Joint Optimization of Time-Slot Allocation and Traffic Steering for Large-Scale Deterministic Networks

Wenhao Wu, Xiaoning Zhang, Jiaming Pan, and Yihui Zhou

***Abstract*—Recently, time-sensitive services have expanded from traditional industrial control systems to more scenarios. Some time-sensitive applications, such as remote surgery, autonomous driving, augmented reality (AR), etc., require deterministic end-to-end delay and jitter in data transmission. deterministic network (DetNet) is proposed as a promising technology for providing deterministic service in wide area networks (WAN). DetNet guarantees deterministic end-to-end delay and jitter by specifying a certain routing path and transmission time-slots for time-sensitive flows. However, how to efficiently steer time-sensitive flows while jointly allocating transmission time-slots is still an open problem. Existing flow scheduling algorithms are limited in the scenarios of local area networks (LAN), and do not consider the impact of propagation delay in large-scale networks. To this end, we study the joint optimization of time-slot allocation and traffic steering, while considering the propagation delay of WAN links. Our objective is to maximize the number of successfully deployed time-sensitive flows under the constraints of required end-to-end delay. Accordingly, we formulate the studied problem as an integer linear programming (ILP) model. Since it is proved to be an NP-hard problem, we design a heuristic algorithm named genetic-based deterministic network traffic scheduling (GDNTS). The solution with the largest number of deployed time-sensitive flows can be obtained from the evolution of chromosomes in GDNTS. Compared with the benchmark algorithms, extensive simulation results show that GDNTS improves the deployed time sensitive-flows number by 22.85% in average.**

***Index Terms*—Deterministic networks, integer linear programming, resource allocation, routing, wide area network.**

I. Introduction

RECENTLY, researchers are exploring new network applications in the context of the 5th generation mobile communication technology (5G) [1]. Some time-sensitive applications, such as augmented reality (AR) [2], virtual reality (VR) [3], Internet of vehicles (IoV) [4], remote surgery [5], etc., demand deterministic end-to-end delay and jitter. However, in the wide area network (WAN) scenarios, network delay will significantly affect the performance of such applications. As shown in Fig. 1, critical health-care applications like remote surgery require quick request-response and feedback control cycles with high availability and reliability. In remote surgery, the control signal sent from the operator is transmitted to the medical equipment through WAN [6]. Uncertain delay and jitter cause a noticeable increase in surgical risk [7]. The jitter larger than the interval time between two packets will cause the system instability. Generally, the jitter of end-to-end delay should be less than 1 ms [8]. Such long-distance time-sensitive applications have brought new challenges to existing networks.

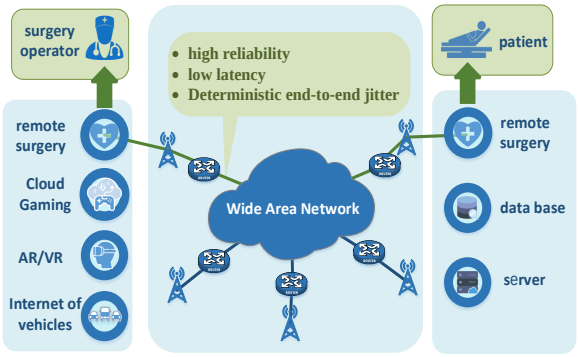


Fig. 1. Illustration of emerging applications in large-scale networks.

Unfortunately, the existing network forwarding strategy can not guarantee deterministic end-to-end delay and jitter for time-sensitive applications. Traditional Internet protocol (IP) networks provide best-effort packet delivery service and lack unified scheduling for the behavior of packets. The end-to-end delay of packets has a long tail effect in IP networks, which means the end-to-end delay of some packets significantly exceeds the average delay due to network congestion or other issues [9]. Therefore, how to provide deterministic end-to-end delay for emerging time-sensitive applications in large-scale networks is a brand new topic for research.

Extensive research has been conducted for deterministic transmission in local area networks (LAN). To meet the deterministic quality of service (QoS) requirements of time-sensitive applications, IEEE 802.1 time-sensitive networking (TSN) task group has sought to provide link layer support for ultra-low latency (ULL) networking. A set of technical standards to guarantee the deterministic delay in the physical layer and link layer has been put forward [10]. In TSN standards, IEEE 802.1 AS [11] proposed the timing and synchronization mechanism. IEEE 802.1Qat [12], Qcc [13] and Qca [14] proposed resource reservation and path control mechanisms. IEEE 802.1 Qch [15] proposed a queueing model named cyclic queuing and forwarding (CQF), which uses

Manuscript received November 21, 2022; revised May 17, 2023; approved for publication by Jamalipour, Abbas, Division 3 Editor, Feburary. 22, 2023.

The authors are with School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu, China, email: {wu_wenhao555, pjm20211202, 13723877647}@163.com, xnzhang@uestc.edu.cn.

W. Wu is the corresponding author.

Digital Object Identifier: 10.23919/JCN.2023.000047

1229-2370/23/$10.00 © 2023 KICS

two queues opened alternately to provide deterministic end-to-end delay. TSN provides mechanisms for guaranteeing deterministic end-to-end delay in the link layer that well suited for small networks.

However, mechanisms in TSN have strict requirements on propagation delay and clock synchronization. The effect of the propagation delay of the links and the precision limitation of the clock synchronization makes the mechanism suitable for TSN difficult to effectively apply in WAN. In order to overcome the limitations of LAN-based TSN standards, IETF deterministic networking group took TSN mechanisms as the underlying transmission solution and proposed deterministic networking (DetNet) architecture to provide deterministic transmission services for time-sensitive flows in layer-3 and multi-protocol label switching (MPLS) based deterministic transmission technology has been standardized [16], [17]. A desirable outcome of the DetNet is the ability to establish a multi-hop path over the IP or MPLS network for a particular flow while meeting the deterministic requirement for delay and jitter [18]. Techniques like resource allocation, service protection, and explicit routes have been used by DetNet to provide deterministic QoS. The IETF DetNet group has proposed the overall architecture [19], framework for data plane [20], and YANG model for configuration and operational data [21]. Some underlying transmission mechanisms are not sensitive to clock synchronization are also proposed to adapt to the features of wide-area transmissions, such as three-buffer CQF and CSQF [24]. Moreover, emerging network technologies such as 5G and Software Defined Networks (SDN) are also actively seeking integration with DetNet [22], [23]. Mohaje *et al.* proposed a dynamic optimization model to optimizing carrier allocation and power utilization with the highest level of energy efficiency [40], [41]. Dong *et al.* [42] proposed a novel adaptive backhaul topology with the ability to adapt to different traffic patterns, which provides the possibility of effective channel allocation to each backhaul link to meet capacity and QoS demands.

The key point of providing deterministic end-to-end delay in DetNet is to allocate certain forwarding paths and transmission time-slots for time-sensitive flows. However, underlying transmission mechanisms like CSQF and CQF only specify the layer-2 primitive while the problem of traffic routing and scheduling remains undefined. Compared with the LAN scenario, the difficulty of traffic scheduling and time-slot allocation in the WAN will be increased due to the asynchronous propagation delay and clock asynchronization. Fig. 2(a) shows a network instance in LAN, in which the CQF is adopted as the underlying transmission solution. The sending and receiving of a packet can be completed in the same time-slot. The core idea of the scheduling algorithm design in the LAN is to guarantee a deterministic queuing delay and processing delay while the propagation delay can be ignored. Fig. 2(b) shows a network instance in WAN. The packet is received at time-slot t_1+n after being sent to the link at time-slot t_1, where n represents the propagation delay between $Router_1$ and $Router_2$. This feature makes it necessary to consider the link propagation delay in the scheduling algorithm of WAN. Variations in propagation delay and clock asynchronization problems also cause a clock

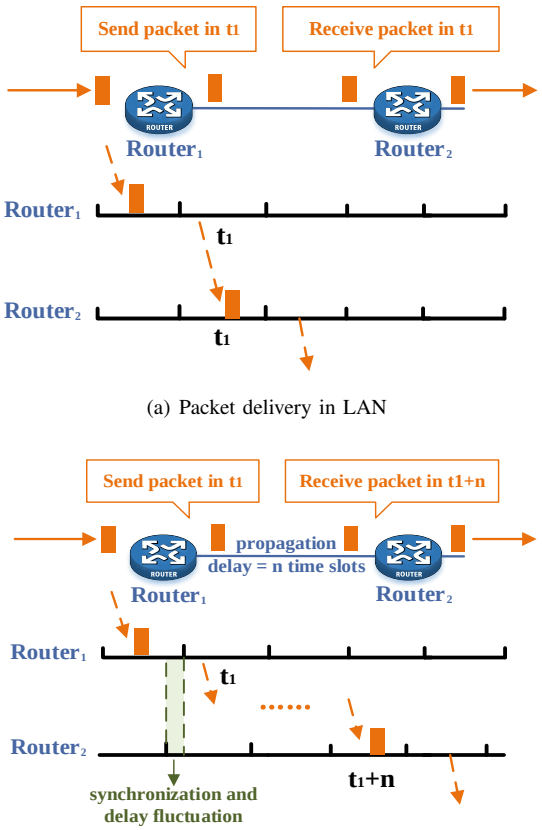


Fig. 2. Difference of packet delivery between WAN and LAN.

skew s between two routers, which makes the actual receiving time of the packet inconsistent with the expected calculation result. The complex network environment makes algorithms designed for TSN difficult to find the optimal solution.

To address the above challenges, this paper focuses on the joint optimization of time-slot allocation and traffic steering for large-scale deterministic networks. Our objective is to maximize the number of deployed time-sensitive flows under the constraints of delay and jitter required by time-sensitive flows. We first propose the DetNet architecture based on SDN and an advanced CQF mechanism is used to solve the clock asynchronization problem in WAN. The centralized SDN network architecture can jointly schedule time-sensitive flows in the network and provide deterministic end-to-end delay guarantee. The studied problem is formulated as an integer linear programming (ILP) model and can be reduced from the knapsack problem, which proves that the studied problem is NP-hard. Due to the problem's complexity, we design a heuristic algorithm named genetic-based deterministic network traffic scheduling (GDNTS). GDNTS decides whether to deploy each time-sensitive flow in the network and allocates network resources for successfully deployed time-sensitive flows. In the GDNTS algorithm, the time-sensitive flow deployment result is placed on a specific data structure named chromosome. By using bioinspired operators on multiple chromosomes, such

as mutation, crossover, and selection, numerous feasible deployment results can be produced. The high-quality solutions are retained to the end of the algorithm. Compared with the benchmark algorithms, extensive simulation results show that GDNTS improves the deployed time-sensitive flow number by 22.85% in average.

The main technical contributions of this paper are summarized as follows.

- We apply the centralized network framework to DetNet, and propose the workflow for time-sensitive flow scheduling.
- We propose an advanced CQF mechanism that can tolerate delay variation and clock synchronization in WAN.
- We present the joint optimization problem in large-scale deterministic network, while considering the impact of propagation delay. We formulate the studied problem as an ILP model.
- Since the formulated problem is NP-hard, we design a heuristic algorithm named GDNTS based on genetic algorithm for time-Slot allocation and traffic steering, and our algorithm improves the deployed time-sensitive flows number by 22.85% in average.

The rest of our paper is organized as follows. In Section II, we briefly review the related literatures. In Section III, we propose the architecture of deterministic network and formulate the studied problem as an ILP model in Section IV. Our heuristic algorithm is presented in Section V. We present simulation results in Section VI and conclusions are given in Section VII.

II. Related work

The scenarios considered in the deterministic networks are becoming large and complicated. Realizing the co-existence of multiple deterministic services is necessary. In the past few years, routing and scheduling problems in deterministic networks have been largely covered. We divide these studies into two parts. For the first part, scheduling mechanisms suitable for LAN were proposed. The scheduling algorithm in TSN should ensure that high-priority data packets not conflict with each other. Steiner *et al.* [25], [30] studied the performance of commercial SMT solver in static-scheduling problem. Craciunas *et al.* [31] formulated necessary constraints for schedules in TSN and focused on gate operations. Nayak *et al.* [26], [32] proposed the architecture of TSSDN and considered the static scheduling problems in TSN while formulating the studied problem as an ILP model. Yan *et al.* [33] proposed an Injection Time Planning (ITP) mechanism to allocate resources for time-sensitive flows by managing the injection time-slot of time-sensitive flows. Xue *et al.* [34] solved the scheduling problem with network virtualization and proposed virtual queues to deploy time-sensitive flows with a time offset parameter. Pang *et al.* [35] proposed a flow schedule generation model with the objective of minimizing the end-to-end delay of time-sensitive flows. However, these schemes need a limited maximum network diameter and are difficult to apply to large-scale deterministic networks.

In the second part, large-scale deterministic network architecture and flow scheduling algorithms suitable for WANs were proposed. With the expansion of network scale and equipment quantity, large-scale deterministic networking has to take time asynchronization and propagation delay of links into account. Chen *et al.* [36] investigated the load balance problem in large-scale deterministic networks. They used an extension of the original segment routing (SR) policy for specified queuing and forwarding (CSQF) to meet the deterministic QoS requirements. The studied problem is formulated as an ILP model and load balancing was realized by assigning different SR labels to packets at the source node. Krolikowski *et al.* [28] focused on the joint routing and scheduling problem and proposed an approximate solution algorithm based on column generation to maximize traffic acceptance. Huang *et al.* [37] proposed the cycle tags planning (CTP) mechanism and a scheduling algorithm named flow offset and cycle shift (FO-CS) is designed to compute the time-slot allocation of each time-sensitive flow.

Above works adopt scheduling methods that separate time-slot allocation from routing calculation. Namely, when solving the scheduling problem of time-sensitive flows, it is necessary to predetermine alternative paths, and select a specified one to allocate time-slots. Such methods are generally unable to maximize the number of successful deployed time-sensitive flows. The two difficulties in joint scheduling algorithms for large-scale deterministic networks are (1) the propagation delay and clock asynchronization in WAN cannot be ignored and (2) the joint scheduling problem of routing and time-slot has high complexity in large-scale networks. Our work considers the effect of propagation delay and provides an advanced CQF mechanism as a solution for clock asynchronization. Meanwhile, we propose a flow scheduling algorithm to jointly consider the interdependence of routing and time-slot allocation. Thereby we can deploy more time-sensitive flows to the network and achieve more efficient utilization of network resources.

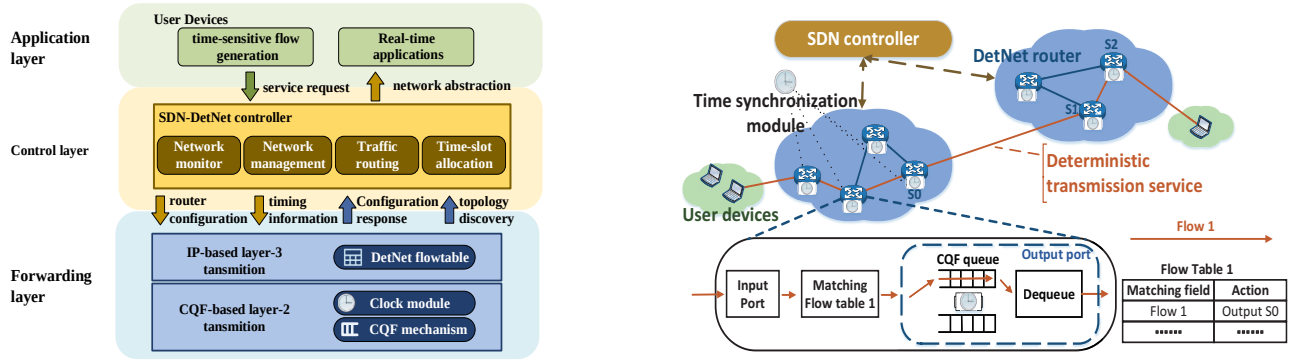
III. Large-scale Deterministic Network Architecture

In this section, we first propose a software-defined deterministic network (SDDN) architecture for WAN. Then we give the workflow of this architecture. Afterward, we provide a deterministic forwarding strategy to guarantee deterministic end-to-end delay and jitter.

A. System Model

The SDDN architecture has the characteristics of centralized user configuration under the management of the SDDN controller. It consists of forwarding layer, control layer and application layer. Fig. 3(a) illustrates the SDDN architecture.

The forwarding layer consists of SDDN routers and is responsible for forwarding time-sensitive packets. Compared to the basic SDN architecture, the routers in SDDN take an advanced cycling queue forward (CQF) queue with buffering mechanism as the underlying transmission solution. This transmission mechanism can guarantee deterministic delay and



(a) SDDN Architecture (b) SDDN in WANs

Fig. 3. Illustration of SDDN Architecture.

jitter in the case of high propagation delay and clock asynchronization. The forwarding layer can receive configuration information from the control layer, and can also feed back network topology change and propagation delay of links to the upper layer.

The control layer is responsible for routing calculation, time-slot allocation, network management, and network monitoring. The controller can collect request information from the application layer and feedback information from the forwarding layer, and generate control information for packet forwarding by executing the scheduling algorithm.

The application layer consists of time-sensitive applications. Users can program and deploy new time-sensitive flows without caring about the underlying operations. These applications can submit the requirement to the controller through the northbound interface. At the same time, the applications can also abstract and encapsulate their own functions to provide a northbound interface for users.

Fig. 3(b) illustrates a deterministic transmission across the WAN requested by the user devices. There are three main entities in SDDN: (1) The SDDN controller is a logically centralized module and is responsible for path calculation and time-slot allocation. The SDDN controller also needs to configure the routers according to calculation results; (2) The router in the data plane can forward packets according to the forwarding rules in DetNet flow table and the packet is sent to the designated port. The queuing behavior of the packet is automatically controlled by the CQF queue. The queue controlled by a high-precision clock with CQF mechanism can guarantee deterministic end-to-end delay and jitter. (3) User devices in the network are the source and destination of time-sensitive flows, there are a large number of user devices directly connected to SDDN routers in the network. And the user devices can receive the information from the controller to specify the sending time of its time-sensitive flows.

B. Deterministic Forwarding Strategy Based on Advanced CQF

The difference between SDDN and ordinary SDN is that advanced CQF mechanism is used to guarantee the deterministic delay and jitter for data transmission. CQF defined

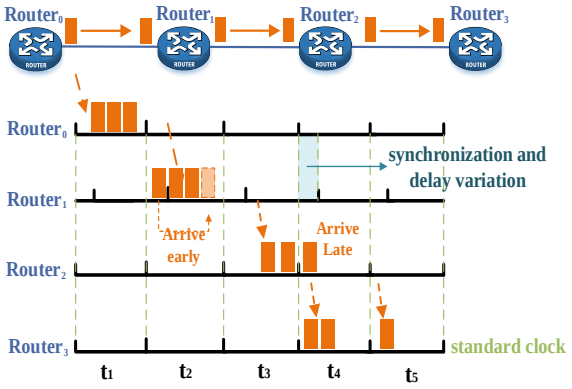


Fig. 4. Packet receiving time deviation.

in 802.1Qch is a widely adopted solution in TSN. CQF mechanism divides the continuous time into time-slots of length d. The basic idea of CQF is that the packet received in a time-slot must be sent to the link for transmission in the next time-slot, which ensures that the queuing delay of the packet is bounded [10]. This mechanism can be achieved by setting two alternately opened queues in the output port of routers. However, the defects of this mechanism are (1) TSN has strict constraints on propagation delay, the propagation delay between two nodes cannot exceed the duration of the time-slot, and (2) CQF mechanism has extremely high requirements of clock synchronization between nodes.

To make this mechanism suitable for WAN, we add two buffer queues to the original CQF to cope with the problem of clock asynchronization and propagation delay variation. The result of the above problem is that the data packet arrives at downstream router earlier than the scheduled time-slot or the data packet arrives later than the scheduled time-slot. Fig.4 shows these two phenomena and their solutions. For the case where the packet arrives early, we use a buffer queue to delay this packet by one slot so that it can be forwarded at the correct time-slot. For the case where the packet arrives late, we send the packet in the next time-slot by frame preemption. This

strategy will not cause traffic collisions in the next time-slot because a portion of the bandwidth has been reserved for these packets during the scheduling algorithm. Such advanced CQF strategy can relax the requirements of clock synchronization, but it is worth noting that this mechanism still requires relatively loose clock synchronization, packets should not be skewed for more than one time-slot.

We also take link propagation delay into account, assuming a transmission path with h hops, the propagation delay in each hop is a_i and the duration of a time-slot is ts. For the ideal case, each packet sent to link with delay a_i in time-slot t will be received by the next node in time-slot $t + \lceil a_i/ts \rceil + 1$, and this process will be repeated h times. Meanwhile, we note that if a packet can be sent to the link at the end of a time-slot at the source node and received at the beginning of a time-slot at the destination node, the delay of this packet can be reduced by one time-slot. In this case, the minimum delay of the packet is:

$$D_{Min} = h \cdot ts + \sum_{i=1}^{h} a_i - ts. \tag{1}$$

Since the propagation delay is usually much larger than the duration of the time-slot, we ignore the ceiling symbol in calculating the time-slot consumption on the link. Noting that this simplification does not affect the correctness of our conclusion, because our advanced CQF strategy can tolerate calculation error within one time-slot.

For the worst case, each packet will arrive later than the scheduled time-slot in each hop of transmission due to clock skew and propagation delay variation. This phenomenon has been illustrated in Fig. 4. The packet sent to link with delay a_i in time-slot t will be received by the next node in time-slot $t + \lceil a_i/ts \rceil + 2$. This process will be repeated h times, and if the packet is sent to the link at the beginning of the time-slot at the source node and received at the end of the time-slot at the destination node, the delay of this packet will increase by one time-slot. In this case, the maximum delay of the packet is:

$$D_{Max} = 2h \cdot ts + \sum_{i=1}^{h} a_i + ts. \tag{2}$$

Therefore, we can limit the end-to-end delay jitter within $(h+2) \cdot ts$.

Our mechanism can be implemented by adding two additional queues to the CQF. Fig. 5 shows the transmission process of a time-sensitive flow in the SDDN. The packets in the buffer queue and the CQF queue will be sent sequentially in each time slot. When a packet arrives at router S0 in time-slot i, the router first checks if the packet arrives at the correct time-slot. If the packet arrives in the correct time-slot, the packet will be forwarded to CQF queue Q0 (Q0 is open at input port in time-slot i), and sent to the link in time-slot $i+1$. If the packet arrives earlier than the scheduled time-slot, the packet will be forwarded to buffer queue Qb1 and sent to the link at time-slot $i+2$, so that the packet is back to the correct time-slot. If the packet arrives later than the scheduled time-slot, the packet will be forwarded to CQF queue Q0 and be sent at time slot i+1, this packet is delayed by one time-slot.

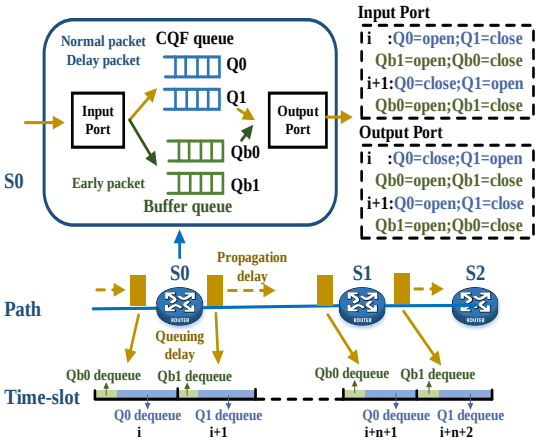


Fig. 5. Deterministic Forwarding Strategy Based on CQF.

When the packet arrives at router S1 in time-slot $i+n$, it is sent to S2 in time-slot $i+n+1$. There is an additional propagation of $n-1$ time-slots between routers S0 and S1. The same process as router S0 will be performed on router S1.

C. Network Measurement and Time-Sensitive Flow Setup

The workflow of SDDN requires cooperation between the SDDN controller and the forwarding device. When the system is started, the controller does not have the information of the topology and propagation delay. To get this information, the SDDN router will periodically send probe packets to measure the propagation delay of links. After the link measurement is completed, each router can obtain the adjacent information and propagation delay information. When the SDDN controller needs to perform time-sensitive flow scheduling, it will actively send a query packet to routers through the out-band control channel. Routers reply to the controller with their adjacency and propagation delay information. The controller will restore the complete network topology based on the information from each router.

When users have the need for deterministic transmission service, the user devices interact with the SDDN controller through an application programming interface (API). Fig. 6 shows the workflow of time-sensitive flow establishment and transmission. The routers will periodically perform link status detection, and upload the link status and topology information to the SDDN controller (step 1). At the beginning of the transmission, the user sends the deterministic transmission service requirement to the gateway router. Information in the requirement includes the source node, destination node, duration, end-to-end delay constraint, and rate of time-sensitive flow. The gateway router forwards the request information to the controller (Step 2). The SDDN controller decides whether to deploy each flow into the network by running the flow scheduling algorithm, and records the status of each link in the network. After the scheduling is completed, the SDDN controller sends configuration information to each SDDN router. The configuration information includes a forwarding method corresponding to a flow ID and a forwarding time-slot allocated to this time-sensitive flow (step 3). The routers record

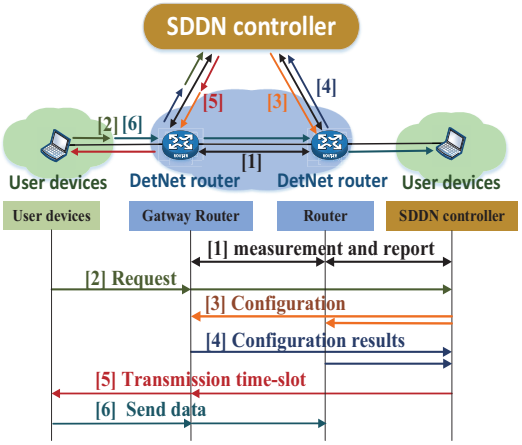


Fig. 6. Workflow of SDDN.

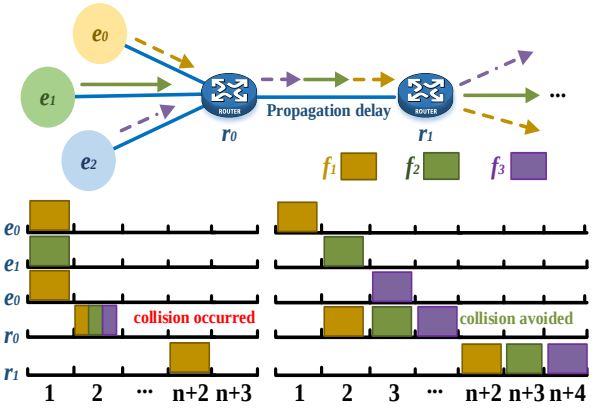


Fig. 7. Scenario with two time-sensitive flows in one edge node.

the forwarding rules of each time-sensitive flow in the flow table according to the configuration information. The router that configured successfully returns confirmation information to the SDDN controller (step 4). After that, the controller sends a deployed result message to the user's device (step 5). The user device will send packets to the gateway router in a specified time-slot, and the router in the network will forward packets according to the pre-configured flow table (step 6).

D. Time-Slot Allocation and Flow Collision

The following part will show the time-slot allocation problem that needs to be solved in large-scale deterministic network flow scheduling. Under the CQF mechanism, time in DetNet is divided into time-slots with a duration of ts. Multiple continuous time-slots constitute a scheduling cycle c. Each time-sensitive flow sends packets in a specified time-slot in scheduling cycle. The goal of SDDN is to enable successfully deployed time-sensitive flows to have deterministic end-to-end delay and low packet loss rate. If the number of packets received in a time-slot exceeds the queue capacity, some packets will be dropped and greatly affect the performance of large-scale deterministic networks. The number of data packets that can be accommodated in each time-slot depends on the rate of the outport and the duration of the time-slot. During the network measurement process, the transmission rate of output port bw will be reported to the controller. The time-slot duration ts and the duration of a scheduling cycle c are determined in the controller and shared with the forwarding devices during the router configuration process. When the rate of packets arriving exceeds bw, some packets will be dropped due to the queue capacity.

To explain this issue more clearly, an example can be given in Fig. 7 to illustrate the collision of three flows. There are three edge nodes e_0, e_1, e_2 and two SDDN routers r_0, r_1 in the network. Time-sensitive flows f_1, f_2 and f_3 need to be deployed. The path assigned by SDDN controller to f_1 is e_0-rt_0-rt_1, the path of f_2 is e_1-rt_0-rt_1, and the path of f_3 is e_2-rt_0-rt_1. All of three flows send packets at the first time-slot of the scheduling cycle, and the router in the network can accept only one packet per time-slot. The packets that exceed the queue capacity will arrive r_0 at time-slot 2, the packets of f_2 and f_3 are dropped. If we specify f_2 send packets in the second slot of a scheduling cycle, and f_3 send packets in the third time-slot, the collision can be avoided. The scheduling algorithm in SDDN controller is necessary to consider the time-slots allocation of flows while calculating routing. Previous works solving such time-slot allocation problems separated from the routing calculation, which leads to poor performance of the scheduling result. We jointly consider the interdependence of path routing and time-slot allocation in our algorithm.

IV. Problem Formulation

In this section, we formulate the joint optimization problem of time-Slot allocation and traffic steering as an ILP model. All the notations used in our paper are listed in Table I. Our objective is to maximize the number of successfully deployed time-sensitive flows.

A. Defination of Network Topolgy, and Time-Slots

The inputs of the ILP model is the topology of the network and the set of time-sensitive flows that need to be scheduled. The DetNet network is modeled as $G(V, L)$, where V denotes the set of nodes and L denotes the set of links. The set of time-sensitive flows is F, where time-sensitive flow $f_i \in F$. Each time-sensitive flow needs to be transmitted from source node src_i to destination node dst_i. $interval_i$ is the interval between data sending, which is a fixed value. $slot_i$ is the specified transmission time-slot at the source node. $delay_i$ denotes the worst-case end-to-end delay requirements. In this paper, we define the duration of time-slot and scheduling cycle according to the $interval_i$ of each flow.

$$ts = GCD(interval_1, interval_2, \cdots, interval_n), \quad (3)$$

$$c = LCM(interval_1, interval_2, \cdots, interval_n), \quad (4)$$

where ts represents the duration of a time slot, c represents the duration of a scheduling cycle, and the functions GCD and LCM represent the operations of the least common multiple and the greatest common divisor, respectively.

TABLE I
NOTATIONS USED IN THE PAPER.

Notation	Description
$G(V, L)$	The network topology, where V denotes the set of nodes and L denotes the set of links
a_{uv}	The propagation delay from node u to node v
bw_{uv}	The bandwith of link $(u, v) \in L$
F	The set of time-sensitive flows which send package periodically
n	The number of time-sensitive flows in F
f_i	The time-sensitive flow i in F, where $i<n$
c	The duration of a scheduling cycle
ts	The duration of a time-slot
x_{iuv}	A binary variable, it takes 1 if f_i forwarded on link (u, v), 0 otherwise, where $f_i \in F$
src_i	The source of time-sensitive flow f_i, where $f_i \in F$
dst_i	The destination of time-sensitive flow f_i, where $f_i \in F$
$interval_i$	The transmission interval between two packets of time-sensitive flow f_i, where $f_i \in F$
r_i	The sending rate of time-sensitive flow f_i, where $f_i \in F$
$delay_i$	The maximum end-to-end delay requirement of time-sensitive flow f_i, where $f_i \in F$
$slot_i$	The transmission slot of time-sensitive flow f_i at the edge node, where $f_i \in F$
$path_i$	The set of links that f_i passed through, where $f_i \in F$
hop_i	The number of hops from source node to destination node in f_{path}, where $f_i \in F$. it takes 0 if $path_i$ is not exist
$hop_{src_i,u}$	The number of hops from source node to node u in $path_i$ of f_i, where $f_i \in F$. it takes 0 if $path_i$ is not exist
M^t_{ijk}	A binary variable, it takes 1 if f_i occupies the slot t in a scheduling cycle of link (j, k), 0 otherwise, where $f_i \in F$
B_i	A binary variable, it takes 1 if f_i is successfully deployed into network

B. Constraints and Objective

To deploy time-sensitive flows in the network, the controller needs to allocate a reliable path and a specific transmission time-slot for each time-sensitive flow. Scheduling schemes that can be used by time-sensitive flows need to satisfy the following constraints.

Flow conservation constraint: For all nodes in the network, the flow conservation constraint needs to be satisfied. We define a binary variable x_{iuv}. It takes 1 if f_i link (u, v) in the path of f_i, 0 otherwise, where $f_i \in F$. If u and v are not the source node or destination node of flow f_i, each flow entering node u leaves in another link. So the amount of data entering node u the same as the data leaving node u. For an adjacent node v of u, x_{iuv} indicates whether flow f_i enters node u from link (u, v), and x_{ivu} indicates whether flow i is sent out from link (u, v). The following constraints should be satisfied:

$$\sum_{v \in V|(u,v) \in L} x_{iuv} - \sum_{v \in V|(v,u) \in L} x_{ivu} = 0. \tag{5}$$

If the u is the source node of f_i, f_i will be sent from node u to other nodes, but will not enter node u from other nodes. The flows be sent out from node u occupies one more links than flows enter into node u. The following constraints should be satisfied:

$$\sum_{v \in V|(u,v) \in L} x_{iuv} - \sum_{v \in V|(v,u) \in L} x_{ivu} = 1. \tag{6}$$

If the u is the destination node of f_i, The flows enter into the node u occupies one more links than flows be sent out from node u. The following constraints should be satisfied:

$$\sum_{v \in V|(v,u) \in L} x_{ivu} - \sum_{v \in V|(u,v) \in L} x_{iuv} = 1. \tag{7}$$

Loop-free constraint: In the scheduling of time-sensitive flows, loops in path will not help the deployment of time-sensitive flows, but will occupy more network resources. When there is a loop in the path, time-sensitive flow will enter or leave a particular node twice. To avoid loops in the path of f_i, each link can only be passed once:

$$\sum_{v \in V|(v,u) \in L} x_{ivu} \leq 1. \tag{8}$$

Collision-free constraint: The scheduling of time-sensitive flows should satisfy the collision-free constraints. We define the binary variable M^t_{iuv}. If the f_i occupies the time-slot t of the link (u, v) in a scheduling cycle, $M^t_{iuv} = 1$, 0 otherwise. A binary variable B_i takes 1 if f_i is successfully deployed into the network, 0 otherwise. The collision-free constraint of link (u, v) is:

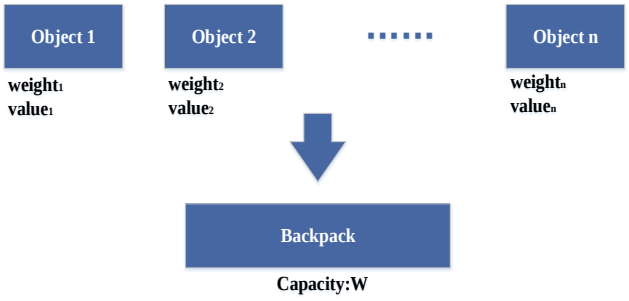
$$\sum_{i=0}^{n-1} M^t_{iuv} B_i r_i \leq bw_{uv}, \tag{9}$$

where r_i represents the rate of time-sensitive flow f_i, and bw_{uv} represents the bandwidth of the link (u,v). The meaning of this constraint is that the total rates of all time-sensitive flows assigned to the time-slot cannot exceed the link bandwidth. Note that before the time-sensitive flow f_i is finished, M^t_{iuv} and r_i are maintained in the SDDN controller to record the current status of the network. When a new time-sensitive flow scheduling is performed, the impact of existing time-sensitive flows will also be taken into account by considering the M^t_{iuv} and r_i of existing time-sensitive flows.

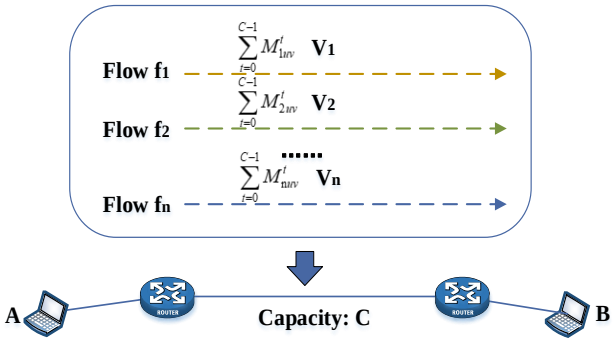
Due to the propagation delay in WAN, the packets sent within one cycle may be received within the next cycle, and the value of M^t_{iuv} should meet the following rule:

$$\begin{gathered} \forall f_i \in F, i \in [0, n-1], \forall u \in E, \forall v \in E, \forall t \in \left[0, \frac{c}{ts} - 1\right] \\ M^t_{iuv} = 1 \\ \text{s.t.} \\ (u, v) \in path_i, \alpha \in \left[0, \frac{c}{ts} - 1\right] \\ t_1 = \left(slot_i + \alpha[\frac{interval_i}{ts}] + hop_{src_i u}\right) \bmod \left(\frac{c}{ts}\right) \\ M^{t_1}_{iuv} = 1. \end{gathered} \tag{10}$$

End-to-end delay constraint: The scheduling of time-sensitive flows needs to meet the worst-case end-to-end delay constraint. The end-to-end delay mainly includes the queuing delay in the router and the propagation delay in the link. The queuing delay is caused by CQF mechanism, and can be calculated from the duration of time-slot and the number of hops, where the hops of the flow f_i is hop_i. The propagation delay of each link can be measured and stored in the SDDN controller, where the propagation delay of the link (u, v) is



(a) Knapsack problem



(b) Joint optimization problem in large-scale deterministic networks

Fig. 8. Reduction from Knapsack to joint optimization problem in large-scale deterministic networks.

a_{uv}, and the end-to-end delay requirement $delay_i$ are provided by users. The worst-case end-to-end delay constraint is:

$$hop_i * ts + \sum_{u \in V | v \in V} x_{ivu} a_{uv} \leq \lfloor \frac{delay_i}{ts} \rfloor. \tag{11}$$

Object: Our objective is to maximize the number of successfully deployed time-sensitive flows in the network. The objective of the model is formulated as follows:

$$max \sum_{i=0}^{n-1} V_i B_i, \tag{12}$$

where V_i is used to measure the priority of f_i. To simplify the problem, our paper considers that all time-sensitive flow priorities are the same, and the objective function is equivalent to maximizing the number of successfully deployed time-sensitive flows.

C. Complexity Analysis

The scheduling problem of time-sensitive flows is an NP-hard optimization problem. Theorem 1 shows the NP-hardness of joint optimization of traffic steering and time-slot allocating problem.

Theorem 1. The joint optimization problem of time-slot allocation and traffic routing for large-scale deterministic networks is NP-hard.

Proof. We can prove that the joint optimization problem in large-scale deterministic networks is an NP-hard problem by reducing the knapsack problem to our problem. The knapsack problem has been proved as NP-hard in [39]. Given a set of objects, each with a weight and a value, the knapsack problem determines whether to put items in the knapsack to get the maximum total value while the total weight below the knapsack capacity. We give an instance of the knapsack problem in Fig. 8(a). There are n objects with different values $value_i$ and weight $weight_i$. The capacity of the knapsack is W, the set of objects is O, where the ith item is O_i, with value $value_i$ and weight $weight_i$. The objective function of the knapsack problem is:

$$max \sum_{i=0}^{n-1} x_i value_i, \tag{13}$$

where x_i is a binary variable, it takes 1 if the item O_i is selected in the knapsack, 0 otherwise.

The knapsack problem should satisfy the following constraint:

$$\sum_{i=0}^{n-1} x_i weight_i < W. \tag{14}$$

With the instance of the knapsack problem, an instance of a simplified joint optimization problem of time-slot allocation and traffic routing can be constructed in polynomial time as follows: (1) Create a network topology $G(V, L)$ with two SDDN routers and one physical link and there are C time slots in a scheduling cycle of the link. All time-sensitive flows are generated in the hosts, which connect to the routers. (2) Each object that needs to be put into the knapsack can map to a time-sensitive flow from A to B. The set of time-sensitive flows is F, f_i is the ith time-sensitive flow, and the end-to-end delay requirements are infinite for f_i. (3) Allocate the time-sensitive flows in the network to get the weighted maximum number of successfully deployed time-sensitive flows.

The constructed time-sensitive flow scheduling problem is shown in Fig. 8(b), where the objective function is as follow:

$$max \sum_{i=0}^{n-1} V_i B_i. \tag{15}$$

Considering that there is only one link in the network, our problem should satisfy (10) and (11) . In a scheduling cycle with W time-slots, the constraint of our problem is as follow:

$$\sum_{t=0}^{C-1} \sum_{i=0}^{n-1} M_{iuv}^t B_i \leq C. \tag{16}$$

The mapping relationship between the variables of the knapsack problem and our problem:

$$\begin{cases} B_i \leftarrow x_i \\ \sum_{t=0}^{C-1} M_{iuv}^t \leftarrow [\frac{W}{weight_i}] \\ C \leftarrow W \\ V_i \leftarrow value_i \end{cases}. \tag{17}$$

For a knapsack problem, we can reduce it to a simplified joint scheduling problem of time-slot allocation and traffic

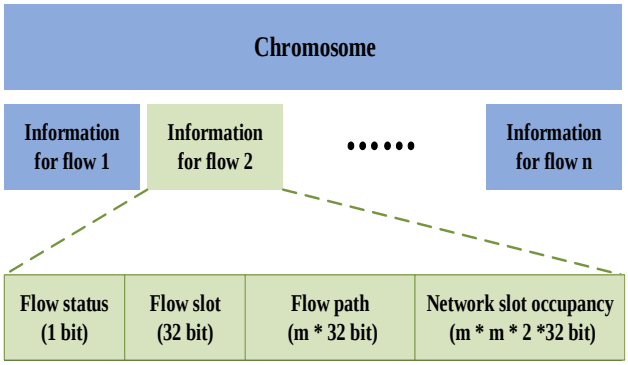


Fig. 9. Chromosome architecture.

routing. Obviously, the studied problem in large-scale networks has higher complexity. So the problem can prove to be an NP-hard problem. Furthermore, it is clear that our problem belongs to NP. For a given time-sensitive flow scheduling scheme, the validity can be checked in polynomial time by formula (5–11).

V. Algorithm Design

Since the studied problem is proved to be NP-hard, we design a heuristic algorithm named GDNTS to solve the ILP model. The GDNTS algorithm maximizes the number of successfully deployed time-sensitive flows in the network while guaranteeing the end-to-end delay requirements.

A. Overview of GDNTS

Genetic algorithm is a heuristic algorithm that simulates natural genetic operations. Our GDNTS is based on the idea of genetic algorithm and starts with a set of randomly generated chromosomes. New generation of chromosomes is continuously generated through evolution. In this process, a better chromosome has more opportunities to retain their information to the next generation. After multiple evolutions, the chromosome with the maximum value of the fitness function is the output of the algorithm. Some key definitions in this algorithm are as follows:

(i) *Chromosome*: Chromosome is a special data structure that carries information. In GDNTS, the information carried by the chromosome is the scheduling results of all flows. The data structure of chromosome is shown in Fig. 9. The information of chromosome includes binary variables B_i of whether to deploy a flow (flow status), the transmitting time-slot $slot_i$ (flow slot), the path $path_i$ of f_i (flow path) and the time-slot occupancy in the entire network (network slot occupancy).

(ii) *Population*: The set of chromosomes is called population and each population contains pop_{size} chromosomes. pop_{cur} represents the set of the current chromosomes and pop_{next} represents the set of the next generation chromosomes.

Algorithm 1 GDNTS algorithm

Input: $G(V, E)$, F;
Output: Deployed results of all flows;
Execute $Initialize()$ to get the initial population;
$K \leftarrow 0$;
while K<**the maximal iteration number do**
  $pop_{\text{next}} \leftarrow$ Select chromosomes with high fitness function values from pop_{cur};
  $pop_{\text{next}} \leftarrow$ Select pairs from pop_{next} and cross their information to generate new chromosome;
  Select chromosome from pop_{next}, randomly mutate the information;
  Calculate the fitness value of pop_{next};
  $Chromosome_{best} \leftarrow \max(F_{\text{fit}}(pop_{\text{next}}))$;
  $pop_{cur} \leftarrow pop_{\text{next}}$;
  $pop_{\text{next}} \leftarrow \varnothing$;
  if convergence evaluation parameters Δ small enough **then**
    break;
  end if
  $K \leftarrow K + 1$;
end while
return deployed results of all flows;

Algorithm 2 Initialize function

Input: Current population pop_{cur};
Output: Current population pop_{cur} with randomly generated deployed results of all flows ;
$pop_{cur} \leftarrow \varnothing$ $i \leftarrow 0$;
for Chromosome $i \in$ the population pop_{cur} **do**
  for $f_j \in F$ **do**
    Initialize $slot_j$;
    $path_j \leftarrow$ random-routing(G, f_j);
    if $path_j$ meets constraints **then**
      Deploy f_j into network;
      Update information in i;
    end if
  end for
end for

(iii) *Evolution*: Evolution is the operation of generating the next generation population pop_{next} based on the current population pop_{cur}. In evolution, the algorithm first selects some chromosomes from pop_{cur} with higher fitness value and inherits them to pop_{next}. After that, a crossover operation is performed to select pairs of chromosomes and exchange part of their information to generate new individuals. Finally, mutation operation randomly changes the information in some chromosomes with a certain probability P_m.

(iv) *Fitness function*: The fitness function is used to evaluate the probability that the information of chromosomes is selected in the genetic process. In GDNTS, the objective function of ILP model is a non-negative maximum func-

Algorithm 3 Random-Routing function

1: **Input:**$G(V,E)$, f_i;
2: **Output:** $path_i$;
3: $flag \leftarrow false$, visited.push(src_i);
4: $alter \leftarrow \varnothing$;
5: **for** $node\ i \notin visited$ & link ($visited.back()$,i) exists **do**
6: $alter.push(i)$;
7: **end for**
8: Randomly disrupt the order of elements in $alter$;
9: **for** $node\ j \in alter$ **do**
10: $visited.push(j)$;
11: **if** visited.top() = dst_i **then**
12: flag ← true;
13: **end if**
14: **if** flag = true **then**
15: break;
16: **end if**
17: random-routing($G(V,E), f_i$);
18: $visited.pop(j)$;
19: **end for**
20: return $path_i$;

tion, which can be directly used as the fitness function

$$F_{\text{fit}}(ch) = \sum_{i=0}^{n+1} B_i. \tag{18}$$

B. GDNTS Mechanism

Algorithm 1 shows the framework of the proposed algorithm. During the implementation of the algorithm, a population is randomly initialized with a predefined population size, and the proposed coding strategy is used to encode the randomly generated population (line 3). Then the algorithm enters the loop process. Each iteration will generate a new generation of population. The next generation of population pop_{next} is generated from the current population pop_{cur} by the selection algorithm in line 6. The pop_{next} then crossed in line 7, mutated in line 8, and updated fitness values in line 9. After that, the chromosome with the optimal solution will be retained, and the next iteration starts. The algorithm ends when the number of iterations reaches the upper limit or convergence parameter Δ is small enough, where Δ is used to evaluate the convergence of the algorithm. That is, when the iteration of all chromosomes in the current population can not bring improvement in the average fitness, the algorithm terminates. Δ is shown as follow:

$$\Delta = \left| \frac{\sum_{ch \in pop_{\text{next}}}(F_{\text{fit}}(ch)) - \sum_{ch \in pop_{cur}}(F_{\text{fit}}(ch))}{\sum_{ch \in pop_{\text{next}}}(F_{\text{fit}}(ch)) + \sum_{ch \in pop_{cur}}(F_{\text{fit}}(ch))} \right|. \tag{19}$$

At the beginning of the GDNTS, a set of initial solutions needs to be generated randomly. We designed an algorithm to generate a random solution for each flow. Algorithm 2 shows the details of the population initialization. For each flow, the random-routing algorithm is called to randomly generate a path in line 7. The generated path is not guaranteed to meet the constraints in the ILP model. If the path meets the constraints of ILP, the output result will be recorded in the chromosome in line 10.

The process of the random-routing algorithm is shown in Algorithm 3. First, initialize the binary variable $flag$ as a sign of whether a feasible path has been found. It takes true if a path has been found, false otherwise. The array $visited$ stores the visited nodes of a flow and the source node is added to the array at the beginning. The algorithm traverses the unvisited nodes among the neighboring nodes of the last element in $visited$ in random order, and pushes the node need to visit in $visited$ in line 10. If node j is the destination node, set $flag$ as true in line 12 which means a path has been found. If the path has not been found, calls the random-routing algorithm recursively. At the end of the algorithm, a randomly generated path is returned.

We take the selection mechanism that combines the best individual retention strategy and the roulette wheel selection to select the appropriate chromosome (line 6 in Algorithm 1). The best individual retention strategy means directly copying chromosome with the highest value of fitness function to the next generation without participating in crossover and mutation. The remaining individuals are selected using the roulette wheel selection algorithm. The greater the fitness value, the greater the probability of being selected.

The crossover operation is executed on line 7 in Algorithm 1. The information of the same flow is randomly selected from two chromosomes. A probability parameter P_c decides whether to exchange information of this flow in the two chromosomes. When an exchange event occurs, the two chromosomes exchange information of the same time-sensitive flow. If this flow is not deployed in either chromosome, the crossover operation will not generate a new chromosome. If the flow is successfully deployed in any chromosome, the two chromosomes exchange all the information of this flow and check whether the deployment of the flow meets the constraints.

The mutation operation is executed on line 8 in Algorithm 1. A chromosome is selected from the population randomly. The probability P_m determines whether the chromosome needs to mutation. When the mutation occurs, we randomly select an undeployed flow in the chromosome and call the Random-Routing algorithm to generate a path for this flow.

C. Algorithm Complexity Analysis

For random-routing algorithm, this algorithm will be recursed m times in the worst case, where m is the number of nodes in the network topology. In each execution, the algorithm should traverse all the neighbors of the current node, and randomly disrupt the order of neighbors. In this process, we shuffle node order by generating random numbers corresponding to the number of neighbors and sorting them. Therefore, the complexity of the Random-Routing algorithm is

$$O\left(|m|^2 \log |m|\right). \tag{20}$$

In population initialization, random-routing and path validity checks are performed n times, where n is the number of time-

sensitive flows that need to be deployed. For path validity checks, the number of executions will not exceed m. The time complexity of generating initial population is

$$O(|n|(|m| + |m|^2 \log |m|)) = O\left(|n||m|^2 \log |m|\right). \quad (21)$$

When the GDNTS algorithm starts, selection, crossover, and mutation operations will be performed in sequence. The selection operation can be completed in constant time. The complexity of the crossover operation is $O(|m|)$. The random routing algorithm is called in the mutation, the complexity is $O\left(|m|^2 \log |m|\right)$. So that the time complexity of each iteration is:

$$O(|m| + |m|^2 \log |m|) = O(|m|^2 \log |m|). \quad (22)$$

In the worst case, the evolution process will loop $|K|$ times. The total time complexity of the algorithm is:

$$O\left(|n||m|^2 \log |m|\right) + O(|K||m|^2 \log |m|) = O(|n||m|^2 \log |m|). \quad (23)$$

VI. Performance Evaluation

In this section, the performance of GDNTS is stuied. We first introduce the simulation setting. After that, the simulation results are presented.

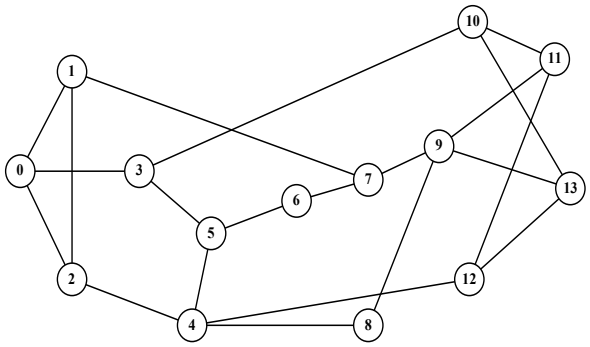
A. Methodology and Simulation Setting

Set up: Considering that the large-scale network topology is complex and diverse, we used two typical network topologies in our simulation, as shown in Fig. 10, including the NSFNET topology (14 nodes, 21 links) and the USNET topology (24 nodes, 43 links). The propagation delay of all links is randomly generated in [1 ms, 10 ms] and the bandwidth of each link is 1000 Mbps. All nodes in the network are SDDN routers with advanced CQF queues, and each time-slot in the network can only be occupied by one time-sensitive flow. In the parameter settings of the algorithm, we set the population size to 80, the crossover probability to 0.5, and the mutation probability to 0.05. To eliminate statistical fluctuations, each group of results is obtained from the averaging on 10 experiments with randomly generated time-sensitive flow requests.

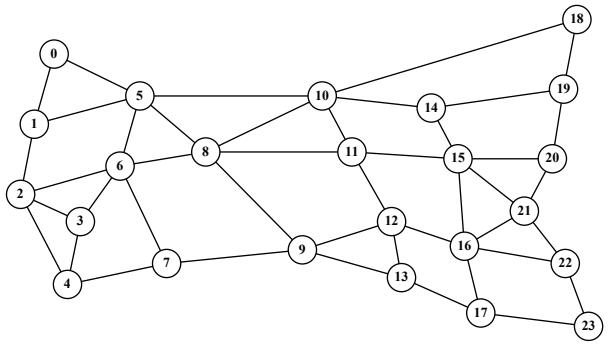
Time-sensitive flows generation: All time-sensitive flows in the experiment are generated before the scheduling algorithm executes. Each time-sensitive flow's source node and destination node are randomly designated. The delay requirements of users are uniformly distributed in [10 ms, 60 ms], and the interval between two packets in a time-sensitive flow is uniformly distributed in [100 us, 600 us].

Benchmark solutions: Load balancing routing and shortest routing are common routing methods in existing wide area networks. We compare the following benchmarks with the proposed GDNTS.

- **Shortest-routing fixed routing (SRFR):** In the SRFR algorithm, all feasible paths are determined through the shortest path algorithm. Time-sensitive flows with longer paths will not be deployed when time-slot conflicts occur.



(a) NSFNET topology



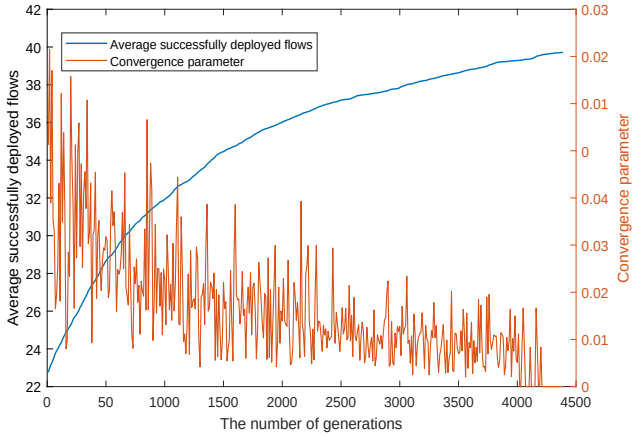
(b) USNET topology

Fig. 10. Simulation topology.

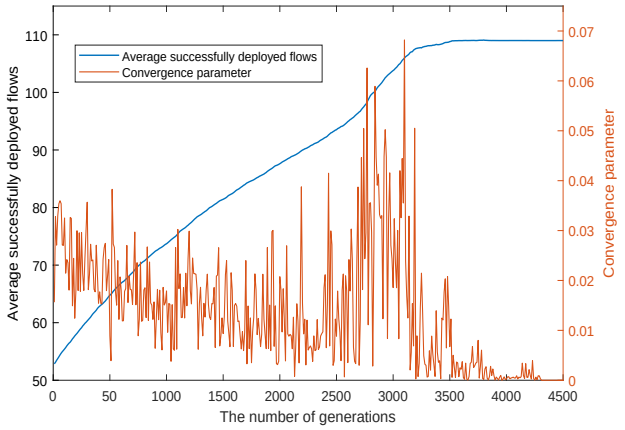
- **Load-balance fixed routing (LBFR):** In the LBFR algorithm, Time-sensitive flows will be allocated to as more links as possible. The idle links will be preferred for path assignment. A conflicting time-sensitive flow will be randomly deleted when a time-slot conflict occurs.

The most significant difference between the benchmark algorithms and our GDNTS is that the routing calculation and time-slot allocation are carried out separately. The benchmark algorithms use different strategies to find paths for time-sensitive flows in advance. After that, time-slot allocation is performed under the fixed route. In fact, time-slot allocation under fixed route is still an NP-hard problem [29]. We use selection and crossover operations in genetic algorithms for heuristic search.

Performance metrics: We use the following performance metrics to evaluate the proposed algorithm: (1) *Convergence and time cost*: The convergence rate and time cost of scheduling algorithms; (2) *Number of successfully deployed time-sensitive flows*: The total number of time-sensitive flows which successfully deployed in the network; (3) *Time-slot utilization ratio*: The ratio of the occupied time-slot to the total number of time-slot in all links; (4) *Throughput of SDDN*: The total throughput of all time-sensitive flows in the network; (5) *Hops*



(a) Convergence in NSFNET topology



(b) Convergence in USNET topology

Fig. 11. Convergence results.

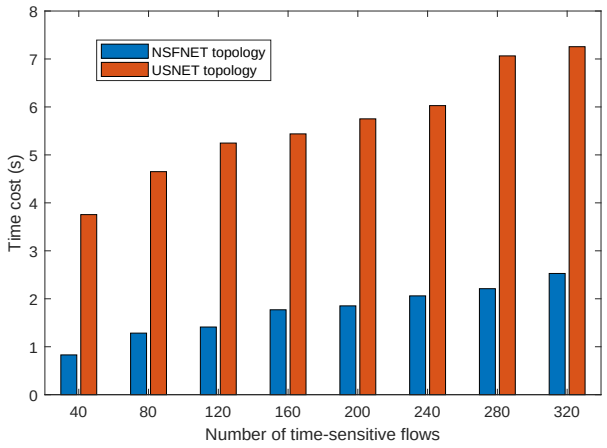


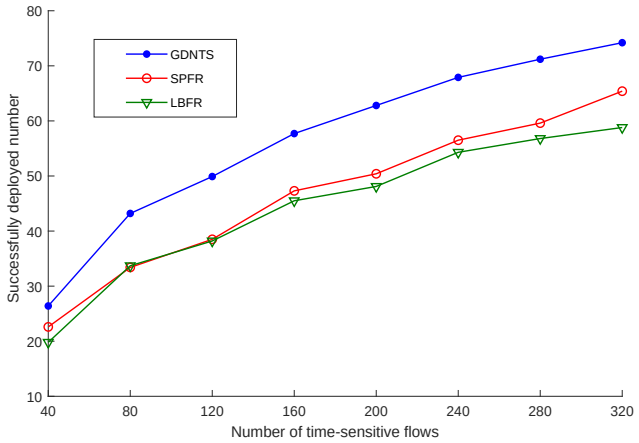
Fig. 12. Time cost.

of successfully deployed time-sensitive flows: The hops of all successfully deployed flows in the scheduling result.

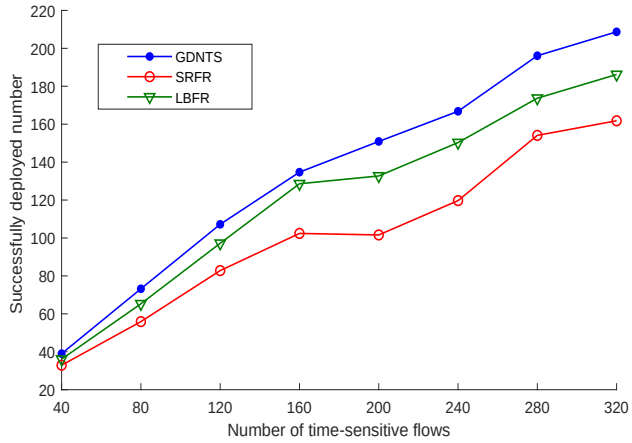
B. Numerical Results

Convergence and time cost: Figs. 11(a) and 11(b) show the execution procedure of successfully deployed time-sensitive flows in the network during the iteration of the GDNTS algorithm. The results in the figures are the convergence of the GDNTS when the number of time-sensitive flows is set as 120 in the NSFNET topology and the USNET topology, respectively. The results show that as the number of iterations increases, the average successfully deployed flows of all chromosomes in the current population increases and eventually converges to a fixed value. Meanwhile, we show the change of the convergence parameter Δ in (19). When the value of Δ is small enough, the algorithm terminates. GDNTS in both networks can end within 4500 iterations and get an acceptable result.

In Fig. 12, we show the actual time cost of GDNTS under different topologies and flow numbers using a server with Intel core i7-10700 CPU and 16 GB of RAM. As far as we know, such kind of scheduling tasks typically take seconds to several minutes [33], which is affected by server performance. From the test results, we can see that the time cost significantly increases with the increase of the number of nodes, and linearly increases with the increase of the number of streams, which is consistent with our conclusion in (23).



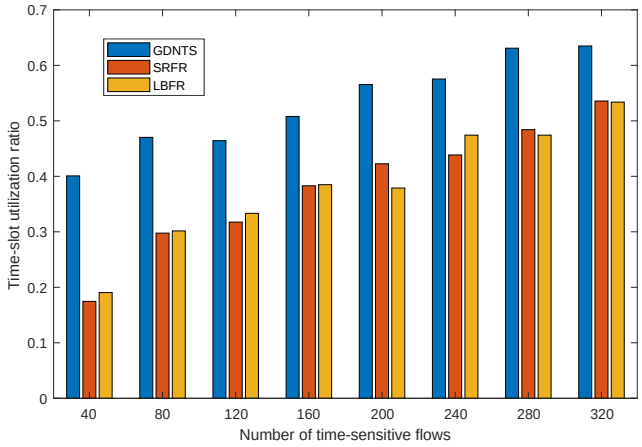
(a) Number of successfully deployment flows in NSFNET topology



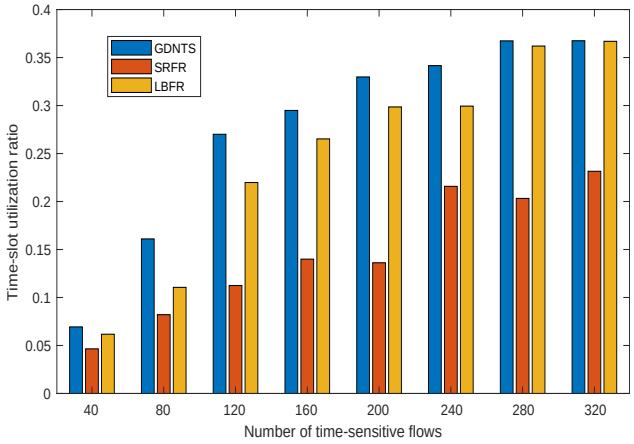
(b) Number of successfully deployment flows in USNET topology

Fig. 13. Number of successfully deployment flows results.

Number of successfully deployed time-sensitive flows: Fig. 13(a) and 13(b) show the number of successfully deployed

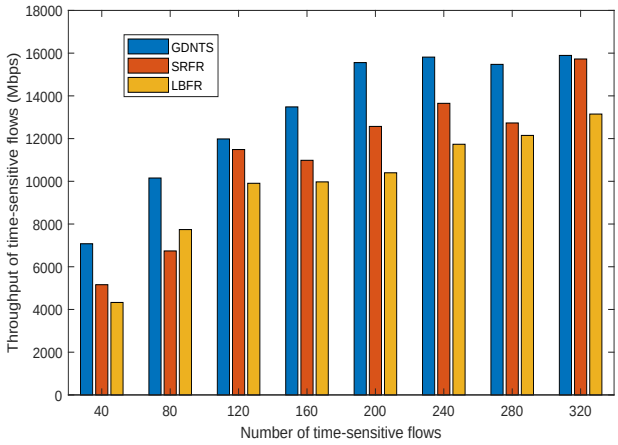


(a) Time-slot utilization ratio in NSFNET topology

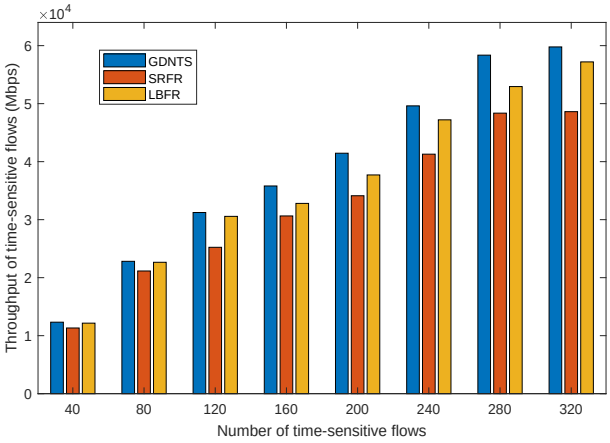


(b) Time-slot utilization ratio in USNET topology

Fig. 14. Time-slot utilization ratio results.



(a) Throughput of algorithms in NSFNET topology



(b) Throughput of algorithms in USNET topology

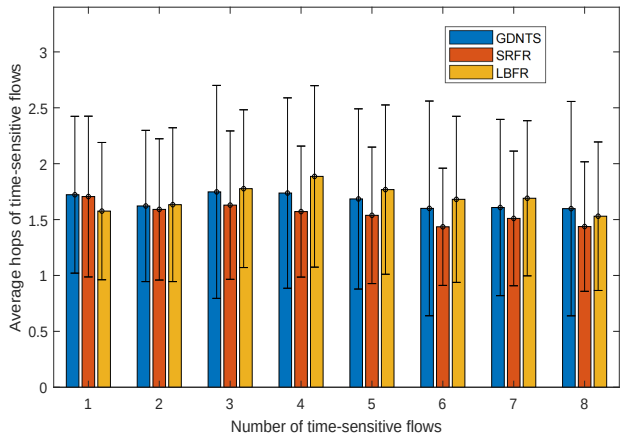
Fig. 15. Throughput results.

time-sensitive flows of different algorithms in NSFNET and USNET. From the results, we can see that the number of successfully deployed flows for each algorithm increases with the growth of time-sensitive flow requests. Compared with SRFR and LBFR, the number of successful deployment flows in GDNTS algorithm has an average increase of 20.52% and 27.18% in NSFNET, an average increase of 11.24% and 32.46% in USNET. The performance gap grows when more time-sensitive flows are required in the networks. The reason is that GDNTS can continuously generate new scheduling results through mutation operations in the algorithm. Therefore, GDNTS has a wider search domain than other algorithms. In SRFR, the paths found by the algorithm are concentrated on a few shorter paths, causing the collision of flows. In LBFR, to achieve load balancing, numerous paths do not meet the end-to-end delay constraint. The results illustrate the progress in the number of successfully deployed flows of GDNTS.

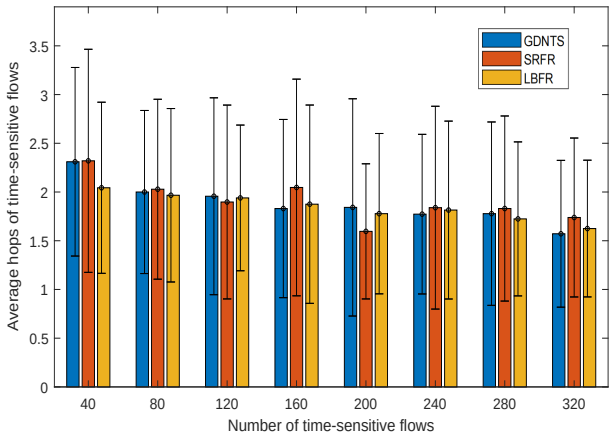
Time-slot utilization ratio: Fig. 14(a) and 14(b) show the average time-slot utilization ratio of all links in NSFNET and USNET for different algorithms. We can see that the time-slot utilization ratio increases with the growth of time-sensitive flow requests. Compared with SRFR and LBFR, the proposed GDNTS improves 27.88% and 17.45% time-slot utilization ratio in average, respectively. This is because GDNTS eliminates the collision of different time-sensitive flows in time-slots through multiple iterations, thereby deploying more time-sensitive flows. The SRFR and LBFR calculate routes for time-sensitive flows in advance, while ignoring the interdependence between time-slots allocation and routing. Thus, GDNTS can use the time-slot resources in the network more efficiently.

Throughput: Fig. 15(a) and 15(b) show the throughput of all time-sensitive flows in NSFNET and USNET using different algorithms. In Fig. 15(a), we can see that when the number of time-sensitive flows is less than 200, the total throughput increases with the increase of time-sensitive flow requests. However, when the number of time-sensitive flows is greater than 200, time-sensitive flow requests will no longer bring an increase in throughput since most links in the NSFNET are already saturated. In Fig. 13(b), throughput increases as the number of time-sensitive flow requests increases. Unlike NSNSFNET, the throughput of LBFR in USNET exceeds that of SRFR, because as the number of network nodes increases, load-balancing algorithms can utilize network resources more effectively. To sum up, GDNTS can always achieves higher total throughput in each case.

Hops of successfully deployed time-sensitive flows:



(a) Hops of algorithms in NSFNET topology



(b) Hops of algorithms in USNET topology

Fig. 16. Hops of successfully deployment flows.

Fig. 16(a) and 16(b) show the hops of all successfully deployed flows in NSFNET and USNET for different algorithms. Compared with SRFR and LBFR, the average hops in GDNTS scheduling results are larger and the maximum performance gap is 0.471 hops and 0.4149 hops, respectively. The reason for this result is that GDNTS allocates longer paths for some time-sensitive flows within the end-to-end delay constraint to deploy more flows. SRFR and LBFR reject some of the time-sensitive flows that cannot be deployed into the network, so that the successfully deployed flows have a shorter average number of hops. Since all the time-sensitive flows in GDNTS still meet the end-to-end delay constraint, this performance drop will not affect the practicality of GDNTS.

Meanwhile, we can point out that among the three algorithms, the standard deviation of hops obtained by GDNTS is larger than that of SRFR and LBSR. Since GDNTS algorithm has no obvious preference for the hops during path scheduling, the results of GDNTS may simultaneously include paths with fewer hops and paths with more hops to maximize the deployment of time-sensitive flows. In contrast, SRFR tends to adopt the path with the smallest number of hops, while LBFR generates hops with higher variance for load balancing.

VII. Limitations and Future Works

A. Limitations

Our solution is based on an ILP model, which may have high computational complexity and scalability issues. As the network size and traffic demand increase, our solution may take longer time to solve the problem or require more computational resources. Our solution also requie the accuracy and completeness of the input data, such as the network topology, traffic demand, and service requirements. If the input data is inaccurate or incomplete, our solution may not be able to find the optimal or feasible solution.

B. Future Works

Collaborative optimization of control plane and data plane: Current hardware implementations and control methods to support deterministic networks remain an open challenge. The implementation of queue scheduling in DetNet utilizes technologies such as network calculus and software-defined queues to schedule at the data layer. By calculating the upper bound of the worst delay of each flow, a routing path and scheduling method that meet the delay requirements are selected for each flow to ensure Deterministic delivery at a per-flow, per-packet level. Or use cycle-based cyclic queue scheduling and other technologies to schedule at the control layer, and use methods such as synchronous packet sending, segment routing, and cyclic queuing forwarding to ensure that the worst end-to-end delay is bounded and reduce delay variation (jitter). Future research work needs to focus on how to achieve collaborative scheduling between different layers to improve network resource utilization and service quality.

Programmable DetNet: The main challenge of DetNet is how to ensure the programmability of paths, that is, how to achieve fast path switching, load balancing, and fault recovery without affecting the application service quality. To this end, multiple aspects need to be considered, such as path selection algorithms, traffic engineering, congestion control, fault detection, state synchronization, etc. The development of programmable data planes has brought new opportunities for DetNet, how to use programmable data planes to achieve reliable queue scheduling still remains an open challenge.

VIII. Conclusion

In this paper, we study the joint optimization of time-slot allocation and traffic steering for large-scale deterministic networks. We first propose the SDDN architecture. With the SDDN architecture, we analyze the problem of traffic steering and time-slot allocation for time-sensitive flows. We formulate the studied problem as an ILP model, and prove that the problem is NP-hard by reducing the knapsack problem to a simplified case of our problem. Accordingly, a heuristic algorithm GDNTS based on genetic algorithm to maximize the number of successfully deployed flows is proposed. Compared with three existing benchmark algorithms, simulation results show that the proposed GDNTS improves the number of successfully deployed flows by 22.85% in average.

IX. ACKONOWLEDGEMENT

This work was supported in part by the Natural Science Foundation of Sichuan (No. 2022NSFSC0543), in part by the National Natural Science Foundation of China (NSFC) (No. 62001087, 62171085, 61871097, 62272428, and U20A20156), in part by Hefei Municipal Natural Science Foundation (No. 2022004), in part by Anhui Provincial Natural Science Foundation (No. 2208085MF167), in part by the Open Research Projects of Zhejiang Lab (No. 2021LC0AB04), and in part by Natural Science Foundation of Jiangsu Province (No. BK20221261).

REFERENCES

[1] M. Agiwal, A. Roy, and N. Saxena, "Next generation 5G wireless networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 1617–1655, 2016.

[2] H. Cho, S. Jung, and H. Jee, "Real-time interactive AR system for broadcasting," in *Proc. IEEE VR*, 2017.

[3] M. S. Elbamby, C. Perfecto, M. Bennis, and K. Doppler, "Toward low-latency and ultra-reliable virtual reality," *IEEE Netw.*, vol. 32, no. 2, pp. 78–84, 2018.

[4] D. Kombate and Wanglina, "The Internet of vehicles based on 5G communications," in *Proc. IEEE iThings, IEEE GreenCom,IEEE CPSCom, and IEEE SmartData*, 2016.

[5] L. Kovacs, T. Haidegger, and I. Rudas, "Surgery from a distance—application of intelligent control for telemedicine," in *Proc. IEEE SAMI*, 2013.

[6] H. Laaki, Y. Miche, and K. Tammi, "Prototyping a digital twin for real time remote control over mobile networks: Application of remote surgery," *IEEE Access*, vol. 7, pp. 20325–20336, 2019.

[7] M. Perez *et al,*, "Impact of delay on telesurgical performance: Study on the robotic simulator dV-Trainer," *Int. J. Comput. Assisted Radiology Surgery*, vol. 11, pp. 581–587, 2016.

[8] C. She *et al.*, "A tutorial on ultrareliable and low-latency communications in 6G: Integrating domain knowledge into deep learning," *Proc. IEEE*, vol. 109, no. 3, pp. 204–246, 2021.

[9] J. Kommineni and A. Malinowski, "Study of time delay distribution in IP protocol family for IP-network based control applications," in *Proc. IEEE IES*, 2005.

[10] A. Nasrallah, V. Balasubramanian, A. Thyagaturu, M. Reisslein, and H. Elbakoury, "Cyclic queuing and forwarding for large scale deterministic networks: A survey," in *arXiv e-prints arXiv:1905.08478,* 2019.

[11] "WG802.1. IEEE Standard for Local and Metropolitan Area Networks-Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks," IEEE Std 802.1AS, 2011.

[12] "WG802.1. IEEE Standard for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks Amendment 14: Stream Reservation Protocol(SRP)," IEEE Std 802.1Qat, 2010.

[13] "IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks – Amendment 31: Stream Reservation Protocol(SRP) Enhancements and Performance Improvements," IEEE Std 802.1Qcc-2018, 2018.

[14] "IEEE Standard for Local and Metropolitan Area Networks - Bridges and Bridged Networks - Amendment 24: Path Control and Reservation," IEEE Std 802.1Qca-2015, 2016.

[15] "IEEE Standard for Local and Metropolitan Area Networks: Cyclic Queuing and Forwarding," IEEE 802.1Qch-2017.

[16] J. Joung, J.-D. Ryoo, T.-S. Cheung, Y. Li, and P. Liu, "Asynchronous Deterministic Networking Framework for Large-Scale Networks," Internet Engineering Task Force, 2022, draft-joung-detnet-asynch-detnet-framework-00.

[17] B. Varga, J. Farkas, A. G. Malis, and S. Bryant. "Deterministic Networking (DetNet) Data Plane: IP over IEEE 802.1 Time-Sensitive Networking (TSN)," 2021, RFC 9023.

[18] N.Finn and P.Thubert, "Deterministic Networking Problem Statement," Internet Engineering Task Force,2019 Internet-Draft draft-ietf draft-ietf-detnet-problem-statement.

[19] N.Finn, P.Thubert, B.Varga, and J.Farkas, "Deterministic Networking Architecture," Internet Engineering Task Force, 2019, Internet-Draft draft-ietf detnet-architecture-13.

[20] B. Varga, J. Farkas, L. Berger, A. G. Malis, and S. Bryant, "Deterministic Networking (DetNet) Data Plane Framework," IETF RFC8938,2020; rfc-editor.org/rfc/rfc8938.txt

[21] X. Geng, M. Chen, Y. Ryoo, D. Fedyk, R. Rahman, and Z. Li, "Deterministic Networking (DetNet) YANG Model," Internet Engineering Task Force, 2021, draft-ietf-detnet-yang-12

[22] N. Shibata, S. Kaneko, K. Honda, and J. Terada, "Deterministic layer-2 ring network with autonomous dynamic gate shaping for multi-service convergence in 5G and beyond," in *Proc. OFC*, 2020.

[23] M. Hernández, R.Tombi, S. Kopp, D. Batista, C.Margie, and R. Silveira, "An analytic-deterministic model for traffic prioritization in software defined networks with network calculus," in *Proc. SBRC*, 2015.

[24] Y. Li, S. Ren, G. Li, F. Yang, J.-D. Ryoo, and P. Liu, "IPv6 Options for Cyclic Queuing and Forwarding Variants," 2022, draft-yizhou-detnet-ipv6-options-for-cqf-variant-00.

[25] W. Steiner, "An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks," in *Proc. IEEE RTSS*,2010.

[26] N. Nayak, F. Durr, and K. Rothermel, "Time-sensitive software defined network (TSSDN) for real-time applications," in *Proc. RTNS*,2016.

[27] M. Ibrar *et al.*, "IHSF: An intelligent solution for improved performance of reliable and time-sensitive flows in hybrid SDN-based FC IoT systems," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3130–3142, 2020.

[28] Krolikowski *et al.*, "Joint routing and scheduling for large-scale deterministic IP networks," *Comput. commun.*, vol. 165, pp. 33–42, 2021

[29] A. Atallah, G. B. Hamad, and O. A. Mohamed, "Routing and scheduling of time-triggered traffic in time-sensitive networks," *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4525–4534, 2020.

[30] W. Steiner, S. S. Craciunas, and R. S. Oliver, "Traffic planning for time-sensitive communication," *IEEE Commun. Standards Mag.*, vol. 2, no. 2, pp. 42–47, 2018.

[31] S. Craciunas, R.Oliver, and W.Steiner. "Formal scheduling constraints for time-sensitive networks." *arXiv e-prints arXiv:1712.02246*, 2017.

[32] N. Nayak, F. Durr, and K. Rothermel, "Incremental flow scheduling and routing in time-sensitive software-defined networks," *IEEE Trans. Ind. Informat.*, vol. 14, no. 5, pp. 2066–2075, 2018.

[33] J. Yan *et al.*, "Injection time planning: Making CQF practical in time sensitive networking," in *Proc. IEEE INFOCOM*, 2020.

[34] J. Xue, G. Shou, Y. Liu, Y. Hu, and Z. Guo, "Time-aware traffic scheduling with virtual queues in time-sensitive networking," in *Proc, IFIP/IEEE IM*, 2021.

[35] Z. Pang *et al.*, "Flow scheduling for conflict-free network updates in time-sensitive software-defined networks," *IEEE Trans. Ind. Informat.*, vol. 17, no. 3, pp. 1668–1678, 2021.

[36] S. Chen, J. Leguay, S. Martin, and P. Medagliani, "Load balancing for deterministic networks," in *Proc. IFIP Networking*, 2020.

[37] Towards Network-Wide Scheduling for Cyclic Traffic in IP-based Deterministic Networks

[38] M. Seaman. *Paternoster policing and scheduling, Revision 2.1.* [Online]. Available: http://www.ieee802.org/1/files/public/docs2019/crseaman-paternoster-policing-scheduling-0519-v04.pdf

[39] M. Garey and D. Johnson, "Computers and intractability: A guide to the theory of NP-Completeness," in *Proc. W. H. Freeman and Co*, 1979.

[40] Mohajer *et al.*, "Heterogeneous computational resource allocation for NOMA: Toward green mobile edge-computing systems," *IEEE Trans. Services Comput.*, vol. 16, no. 2, pp. 1225–1238, 2022.

[41] Mohajer *et al.*, "Energy-aware hierarchical resource management and backhaul traffic optimization in heterogeneous cellular networks," *IEEE Syst. J.* vol. 16, no. 4, pp. 5188–5199, 2022.

[42] Dong *et al.*, "Energy-efficient hierarchical resource allocation in uplink-downlink decoupled NOMA HetNets," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 2, pp. 3380–3395 , 2023.

Wenhao Wu is currently pursuing the Bachelor degree in Internet of Things engineering with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu, China. His research interests include deterministic network, software defined networks, and network function virtualization.

Xiaoning Zhang received the B.S., M.S., and Ph.D. degrees in Communication and Information Engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 2002, 2005, and 2007, respectively. He is currently an Professor with School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu, China. His research interests include network design, software defined networks and network function virtualization.

Jiaming Pan is currently pursuing the Master degree in Information and Communication Engineering with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu, China. His research interests include deterministic network, network optimization, software defined networks, and network function virtualization.

Yihui Zhou is currently pursuing the Bachelor degree in Internet of Things Engineering with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu, China. His research interests include deterministic network, network optimization, software defined networks, and network function virtualization.