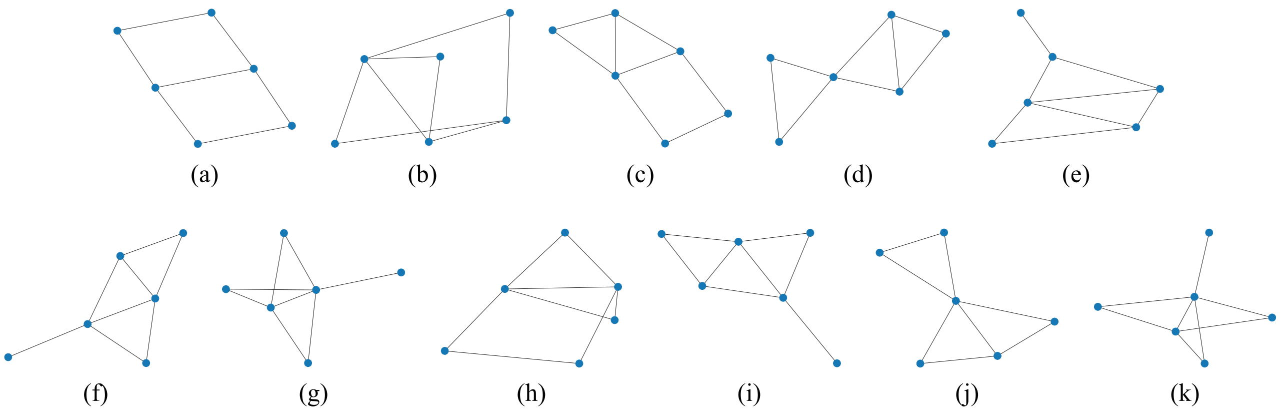


Fig. 7. Topologies used in the experiments. (a) A two-by-three grid graph topology used to present temporal and statistical results; (b)-(k) Randomly generated topologies used to further examine the effectiveness of the proposed method.

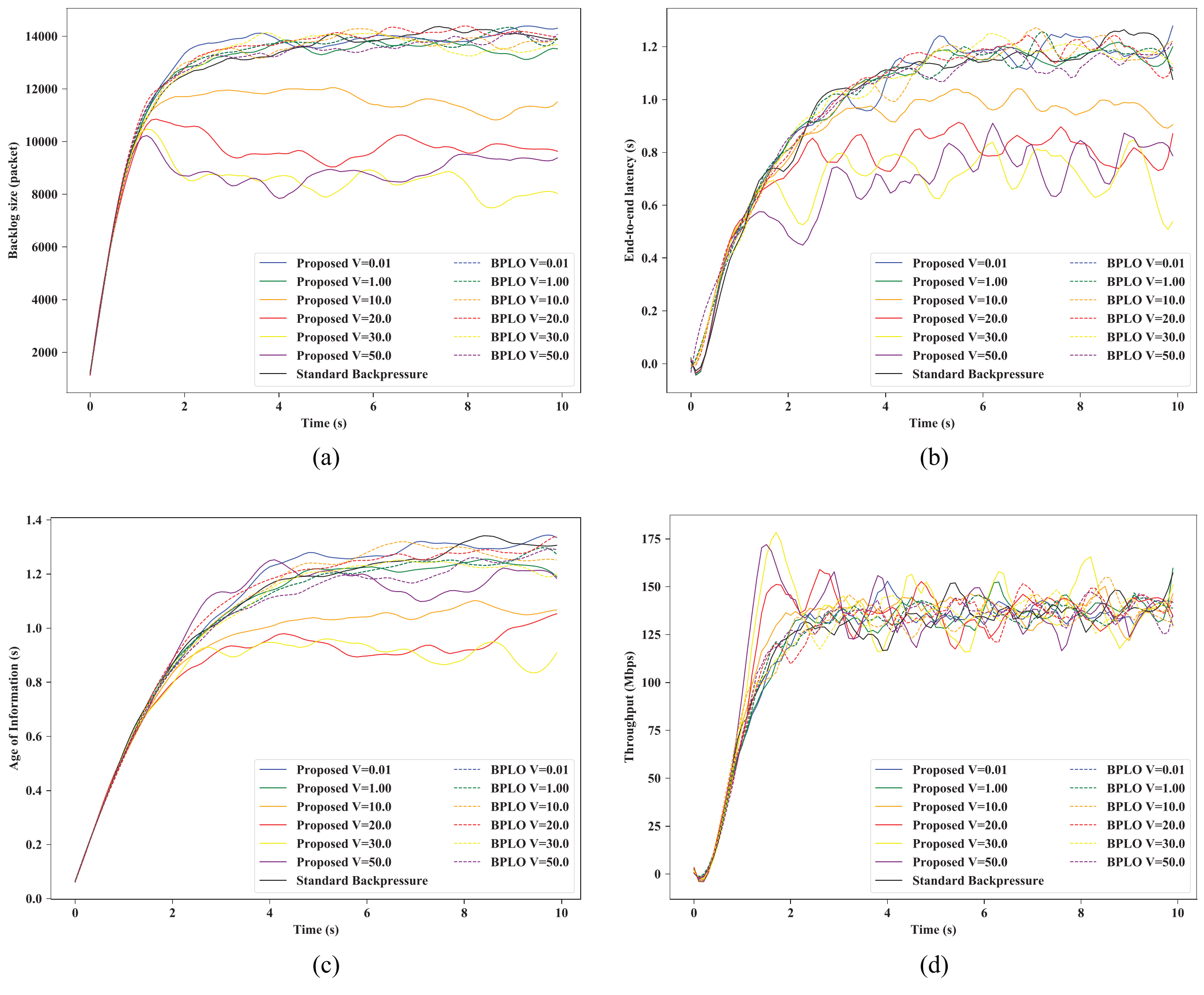


Fig. 8. Temporal results under burstiness $p = 0.5$. The results with different values of preference parameter V are shown. Data curves are smoothed using the Savitzky–Golay filter with window size 15 and polynomial order 4. BPLO stands for the backpressure routing method with Lyapunov optimization. (a) Queue backlog size versus time. (b) End-to-end latency versus time. (c) AoI versus time. (d) Throughput versus time.

Age of Information: The sum of the ages of packets since their generation over the whole network is used to measure the freshness of the network's information, or the timeliness of the network. This metric includes the age of packets that have been sent from the previous node but yet to be received by the next node. A lower AoI would show that the information inside the system is fresher and the networked system's control and routing policy provides more timeliness.

Throughput: The sum of all end-to-end throughput over the whole network. Since backpressure routing is known to

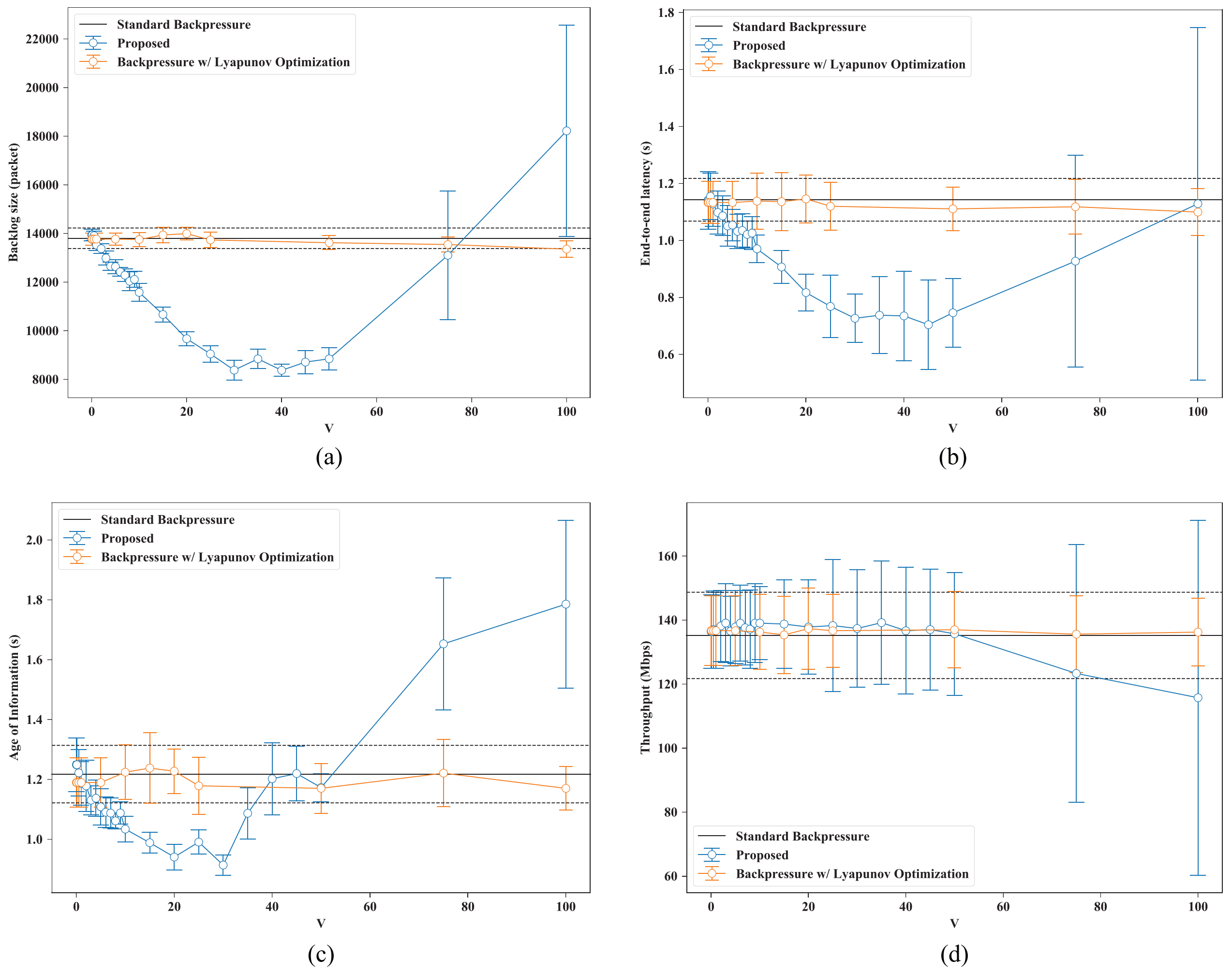


Fig. 9. Statistical results over various tested methods. Error bar shows the one standard deviation region for each data point. The solid line shows the mean evaluation metrics for the baseline backpressure algorithm, and the dashed line shows its one standard deviation region. (a) Sum of queue backlog size versus preference parameter V. (b) Average latency under different values of the parameter V. (c) Sum of the AoI over the whole network. (d) Sum of end-to-end throughput over the whole network.

provide good throughput results, it is vital to compare the tested methods in this aspect. A higher throughput shows that the system is able to route and deliver more information across the network.

C. Simulation Results and Analysis

Fig. 8 shows the temporal results of the proposed methods under different settings. Fig. 8(a)–8(d) shows the results in terms of the queue backlog size, end-to-end latency, the AoI, and the throughput, respectively. The first three figures also show that the proposed Lyapunov-based DRL routing model converges faster than existing solutions. The convergence time is also listed in Table II, where we can see an improvement as in relatively reduced convergence time ranging from 18.2% to 53.2%, depending on the metric selected.

Fig. 9 further shows the statistical results of the tested methods over different settings of parameter V. Similarly, Fig. 9(a)–(d) reveals the performance in the metric of the queue backlog size, end-to-end latency, the AoI, and the throughput, respectively. The results show that the

TABLE II
TYPICAL CONVERGENCE TIME

	Backlog size	Latency	AoI	Throughput
Backpressure	2.2 s	6.2 s	5.6 s	2.1 s
Proposed	1.8 s	2.9 s	3.9 s	2.1 s
Improvement	18.2 %	53.2 %	30.4 %	-

backpressure routing with the Lyapunov optimization performs slightly better and almost identical to the standard backpressure routing, and typically the proposed Lyapunov-based DRL routing model outperforms both the existing solutions over all the evaluated metrics.

By taking a deeper dive into the results illustrated by Fig. 9, we can find an interesting observation that although the end-to-end latency estimation is used as the penalty function, typically (when $V <= 50$), the increased emphasis on penalty reduction (increased value of V) leads to not only reduced end-to-end latency, but also better stability with reduced queue backlog size. It is because reducing the overall end-to-end latency requires the queue backlog to be sufficiently small, and thus

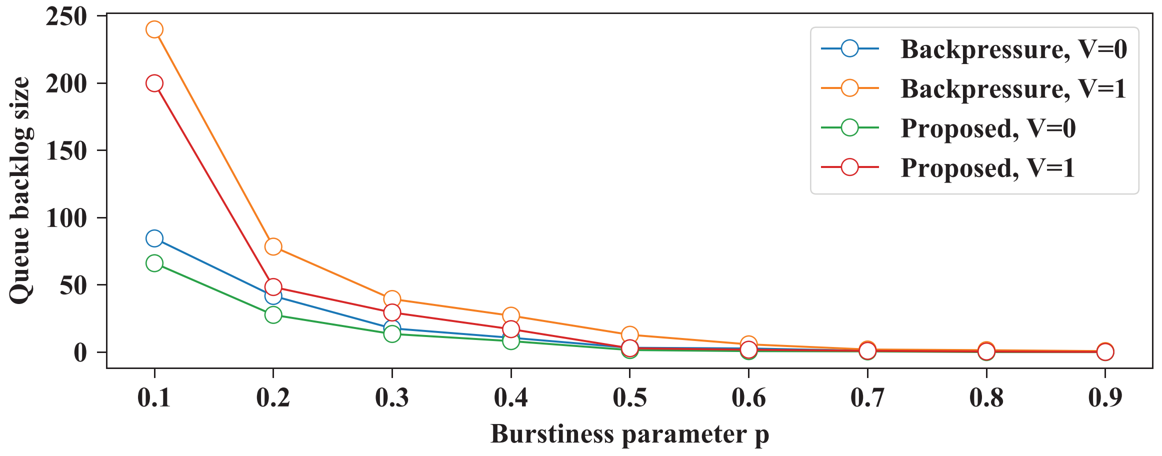


Fig. 10. Average queue backlog size versus different burstiness level governed by parameter p. The lower the value of p is, the higher the burstiness there will be in the system.

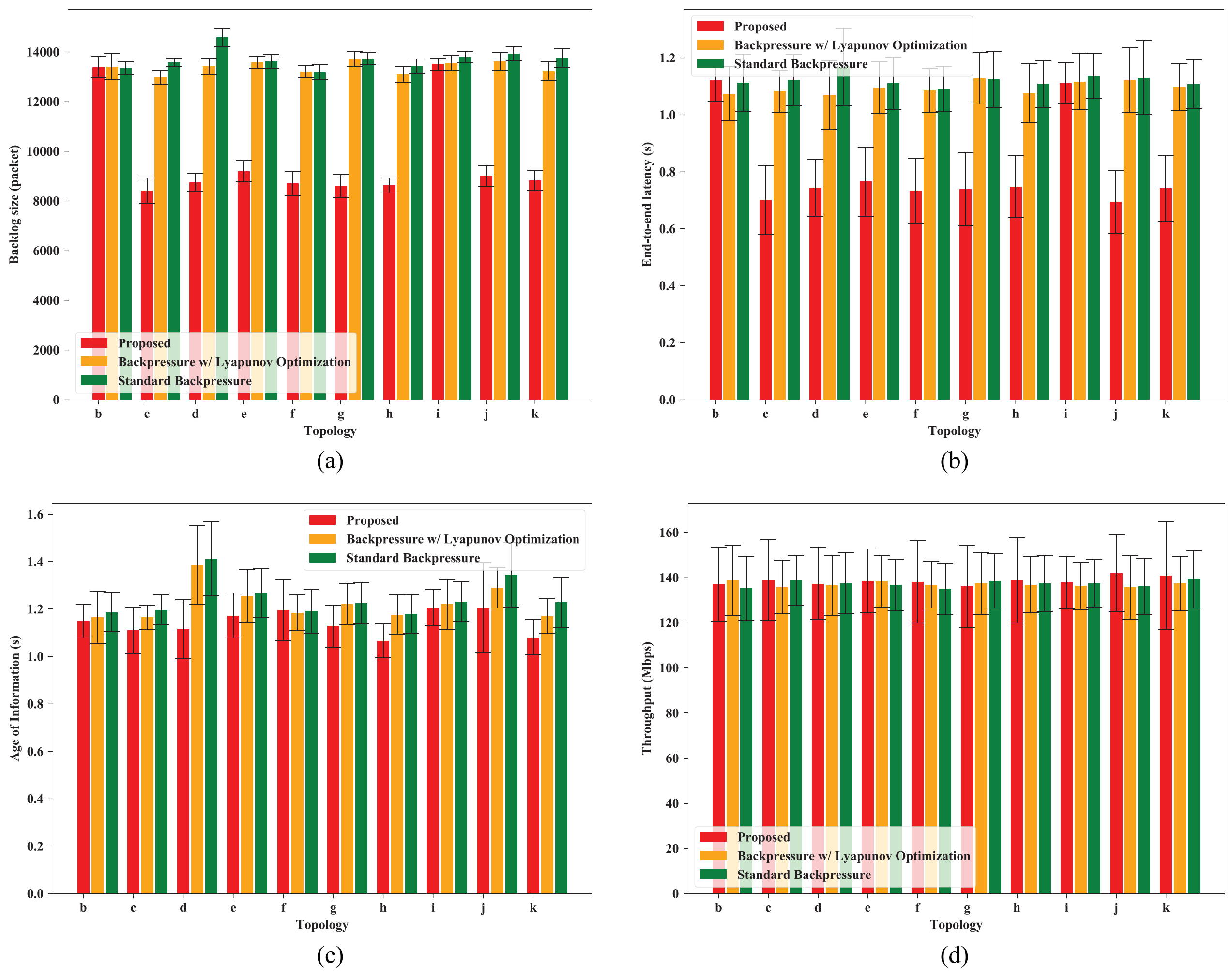


Fig. 11. Statistical results under different topologies. Error bar shows the one standard deviation region for each data point. (a) Sum of queue backlog size. (b) Average end-to-end latency. (c) Sum of the AoI over the whole network. (d) Sum of end-to-end throughput over the whole network.

enforcing the queue stability. In the existing literature, the penalty is designed to pursuing either better overall network throughput or better fairness among the services, which triggers more demand for queue backlog. This is also why we did not see a tradeoff of upper bound accuracy $O(1/V)$ for queue stability and $O(V)$ for the penalty as in other Lyapunov optimization applications.

In addition, Fig. 9(a) and (b) also shows that the performance hits a plateau when the preference parameter $V \in [30, 50]$ in terms of queue backlog size and average

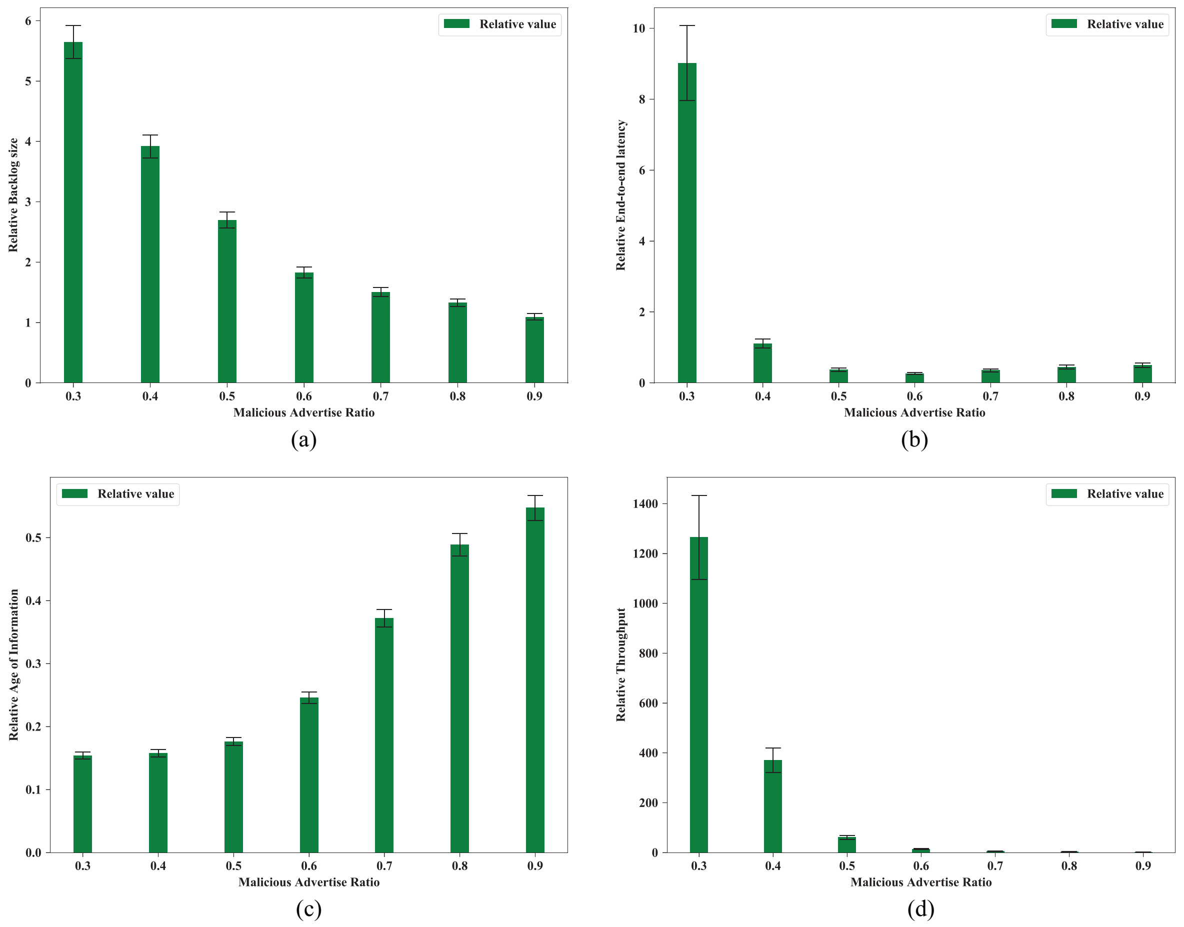


Fig. 12. Statistical results under different malicious node advertise ratio. Error bar shows the one standard deviation region for each data point. (a) Relative queue backlog size. (b) Relative end-to-end latency. (c) Relative AoI over the whole network. (d) Relative end-to-end throughput over the whole network.

end-to-end latency. This is because the algorithm becomes more and more emphasized on the latency reduction as the parameter V increases. At $V = 30$, the latency hits a bottleneck as the best forwarding links are reaching to their capacity limits. However, the proposed method still manages to stabilize the system by balancing the traffic across the whole network for $V \in [30, 50]$, as the overall AoI increases shown by Fig. 9(c). For $V > 50$, the control policy becomes overly focused on reducing latency and tasks are competing for a few key links. The links become overloaded and it results in performance degradation. The system is no longer stable, and the one standard deviation region increases dramatically.

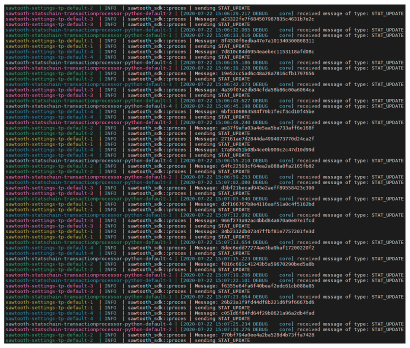
In practice, the parameter V can be changed at will, wither manually or through an automatic procedure. We have tested one possible procedure that initially set $V := 1$ at the start and gradually increase V until the end-to-end latency breaks its last-known best value. Then the parameter is reset to $V := V/2$ and repeat the procedure again.

Fig. 9(c) further shows the AoI over the network and that the aforementioned improvement comes from optimally routing the packets to different links. This metric measures not only the AoI residing in the nodes, but also those over the links. The proposed method effectively reduces the AoI as the parameter V increases from 0.01 to 30. The link transportation delay is usually much larger than the port transmission delay and thus the links can be used as a big buffer. As the parameter V goes beyond 30 and toward 50, although the performance is saturated, the links can absorb the side effects of penalty reduction being overly emphasized. The DRL module implicitly takes the links into consideration and learns a better estimation of the penalty function, thus providing better results.

Fig. 10 shows queuing stability by queue backlog size for three different models. The proposed method achieves a better result than the baseline backpressure method under all circumstances. When there is more burstiness in the system (p is smaller), the improvement in queuing stability is more significant.

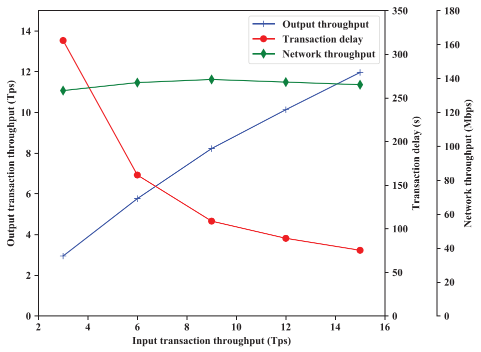
Fig. 11(a)–(d) shows the statistical results under ten different topologies from Fig. 7(b)–(k). In most cases, the proposed method outperforms the other methods in terms of smaller backlog size, smaller end-to-end latency, and higher throughput. In some cases (topology Fig. 7(b) and (i), the proposed method achieves comparable results in terms of the aforementioned metrics. In all cases, the proposed method achieves better system responsiveness as in smaller measured AoI.

Fig. 12(a)–(d) shows the relative statistical results with the proposed blockchain information exchange protocol compared to those without it. The malicious node would advertise its queue backlog size to the other peers proportional to the actual



```
sawtooth-settings-tp-default-2  | INFO  | sawtooth_sdk::proces | sending STAT_UPDATE
sawtooth-statschain-transactionprocessor-python-default-2 | [2020-07-22 15:06:29.217 DEBUG    core] received message of type: STAT_UPDATE
sawtooth-settings-tp-default-3  | INFO  | sawtooth_sdk::proces | Message: a23322fe7f684507987835c4631b7e2c
sawtooth-settings-tp-default-3  | INFO  | sawtooth_sdk::proces | sending STAT_UPDATE
sawtooth-statschain-transactionprocessor-python-default-3 | [2020-07-22 15:06:32.065 DEBUG    core] received message of type: STAT_UPDATE
sawtooth-statschain-transactionprocessor-python-default-1 | [2020-07-22 15:06:33.616 DEBUG    core] received message of type: STAT_UPDATE
sawtooth-settings-tp-default-1  | INFO  | sawtooth_sdk::proces | Message: 8f4330f6edba47e7a10134030086831l
sawtooth-settings-tp-default-1  | INFO  | sawtooth_sdk::proces | sending STAT_UPDATE
sawtooth-settings-tp-default-4  | INFO  | sawtooth_sdk::proces | Message: 7d810c84d6954eaebec1153118afd08c
sawtooth-settings-tp-default-4  | INFO  | sawtooth_sdk::proces | sending STAT_UPDATE
sawtooth-statschain-transactionprocessor-python-default-4 | [2020-07-22 15:06:35.186 DEBUG    core] received message of type: STAT_UPDATE
sawtooth-statschain-transactionprocessor-python-default-2 | [2020-07-22 15:06:39.228 DEBUG    core] received message of type: STAT_UPDATE
sawtooth-settings-tp-default-2  | INFO  | sawtooth_sdk::proces | Message: 19e52cc5ad6c48a28a7816cfb1797656
sawtooth-settings-tp-default-2  | INFO  | sawtooth_sdk::proces | sending STAT_UPDATE
sawtooth-statschain-transactionprocessor-python-default-3 | [2020-07-22 15:06:42.073 DEBUG    core] received message of type: STAT_UPDATE
sawtooth-settings-tp-default-3  | INFO  | sawtooth_sdk::proces | Message: 4a39f07a2db84cfda58b86c00a6064ca
sawtooth-settings-tp-default-3  | INFO  | sawtooth_sdk::proces | sending STAT_UPDATE
sawtooth-statschain-transactionprocessor-python-default-1 | [2020-07-22 15:06:43.627 DEBUG    core] received message of type: STAT_UPDATE
sawtooth-statschain-transactionprocessor-python-default-4 | [2020-07-22 15:06:45.198 DEBUG    core] received message of type: STAT_UPDATE
sawtooth-settings-tp-default-4  | INFO  | sawtooth_sdk::proces | Message: 6bf3106863584f70b1fecf3cd10f45be
sawtooth-settings-tp-default-4  | INFO  | sawtooth_sdk::proces | sending STAT_UPDATE
sawtooth-statschain-transactionprocessor-python-default-2 | [2020-07-22 15:06:49.240 DEBUG    core] received message of type: STAT_UPDATE
sawtooth-settings-tp-default-2  | INFO  | sawtooth_sdk::proces | Message: ae37f9afa03a4e5aa5ba733aff6e168f
sawtooth-settings-tp-default-2  | INFO  | sawtooth_sdk::proces | sending STAT_UPDATE
sawtooth-settings-tp-default-1  | INFO  | sawtooth_sdk::proces | Message: 27161ae7d2844da4994073770d24ca2f
sawtooth-settings-tp-default-1  | INFO  | sawtooth_sdk::proces | sending STAT_UPDATE
sawtooth-settings-tp-default-4  | INFO  | sawtooth_sdk::proces | Message: 17a86d51b08b4ce0b909c2c47d10d99d
sawtooth-settings-tp-default-4  | INFO  | sawtooth_sdk::proces | sending STAT_UPDATE
sawtooth-statschain-transactionprocessor-python-default-4 | [2020-07-22 15:06:55.210 DEBUG    core] received message of type: STAT_UPDATE
sawtooth-settings-tp-default-2  | INFO  | sawtooth_sdk::proces | Message: 64fc22503cf64ea2a088a8fa2165fb82
sawtooth-settings-tp-default-2  | INFO  | sawtooth_sdk::proces | sending STAT_UPDATE
sawtooth-statschain-transactionprocessor-python-default-2 | [2020-07-22 15:06:59.253 DEBUG    core] received message of type: STAT_UPDATE
sawtooth-statschain-transactionprocessor-python-default-3 | [2020-07-22 15:07:02.080 DEBUG    core] received message of type: STAT_UPDATE
sawtooth-settings-tp-default-3  | INFO  | sawtooth_sdk::proces | Message: d3bf21becad943e2aeff89558423c390
sawtooth-settings-tp-default-3  | INFO  | sawtooth_sdk::proces | sending STAT_UPDATE
sawtooth-statschain-transactionprocessor-python-default-1 | [2020-07-22 15:07:03.640 DEBUG    core] received message of type: STAT_UPDATE
sawtooth-settings-tp-default-1  | INFO  | sawtooth_sdk::proces | Message: d2f166767b8e4116aaf51a0c4f5162bd
sawtooth-settings-tp-default-1  | INFO  | sawtooth_sdk::proces | sending STAT_UPDATE
sawtooth-statschain-transactionprocessor-python-default-3 | [2020-07-22 15:07:12.092 DEBUG    core] received message of type: STAT_UPDATE
sawtooth-settings-tp-default-3  | INFO  | sawtooth_sdk::proces | Message: 966f273a92ac4bbd84a670a0e07e1fcd
sawtooth-settings-tp-default-3  | INFO  | sawtooth_sdk::proces | sending STAT_UPDATE
sawtooth-settings-tp-default-1  | INFO  | sawtooth_sdk::proces | Message: 34b2312db07347ffbf81e7757201fe3d
sawtooth-settings-tp-default-1  | INFO  | sawtooth_sdk::proces | sending STAT_UPDATE
sawtooth-statschain-transactionprocessor-python-default-1 | [2020-07-22 15:07:13.654 DEBUG    core] received message of type: STAT_UPDATE
sawtooth-settings-tp-default-4  | INFO  | sawtooth_sdk::proces | Message: 8dec6edd77274ae3ba9baf17200220f2
sawtooth-settings-tp-default-4  | INFO  | sawtooth_sdk::proces | sending STAT_UPDATE
sawtooth-statschain-transactionprocessor-python-default-4 | [2020-07-22 15:07:15.223 DEBUG    core] received message of type: STAT_UPDATE
sawtooth-settings-tp-default-2  | INFO  | sawtooth_sdk::proces | Message: f816c601c41243b5a59670290bed5a8b
sawtooth-settings-tp-default-2  | INFO  | sawtooth_sdk::proces | sending STAT_UPDATE
sawtooth-statschain-transactionprocessor-python-default-2 | [2020-07-22 15:07:19.266 DEBUG    core] received message of type: STAT_UPDATE
sawtooth-statschain-transactionprocessor-python-default-3 | [2020-07-22 15:07:22.101 DEBUG    core] received message of type: STAT_UPDATE
sawtooth-settings-tp-default-3  | INFO  | sawtooth_sdk::proces | Message: f6355e64fa6f40beaf2edc61cb088e85
sawtooth-settings-tp-default-3  | INFO  | sawtooth_sdk::proces | sending STAT_UPDATE
sawtooth-statschain-transactionprocessor-python-default-1 | [2020-07-22 15:07:23.664 DEBUG    core] received message of type: STAT_UPDATE
sawtooth-settings-tp-default-1  | INFO  | sawtooth_sdk::proces | Message: 26b23a1f9fd44df8b221d6f9f6667bd6
sawtooth-settings-tp-default-1  | INFO  | sawtooth_sdk::proces | sending STAT_UPDATE
sawtooth-settings-tp-default-4  | INFO  | sawtooth_sdk::proces | Message: c051d6f84fd64f29b0621a96a2db4fad
sawtooth-settings-tp-default-4  | INFO  | sawtooth_sdk::proces | sending STAT_UPDATE
sawtooth-statschain-transactionprocessor-python-default-4 | [2020-07-22 15:07:25.234 DEBUG    core] received message of type: STAT_UPDATE
sawtooth-statschain-transactionprocessor-python-default-2 | [2020-07-22 15:07:29.276 DEBUG    core] received message of type: STAT_UPDATE
sawtooth-settings-tp-default-2  | INFO  | sawtooth_sdk::proces | Message: 770bf794a0ee4a2ba528d4b73ffa7428
sawtooth-settings-tp-default-2  | INFO  | sawtooth_sdk::proces | sending STAT_UPDATE
```

(a)



(b)

Fig. 13. Performance of blockchain implementation tested. (a) Blockchain system screenshot. (b) Performance.

value. The malicious advertise ratio is noted in the x-axis of each figure. An extremely small value of this malicious advertise ratio would create a blackhole at the position of the malicious node and result in a congestion there. The result shows that the proposed blockchain-based network statistics exchange can protect the proposed network routing framework from this kind of attacks. Thus, the system becomes more robust and reliable.

Finally, we show the performance of blockchain implementation in Fig. 13. Five blockchain nodes are deployed and the input transaction speed ranges from 3 to 15 tps, which roughly translate to 0.6–3 samples/s. The blockchain system reaches its capacity for input transaction rate higher than 9 tps. The network throughput is not affected much as the queue backlog size information is more frequently exchange between direct neighbors, regardless of the status of the blockchain.

D. Discussion on Implementation Details

Scalability: The following methods may be of help to improve scalability.

1) The proposed method is implemented in a distributed way and there is one DRL agent per node so that the action space is reduced.
2) The computationally intensive model training is offloaded to nearby MEC clusters.
3) Potentially a compressed feature embedding can be extracted from the middle layer of jointly trained models, reducing state space on network nodes.

Routing loops: The distributed hop-by-hop implementation of DRL agents increases the risk of routing loops, and we use the following methods to mitigate it.

1) In the early bootstrap stage, use the actual queue backlog size difference to estimate the pressure as it is done in standard backpressure routing. Once the bootstrap is completed, the algorithm will switch back to using the output of DRL models.
2) When the DRL agent selects a nonoptimal route for exploration, it should also mark the packet so that the following hops do not explore again, and thus preventing routing loops caused by exploration.

VI. Conclusion

In this article, we considered the network routing control with time-variant propagation delays, and modeled it as a Lyapunov optimization problem. In addition, we identified that the time-slot interval has a big effect, and there is a trade-off between optimization performance and modeling accuracy for queuing dynamics. To tackle the issues, we proposed a DRL-based adaptive routing control method that combines Lyapunov drift and penalty and uses it as the objective function. By using BMAP to model the network traffic pattern, we prove that the network routing problem with highly variant, bursty, and stochastic input can be modeled as a Markov decision process and that we can use value-iteration-based RL methods to solve it. Moreover, the DRL module can effectively learn a better estimation of long-term Lyapunov drift and penalty functions, and thus it provides superior results in terms of the backlog size, end-to-end latency, AoI, and throughput. Furthermore, the proposed method outperforms the baseline backpressure method in multiple settings, and converges faster than existing methods. Extensive experiments also show that the proposed model performs well under various topologies, and thus the proposed model can be used in general cases. At last, the proposed blockchain-based information exchange protocol can provide trustworthy network statistics to the routing framework.

References

[1] E. Bertino and N. Islam, "Botnets and Internet of Things security," *Computer*, vol. 50, no. 2, pp. 76–79, Feb. 2017.
[2] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, Jul. 2017.
[3] S. Verma, Y. Kawamoto, Z. M. Fadlullah, H. Nishiyama, and N. Kato, "A survey on network methodologies for real-time analytics of massive IoT data and open research issues," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1457–1477, 3rd Quart., 2017.
[4] B. Liang, *Mobile Edge Computing*, vol. 16. Cambridge, U.K.: Cambridge Univ. Press, Apr. 2017, pp. 1397–1411.
[5] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1657–1681, 3rd Quart., 2017.
[6] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, "Joint service placement and request routing in multi-cell mobile edge computing networks," in *Proc. IEEE Conf. Comput. Commun.*, Paris, France, Apr. 2019, pp. 10–18.
[7] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Netw.*, vol. 33, no. 5, pp. 156–165, Sep./Oct. 2019.
[8] R. Li, L. Wang, Y. Gong, M. Song, M. Pan, and Z. Han, "Dynamic cache placement, node association, and power allocation in fog aided networks," in *Proc. IEEE Global Commun. Conf.*, Waikoloa, HI, USA, Dec. 2019, pp. 1–6.

[9] K. Thar, T. Z. Oo, Z. Han, and C. S. Hong, "Meta-learning-based deep learning model deployment scheme for edge caching," in *Proc. 15th Int. Conf. Netw. Serv. Manag. (CNSM)*, Halifax, NS, Canada, Oct. 2019, pp. 1–6.

[10] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 1, no. 2, pp. 89–103, Jun. 2015.

[11] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[12] Y. Mao, J. Zhang, S. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 16, no. 9, pp. 5994–6009, Sep. 2017.

[13] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *Proc. IEEE Conf. Comput. Commun.*, Orlando, FL, USA, Mar. 2012, pp. 2731–2735.

[14] T. Chen, S. Barbarossa, X. Wang, G. B. Giannakis, and Z.-L. Zhang, "Learning and management for Internet of Things: Accounting for adaptivity and scalability," *Proc. IEEE*, vol. 107, no. 4, pp. 778–796, Apr. 2019.

[15] A. M. Bedewy, Y. Sun, and N. B. Shroff, "Minimizing the age of information through queues," *IEEE Trans. Inf. Theory*, vol. 65, no. 8, pp. 5215–5232, Aug. 2019.

[16] M. Chen, Y. Miao, Y. Hao, and K. Hwang, "Narrow band Internet of Things," *IEEE Access*, vol. 5, pp. 20557–20577, 2017.

[17] B. Zong, C. Fan, X. Wang, X. Duan, B. Wang, and J. Wang, "6G technologies: Key drivers, core requirements, system architectures, and enabling technologies," *IEEE Veh. Technol. Mag.*, vol. 14, no. 3, pp. 18–27, Sep. 2019.

[18] Z. Zhang *et al.*, "6G wireless networks: Vision, requirements, architecture, and key technologies," *IEEE Veh. Technol. Mag.*, vol. 14, no. 3, pp. 28–41, Sep. 2019.

[19] L. Georgiadis, M. J. Neely, and L. Tassiulas, *Resource Allocation and Cross-Layer Control in Wireless Networks*. Hanover, Germany: Now Publ., Inc., 2006.

[20] M. J. Neely, "Stability and probability 1 convergence for queueing networks via Lyapunov optimization," *J. Appl. Math.*, vol. 2012, Jul. 2012, Art. no. 831909.

[21] H. Okamura, T. Dohi, and K. S. Trivedi, "Markovian arrival process parameter estimation with group data," *IEEE/ACM Trans. Netw.*, vol. 17, no. 4, pp. 1326–1339, Aug. 2009.

[22] F. Rezaei, A. Momeni, and B. H. Khalaj, "Delay analysis of network coding in multicast networks with markovian arrival processes: A practical framework in cache-enabled networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 8, pp. 7577–7584, Aug. 2018.

[23] M. Gadaleta, F. Chiariotti, M. Rossi, and A. Zanella, "D-DASH: A deep Q-learning framework for DASH video streaming," *IEEE Trans. Cogn. Commun. Netw.*, vol. 3, no. 4, pp. 703–718, Dec. 2017.

[24] S. Chinchali *et al.*, "Cellular network traffic scheduling with deep reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, New Orleans, LA, USA, Apr. 2018, pp. 766–774.

[25] C. Tunc and N. Akar, "Markov fluid queue model of an energy harvesting IoT device with adaptive sensing," *Perform. Eval.*, vol. 111, pp. 1–16, May 2017.

[26] H. Ye and G. Y. Li, "Deep reinforcement learning for resource allocation in V2V communications," in *Proc. IEEE Int. Conf. Commun.*, Kansas City, MO, USA, May 2018, pp. 1–6.

[27] C. M. Lagoa, H. Che, and B. A. Movsichoff, "Adaptive control algorithms for decentralized optimal traffic engineering in the Internet," *IEEE/ACM Trans. Netw.*, vol. 12, no. 3, pp. 415–428, Jun. 2004.

[28] D.-Y. Kim, S. Kim, H. Hassan, and J. H. Park, "Adaptive data rate control in low power wide area networks for long range IoT services," *J. Comput. Sci.*, vol. 22, pp. 171–178, Sep. 2017.

[29] Z. Zhou, T. Zhang, and A. Kwatra, "NFV closed-loop automation experiments using deep reinforcement learning," in *Proc. IEEE Conf. Comput. Commun. Workshops*, Paris, France, Apr. 2019, pp. 696–701.

[30] X. Fu, F. R. Yu, J. Wang, Q. Qi, and J. Liao, "Service function chain embedding for NFV-enabled IoT based on deep reinforcement learning," *IEEE Commun. Mag.*, vol. 57, no. 11, pp. 102–108, Nov. 2019.

[31] X. Wang, C. Wang, X. Li, V. C. M. Leung, and T. Taleb, "Federated deep reinforcement learning for Internet of Things with decentralized cooperative edge caching," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9441–9455, Oct. 2020.

[32] Q. Guan, F. R. Yu, S. Jiang, V. C. Leung, and H. Mehrvar, "Topology control in mobile ad hoc networks with cooperative communications," *IEEE Wireless Commun.*, vol. 19, no. 2, pp. 74–79, Apr. 2012.

[33] Q. Guan, F. R. Yu, S. Jiang, and V. C. Leung, "Joint topology control and authentication design in mobile ad hoc networks with cooperative communications," *IEEE Trans. Veh. Technol.*, vol. 61, no. 6, pp. 2674–2685, Jul. 2012.

[34] G. Stampa, M. Arias, D. Sanchez-Charles, V. Muntés-Mulero, and A. Cabellos, "A deep-reinforcement learning approach for software-defined networking routing optimization," Sep. 2017. [Online]. Available: arXiv:1709.07080.

[35] J. Suarez-Varela *et al.*, "Feature engineering for deep reinforcement learning based routing," in *Proc. IEEE Int. Conf. Commun.*, Shanghai, China, May 2019, pp. 1–6.

[36] Z. Xu *et al.*, "Experience-driven networking: A deep reinforcement learning based approach," in *Proc. IEEE Conf. Comput. Commun.*, Honolulu, HI, USA, Apr. 2018, pp. 1871–1879.

[37] A. Klemm, C. Lindemann, and M. Lohmann, "Modeling IP traffic using the batch Markovian arrival process," *Perform. Eval.*, vol. 54, no. 2, pp. 149–173, Oct. 2003.

[38] M. J. Neely, E. Modiano, and C. E. Rohrs, "Power allocation and routing in multibeam satellites with time-varying channels," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 138–152, Feb. 2003.

[39] I. Kadota and E. Modiano, "Minimizing the age of information in wireless networks with stochastic arrivals," *IEEE Trans. Mobile Comput.*, early access, Dec. 16, 2019, doi: 10.1109/TMC.2019.2959774.

[40] A. M. González, A. S. Roque, and J. García-González, "Modeling and forecasting electricity prices with input/output hidden Markov models," *IEEE Trans. Power Syst.*, vol. 20, no. 1, pp. 13–24, Feb. 2005.

[41] R. E. Kalman and J. E. Bertram, "Control system analysis and design via the 'second method' of Lyapunov: I—Continuous-time systems," *J. Basic Eng.*, vol. 82, no. 2, pp. 371–393, Jun. 1960.

[42] Y. Mao, J. Zhang, and K. B. Letaief, "A Lyapunov optimization approach for green cellular networks with hybrid energy supplies," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 12, pp. 2463–2477, Dec. 2015.

[43] L. Zheng and L. Cai, "A distributed demand response control strategy using Lyapunov optimization," *IEEE Trans. Smart Grid*, vol. 5, no. 4, pp. 2075–2083, Jul. 2014.

[44] M. J. Neely, *Stochastic Network Optimization With Application to Communication and Queueing Systems*. San Rafael, CA, USA: Morgan Claypool Publ., 2010.

[45] H. Yu and M. J. Neely, "A new backpressure algorithm for joint rate control and routing with vanishing utility optimality gaps and finite queue lengths," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1605–1618, Aug. 2018.

[46] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," in *Proc. IEEE Conf. Decis. Control*, Honolulu, HI, USA, Dec. 1990, pp. 2130–2132.

[47] A. Kabou, N. Nouali-Taboudjemat, S. Djahel, S. Yahiaoui, and O. Nouali, "Lifetime-aware backpressure—A new delay-enhanced backpressure-based routing protocol," *IEEE Syst. J.*, vol. 13, no. 1, pp. 42–52, Mar. 2019.

[48] M. J. Neely and R. Urgaonkar, "Optimal backpressure routing for wireless networks with multi-receiver diversity," *Ad Hoc Netw.*, vol. 7, no. 5, pp. 862–881, Mar. 2009.

[49] L. Ying, S. Shakkottai, A. Reddy, and S. Liu, "On combining shortest-path and back-pressure routing over multihop wireless networks," *IEEE/ACM Trans. Netw.*, vol. 19, no. 3, pp. 841–854, Jun. 2011.

[50] N. Stefanovic and L. Pavel, "A stability analysis with time-delay of primal–dual power control in optical links," *Automatica*, vol. 45, no. 5, pp. 1319–1325, May 2009.

[51] M. J. Neely, "Opportunistic scheduling with worst case delay guarantees in single and multi-hop networks," in *Proc. IEEE Int. Conf. Comput. Commun.*, Shanghai, China, Apr. 2011, pp. 1728–1736.

[52] J. Núñez-Martínez and J. Mangues-Bafalluy, "Distributed Lyapunov drift-plus-penalty routing for WiFi mesh networks with adaptive penalty weight," in *Proc. IEEE Int. Symp. WoWMoM*, San Francisco, CA, USA, Jun. 2012, pp. 1–6.

[53] D. M. de la Peña and P. D. Christofides, "Lyapunov-based model predictive control of nonlinear systems subject to data losses," *IEEE Trans. Autom. Control*, vol. 53, no. 9, pp. 2076–2089, Oct. 2008.

[54] L. Huang *et al.*, "When backpressure meets predictive scheduling," *IEEE/ACM Trans. Netw.*, vol. 24, no. 4, pp. 2237–2250, Aug. 2016.

[55] Z. Xiong, Y. Zhang, D. Niyato, P. Wang, and Z. Han, "When mobile blockchain meets edge computing," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 33–39, Aug. 2018.

[56] M. S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli, and M. H. Rehmani, "Applications of blockchains in the Internet of Things: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1676–1717, 2nd Quart., 2019.

[57] M. Liu, Y. Teng, F. R. Yu, V. C. M. Leung, and M. Song, "A mobile edge computing (MEC)-enabled transcoding framework for blockchain-based video streaming," *IEEE Wireless Commun.*, vol. 27, no. 2, pp. 81–87, Apr. 2020.

[58] M. Liu, F. R. Yu, Y. Teng, V. C. Leung, and M. Song, "Performance optimization for blockchain-enabled industrial Internet of Things (IIoT) systems: A deep reinforcement learning approach," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3559–3570, Jun. 2019.

[59] R. Yang, F. R. Yu, P. Si, Z. Yang, and Y. Zhang, "Integrated blockchain and edge computing systems: A survey, some research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1508–1532, 2nd Quart., 2019.

[60] J. Yang, S. He, Y. Xu, L. Chen, and J. Ren, "A trusted routing scheme using blockchain and reinforcement learning for wireless sensor networks," *Sensors*, vol. 19, no. 4, p. 970, Feb. 2019.

[61] M. Saad, A. Anwar, A. Ahmad, H. Alasmary, M. Yuksel, and A. Mohaisen, "RouteChain: Towards blockchain-based secure and efficient BGP routing," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency*, Seoul, South Korea, May 2019, pp. 210–218.

[62] W. Wang *et al.*, "A survey on consensus mechanisms and mining strategy management in blockchain networks," *IEEE Access*, vol. 7, pp. 22328–22370, 2019.

[63] J. Li, T. Liu, D. Niyato, P. Wang, J. Li, and Z. Han, "Contract-based approach for security deposit in blockchain networks with shards," in *Proc. IEEE Int. Conf. Blockchain*, Atlanta, GA, USA, Jul. 2019, pp. 75–82.

[64] H. Dang, T. T. A. Dinh, D. Loghin, E.-C. Chang, Q. Lin, and B. C. Ooi, "Towards scaling blockchain systems via sharding," in *Proc. Int. Conf. Manage. Data*, Amsterdam, The Netherlands, Jun. 2019, pp. 123–140.

[65] G. Casale, "Building accurate workload models using markovian arrival processes," in *Proc. ACM SIGMETRICS Joint Int. Conf. Meas. Model. Comput. Syst.*, San Jose, CA, USA, Jun. 2011, pp. 357–358.

[66] S. Pacheco-Sanchez, G. Casale, B. Scotney, S. McClean, G. Parr, and S. Dawson, "Markovian workload characterization for QoS prediction in the cloud," in *Proc. IEEE Int. Conf. Cloud Comput.*, Washington, DC, USA, Jul. 2011, pp. 147–154.

[67] C. Liaskos, X. Dimitropoulos, and L. Tassiulas, "Backpressure on the backbone: A lightweight, non-intrusive traffic engineering approach," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 1, pp. 176–190, Mar. 2017.

[68] R. Bellman, "A markovian decision process," *J. Math. Mech.*, vol. 6, no. 5, pp. 679–684, Apr. 1957.

[69] G. Shani, J. Pineau, and R. Kaplow, "A survey of point-based POMDP solvers," *Auton. Agents Multi-Agent Syst.*, vol. 27, no. 1, pp. 1–51, Jul. 2013.

[70] K. Olson, M. Bowman, J. Mitchell, S. Amundson, D. Middleton, and C. Montgomery, *Sawtooth: An Introduction*, Linux Found., San Francisco, CA, USA, Jan. 2018.

[71] L. Chen, L. Xu, N. Shah, Z. Gao, Y. Lu, and W. Shi, "On security analysis of proof-of-elapsed-time (PoET)," in *Proc. Int. Symp. Stabilization Safety Security Distrib. Syst.*, Oct. 2017, pp. 282–297.

[72] A. Corso, "Performance analysis of proof-of-elapsed-time (PoET) consensus in the sawtooth blockchain framework," M.S. thesis, Dept. Comput. Inf. Sci., Univ. Oregon, Eugene, OR, USA, 2019.

Zirui Zhuang (Member, IEEE) received the B.S. degree in electronic information engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 2015, where he is currently pursuing the Ph.D. degree in information and communication engineering with the State Key Laboratory of Networking and Switching Technology.

In 2019, he visited the Department of Electrical and Computer Engineering, University of Houston, Houston, TX, USA, doing research on network routing and optimization. His research interests include network routing and management for next generation network infrastructures, using machine learning and artificial intelligence techniques, including deep learning, reinforcement learning, graph representation, multiagent system, and Lyapunov-based optimization.

Jingyu Wang (Member, IEEE) received the Ph.D. degree from Beijing University of Posts and Telecommunications, Beijing, China, in 2008.

He is currently a Professor with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. He has published more than 50 papers in international journal, including *IEEE Communication Magazine*, and IEEE SYSTEMS JOURNAL. His research interests span broad aspects of SDN, big data processing and transmission, overlay networks, multimedia services, and traffic engineering.

Qi Qi (Member, IEEE) received the Ph.D. degree from Beijing University of Posts and Telecommunications, Beijing, China, in 2010.

She is currently an Associate Professor with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. She has published more than 30 papers in international journal, and obtained two National Natural Science Foundations of China. Her research interests include edge computing, mobile cloud computing, Internet of Things, ubiquitous services, deep learning, and deep reinforcement learning.

Jianxin Liao (Member, IEEE) received the Ph.D. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 1996.

He is currently the Dean of Network Intelligence Research Center and a Full Professor with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China. He has published hundreds of research papers and several books. His main research interests include cloud computing, mobile intelligent network, service network intelligent, networking architectures and protocols, and multimedia communication.

Dr. Liao has won a number of prizes in China for his research achievements, which include the Premier's Award of Distinguished Young Scientists from National Natural Science Foundation of China in 2005, and the Specially-invited Professor of the "Yangtse River Scholar Award Program" by the Ministry of Education in 2009.

Zhu Han (Fellow, IEEE) received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 1997, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Maryland, College Park, MD, USA, in 1999 and 2003, respectively.

From 2000 to 2002, he was an R&D Engineer with JDSU, Germantown, Maryland, Germantown, MD, USA. From 2003 to 2006, he was a Research Associate with the University of Maryland. From 2006 to 2008, he was an Assistant Professor with Boise State University, Boise, ID, USA. He is currently a John and Rebecca Moores Professor with the Electrical and Computer Engineering Department as well as in the Computer Science Department, University of Houston, Houston, TX, USA. His research interests include wireless resource allocation and management, wireless communications and networking, game theory, big data analysis, security, and smart grid.

Dr. Han received the NSF Career Award in 2010, the Fred W. Ellersick Prize of the IEEE Communication Society in 2011, the EURASIP Best Paper Award for the Journal on Advances in Signal Processing in 2015, IEEE Leonard G. Abraham Prize in the field of Communications Systems (best paper award in IEEE JSAC) in 2016, and several best paper awards in IEEE conferences. He is also the winner of 2021 IEEE Kiyo Tomiyasu Award, for outstanding early to mid-career contributions to technologies holding the promise of innovative applications, with the following citation: "for contributions to game theory and distributed management of autonomous communication networks." He was an IEEE Communications Society Distinguished Lecturer from 2015 to 2018, and has been an AAAS Fellow since 2019 and an ACM distinguished Member since 2019. He has been 1% highly cited researcher since 2017 according to Web of Science.