Network Calculus-based Routing and Scheduling in Software-defined Industrial Internet of Things

Conference Paper · June 2022

DOI: 10.1109/INDIN51773.2022.9976177

Luyue Ji
Zhejiang University
8 PUBLICATIONS **224** CITATIONS
SEE PROFILE

Chaojie Gu
Zhejiang University
35 PUBLICATIONS **342** CITATIONS
SEE PROFILE

Jichao Bi
Chongqing University
25 PUBLICATIONS **103** CITATIONS
SEE PROFILE

Shibo He
Zhejiang University
192 PUBLICATIONS **5,756** CITATIONS
SEE PROFILE

Network Calculus-based Routing and Scheduling in Software-defined Industrial Internet of Things

Luyue Ji, Wenjie Wu, Chaojie Gu, Jichao Bi, Shibo He, Zhiguo Shi
The State Key Laboratory of Industrial Control Technology
Zhejiang University, Hangzhou, China
{july_fish, wuwenj, gucj, jonny.bijichao, s18he, shizg}@zju.edu.cn

***Abstract*—With the emergence of Industry 5.0, it is significant to enable efficient cooperation between humans and machines in the Industrial Internet of Things (IIoT). However, achieving real-time and reliable transmission of data flows deriving from time-sensitive applications in IIoT remains an open challenge. In this paper, we propose a three-layer software-defined IIoT (SDIIoT) architecture to enable multiple industrial services and flexible network configuration. In particular, when network services change frequently in SDIIoT, the delay of the control plane has a great influence on the end-to-end delay of data flows. To address this issue, we portray two different service curves of OpenFlow switches to adapt to dynamic network status based on Network Calculus (NC). To elevate resource efficiency and comply with friendly environments, we minimize the total worst-case network cost under strict resource constraints and transmission requirements by exploiting the joint flow routing and scheduling algorithm (JFRSA). Our numerical simulation results demonstrate the effectiveness and efficiency of our solution.**

***Index Terms*—Network calculus, Industrial Internet of Things, Software-Defined Networking, network configuration, routing**

I. Introduction

Industry 5.0, the new generation of the value-driven industrial revolution, aims to establish a sustainable, human-centric, and resilient industry [1]. It focuses on enabling creative human experts to collaborate with efficient and precise machines with emerging technologies [2], e.g., Internet of Things, artificial intelligence, digital twin. The human-machine collaboration requires reliable and low latency data interactions. Thus, Industry 5.0 poses stringent performance requirements for the industrial Internet of Things (IIoT) system [3].

On the one hand, in order to improve the connectivity and manageability of the IIoT system, software-defined networking (SDN) is applied for enabling multi-service communication, flexible equipment control, and orchestration. SDN decouples the network's control plane and data plane, which offers flexible, dynamic, and programmable functionality of the IIoT system. The software-defined industrial Internet of Things (SDIIoT) allows runtime network configuration and provides application-dependent Service Level Agreements (SLAs). On the other hand, time-sensitive applications are increasingly available in Industry 5.0. It is important to ensure they have guaranteed end-to-end (E2E) delay, a crucial parameter of Quality of Service (QoS). Previous studies adopted Queuing theory (QT) [4] for network E2E delay analysis. However, QT assumes that the network is in a steady state to quantitatively describe the network features. The analytical results computed by QT only reflect the average rather than the upper bound of the network E2E delay, which means QT is not applicable for time-sensitive applications. To address the issue, researchers proposed to use Network Calculus (NC) [5] to precisely calculate the upper bound of the E2E network delay. NC models the arrival curve of network traffic and the service curve of network servers to portray the flow behavior of network elements. With the assistance of the arrival curve and service curve, we can calculate the upper bound of the delay and the backlog bound of flows on SDN switches, which can further guarantee service reliability and real-time performance.

However, existing studies on NC do not investigate the impact of control plane delay in an SDN system. Compared with the data plane delay, the control plane delay is negligible when the number and types of services are stable. In practice, when network services change frequently, it costs the SDN controller extra time to re-compute and update flow tables when a new flow enters the network. The SDN switch drops many packets due to the mismatch in its out-of-date flow table, and thus exhibits a non-constant pattern in its service curve. Therefore, the network controller cannot accurately estimate the service rate and the upper bound of the E2E delay using NC, and fail to schedule the time-sensitive services.

To this end, in this paper, we consider states of network services, i.e., packet hit and loss, and characterize the corresponding service curves in SDN switches to accurately estimate the upper bound of the E2E delay. We further design an NC-based routing and scheduling scheme for SDIIoT. Since time-critical applications occupy plenty of network resources, we jointly optimize the routing and scheduling strategies of multi-service flows to minimize the total network costs, subject to resource constraints and delay constraints. We propose a heuristic algorithm to solve the objective problem and verify its efficiency using simulation with multiple metrics. The main contributions of this work are summarized as follows:

- Exploiting SDN's logical centrality and programmability, we propose an SDIIoT system architecture to support multiple industrial services with diverse priorities and SLAs.

This work was partially supported by National Key R&D Program of China under Grant 2018YFB1702100, the National Natural Science Foundation of China under Grant U1909207, the State Key Laboratory of Industrial Control Technology, Zhejiang University, China (No.ICT2022A01), and the Major Key Project of PCL under Grant PCL2021A09.

- Considering that the packet forwarding process has different states (i.e., hit or loss), we formulate corresponding service curves of the SDN switch to accurately estimate the upper bound of the E2E delay for time-sensitive services.
- We propose a low-complexity heuristic routing and scheduling algorithm for time-sensitive services to minimize the network cost of an SDIIoT system with limited resources.

The rest of this paper is organized as follows. §II reviews the related literature. §III presents the system model of the proposed SDIIoT architecture. §IV formulates the optimization problem and designs a heuristic algorithm. §V evaluates the performance of the proposed algorithm. §VI concludes this paper.

II. Related Work

A. Model optimization on device service curves

A novel NC-based approach is demonstrated [6] to model SDN-based devices despite their increased complexity. Specifically, fundamental modules of SDN devices are modeled and combined into a single model, which achieves modeling a maximal amount of devices with a minimal number of measurements. Geyer et al. [7] extended Tandem Matching Analysis through coupling graph-based neural networks with NC to achieve almost constant execution times and lower maximum relative error. Deterministic NC model, distance graph, and resource residual graph for the SDN network are implemented [8] to improve the network performance. Furthermore, the hierarchical token-bucket queuing discipline is applied to output ports to manage bandwidth resources and the E2E QoS routing algorithm was implemented in Mininet [8]. Koohanestani et al. [9] estimate the delay bound using service curves according to the similarities between the caches and flow tables in OpenFlow switches.

B. E2E communication network performance optimization

For industrial scenarios, the worst-case E2E network delay performance is crucial to realize industrial time-critical applications. Bouillard [10] computes deterministic performance bounds in FIFO networks using NC and proposes a new algorithm based on linear programming contributing to a trade-off between accuracy and tractability. Zhao et al. [11] propose an extensional timing analysis method in Time-Sensitive Networking (TSN) to obtain the credit bounds for multiple classes of flow with frozen and non-frozen behaviors of credit during guard band. Combining credit-based shaper and strict priority, latency upper bound is portrayed [12] for industrial automation scenario based on NC for the specific QoS-based shaping mechanism of TSN. A service discipline of dynamic bandwidth scheduling within OpenFlow switches is proposed in SDN [13] to dynamically allocate data rates to flows regarding QoS.

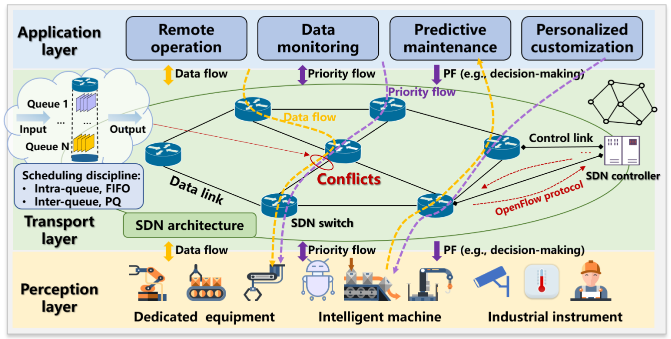

Fig. 1. The software-defined industrial Internet of Things (SDIIoT) system.

III. System model

A. Arrival curve

As shown in Fig. 1, the SDIIoT system architecture has three layers, i.e., the perception layer, transport layer, and application layer. In particular, industrial equipment senses or generates data in the perception layer. The service flows traverse the SDN network in the transport layer and various industrial services are implemented in the application layer. An SDN network topology ($\mathcal{G}$) is considered with N OpenFlow switches and a set of token-bucket [14] service flows ($\mathcal{F}$) with specified features and transmission requirements. In the token-bucket algorithm, when a packet of size a has to be forwarded, a tokens are removed from the token bucket. If there are not enough tokens in the bucket, the switch will drop the packet. The network is modeled as a directed graph $\mathcal{G}(\mathcal{V},\mathcal{L})$, which consists of a set of nodes $\mathcal{V}$ and a set of links $\mathcal{L}$. The set $\mathcal{L}$ is composed of multiple ordered pair of nodes (u,v), $\forall u,v \in \mathcal{V}$ and $u \neq v$. We define that $N = |\mathcal{V}|$ is the number of nodes and (u,v) is the directed physical link from node u to v. The service flow set $\mathcal{F} = \{f_1,...,f_i,...,f_M\}$ is composed of M flows. Specifically, we define a 7-tuple to describe a service flow f_i, i.e., $(s_i,d_i,r_i,b_i,D_i,p_i,Type_i)$, which respectively denotes the source node, destination node, arrival rate, burst size, delay bound requirement, priority, and service type. In particular, every flow f_i is token-bucket constrained (r_i,b_i). It exhibits that r_i is the maximal long-term arrival rate of data and b_i is the maximal amount of data that arrives simultaneously. The arrival curve of each flow is expressed as

$$\alpha_i(t) = r_i t + b_i,\ \ i = 1,2,...,M. \tag{1}$$

According to the aggregation property, if there are I token-bucket flows concurrently reaching the same ingress on a switch, the arrival curve of the aggregated flow can be expressed as

$$\alpha(t) = \sum_{i=1}^{I} \alpha_i(t) = \sum_{i=1}^{I} r_i t + \sum_{i=1}^{I} b_i. \tag{2}$$

However, the collision occurs when two or more flows are simultaneously sent to the same egress. In order to reduce the collision between services, we create Q virtual queues for each link (u, v) and schedule these flows through jointly exploiting the priority queuing (PQ) and first-in-first-out (FIFO) scheduling. The low-priority flow is transmitted when there is no flow in the high-priority queue. Fig. 1 shows that Queue 1 has the highest priority. The flows in the Queue are transmitted according to FIFO rule. We denote (u, v, q) as the qth virtual queue of physical link (u, v). The used bandwidth and buffer of virtual queue (u, v, q) is $r_{u,v,q} = \sum_{i=1}^{M} x_i(u, v, q) r_i$ and $b_{u,v,q} = \sum_{i=1}^{M} x_i(u, v, q) b_i$, respectively. If the ith flow is buffered at virtual queue (u, v, q), $x_i(u, v, q) = 1$. Otherwise, $x_i(u, v, q) = 0$.

B. Service curve

Each network element can be regarded as a server that provides services at a certain rate in the NC framework. The rate-latency service curve of the OpenFlow switch is expressed as [5]

$$\beta_n(t) = R_n [t - T_n]^+, \; n = 1, 2, ..., N, \tag{3}$$

where T_n is the latency until the server becomes active and R_n is its minimal service rate after this latency. Considering non-preemptive PQ scheduling, the transmission of a packet is not interrupted once it has begun. Therefore, the service curve for priority Queue q is expressed as

$$\beta_{u,v,q}(t) = \left[\left(R_{u,v} - \sum_{j=1}^{q-1} r_{u,v,j} \right) t - \sum_{j=1}^{q-1} b_{u,v,j} - 2l^{\max} \right]^+, \tag{4}$$

where $R_{u,v}$ denotes the overall capacity of the link (u, v) and $l^{\max}$ represents the maximum packet size of flows. Thus, $(R_{u,v} - \sum_{j=1}^{q-1} r_{u,v,j})$, the difference between the link capacity and the total bandwidth used by higher priority flows, denotes the available bandwidth of virtual queue (u, v, q). The $(\sum_{j=1}^{q-1} b_{u,v,j} + 2l^{\max})$ represents the maximum burst size that this queue may have buffered. Specifically, $2l^{\max}$ is derived from non-preemptive PQ scheduling and store-and-forward behavior of switches.

When a flow traverses two switches (with the service curves β_1 and β_2) by sequence, the concatenation of two switches, denoted by β, offers the service curve with min-plus convolution [5]

$$\beta(t) = \beta_1 \otimes \beta_2(t) = \inf_{0 \le s \le t} (\beta_1(s) + \beta_2(t - s)), \tag{5}$$

where $\otimes$ is min-plus convolution. The aggregated service curve has several properties, including associativity, commutativity, and distributivity.

The process of packet forwarding in an SDN architecture is shown in Fig. 2. The network is decoupled into a control plane and a data plane. The Ryu controller [15] in the control plane exchanges data with the Open VSwitch (OVS) [16] in the data plane through the OpenFlow protocol. The core components of the OVS switch include ovs-vswitchd, ovsdb-server, and Kernel Datapath, which forwards packets according to flow table rules from the controller. The ovs-vswitchd module reads the configuration information in the ovsdb-server module at startup, and then configures the datapaths in the Kernel Datapath module. Each datapath owns multiple vports, which are set up by associating flow tables.

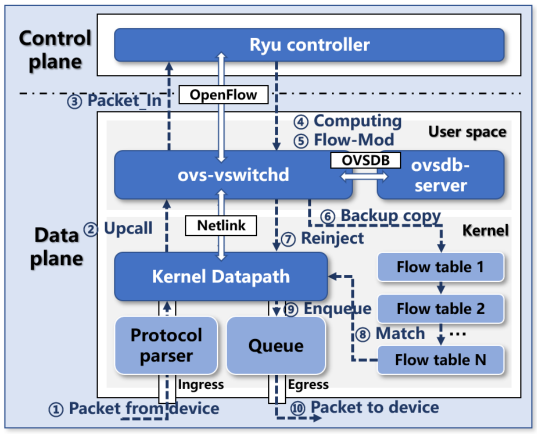

Fig. 2. The process of packet forwarding in an SDN architecture.

When a packet enters the network, the corresponding datapath will first parse the packet header to obtain the source node IP, destination node IP, MAC source address, priority, and other information of the packet. The packet is then matched against the flow table in ovs-vswitchd. If a matching entry is found, the corresponding action (e.g., forward, enqueue, drop, and modify-field) is performed on the packet (steps ①⑧⑨⑩). At the same time, ovs-vswitchd will back up this flow entry to the Kernel Datapath, so that flows of the same type directly perform actions in the Kernel, thereby reducing the matching time (step ⑥). If the match fails, ovs-vswitchd sends the whole packet or its header to Ryu controller through Packet-In message. The Ryu controller computes a rational forwarding strategy relying on the global network topology and network state information, and returns the result to ovs-vswitchd (steps ③④⑤). Finally, the datapath forwards the packet to the egress queue according to the flow table rules (step ⑨). Therefore, when a packet enters the switch node, the service curve of the switch varies due to the freshness of this packet. We portray these two different service curves in the following two situations.

1) Hit: As mentioned above, only the steps ①⑧⑨⑩ are performed when a matching entry is found. It signifies that the only factor that affects the service rate of the flow in switches is the latency caused by flow table matching and action execution. The service curve of OpenFlow switch is $\beta_{hit} = \beta_{\tau_{ma}} \otimes \beta_{u,v,q}$, where τ_{ma} is the delay caused by the match-and-action.

2) Loss: The controller determines the scheduling strategy by considering the current network status and the transmission requirements of this flow if there is no matching entry. Then, three additional processes will be added, including the twoway communication between the switch and the controller, strategy generation, and flow table installation. The service curve of OpenFlow switch is $\beta_{loss} = \beta_{\tau_{nf}} \otimes \beta_{\tau_{ma}} \otimes \beta_{u,v,q}$, where τ_{nf} represents the time costed by requesting, computing, and installing new flow table entries. It is worth noting that matching, action, new flow table installation are related to the performance of the device. Hence, the delay can be regarded as a constant, which is expressed as a delay function $\delta_\tau(t) = +\infty \cdot 1_{\{t \geq \tau\}}$.

C. End-to-end delay model

If a α-constrained flow is served in network element with service curve $\beta(t)$, the delay bound [14] is the maximum distances between the two curves in the vertical directions, $h(\alpha, \beta) = \frac{b}{R} + T$. The E2E delay bound of a specific flow is the total worst-case delays of the servers on its path. We define $path_i = \left\{(u_1^i, v_1^i, q_1^i), ..., (u_{L_i}^i, v_{L_i}^i, q_{L_i}^i)\right\}$ is the path of flow f_i, where L_i denotes the length of $path_i$. Hence, according to the concatenation and pay-burst-only-once principle, the worst-case E2E delay of flow f_i is bounded by

$$\begin{aligned} d_i = & \sum_{(u,v,q)\in path_i} \left(T_{u,v,q}^i + \tau\right) \\ & + \frac{\sum\limits_{f_j \in equal_i} b_j}{\min\limits_{(u,v,q)\in path_i} \left\{ R_{u,v} - \sum\limits_{j=1}^{q_i-1} r_{u,v,j} \right\}} + d_{con}, \end{aligned} \tag{6}$$

where $T_{u,v,q}^i = \frac{\left(\sum_{j=1}^{q_i-1} b_{u,v,j} + 2l^{\max}\right)}{\left(R_{u,v} - \sum_{j=1}^{q_i-1} r_{u,v,j}\right)}$ denotes the waiting time of flow f_i in virtual queue (u, v, q), which is derived from (4). $\tau = \tau_{ma}$ denotes the delay caused by flow table match in one switch. If the flow arriving at the switch is new, $\tau = \tau_{nf} + \tau_{ma}$. The delays τ_{ma} and τ_{nf} are constants determined by the hardware performance. A new set $equal_i$ is defined to represent all flows which choose the same physical link (u, v) as flow f_i (i.e., $x_j(u, v, q_k) = 1$) does. The constant delay d_{con} results from the packet propagation and processing.

IV. Formulation and Optimization

A. Problem formulation

In smart factories of Industrial 5.0, it costs plenty of network resources to guarantee the real-time and reliability of time-critical applications, such as human-machine interaction, remote configuration, decision making, etc. Therefore, we aim to minimize the total worst-case network cost (MWCNC) of all services in our system, subject to five constraints, including restricted distribution, flow conservation, link capacity, buffer capacity, and delay limit. Due to limited physical network resources, the objective function is the sum of link cost and node cost (i.e., bandwidth consumption and buffer consumption). We define ω_c as the cost required to allocate unit bandwidth. $C_u(\cdot)$ denotes the cost function of resources required to buffer burst on node u, which is a monotone increasing function. Thus, our MWCNC problem is formulated as

$$\min \sum_{i=1}^{M} \sum_{q=1}^{Q} \left(\sum_{(u,v)\in\mathcal{L}} x_i(u,v,q)\, r_i \omega_c + \sum_{u\in\mathcal{V}} x_i(u,v,q)\, C_u(b_i) \right) \tag{7}$$

subject to

$$\sum_{q=1}^{Q} x_i(u,v,q) \leq 1, \forall (u,v) \in \mathcal{L}, x_i(u,v,q) \in \{0,1\}, \tag{8}$$

$$\sum_{u\in\mathcal{V}} \sum_{q=1}^{Q} \left[x_i(u,v,q) - x_i(v,u,q)\right] = \begin{cases} -1, & if\ v = s_i \\ 1, & if\ v = d_i \\ 0, & otherwise \end{cases}, \tag{9}$$

$$\sum_{i=1}^{M} \sum_{q=1}^{Q} x_i(u,v,q)\, r_i \leq R_{u,v}, \forall (u,v) \in \mathcal{L}, \tag{10}$$

$$\sum_{i=1}^{M} \sum_{q=1}^{Q} x_i(u,v,q)\, b_i \leq B_u, \forall u \in \mathcal{V}, \tag{11}$$

$$d_i \leq D_i, \forall i \in \mathcal{F}. \tag{12}$$

The feasible region for optimization variable $x_i(u, v, q)$ is given by constraints (8)-(12). Constraint (8) ensures that each flow on the link (u, v) is assigned to at most one virtual queue. Constraint (9) requires that the number of inflows equals to the number of outflows on each node, excepting s_i and d_i. Constraint (10) and (11) represent that the occupied bandwidth and buffer cannot exceed the link capacity of (u, v) and the node buffer capacity of u, respectively. Constraint (12) guarantees the worst-case delay requirement for each flow.

B. Flow routing and scheduling algorithm

The proposed MWCNC problem is regarded as two integrated selection problems, i.e., path selection for service flows with different SLAs (routing) and queue selection for flows with various priorities (scheduling). The delay-constrained shortest path problem is NP-hard whereby our shortest routing problem with a flow scheduling is still NP-hard [17]. Considering the surging of industrial flow and network size, we propose a joint flow routing and scheduling algorithm (JFRSA) by exploiting the natural aggregation algorithm (NAA) [18].

Resource contention occurs when two flows are waiting to be forwarded on one egress. According to the QP principle, only the flows with a higher priority affect the service rate of other flows. In consequence, we divide the JFRSA into several phases. As shown in Algorithm 1, flows first are grouped by service type and sorted by priorities in the subset $\mathcal{F}_{Type_k} \in \mathcal{F}$ (**line 1**). Afterwards, we reduce the dimension of variables by constructing a map: $(\mathbf{x}, \mathbf{y}) \rightarrow \mathbf{Z}$, where the auxiliary matrices $\mathbf{x} = [p_1, p_2, ..., p_H, s]$ and $\mathbf{y} = [q_1, q_2, ..., q_{s-1}]$ respectively denote the path and queue of specific flow in physical networks. It is noted that there is usually a maximum number of hops (a fraction of the number of nodes) for information transmission in a network. Hence, H and $(s-1)$ are set as the maximum number of hops and

Algorithm 1 Joint flow routing and scheduling algorithm (JFRSA)

```
Input: Set of network services $\mathcal{F}$, physical network $G(V, L)$.
Output: Path selection matrix $X^*_{M\times(H+1)}$, queue selection matrix $Y^*_{M\times(s-1)}$, minimum worst-case network cost $C^*$.
1: Classify flows by type ($K$ subsets) and sort flows in $\mathcal{F}_{type_k}$ by priority.
2: for all $\mathcal{F}_{type_k} \in \mathcal{F}$ do
3:     for all $f_i \in \mathcal{F}_{type_k}$ do
4:         Compute the optimal ${C_i}^*$ by invoking NAA subject to the constraints (10)-(12);
5:         If constraints (8)-(9) are satisfied, $flag = 1$. Otherwise, $flag = 0$;
6:         while $flag$ is FALSE do
7:             Add $\mathbf{Z_i}(\mathbf{x_i}, \mathbf{y_i})! = \mathbf{Z_i^*}(\mathbf{x_i^*}, \mathbf{y_i^*})$ as a new constraint to the original problem;
8:             Recompute the optimal $\mathbf{Z}^*$ based on NAA;
9:         end while
10:    end for
11: end for
```

TABLE I
PARAMETER SETTING

Notion	Value range
Number of nodes, N	$\{50, 75, 100, 125, 150\}$
Number of flows, M	$\{60, 80, 100, 120, 140\}$
Capacity of the link(u, v) , $R_{u,v}$	200 Mbps
Buffer capacity of node u , B_u	200 Mb
Delay bound requirement of f_i , D_i	200 $\sim$ 400 ms
Arrival rate of f_i , r_i	6 $\sim$ 8 Mbps
Burst size of f_i , b_i	4 $\sim$ 6 Mb
Maximum packet size of flows , l^{max}	1500 Byte
Constant delay , d_{con}	1 $\sim$ 2 ms

length of the path. For instance, $\mathbf{x_i} = [3, 5, 7, 8, 0, 0, 4]$ and $\mathbf{y_i} = [1, 2, 2]$ indicate that the selected path of flow f_i is $path_i = \{(3, 5, 1), (5, 7, 2), (7, 8, 2)\}$. Finally, we sequentially compute the sub-optimal solution for all flows with various types by invoking NAA (**line 4-8**). If there is no feasible solution, the flow is denied access to the network. The time complexity of JFRSA is $O(M \times (H + 1) \times P)$, where P is the product of the number of populations and the number of iterations of NAA.

V. EVALUATION

We compare the performance of JFRSA using MATLAB numerical simulation with multiple metrics, including average cost, access ratio, running time, and utilization variance. Specifically, we conduct the simulation on a laptop, with a 1.6 GHz CPU and 8 GB RAM. Unless otherwise stated, the parameters used in the simulation are listed in Table I. In addition, the cost function is regarded as an affine function, exponential function, or staircase function according to diverse scenarios.

We first evaluate the performance under different network scales, business scales, and delay bound requirements. As illustrated in Fig. 3(a) and 3(b), as the network scale expands,

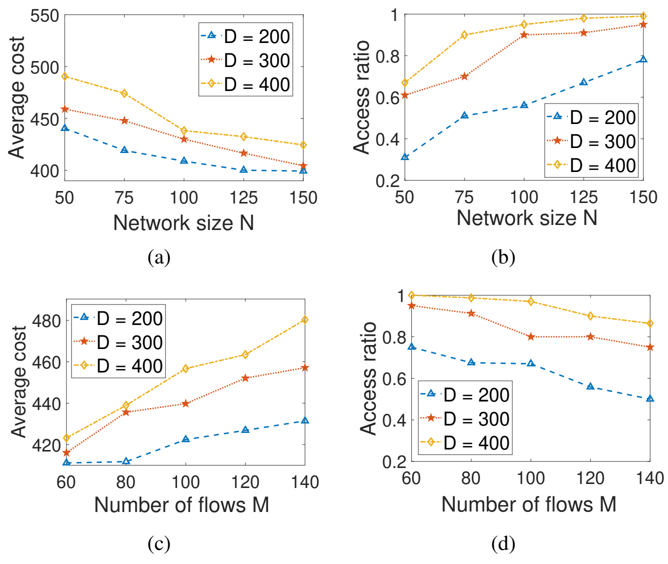

Fig. 3. Impacts of network scale and service scale on JFRSA performance. (a) Average cost versus network size. (b) Access ratio versus network size. (c) Average cost versus service scale. (d) Access ratio versus service scale.

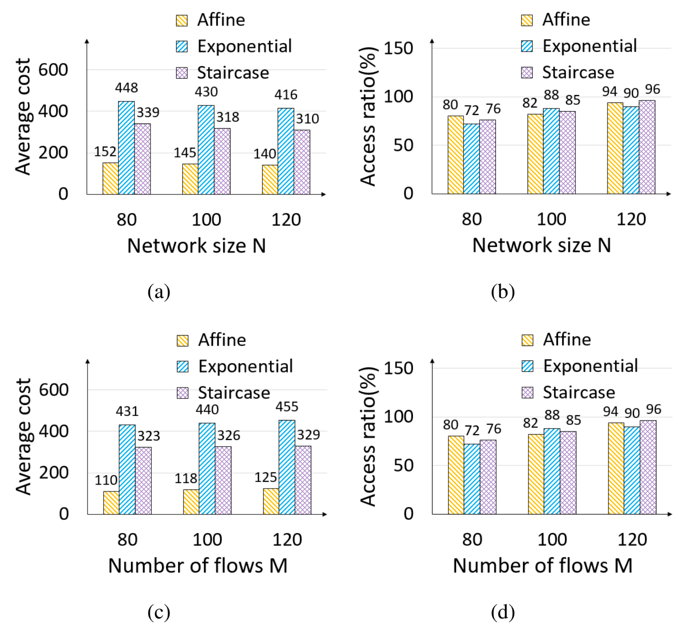

Fig. 4. Performance under various cost functions. (a) Average cost versus network size. (b) Access ratio versus network size. (c) Average cost versus service scale. (d) Access ratio versus service scale.

the average cost and access rate becomes smaller and larger, respectively. This is because the SDIIoT system with abundant available network resources simultaneously satisfies the transmission requirements of more time-sensitive applications. On the contrary, Fig. 3(c) and 3(d) depict that more network services result in a larger average cost and a smaller access ratio. This is mainly because the limited network resources cannot support excessive network services resulting in some services being denied access to the network.

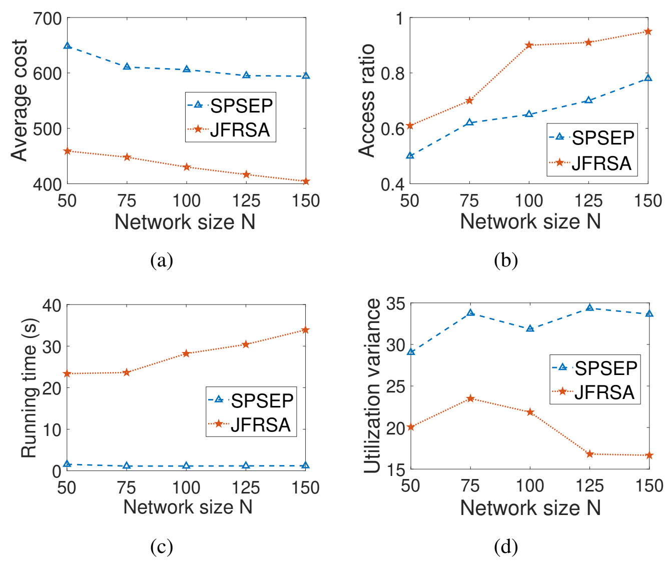

Fig. 5. Performance comparison of algorithms. (a) Average cost versus network size. (b) Access ratio versus network size. (c) Running time versus network size. (d) Utilization variance versus network size.

We use three types of cost functions (defined in §IV-A) to evaluate the performance under different scenarios. The affine function is $C_u(x) = \alpha_1 \frac{x}{\alpha_2 B_u}, x \in [0, \alpha_2 B_u]$. The exponential function is defined as $C_u(x) = \beta_1(exp(\frac{x}{\beta_2 B_u}) - 1), x \in [0, \beta_2 B_u]$, and the staircase function is $C_u(x) = \gamma_1 \lfloor \frac{x}{\gamma_2 B_u} \rfloor, x \in [0, 5\gamma_2 B_u]$. We set $\alpha_1 = 320$, $\alpha_2 = 0.1$, $\beta_1 = 186.23$, $\beta_2 = 0.1$, $\gamma_1 = 64$, $\gamma_2 = 0.02$, and $B_u = 200$, which are regulatory factors determined by the physical network status. Fig. 4 depicts that under the identical network scale and service scale, the application of the affine function as the cost function obtains a superior routing and scheduling scheme. The staircase function is the second, and the exponential function is the worst. This demonstrates that our algorithm is more suitable in which the cost function has a linear trend.

Finally, we compare JFRSA with the shortest path and sequential enqueueing based on the priority (SPSEP) algorithm. SPSEP is derived from delay-constrained least-cost routing [19]. It first computes the shortest paths of all service flows, and then schedules the flows according to the priorities if the constraints are satisfied. Fig. 5 demonstrates that the proposed JFRSA achieves preferable performance in multiple dimensions, including lower average cost, higher access ratio, and smaller utilization variance. Note that the exhaustive search of JFRSA results in more running time. JFRSA trades real-time performance for computational accuracy and resource efficiency. However, this concern can be mitigated by the fact that the SDN controller can proactively compute policies and install flow tables based on services prediction.

VI. Conclusion

In this paper, we studied the multi-flow routing and scheduling in the SDIIoT system. Configuration of flow tables in OVS switches causes the service delay due to the radically dynamic industrial environment and service flows. Therefore, we respectively characterized OVS switch service curves in two states of network services (packet hit and loss). In addition, we formulated a worst-case network cost minimization problem and proposed a heuristic algorithm to optimize the flow routing and scheduling strategy. Numerical results showed the superior performance of JFRSA.

References

[1] P. K. R. Maddikunta, Q.-V. Pham, P. B, N. Deepa, K. Dev, T. R. Gadekallu, R. Ruby, and M. Liyanage, "Industry 5.0: A survey on enabling technologies and potential applications," *Journal of Industrial Information Integration*, vol. 26, p. 100257, 2022.

[2] B. Bajic, A. Rikalovic, N. Suzic, and V. Piuri, "Industry 4.0 implementation challenges and opportunities: A managerial perspective," *IEEE Systems Journal*, vol. 15, no. 1, pp. 546–559, 2021.

[3] X. Xu, Y. Lu, B. Vogel-Heuser, and L. Wang, "Industry 4.0 and industry 5.0—inception, conception and perception," *Journal of Manufacturing Systems*, vol. 61, pp. 530–535, 2021.

[4] V. V. Kalashnikov, *Mathematical methods in queuing theory*. Springer Science & Business Media, 2013, vol. 271.

[5] J. Y. Le Boudec and P. Thiran, *Network calculus: a theory of deterministic queuing systems for the internet*. Springer, 2001.

[6] M. Helm, H. Stubbe, D. Scholz, B. Jaeger, S. Gallenmüller, N. Deric, E. Goshi, H. Harkous, Z. Zhou, W. Kellerer, and G. Carle, "Application of network calculus models on programmable device behavior," in *2021 33th International Teletraffic Congress (ITC-33)*, 2021, pp. 1–9.

[7] F. Geyer and S. Bondorf, "Graph-based deep learning for fast and tight network calculus analyses," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 1, pp. 75–88, 2021.

[8] S. Ren, Q. Feng, and W. Dou, "An end-to-end qos routing on software defined network based on hierarchical token bucket queuing discipline," in *Proceedings of the 2017 International Conference on Data Mining, Communications and Information Technology*, 2017, pp. 1–5.

[9] A. K. Koohanestani, A. G. Osgouei, H. Saidi, and A. Fanian, "An analytical model for delay bound of openflow based sdn using network calculus," *Journal of Network and Computer Applications*, vol. 96, pp. 31–38, 2017.

[10] A. Bouillard, "Trade-off between accuracy and tractability of network calculus in fifo networks," *Performance Evaluation*, vol. 153, p. 102250, 2022.

[11] L. Zhao, P. Pop, Z. Zheng, H. Daigmorte, and M. Boyer, "Latency analysis of multiple classes of avb traffic in tsn with standard credit behavior using network calculus," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 10, pp. 10 291–10 302, 2021.

[12] J. Zhang, L. Chen, T. Wang, and X. Wang, "Analysis of tsn for industrial automation based on network calculus," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2019, pp. 240–247.

[13] A. Lee, P. Wang, S.-C. Lin, I. F. Akyildiz, and M. Luo, "Dynamic bandwidth allocation in sdn based next generation virtual networks: A deterministic network calculus approach," in *Proceedings of the 2018 Conference on Research in Adaptive and Convergent Systems*, ser. RACS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 80–87.

[14] A. Van Bemten and W. Kellerer, "Network calculus: A comprehensive guide," 2016.

[15] "Ryu," Ryu SDN Framework Community, Accessed: Mar. 2022. [Online]. Available: https://ryu-sdn.org

[16] "Open vswitch," Linux Foundation, Accessed: Mar. 2022. [Online]. Available: https://www.openvswitch.org

[17] M. Jia, W. Liang, M. Huang, Z. Xu, and Y. Ma, "Routing cost minimization and throughput maximization of nfv-enabled unicasting in software-defined networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 2, pp. 732–745, 2018.

[18] F. Luo, Z. Y. Dong, Y. Chen, and J. Zhao, "Natural aggregation algorithm: A new efficient metaheuristic tool for power system optimizations," in *2016 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, 2016, pp. 186–192.

[19] D. S. Reeves and H. F. Salama, "A distributed algorithm for delay-constrained unicast routing," *IEEE/ACM Transactions on networking*, vol. 8, no. 2, pp. 239–250, 2000.