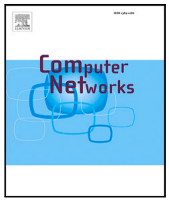
Computer Networks 233 (2023) 109863

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



Congestion-aware delay-guaranteed scheduling and routing with renewal optimization

Xiangyu Ren [a,*], Lin Cai [a], Pu Yang [a], Jiequ Ji [b]

[a] Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada
[b] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu, China

ARTICLE INFO

Keywords:
Delay-guaranteed scheduling and routing
Cross-layer design
Congestion-aware
Renewal optimization

ABSTRACT

The emergence of time-critical applications imposes great challenges on traditional best-effort data networks. Such applications demand delay-guaranteed services with different granularity. In this paper, we propose a novel fully distributed approach that provides delay-guaranteed services at the packet level within an autonomous system. Specifically, we adopt priority queues with fixed buffer sizes to provide differentiated services and per-hop delay upper bound and explore path diversity in the network to achieve multiplex gain. Network congestion is a major cause of long delay. To address this issue, a virtual queue manager is deployed at each node in the network to exchange their local queue information with neighbors periodically. The exchanged queue information reflects the congestion status of the neighborhood so the nodes can avoid congested routes in making routing decisions. Given the rich control space including routing and queuing decisions, we aim at maximizing the overall network utility. Due to the randomness caused by network dynamics, we transform the utility maximization problem into renewal optimization which is solved at each node. A delay laxity-based reward function and a weighted queue time cost are designed to characterize each decision. To solve the renewal optimization problem, an algorithm named DSROpt is proposed using an iterative approach. Extensive experiments are conducted to verify the performance of the proposed solution using NS-3. Simulation results show that the proposed solution can guarantee packet-level delay while achieving significant performance improvements in goodput and network utility over the state-of-the-art.

1. Introduction

The Internet is facing new challenges and requirements characterized by stringent QoS requirements such as extremely low latency and extremely high reliability [1,2]. Typical applications including metaverse, digital twin, and real-time control involve large volumes of both delay-sensitive (DS) and non-delay sensitive (NDS) data exchange [3–5]. Although bandwidth over-provisioning has been adopted as an engineering solution to satisfy the stringent QoS requirements, how to effectively and efficiently support both DS and NDS applications remains an open issue [6,7].

With the existing best-effort Internet services, packets may experience packet loss and unexpected long latency caused by congestions or software/hardware failures [8,9]. The distinctions among packets with different priorities, such as delay requirements, are ignored on the Internet where all packets are treated equally. Such design is undesirable and inefficient with the growing demand for DS applications. It is crucial to introduce priority to networks and provide differentiated services (DiffServ) to different applications to ensure service quality and high network resource efficiency.

The routing and scheduling problem to provide end-to-end (e2e) delay-guaranteed service for DS applications has been extensively studied from different aspects. In recent years, the deterministic network (DetNet) led by the IETF DetNet working group has become a hot research topic to provide delay-guaranteed services on a per-deterministic-flow basis by exploring explicit data paths [10,11]. The DetNet takes the joint effort of both layer three routed segments and layer two bridged segments using technologies such as multiprotocol label switching (MPLS), software-defined network (SDN), and IEEE 802.1 Time-Sensitive Networking (TSN) [12–14] in a centralized manner. The data traffic of each deterministic flow is delivered with guaranteed delay and low delay variation constraint via resource reservation [15]. To find the optimal routes with guaranteed delay in a given network, several routing algorithms have been proposed using the worst-case delay [16–18]. However, the existing approaches have various limitations. Resource reservation-based solutions guarantee delay at the cost of multiplexing gain. The worst-case delay-based routing algorithms can be too conservative to serve DS applications with low latency requirements. The flow-based traffic engineering solutions face scalability

* Corresponding author.
E-mail addresses: jamesrxy@uvic.ca (X. Ren), cai@ece.uvic.ca (L. Cai), puyang@uvic.ca (P. Yang), jiequ@nuaa.edu.cn (J. Ji).

https://doi.org/10.1016/j.comnet.2023.109863
Received 15 February 2023; Received in revised form 2 May 2023; Accepted 2 June 2023
Available online 8 June 2023

issues and cannot guarantee delay at the packet level. In addition, these solutions are unaware of the deadlines of each packet when making routing decisions and fail to handle congestion caused by network dynamics.

On the other hand, the distributed e2e delay-guaranteed solution at the packet level is underexplored given the following challenges. First, it is difficult to handle large variations in the e2e delay. The highly dynamic feature in e2e delay is mainly caused by the varying queuing delay at each hop in the network and is difficult to predict due to the randomness in network traffic. Second, the diversity in delay requirements. For most DS applications, packets belonging to the same flow can have different priorities or delay requirements. For example, I/B/P frames have different importance for MPEG videos [19]. How to effectively guarantee various delay requirements at the same time, i.e., satisfying high-priority packets without starving the low-priority packets, remains a critical task. Last but not least, how to efficiently handle the bursty traffic while guaranteeing delay with good load balance is challenging. The traditional shortest-path routing algorithms can result in heavy congestion in certain paths due to bursty traffic causing long queuing delay and high packet loss rate.

To bridge the gap, we propose a novel distributed, delay-guaranteed, and congestion-aware network architecture called delay-guaranteed scheduling and routing (DSR) which provides DiffServ to various applications and guarantees the packet-level end-to-end delay. DSR explores a new possibility that enables collaboration across layers and among neighboring routers. The solution is feasible to be adopted in an autonomous system and is backward-compatible with existing network infrastructure with acceptable modifications. Specifically, each DS packet carries a maximum tolerable delay requirement called *delay budget* specified by the source application. The network is responsible for delivering the packet within its delay budget using information collected from both the network layer and the link layer. The priority queue is adopted to provide DiffServ to packets with different delay requirements. Given the randomness of network traffic, we rely on the per-hop delay upper bound to guarantee the e2e delay, where each priority queue is assigned a fixed buffer size and service rate. Moreover, to avoid congestion due to bursty traffic in the network, neighboring routers exchange their congestion information periodically so the upstream node can adjust routing decisions in advance.

In addition, an efficient packet scheduling algorithm is fundamental to a distributed routing protocol to achieve high network utility. Although many routing and scheduling algorithms have been heavily explored in the literature to maximize throughput [20–22], these throughput-optimal scheduling may suffer severe queuing delays because only the backlog-gradient are considered and they are unaware of the temporal correlation between each scheduling decision.

To address these issues, we develop a scheduling and routing solution named DSROpt that makes routing decisions and schedules packets to different priority queues by jointly considering their delay budgets, queuing delay, and neighborhood congestion status. For simplicity, we refer to the scheduling and routing decision space for each packet as a forward decision set and each scheduling decision as a forward decision. To this end, we formulate an optimization problem that maximizes the overall network utility under the constraint of the e2e delay requirement of each packet. We solve the problem in a fully distributed manner where each router determines a suitable forward decision for each packet. The main contributions of this work are three-fold:

- We achieve the e2e delay-guaranteed service using a cross-layer approach, where the link layer queue management and the network layer distributed routing work together to guarantee e2e delay at the packet level.
- We propose a novel scheduling and routing algorithm to improve network utility by solving a renewal optimization problem at each node. The proposed algorithm is run by each router which selects the optimal forward decision for each packet by considering the delay requirement, queue status of the current node, and the congestion status of the downstream nodes. The proposed algorithm maximizes the reward per unit time cost ratio using an iterative approach to improve the overall network utility.
- To verify the performance of the proposed solution, we build a prototype in the NS-3 simulator. Extensive simulations are conducted to evaluate the proposed DSR solution with comparisons to the state-of-the-art and benchmarks. The results show that DSROpt is capable of providing DiffServ, guaranteeing e2e delay, reducing packet loss caused by congestion, and achieving high network utility.

The rest of the paper is organized as follows: Section 2 introduces the related work. Section 3 presents the system model and problem formulation. The design details and working mechanisms of DSR are explained in Section 4. In Section 5, the simulation settings and experimental results are explained and analyzed. Finally, conclusions and future research issues are discussed in Section 6.

2. Related works

2.1. Delay-guaranteed network architecture

In [23], a latency-based forwarding (LBF) solution was proposed to achieve high-precision latency objectives. LBF focused on a link layer forwarding strategy where each node takes different actions on receiving the packet. LBF consists of two major parts, i.e., latency budget determination and QoS Action. The actions taken at each node to guarantee delay depend on the packet's remaining latency budget, destination, and the latency encountered. The Push-In First Out (PIFO) queue was adopted to insert the packet to a specific position that satisfies the latency budget's lower bound. However, the LBF did not involve routing and requires prior knowledge of the routing path for each packet, which may cause delay guarantee failure due to network dynamics or router failure. In addition, the PIFO enqueue strategy applied in LBF may cause strong decision interference among successive packets and result in longer delay than expected.

An adaptive routing protocol was proposed in [24]. The solution assumes that the remaining delay budget for each arriving flow is known to the network. The delay upper bound and the typical delay of each link are exchanged in the network to generate a lookup table indicating the guaranteed delay bound to reach the destinations. In the run-time phase, the node selects the path with the smallest typical delay and checks whether the delay budget of the path is within the remaining delay budget. However, the typical delay is highly dynamic. Therefore, it either requires a high overhead to collect this information, leading to high oscillations of path selection and congestion; or the typical delay used in the algorithm is inaccurate leading to poor path selection. Moreover, the maximum delay of each hop is very high, so the algorithm may frequently fail to find a suitable delay-guaranteed path even if such a path exists.

2.2. Delay-based scheduling and routing protocol

The backpressure (BP) routing algorithm has been widely adopted to achieve throughput optimality by stabilizing the network [20,21]. The original BP algorithm used the backlog gradient to formulate the throughput problem, which, however, suffers from severe queuing delay. To address this problem, many extensions of the BP algorithm have been developed in terms of different congestion gradient metrics. In [22], a sojourn time-based BP algorithm (STBP) was proposed. The accumulated sojourn time of buffering packets describes the congestion status of each queue instead of queue length. The sojourn time backlog measurement has the advantages of faster increment, reducing random walk packet delay and last packet delay. More recently, a routing algorithm that takes the advantages of both BP and max-weight scheduling algorithms named MW+BP was proposed to achieve

Table 1
Notations and definitions.

Symbol	Definitions
$\mathcal{N}$	Set of nodes
$\mathcal{K}$	Set of priority queues
$\mathcal{H}$	Set of forward decision
$\mathcal{P}$	Set of profits achieved by each queue
T_{budget}	End-to-end delay requirement (delay budget)
R	Link rate
T_{WRR}	Total service time of each round-robin period
T_k	Delay upper bound of queue k
$\hat{\mathcal{H}}$	Set of simplified forward decision
$h^k_{i,j}$	Forward decision for each packet, $i,j \in \mathcal{N},\ k \in \mathcal{K}$
$Q^k_{i,j}[t]$	Queue length of queue k at port j of node i at time slot t
B_k	Buffer size of queue k
w_k	Portion of service time assigned to queue k
GI	Delay upper bound of the highest priority queue
Cost$(\cdot)$	Routing cost (delay) of a path
$g_d(\cdot)$	Network utility function for delay-sensitive packet
$g_n(\cdot)$	Network utility function for non-delay sensitive packet
T^w_{budget}	Residual delay budget
T_{hop}	Per-hop delay budget
α	Control parameter for queue management
γ	Congestion index
$G^k_{i,j}[t]$	Output gain of $h^k_{i,j}$ at time slot t
$T^k_{i,j}[t]$	Time cost of $h^k_{i,j}$ at time slot t
θ	Reward per unit time cost ratio
$D(\cdot)$	Complete cost-reward pair set
L	Delay laxity
P_k	Profit achieved using queue k

overload balancing [25] in a single-hop network with bounded buffers, where the max-weight part aims to serve longer queues while the BP part balances the load between ingress and egress buffers. However, the existing approaches are insufficient to achieve delay-guaranteed service at the packet level. First, the diverse delay requirements of different packets of the same flow are not considered. Second, the temporal correlation between scheduling decisions is not fully explored. Last, considering the queue delay of the current node only is insufficient because the packet may be dropped due to congestion in the next hop.

3. System model and problem formulation

We first give the design details of the proposed DSR network architecture. Then, we present the design objective and problem formulation. Table 1 summarizes the notations and definitions frequently used in this paper.

3.1. Network architecture

3.1.1. Router structure

The per-hop delay in a network is mainly composed of queuing delay, processing delay, transmission delay, and propagation delay. In this paper, we focus on the queuing delay as it is the major cause of the e2e delay variation of packet delivery. While the instantaneous queuing delay varies quickly due to high uncertainties in packet arrival rate, we adopt the queue delay upper bound for route calculations. Each router in the network maintains the delay upper bound of each queue by fixing the buffer size and service rate. In addition, it exchanges its local queue information with neighbors to advertise congestion information. An overview of the router structure is shown in Fig. 1.

Each port of the router is deployed with a set of priority queues to provide DiffServ for DS and NDS packets. The queues can be classified into two categories in terms of service types, namely delay-guaranteed (DG) queues, and best-effort (BE) queues. DG queues provide delay-guaranteed services for the DS packets while BE queues only serve NDS packets without any delay guarantee. Using multiple priority queues for DS packets helps to provide finer granularity of delay-guaranteed services and a rich forward decision set. To avoid always starving

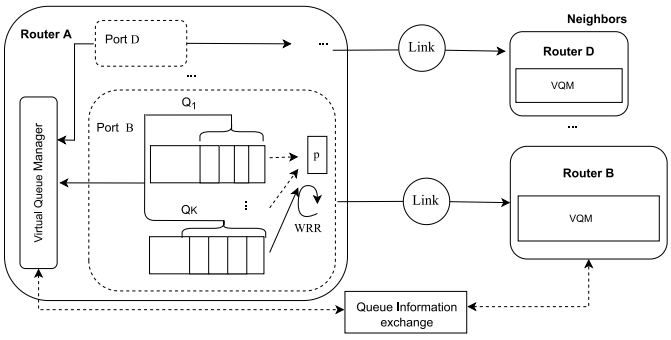


Fig. 1. Router structure overview.

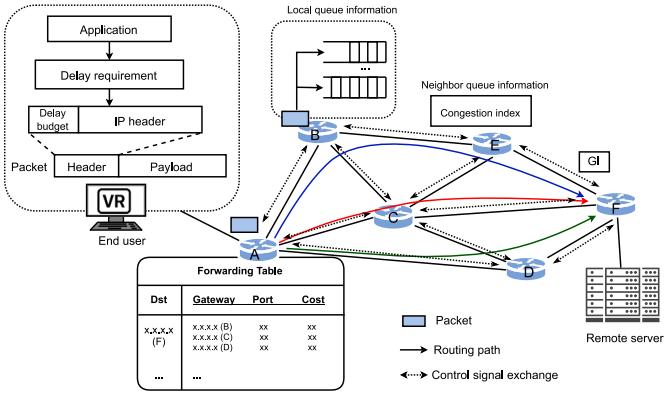


Fig. 2. System model overview.

low-priority queues, the weighted-round-robin (WRR) scheduling approach [26] is adopted where each queue is assigned with a certain portion of service time, i.e., weight, for packet transmission. Thus, the per-hop delay of each DG queue is upper bounded by fixing the buffer size and weight of each queue.[1] Note that in DSR, we assume a shallow buffer size for each DG queue to guarantee a tight delay upper bound.[2] Since the BE queue does not guarantee a delay upper bound, its buffer size can be sufficiently large to store more NDS packets.

However, bounding the per-hop delay alone is insufficient to guarantee the e2e delay performance because packets may be lost due to network congestion [27]. In this paper, we address network congestion in the network layer via neighbor routers information exchange and re-routing. A virtual queue manager (VQM) is designed for collecting and exchanging queue information with neighbors as shown in Fig. 1. In general, VQM in each router collects the local queue information, i.e., the queue length of each priority queue, and exchanges it with neighbors. The queue information essentially reflects the congestion status of each router. In this case, routers are aware of network congestion and are able to adjust forward decisions for each packet in advance before congestion becomes severe.

3.1.2. System model

The system model of DSR network architecture is shown in Fig. 2 following the design principle proposed in [28]. We assume every DS packet carries a delay budget in its packet header before being injected

[1] For example, for a router with the link rate of R Mbps requiring a queue delay upper bound of T_k ms in queue k, the corresponding buffer size should be at most $B_k = 0.125 w_k R T_k$ KB, where w_k is the weight allocated to queue k.

[2] We assume a light weight of DS traffics in the network and the tradeoff between buffer size and packet drop is beyond the scope of this paper.

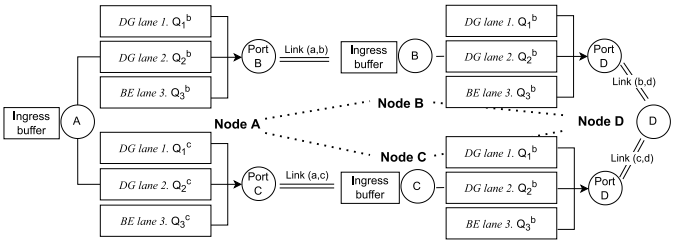


Fig. 3. DSR-enabled network with finite buffers.

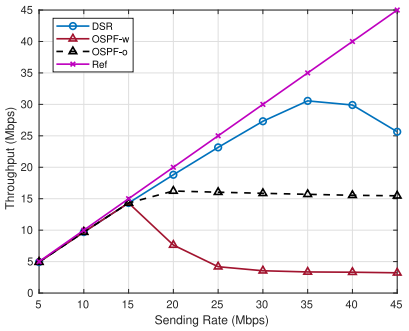


Fig. 4. Numerical analysis of DSR throughput improvements.

into the network. The delay budget is used by routers to make routing and scheduling decisions to guarantee the e2e delay requirements. In the meantime, routers monitor their local queue information while exchanging two types of information with each other, i.e.,

- Link-state information including delay upper bound of the highest priority queue (denoted by GI).[3] GI is broadcast in the entire network similar to the link state information used in the open-shortest-path-first (OSPF) routing protocol [29].
- Neighborhood queue information (NI), i.e., the local queue information shared by neighboring routers. NI is useful for route selection and congestion avoidance. Note that NI is exchanged more frequently at the level of a few milliseconds.[4]

Routers in the DSR network generate forwarding tables based on GI. It is worth noting that unlike single-path link state routing protocols such as OSPF, DSR-enabled routers adopt multipath exploration to reach the destination by computing the shortest paths rooted from their neighbors. For example, for a target pair from A to F as shown in Fig. 2, three shortest paths can be obtained at router A by running the shortest path algorithm from router B, C, and D, respectively, based on the link-state information.

In our routing algorithm, multipath exploration provides larger routing decision space at the cost of increment in time and space complexity. In a network consists of n nodes and an average number of k neighbor nodes. The time complexity of DSR increases to $O(kn \log n)$, while the time complexity of a single-path routing protocol is $O(n \log n)$. The space complexity for maintaining a forwarding table at each router increases from n (for single-path routing protocol) to kn.

We numerically analyze the gain achieved by DSR using a simple 2×2 grid topology as shown in Fig. 3, where the link rate is fixed at 20 Mbps. A source application deployed in node A sends 1000 DS packets to the destination in node D with a constant sending rate. We increase the sending rate by 5 Mbps from 5 Mbps at each simulation. We compare DSR with OSPF and show their throughput performance variations under different sending rates. As shown in Fig. 4, the blue and red curves show the goodput performance of DSR and OSPF, respectively, considering packet delay requirements (i.e., packets exceeding delay budget are ignored. For simplicity, we refer to it as goodput.). The black dashed curve indicates the total throughput achieved by OSPF without considering delay requirements, and the magenta curve indicates the sending rate of the application.

As shown in Fig. 4, both DSR and OSPF achieve high throughput when the sending rate is lower than the link capacity (i.e., 20 Mbps). Their performances diverge from 15 Mbps when the sending rate approaches the link capacity. OSPF in particular only achieves a total throughput of around 15 Mbps and a low goodput of less than 5 Mbps when the sending rate exceeds the link capacity. On the other hand, DSR maintains an increasingly high goodput performance and only starts to decrease when the sending rate exceeds 40 Mbps (2 times the link capacity). The performance gain achieved by DSR is because both paths ($A \to B \to D$ and $A \to C \to D$) are used for packet transmission. Specifically, the scheduler at node A selects a different path if the original path is congested. However, when the sending rate approaches twice the link capacity, both paths become congested and packets will be dropped directly, causing goodput performance degradation.

In summary, DSR adopts priority queues to provide DiffServ while using per-hop delay upper bound maintained by each router to explore multiple routing paths with various delay guarantees. In this context, a rich set of forward decisions including routing and queuing decisions at each hop can be obtained, which allows huge scheduling space to accommodate bursty traffic via load balancing. Therefore, an efficient routing and scheduling algorithm to determine the optimal forward decisions is crucial in the proposed DSR network.

3.2. Network model

In this paper, we consider a network that is modeled as a graph $\mathcal{G} = \langle \mathcal{N}, \mathcal{L} \rangle$, where $\mathcal{N}$ is the set of routers (we use node and router exchangeable in the rest of the paper) and $\mathcal{L}$ is the set of links. Let N and L be the number of nodes and links in the network, respectively. Denote R as the link rate and the link rate between two neighboring nodes i and j is denoted by $R_{i,j}$. Denote $\mathcal{N}_i$ as the set of neighbors of node i with the size N_i. For simplicity, we assume each node is connected with its neighbor node via a specific port and the port ID is the same as the neighbor ID. For example, in Fig. 3, node A is connected to node B via port B at node A.

Let $\mathcal{K}$ be the set of priority queues deployed at each port and has the size of K. The buffer size of each queue is denoted by B_k, $k \in \mathcal{K}$ which guarantees a delay upper bound denoted by T_k, and $T_1 < T_2 < \cdots < T_K$. In addition, we assume each DS packet carries a tuple of information denoted by $\mathcal{C} = \{T_{budget}, T_{start}\}$ in its header, where T_{budget} is the delay budget and T_{start} is the set-off time at the source node.[5] The routers in DSR network select a route and a corresponding queue for each packet. We couple the route and queue selection and refer to it as a forward decision. The forward decision for each DS packet is determined based on $\mathcal{C}$ at the ingress buffer and an NDS packet will be directly injected into the BE queue. Let $\mathcal{H}_i$ be the complete forward decision set at node i, then $\mathcal{H}_i = \{\mathcal{H}_{i,1}, \ldots, \mathcal{H}_{i,j}, \ldots, \mathcal{H}_{i,N_i}\}$, where $\mathcal{H}_{i,j}$ is the forward decision set at port j of node i. The size of $\mathcal{H}_i$ is denoted by H_i and thus, there are at most $(N_i - 1) \times K$ forwarding decisions at each node excluding the ingress port. Finally, let $h^k_{i,j} \in \mathcal{H}_i$ be the forward decision for each packet in node i, which indicates that the packet should be enqueued to queue k at port j.

[3] The highest priority queue is chosen because it is the minimum per-hop delay that can be guaranteed by each router and it allows a finer granularity for path selection, especially for urgent packets with small delay budgets.

[4] For easier implementation in practice, NI of each router can be a few bits piggybacked in the frame header. In this context, the additional bandwidth cost of NI exchange is only a few Kbps.

[5] Here we assume all routers are synchronized using time-synchronization protocols, e.g., the IEEE 802.1AS standard [30].

3.3. Weighted-round-robin scheduling model

The WRR scheduling strategy is adopted to guarantee the per-hop delay upper bound of each DG queue and avoid low-priority queue starvation. Let T_{WRR} be the total service time of each round-robin period. Each queue is assigned with a portion of the service time denoted by w_k, where $w_1 > \cdots > w_K$ and $\sum_{k=1}^{K} w_k = 1$. The delay upper bound guaranteed by each queue can be expressed by $T_k = \frac{B_k}{w_k R}$, $k \in \{1, 2, \ldots, K\}$. We further assume that the WRR scheduler at each port switches to the next queue to serve if the current queue is empty so that no time slot is wasted.

3.4. Queuing model

We consider a discrete-time system where packet arrivals and transmissions occur at the beginning of each normalized time slot. Arriving packets are backlogged at the corresponding queues before transmission. Let $\lambda_{i,j}^{k}[t]$ be the number of packets injected to queue k at port j of node i within time slot t. The number of packets dequeued from queue k to node j is denoted by $\mu_{ij}^{k}[t]$, $j \in \mathcal{N}_i$, where $\mu_{ij}^{k}[t] = 1$ if the packet is successfully transmitted and $\mu_{ij}^{k}[t] = 0$, otherwise. Denote $Q_{i,j}^{k}$ as the queue length of queue k at port j of node i. Then the queue dynamics of queue k can be written as

$$Q_{i,j}^{k}[t+1] = \max\left\{ Q_{i,j}^{k}[t] - \mu_{i,j}^{k}[t], 0 \right\} + \lambda_{i,j}^{k}[t]. \tag{1}$$

The average queue length of all DG queues at port j after one WRR scheduling period is given by

$$\bar{Q}_{i,j} = \frac{1}{K-1} \sum_{k=1}^{K-1} \sum_{\tau=1}^{T_{\mathrm{WRR}}} Q_{i,j}^{k}[\tau], \tag{2}$$

and similarly, the average queue length of all DG queues of node i is written as,

$$\bar{Q}_{i} = \frac{1}{N_i} \sum_{j=1}^{N_i} \bar{Q}_{i,j}, \tag{3}$$

which is useful for reflecting the congestion status of node i and generating NI shared with neighbor nodes.

3.5. Problem formulation

We use a simple 4-nodes network topology to explain our problem as shown in Fig. 3, where all packets at the ingress buffer of node A have the same destination to node D. Each port has three queues including two DG queues and one BE queue. There are two paths to reach node D from node A, i.e., $r_1 := \{A \to B \to D\}$ and $r_2 := \{A \to C \to D\}$, and we assume that the costs of both paths satisfy the delay requirement of the application, i.e., $\max\{\mathrm{Cost}(r1), \mathrm{Cost}(r2)\} \le T_b$, where $\mathrm{Cost}(r) = \sum_{h \in r} \mathrm{GI}_h$ denote the routing cost (i.e. e2e delay) and GI_h is the minimum delay upper bound at each hop h along path r. In this context, the forward decision set at each port of node A has the size $H_{AB} = H_{AC} = 3$ and a total size $H_A = H_{AB} + H_{AC} = 6$.

However, due to queue dynamics, not all queues in $\mathcal{H}_A$ satisfy the per-hop delay budget. Moreover, the congestion status at the downstream nodes (i.e., queue lengths of node B and C) also varies over time. A congested downstream node can lead to long queuing delay and heavy packet losses. Therefore, it is crucial to consider both local queue dynamics and neighbor queue information in routing and scheduling. In addition, since the WRR scheduler is adopted, there exists a mutual impact on queue selection. For example, heavy usage of the high-priority queues results in long queue delays in the low-priority queues because more time slots are used to serve the high-priority queues in each WRR period. Similarly, a busy low-priority queue affects the queuing time of the DS packets in the high-priority queues by using up the entire service time in each period.

Based on the above analysis, we formulate the following routing and scheduling problem aiming at maximizing the overall network utility. In brief, the optimization problem focuses on finding a series of suitable forward decisions for each packet such that its e2e delay requirement is satisfied while achieving load balance among priority queues at each hop. Mathematically, the problem can be formulated as follows

$$\text{P0:} \max_{h_d, h_n \in \mathcal{H}} \sum_{t=0}^{T} g_d(P_d[t]; h_d) + g_n(P_n[t]; h_n), \tag{4a}$$

$$\text{s.t.} \quad \mathrm{Delay}(P_d[t]) \le T_{budget}, \ \forall t \in \{0, 1, \ldots, T\}, \tag{4b}$$

$$Q_{i,j}^{k}[t] \le \alpha B_k, \ \forall i, j \in \mathcal{N}, \ \forall k \in \mathcal{K}, \tag{4c}$$

where $g_d(\cdot)$ and $g_n(\cdot)$ denote the network utility function for DS and NDS packets, respectively; $P_d[t]$ and $P_n[t]$ denote the DS and NDS packets received by the destination at time slot t; h_d and h_n denote the forward decisions made at each hop along the path for DS and NDS packets, respectively; $\mathrm{Delay}(\cdot)$ measures the e2e delay of each DS packet received at the destination, which is calculated by $T_{receive} - T_{start}$; and $\alpha \in [0, 1]$ is the parameter to control the maximum queue length of each queue. In this paper, we fix $\alpha = 1$ for simplicity. Eq. (4a) aims at maximizing the overall network utility for the source–destination connection pair. Constraint (4b) ensures all DS packets received by the destination node should have e2e delay less than their delay budgets while constraint (4c) is used to avoid bufferbloat. Note that a packet will be dropped in the ingress buffer if (4b) or (4c) is not satisfied.

4. Algorithm design

4.1. Delay-guaranteed scheduling and routing protocol

The forward decision set grows with the increase of priority queues and the size of the network causing higher time complexity. However, some forward decisions are detrimental to the network performance causing longer delay and loops [31]. To address this issue, DSR adopts a filter algorithm to filter out the undesirable routes in the forward decision set and determines the optimal forward decision jointly considering local queue information and neighborhood congestion status.

4.1.1. Route filter algorithm

Let Ω_i be the complete set of routes to reach the destination from node i in the forwarding table. Upon receiving a DS packet in the ingress buffer, the node computes a threshold value, a threshold value $T_{thld} = \min\{T_{budget}^{*}, \mathrm{Cost}^{*}(r)\}$ is set as the threshold value to filter out the undesirable routes, where $T_{budget}^{*} = T_{budget} - (T_{now} - T_{start})$ is the residual delay budget of the packet and $\mathrm{Cost}^{*}(r)$ is the routing cost computed in the previous hop.[6] If Ω_i becomes empty after filtering, i.e., there is no available path that satisfies the delay requirement, the packet is dropped directly in the ingress buffer. Hence, the size of $\mathcal{H}_i$ can be reduced to $H_i = (N_i' - 1) \times K$, where N_i' is the size of Ω_i after filtering. The complete algorithm is shown in Algorithm 1 from line 3 to line 16.

In summary, the route filter algorithm ensures all the remaining routes in the forward decision set have at least one queue (i.e., the highest priority queue) that guarantees the delay requirement of the packet. It is also worth noting that the filter also helps to avoid the loop problem in dynamic routing by ensuring the routing cost of each forward decision selected at each hop is in descending order while approaching the destination.

[6] The design rationale of T_{thld} is to ensure the remaining routing paths satisfy the delay requirement, i.e., less than T_{budget}^{*}, and are closer to the destination compared to the previous hop to avoid loop problem. The major cause of loop in the DSR network is the imbalance between a large delay budget and path diversity.

4.1.2. Congestion-aware route selection algorithm

Although the filtered forward decision set $\mathcal{H}_i$ satisfies the delay requirement, packets may be dropped due to network congestion in the downstream nodes. We address this issue by enabling queue information exchange among neighboring nodes. Inspired by the idea from backpressure routing algorithms [20] where the backlog gradient is used for packet scheduling, we develop a congestion-aware route selection algorithm as shown in Algorithm 1 Part 2 from line 18 to line 38.

The algorithm works based on the design of VQM which periodically exchanges local queue information with neighboring nodes. Specifically, the VQM at each node computes the average queue length using Eq. (3) to estimate the current queuing status. An indicator function γ based on the average queue length is designed as follows

$$\gamma_j = \begin{cases} \frac{\bar{Q}_j^1}{\alpha \bar{B}_1} & \bar{Q}_j^1 < \alpha \bar{B}_1, \\ 1 & \bar{Q}_j^1 \geq \alpha \bar{B}_1, \end{cases} \quad j \in \mathcal{N}_i, \tag{5}$$

where $\bar{Q}_j^1$ and $\bar{B}_1$ refers to the average queue length and average buffer size of the all highest priority queues of the node.[7] Such design helps to deliver the congestion status of the downstream node to the upstream node. If node i receives a congestion signal $\gamma_j = 1$ from node j indicating heavy congestion, it removes the route to reach node j from Ω_i to alleviate the congestion of the downstream nodes. Otherwise, it is used as a control parameter for route evaluation.

Next, for each decision $h_{i,j}^k \in \mathcal{H}_i$, the router computes a weighted queue delay denoted by $T_{i,j}^k[t]$, where

$$T_{i,j}^k[t] = (1+\gamma_j) \cdot \frac{Q_{i,j}^k[t] \times \text{PktSize}}{w_k R} \tag{6}$$

and the per-hop delay budget T_{hop} of each output port

$$T_{hop}(j) = T_{budget}^* - \text{Cost}(r_j), \ r_j \in \Omega_i \ j \in \mathcal{N}_i. \tag{7}$$

The weighted queue delay $T_{i,j}^k[t]$ of each forward decision $h_{i,j}^k \in \mathcal{H}_i$ is compared with $T_{hop}(j)$, and $h_{i,j}^k$ will be removed from $\mathcal{H}_i$ if $T_{i,j}^k[t] > T_{hop}(j)$, i.e., the per-hop delay budget is insufficient to support the corresponding forward decision. Note that the packet is dropped directly if the forward decision set is empty, i.e., no available decision exists. Finally, a fully simplified forward decision set $\hat{\mathcal{H}}_i$, $i \in \mathcal{N}$ that guarantees the e2e delay is obtained. Next, the router runs the proposed DSROpt scheduling algorithm to determine the optimal forward decision h_{ij}^{k*}.

Algorithm 1 DSR routing protocol

1: **Input:** forward decision set $\mathcal{H}_i$ at node i
2: **Output:** Forward decision $h_{ij}^{k*} = (route, queue)$
3: Upon receiving a packet at the ingress buffer of node i
4: Read T_{budget}, and T_{start} from the packet and router
5: Compute residual delay budget:
$T_{budget}^* = T_{budget} - (T_{now} - T_{start})$
6: Set $T_{thld} = \min\{T_{budget}^*, \text{Cost}^*(r)\}$
7: (**Part 1: Route filter**)
8: **for** each *route* in Ω_i **do**
9: Let $Cost = \text{Cost}(route)$;
10: **if** $Cost > T_{thld}$ **then**
11: remove *route* from Ω_i
12: **end if**
13: **end for**
14: **if** $\Omega_i = \varnothing$ **then**
15: Drop packet
16: **end if**
17:
18: (**Part 2: Congestion-aware route selection**)
19: Obtain Ω_i
20: **for** route $r_j \in \Omega_i$ **do**
21: **if** $\gamma_j = 1$ **then**
22: Remove r_j from Ω_i
23: **end if**
24: **end for**
25: Compute $T_{hop}(j) = T_{hop}(j) = T_{budget}^* - \text{Cost}(r_j)$
26: **for** $h_{i,j}^k \in \mathcal{H}_i$ **do**
27: Compute $T_{i,j}^k[t] = (1+\gamma_j) \cdot \frac{Q_{i,j}^k[t] \cdot \text{PktSize}}{w_k \cdot R}$
28: **if** $T_{i,j}^k[t] > T_{hop}(j)$ **then**
29: Remove $h_{i,j}^k$ from $\mathcal{H}_i$
30: **end if**
31: **end for**
32: Obtain $\hat{\mathcal{H}}_i$
33: **if** $\hat{\mathcal{H}}_i = \varnothing$ **then**
34: Drop packet
35: **else**
36: Run **DSROpt**($\mathbf{H}_i$)
37: **Return:** Routing decision $h_{ij}^{k*} = (\text{port } j, \text{queue } k)$.
38: **end if**
39: Enqueue packet to queue k at port j

4.2. Network utility maximization

We aim to maximize the network utility as presented in P0. However, since the forward decisions are spatio-temporally correlated, i.e., the current decision is affected by decisions made in previous time slots and upstream nodes, and eventually affects the overall network utility. Therefore, it is difficult if not impossible to find the optimal scheduling policy at every node for every packet given the large scheduling space, stringent delay requirement, and network dynamics.

To this end, we transform the utility maximization problem and let each node computes the optimal forward decision for each packet. Specifically, at time slot t, we represent each forward decision $h_{i,j}^k \in \hat{\mathcal{H}}_i$ by a cost-reward tuple $(T_{i,j}^k[t], G_{i,j}^k[t])$, where $G_{i,j}^k[t]$ is the reward obtained for executing $h_{i,j}^k$. The infinite horizon reward per unit time cost ratio is then given by

$$\theta = \lim_{T\to\infty} \frac{\sum_{t=0}^T G_{i,j}^k[t]}{\sum_{t=0}^T T_{i,j}^k[t]}, \tag{8}$$

assuming the limit exists. In this context, the overall network utility can be improved if θ_i at each node $i \in \mathcal{N}$ is maximized. Note that to avoid the divide-by-zero issue, we set $T_{i,j}^k[t] = 1$ if the queue is empty.

In other words, we decouple the end-to-end network utility maximization problem in P0 into a series of subproblems, where the reward per unit time cost ratio at each node is maximized. Therefore, P0 can be rewritten as follows

$$\text{P1:} \quad \max_{h_{i,j}^k \in \hat{\mathcal{H}}_i, \ i\in\mathcal{N}} \quad \theta_i = \lim_{T\to\infty} \frac{\sum_{t=0}^T G_{i,j}^k[t]}{\sum_{t=0}^T T_{i,j}^k[t]}, \tag{9a}$$

$$\text{s.t.} \quad (T_{i,j}^k[t], G_{i,j}^k[t]) \in \mathcal{D}(\hat{\mathcal{H}}), \ t \in \{0, 1, \ldots, T\}, \tag{9b}$$

where $\mathcal{D}(\hat{\mathcal{H}})$ denotes the set of all possible cost-reward combinations in the forward decision set $\hat{\mathcal{H}}$. Since the forward decision set is only known when the packet's delay requirement is observed and the state of the system is renewed after executing each forward decision, we identify each subproblem as a renewal optimization problem [32,33]. However, due to the randomness of network dynamics and large space of $\mathcal{D}(\hat{\mathcal{H}})$, it is impractical to find the optimal solution to P1 [33]. To address this issue, we propose a heuristic algorithm that maximizes (9a) by learning from past forward decisions.

[7] This is because the highest priority queue guarantees the minimum delay upper bound of each hop and is preserved for urgent packets by design (explained in Algorithm 2), its queue length is sufficient to reflect the congestion status of the corresponding node.

Algorithm 2 DSROpt scheduling algorithm

Input: forward decision set $\hat{\mathcal{H}}_i$, η
Output: forward decision $h_{i,j}^k$
Initialize $\phi, \phi' = 0$, $\theta[t] \in [\theta_{\min}, \theta_{\max}]$,
for each decision $h_{ij}^k \in \hat{\mathcal{H}}_i$ at node i **do**
  Compute $T_{i,j}^k[t]$ and $G_{i,j}^k[t]$ using (6) and (11)
  Let $\phi = G_{i,j}^k[t] - \theta[t]T_{i,j}^k[t]$
  if $\phi \geq \phi'$ **then**
    $\phi' = \phi$, port $= j$, queue $= k$
  end if
end for
Update $\theta[t+1] = \left[\theta[t] + \eta(G_{i,j}^k[t] - \theta[t]T_{i,j}^k[t])\right]_{\theta_{\min}}^{\theta_{\max}}$
Return: decision $h_{i,j}^k = (\text{port } j, \text{queue } k)$

4.2.1. Delay laxity-based reward function

We first discuss the design of our reward function which will be used in our proposed scheduling algorithm. We focus on the scheduling of DS packets in this paper, as the NDS packets are directly forwarded to the BE queue. Since there are multiple priority queues for transmitting DS packets, it is unfair to use the same reward function to measure the output gain of successful transmissions of DS packets using different priority queues.[8] In this regard, we define a metric called *profit* to measure the true return of using each queue. Let $\mathcal{P} = \{P_1, P_2, \dots, P_K\}$ be the set of profit to use each queue from high priority to low priority and $P_1 < P_2 < \cdots < P_{K-1}$.[9] Furthermore, we define a delay laxity as denoted by

$$L_{ij}^k|_{h_{ij}^k} = T_{hop}(j) - T_{i,j}^k, \quad h_{ij}^k \in \hat{\mathcal{H}}_i \tag{10}$$

as the residual per-hop delay budget, which is helpful in measuring the urgency of each DS packet [34,35]. Finally, we define the reward function for successfully transmitting the DS packets of choosing decision h_{ij}^k from $\hat{\mathcal{H}}_i$ as follows

$$G_{i,j}^k|_{h_{ij}^k} = P_k \cdot L_{ij}^k = P_k \cdot (T_{hop}(j) - T_{i,j}^k), \ h_{ij}^k \in \hat{\mathcal{H}}_i. \tag{11}$$

The goal of such design is to first use the lower priority queues when available while preserving the higher priority queues for the urgent packets yet to come.

4.2.2. Renewal optimization based scheduling algorithm

Each router in the DSR network is essentially a renewal system that updates its state, i.e., queue status, after executing each forward decision. But due to network randomness and lack of packet knowledge, it is impractical to find the optimal scheduling strategy that maximizes (9a). To this end, we develop a heuristic scheduling algorithm DSROpt based on renewal optimization as shown in Algorithm 2.

Although packet arrival is unknown, the cost-reward pairs of all possible combinations are bounded. For example, the output gain for each DS packet is bounded by

$$G_{i,j}^k|_{h_{ij}^k} \in [G_{\min}, G_{\max}] = [0, P_K \cdot \max\{T_{hop}\}], \ h_{ij}^k \in \hat{\mathcal{H}}_i.$$

Similarly, the time cost $T_{i,j}^k$ is bounded by $[T_{i,j}^1, T_{i,j}^K]$, where $T_{i,j}^1$ and $T_{i,j}^K$ are the delay upper bounds maintained by the highest priority queue and the lowest priority queue, respectively.

Assume the tuple $\{T_{\min}, T_{\max}, G_{\min}, G_{\max}\}$ is fixed at each node, where

$$T_{\min} = \min\left\{T_{i,j}^k | i,j \in \mathcal{N},\ k \in \mathcal{K}\right\},$$

$$T_{\max} = \max\left\{T_{i,j}^k | i,j \in \mathcal{N},\ k \in \mathcal{K}\right\},$$

$$G_{\min} = \min\left\{G_{i,j}^k | i,j \in \mathcal{N},\ k \in \mathcal{K}\right\},$$

$$G_{\max} = \max\left\{G_{i,j}^k | i,j \in \mathcal{N},\ k \in \mathcal{K}\right\}.$$

Let

$$\theta_{\min} = \min\left\{\frac{G_{\min}}{T_{\max}}, \frac{G_{\min}}{T_{\min}}\right\}, \quad \theta_{\max} = \max\left\{\frac{G_{\max}}{T_{\max}}, \frac{G_{\max}}{T_{\min}}\right\}$$

denote the minimum and maximum reward per unit time cost ratio, respectively. Then the optimal θ^* satisfies the condition $\theta_{\min} \leq \theta^* \leq \theta_{\max}$ if exists.

Note that always choosing the decision which yields the maximum reward does not necessarily maximize θ over time given the temporal correlations between forward decisions. To solve the optimization problem in P2, an iterative approach proposed in [33] is adopted, where θ^* can be approached by constantly selecting the cost-reward pair that maximizes the reward gradient $\phi = G_{i,j}^k[t] - \theta[t]T_{i,j}^k[t]$. In other words, the router determines the optimal forward decision by solving the following problem

$$\text{P2:} \quad \max_{h_{i,j}^k \in \hat{\mathcal{H}}} \ G_{i,j}^k[t] - \theta[t]T_{i,j}^k[t], \tag{12a}$$

$$\text{s.t.} \quad (G_{i,j}^k[t], T_{i,j}^k[t]) \in \mathcal{D}(\hat{\mathcal{H}}). \tag{12b}$$

Here $\theta[t]$ is updated after each decision

$$\theta[t+1] = \left[\theta[t] + \eta[t](G_{i,j}^k[t] - \theta[t]T_{i,j}^k[t])\right]_{\theta_{\min}}^{\theta_{\max}}, \tag{13}$$

where η is the step size of each iteration and $[x]_{\theta_{\min}}^{\theta_{\max}}$ denotes the normalization of x within the range of $[\theta_{\min}, \theta_{\max}]$. The convergence of the update rule (12) is discussed in Appendix. Such an opportunistic online learning approach is effective because the historical decisions are included in the optimization objective. The complete scheduling algorithm is summarized in Algorithm 2. To be more specific, each node $i \in \mathcal{N}$ initializes the reward per time cost ratio $\theta[t]$ by randomly selecting a value from the range $[\theta_{\min}, \theta_{\max}]$ at $t = 0$. To determine the optimal forward decision for a DS packet received at time slot t, the node first computes the weighted queue delay $T_{i,j}^k[t]$ and output gain $G_{i,j}^k[t]$ for each available forward decision $h_{i,j}^k \in \hat{\mathcal{H}}_i$ of the packet. Then, it selects the forward decision that solves P2 and updates $\theta[t]$ according to (13).

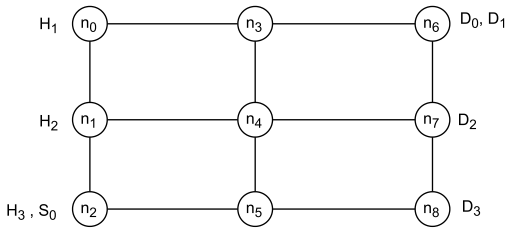


Fig. 5. 3 × 3 grid topology.

5. Performance evaluation

In the experiment, we implemented the proposed DSR network using the NS-3 network simulator [36]. Three priority queues including two DG queues and one BE queue are deployed at each port of each router. Note that the model can be easily extended with more priority queues to achieve a finer-granularity performance.

We compare the performance of DSROpt scheduling algorithm (denoted by DSR in our simulation results) with three BP-based scheduling algorithms such as the finite-buffer max-weight algorithm [21]

[8] Otherwise, the high priority queues are more likely to be selected as they provide smaller delay upper bounds.

[9] For simplicity, we set the profit of each DG queue to be inversely proportional to the weight assigned to them, i.e., $P_k = 1/w_k$.

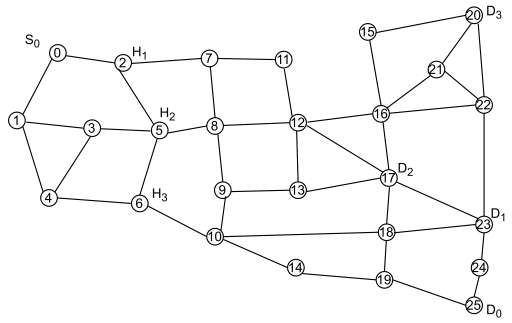


Fig. 6. Simulated U.S. backbone network.

Table 2
Network configurations.

Parameters	DG queue 1	DG queue 2	BE queue
Weight	0.5	0.3	0.2
Buffer size (packet)	6	17	100
Network type	Source	Destination	
Grid	n_2, n_0, n_1	n_6, n_7, n_8	
Mesh	n_0, n_2, n_5, n_6	$n_{17}, n_{20}, n_{23}, n_{25}$	

(denoted by MAX), sojourn-time-based BP algorithm [22] (denoted by TBP), MW+BP algorithm developed in [25] (denoted by MBP), and one greedy algorithm denoted by GRD which always selects the route/queue with the highest reward calculated by Eq. (11). All algorithms are implemented based on the proposed DSR network architecture with priority queues. In addition, we use a single-queue network that adopts OSPF routing protocol (denoted by BEF) for packet delivery as the benchmark in our experiment. Since we assume a static network topology and link cost in our experiment, OSPF provides comparable performance as a centralized routing algorithm in static networks.

5.1. Network configurations

We consider two types of network topology in the experiments, i.e., a 3×3 grid network and a 26-nodes mesh network as shown in Figs. 5 and 6, respectively. Every router in both networks has the same configurations as summarized in Table 2. The buffer size and weight assigned to each queue are $1200, 3400$, and 20 K bytes, and $0.5, 0.3$, and 0.2, respectively. The link rate of all links in both networks is fixed at 20 Mbps unless mentioned otherwise. All packets have the same size of 200 bytes. Therefore, the DG queues guarantee a per-hop delay upper bound of 2 ms and 10 ms, respectively. The propagation delay of each link is randomly selected from a $[3, 5]$ ms range.

We use the (Src, Dst) pair to indicate the source–destination connection in our experiments. In the grid network, (n_2, n_6) is used as our target pair, i.e., (S_0, D_0) as shown in Fig. 5, for analysis while three other pairs (n_0, n_6), (n_1, n_7), and (n_2, n_8) are set to simulate the background traffic in the network. In the mesh network, (n_0, n_{25}) is the target pair. Three pairs (n_2, n_{20}), (n_5, n_{17}), and (n_6, n_{23}) are used for simulating background traffics. Note that in our experiments, we only evaluate the performance of the target pairs.

5.2. Performance metrics

Several performance metrics are defined to measure the performance of each algorithm: (1) the e2e delay of each DS packet, (2) the average e2e delay of all DS packets and their standard deviation, and (3) the goodput calculated by $\frac{\text{total received bits}}{\text{total Tx time}}$. We also define the performance gain as $a_1 N_{\text{DS}} + a_2 N_{\text{NDS}}$, where a_1 and a_2 $(a_1 > a_2)$ are the utility gain for successfully receiving DS packet and NDS packet at the destination. In our experiments, we fix $a_1 = 3, a_2 = 1$. Note that all DS packets that exceed their delay budget are discarded in the network or at the receiver side, i.e., are not considered in performance evaluations.

Table 3
Experiment settings.

Impact of	Target pair	Background pair
Bursty traffic	Base rate: 5 Mbps Bursty range: 1∼15 Mbps Base rate: 1 Gbps Bursty range: 0∼2.5 Gbps	Base rate: 10 Mbps Bursty range: 1∼5 Mbps Base rate: 1 Gbps Bursty range: 0∼0.5 Gbps
Traffic density	Base rate: 10 Mbps	Base rate: 5 Mbps (DS) Increase by 1 Mbps (NDS) Increase by 2 Mbps
Delay budget	Base rate: 10 Mbps Delay budget reduce by 2 ms per epoch	(DS) Base rate: 5 Mbps Bursty range: 1∼15 Mbps (NDS) Base rate: 10 Mbps Bursty range: 1∼5 Mbps
NDS traffics	Base rate: 5 Mbps Increase 2 Mbps per epoch	(DS) Base rate: 5 Mbps (NDS) Base rate: 10 Mbps

5.3. Simulation results and analysis

We designed four experiments to evaluate our proposed delay-guaranteed solution. Specifically, we tested its delay and goodput performance under bursty traffic, different traffic volumes, various delay requirements, and co-existence with NDS traffic. Each experiment consisted of several simulation rounds with different settings, called epochs. All experiments used UDP traffic and their settings are summarized in Table 3. In addition, we select two representative algorithms, TBP and BEF, to show their standard deviations in end-to-end delay. TBP is selected because it leverages the local queuing delay to make scheduling decisions and it performs the second best in our experiments.

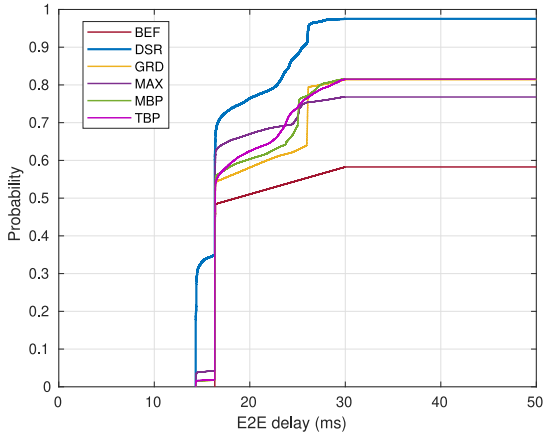
5.3.1. Impact of bursty traffic

In this experiment, we verify the performance of DSROpt scheduling algorithm under bursty traffic using the 3×3 grid topology. Both target and background traffic pairs transmitted DS packets with bursty sending rates. The target source node transmitted $50\,000$ packets at a base rate of 5 Mbps and an additional bursty rate of $1 \sim 15$ Mbps. The background traffic pairs had a base rate of 10 Mbps and an additional bursty rate of $1 \sim 5$ Mbps. All DS packets required a 30 ms end-to-end delay performance.

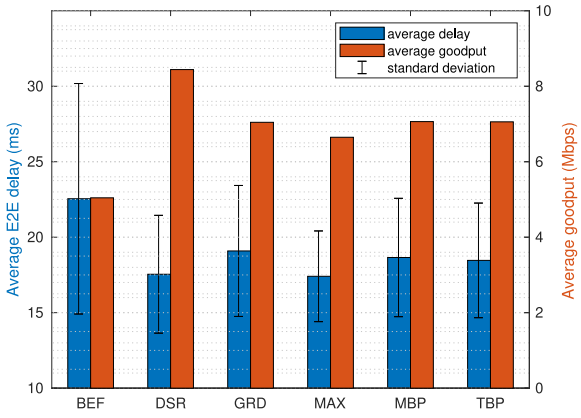
The simulation results are presented in Fig. 7, where our algorithm achieves the best delay and goodput performance compared with other algorithms. As shown in Fig. 7(a), DSROpt guarantees the most DS packets delivery up to 98% within 30 ms. GRD, MBP, and TBP show similar performances guaranteeing around 82% of the DS packets while MAX achieves 76% of e2e delay guarantee. BEF only guarantees 59% of the packets due to severe packet drop caused by congestion and bufferbloat.

Without loss of generality, we also compare the average delay and goodput performances of all algorithms as shown in Fig. 7(b). As expected, DSROpt shows the highest goodput up to 8.2 Mbps while GRD, MBP, and TBP give similar goodput around 7.2 Mbps. MAX shows a worse goodput performance of 6.7 Mbps. BEF gives the worst goodput performance of 5.2 Mbps. As for the average e2e delay, the BP-based scheduling algorithms achieve similar delay performances, among which MAX shows slightly better performance as it achieves load balance using the queue-length gradient. Overall, the proposed priority queue-based network outperforms the single-queue network because the upstream nodes are able to change routes for the DS packets. DSROpt outperforms other algorithms thanks to its congestion-aware design which helps to avoid congested paths to improve goodput.

In addition, we verify the scalability of our network architecture and DSROpt scheduling algorithm using a high link rate network setting. Each link has a rate of 2 Gbps. The target source application transmits DS data packets with a bursty sending rate ranging from $1.0 \sim 3.5$ Gbps

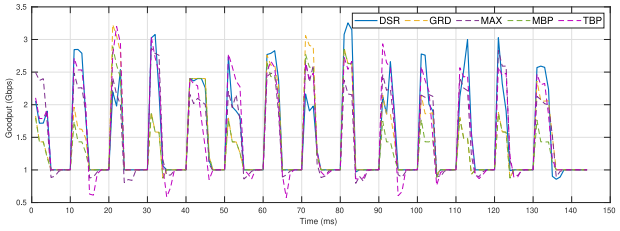


(a) CDF of per-packet e2e delay.

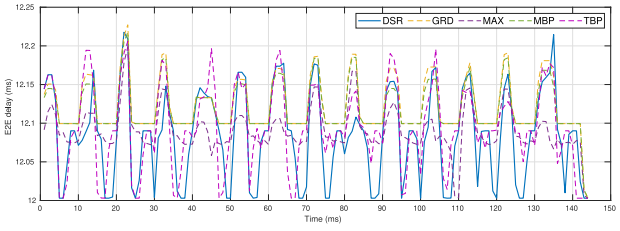


(b) Average QoS performances.

Fig. 7. Comparisons under bursty traffic.

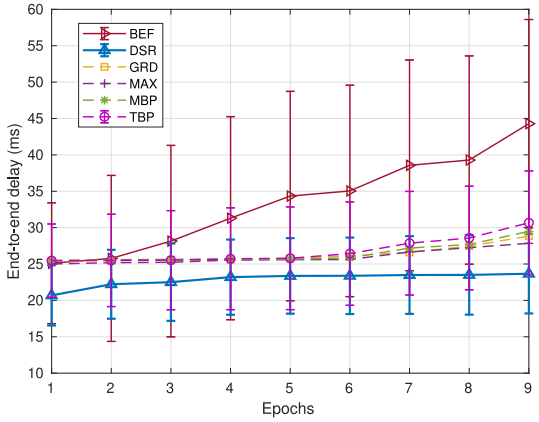


(a) Goodput performance.

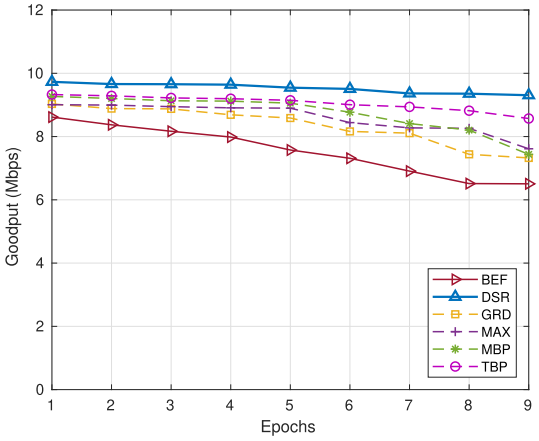


(b) E2E delay performance.

Fig. 8. QoS performances with high link rate.



(a) Average delay performance.



(b) Goodput performance.

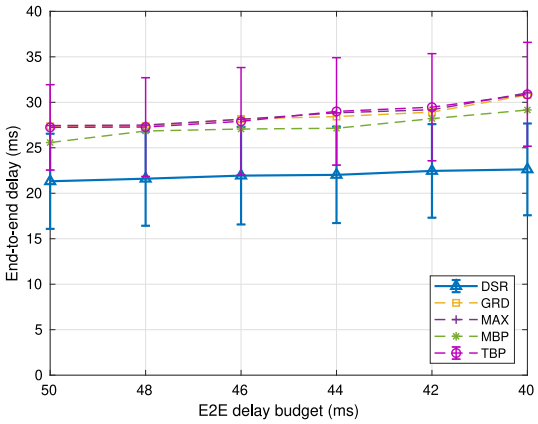
Fig. 9. QoS performance comparison using mesh network.

and the background traffics have a sending rate of 1 ∼ 1.5 Gbps. The goodput and delay performances of DSROpt and other algorithms are shown in Fig. 8. The results show that DSROpt has good scalability and is able to maintain high goodput and low e2e delay in high data rate network settings.

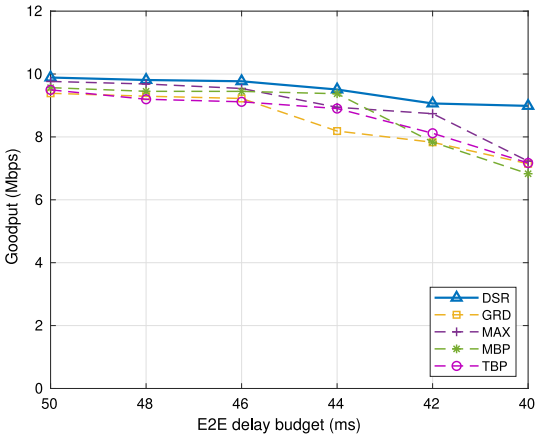
5.3.2. *Impact of traffic density*

In this experiment, we use the mesh network topology to verify the performance of DSROpt under various traffic densities. In the target pair, the source node transmits 10 000 DS packets with a delay budget of 45 ms to the receiver at a fixed sending rate of 10 Mbps in each epoch. As for the background pairs, both DS and NDS applications are deployed. The sending rate of the DS and NDS application increases by 1 Mbps and 2 Mbps at each epoch from 5 Mbps, respectively.

Fig. 9(a) shows the delay performance. The e2e delay of BEF increases quickly because all traffic uses the same queue for packet transmission and OSPF always chooses the shortest path for packets routing, which can easily result in network congestion when the packet arrival rate exceeds the link capacity, leading to a long queue delay and bufferbloat. This also accounts for the large end-to-end delay variance in each epoch. BP-based algorithms (GRD, MAX, TBP, and MBP) show similar performances, degrading from the 5th epoch when the aggregated sending rate exceeds link bandwidth. DSROpt performs best throughout the experiment, reducing end-to-end delay by up to 22.5% compared to BP-based scheduling algorithms.

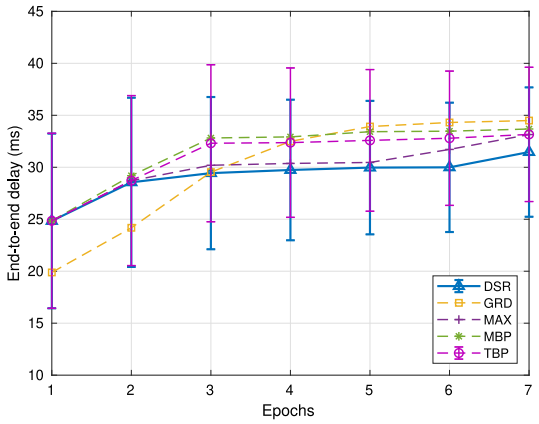


(a) Average e2e delay.

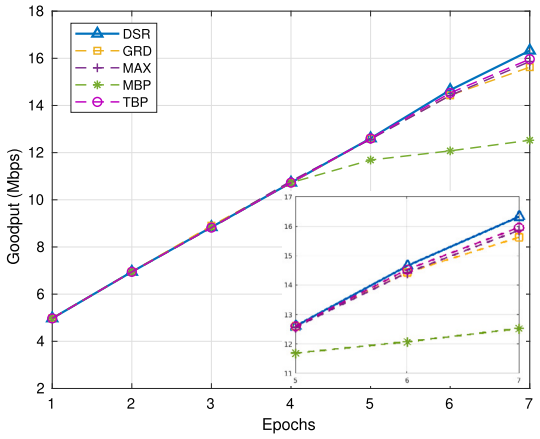


(b) Goodput.

Fig. 10. Comparisons under different delay budgets.



(a) DS application: Average delay.



(b) DS application: Goodput.

Fig. 11. QoS performance comparisons on DiffServ property for DS applications.

Fig. 9(b) shows the goodput performance. BEF performs the worst due to heavy congestion and packet drops as traffic density increases. MAX, GRD, and MBP show similar patterns as the sending rate grows, with GRD experiencing more rapid performance degradation due to congestion. TBP has better goodput performance with slower degradation because it considers each DS packet's sojourn time and schedules urgent packets to higher priority queues, reducing delay outage drops. On the other hand, DSROpt maintains the highest goodput throughout the experiment and is only reduced by 0.8 Mbps until the last epoch. Overall, DSROpt achieves a goodput increase of up to 26.4% over BP-based scheduling algorithms.

5.3.3. Impact of delay requirements

Furthermore, We show the impact of the delay requirements using the mesh network topology. Similar to the previous settings, the target pair transmitted 20 000 DS packets at a fixed rate of 10 Mbps while the background pairs transmitted 10 000 mixed DS and NDS packets at a varying rate of $[10, 30]$ Mbps in each epoch. The delay budget for the target application decreased from 50 ms to 40.[10] The simulation results are shown in Fig. 10.

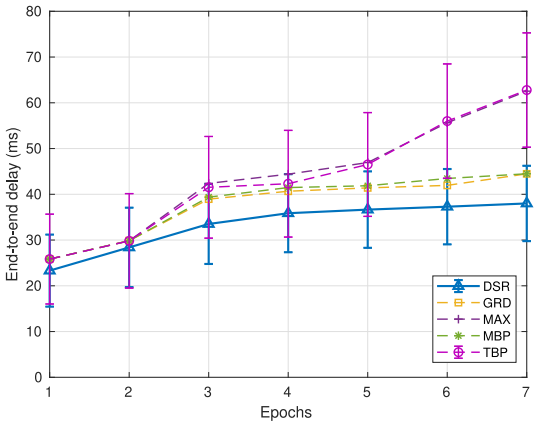
Overall, GRD, MAX, TBP, and MBP show similar delay and goodput performance. All experience rapid goodput degradation when the delay budget is less than 44 ms, approaching the target pair's minimum delay cost to reach the destination. DSROpt maintains a good performance, reducing delay by up to 28.1% and improving goodput by up to 47.5% compared to other BP-based scheduling algorithms. It maintains a goodput of 9 Mbps when the delay budget is 40 ms while BP-based algorithms degrade to less than 7 Mbps. This is because DSROpt considers each packet's delay requirement and temporal correlations among forward decisions for scheduling. On the other hand, BP algorithms aim to minimize per-hop time cost in scheduling packets, frequently occupying high-priority queues and leading to goodput degradation when the delay budget is small.

5.3.4. Differentiated service

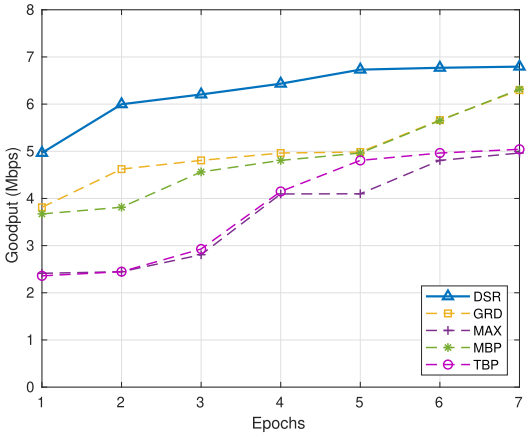
The DSR network architecture can provide DiffServ to both DS and NDS traffic simultaneously and achieves the highest performance gain using DSROpt. We verify this property using the grid topology shown in Fig. 5. Two applications, one DS with a delay requirement of 35 ms and one NDS, are deployed at the target and background pairs. In each epoch, we increase both applications' sending rate by 2 Mbps from 5 Mbps for the target pair. The background pairs' sending rates are fixed at 5 Mbps for the DS application and 10 Mbps for the NDS application. We evaluate both applications' QoS metrics for the target pair and only compare DSROpt with the BP-based scheduling algorithms.

The QoS performances of DS applications are presented in Fig. 11. As shown in Fig. 11(a), the average e2e delay achieved by all algorithms increases in each epoch due to the longer queuing delay. GRD

[10] Note that the minimum e2e cost that can be guaranteed for the target pair is 40 ms.



(a) NDS application: Average delay .



(b) NDS application: Goodput.

Fig. 12. QoS performance comparisons on DiffServ property for NDS applications.

achieves the least delay when the network is less congested but quickly grows to the highest as the sending rate increases due to its greedy exploration strategy causing congestion and long queuing delays. DSROpt performs best when the network becomes more congested because it considers both downstream and current node congestion when making forward decisions. As for the goodput performance shown in Fig. 11(b), all algorithms have similar performances when the network is underloaded. BP-based algorithms gradually diverge when network traffic exceeds link capacity from the 4th epoch. MBP degrades quickly from the 5th epoch when the target pair's aggregated sending rate exceeds 20 Mbps and the rest saturate from the 6th epoch. DSROpt shows an approximately linear goodput increase, reaching up to 16.2 Mbps when the DS packet sending rate is 17 Mbps, with TBP achieving the second-best goodput of 16 Mbps.

In terms of NDS applications, the QoS performance presents a different pattern as shown in Fig. 12. Two groups, GRD and MBP, and MAX and TBP, show similar performances in delay and goodput, respectively. All algorithms show low goodput and increasingly high end-to-end delay as the sending rate grows due to a single queue and shortest path being used for NDS packet transmission. The MAX and TBP group, in particular, sees a sharp delay increase when the sending rate increases because they heavily exploit optimal queues/routes for DS packet transmission while starving low-priority queues transmitting NDS packets. On the other hand, DSROpt achieves the best performances in both e2e delay and goodput compared to the BP-based scheduling algorithms because it considers the mutual impact of

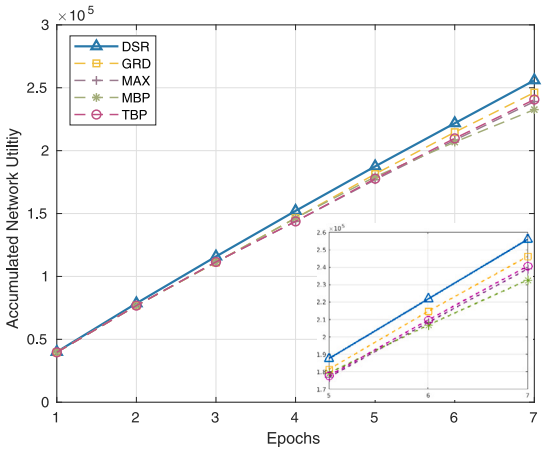


Fig. 13. Performance gain achieved by different algorithms.

forward decisions by maximizing the long-term reward per unit time cost ratio at each node, maintaining good performance for both DS and NDS applications.

We also compared the network performance gain achieved by different scheduling algorithms as shown in Fig. 13. DSROpt achieves the best performance, 10.2% higher than MBP. Although the pattern varies by tuning the utility gain (i.e., a_1 and a_2) of DS and NDS packets, DSROpt always performs best. We can conclude that DSR can provide DiffServ to different applications with different delay requirements.

6. Conclusion

In this paper, we propose DSR, a novel distributed, congestion-aware, and delay-guaranteed solution for time-sensitive applications. DSR adopts priority queues with fixed buffer sizes to provide DiffServ and per-hop delay upper bounds and explores path diversity in the network to achieve multiplex gain. To reduce the time complexity and avoid the loop problem, a filter algorithm is developed to filter out unsuitable routes. In addition, DSR addresses the congestion problem in the network by introducing a virtual queue manager that exchanges the local queue information of each node to reflect the congestion status of the neighborhood and accommodate route selection. Given the rich set of forward decisions at each hop, we aim to maximize the overall network utility by finding the optimal routing and scheduling strategies under the e2e delay constraint. However, due to the randomness of network dynamics and the large decision space, it is impractical to find the optimal forward decision for every packet at each hop. In this regard, we solve the network utility maximization problem using a heuristic approach by decoupling it into a set of subproblems that maximizes the reward per unit time cost ratio at each hop, where each subproblem is essentially a renewal optimization. To this end, we develop a novel scheduling algorithm named DSROpt that solves the renewal optimization problem and determines the optimal forward decision for each packet jointly considering its delay requirement, downstream congestion status, and local queue information.

To verify the performance of DSR and DSROpt, we implement a prototype on NS-3 and compare it with the state-of-the-art via simulations. The results show that DSROpt guarantees the e2e delay requirements while achieving the highest goodput and performance gain in various experiments. In our current simulation, routers exchange the neighbor queue information continuously and instantaneously. Such simplification may affect the performance of DSR in practice, since the queue length varies quickly due to bursty traffic. The delayed congestion status feedback can result in poor forwarding decisions which requires further research. In addition, it is worth further investigating how to leverage more control functions such as congestion control and active queue management to further improve network efficiency.

CRediT authorship contribution statement

Xiangyu Ren: Conceptualization, Methodology, Software, Visualization, Writing – original draft. **Lin Cai:** Methodology, Writing – review & editing, Supervision. **Pu Yang:** Software. **Jiequ Ji:** Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Lin Cai reports financial support was provided by Natural Sciences and Engineering Research Council of Canada, and Compute Canada.

Data availability

Data will be made available on request.

Appendix

For a node in the network, let u be a constant value that satisfies

$$\frac{1}{2}\mathbb{E}[(G[t]-\theta[t]T[t])^2] \le u, \tag{14}$$

where $G[t]$ and $T[t]$ are the reward and time cost of scheduling a packet at time slot t, and $\theta[t]$ is the reward-per-time-cost ratio that is updated after each scheduling decision. u exists as $G[t]$, $T[t]$, and $\theta[t]$ are bounded by design.

Denote $z_t = \frac{1}{2}\mathbb{E}[(\theta[t]-\theta^*)^2]$ as the mean square error of $\theta[t]$ and the optimal θ^* that can be obtained, where $z_t,\ t=0,1,\dots$ satisfies [33]

$$z_{t+1} \le (1-2T_{\min}\eta[t])z_t + \eta[t]^2 u,\ \forall \eta[t]>0, \tag{15}$$

where $\eta[t]$ is the step size defined in (12). When a fixed step size, e.g., $\eta[t]=\frac{1}{2T_{\min}}$, is adopted, we have

$$z_{t+1} \le \frac{u}{4T_{\min}^2}, \tag{16}$$

i.e., z_t is upper bounded. Similarly, the convergence of z_t can be proved when a varying step size is adopted.

Suppose $z_t \le \frac{b}{t\cdot T_{\min}^2}$, then we prove z_{t+1} is also upper bounded. Let $\eta[t]=\frac{1}{(t+2)T_{\min}}$, then

$$\begin{aligned} z_{t+1} &\le (1-2T_{\min}\eta[t])z_t + \eta[t]^2 u \\ &= (\frac{t}{t+2})z_t + \frac{u}{(t+2)^2T_{\min}^2} \\ &\le (\frac{t}{t+2})\frac{u}{t\cdot T_{\min}^2} + \frac{u}{(t+2)^2T_{\min}^2} \\ &= \frac{u(t+3)}{(t+2)^2T_{\min}^2} \\ &\le \frac{u}{(t+1)\cdot T_{\min}^2} \end{aligned} \tag{17}$$

It is obvious that a faster convergence can be achieved using a varying step size.

References

[1] T. Nakamura, 5G evolution and 6G, in: 2020 International Symposium on VLSI Design, Automation and Test (VLSI-DAT), 2020, p. 1.

[2] I. Tomkos, D. Klonidis, E. Pikasis, S. Theodoridis, Toward the 6G network era: Opportunities and challenges, IT Prof. 22 (1) (2020) 34–38.

[3] W. Jiang, B. Han, M.A. Habibi, H.D. Schotten, The road towards 6G: A comprehensive survey, IEEE Open J. Commun. Soc. 2 (2021) 334–366.

[4] W. Saad, M. Bennis, M. Chen, A vision of 6G wireless systems: Applications, trends, technologies, and open research problems, IEEE Netw. 34 (3) (2019) 134–142.

[5] T. Huang, W. Yang, J. Wu, J. Ma, X. Zhang, D. Zhang, A survey on green 6G network: Architecture and technologies, IEEE Access 7 (2019) 175758–175768.

[6] C. Curino, D.E. Difallah, C. Douglas, S. Krishnan, R. Ramakrishnan, S. Rao, Reservation-based scheduling: If you're late don't blame us!, in: Proceedings of the ACM Symposium on Cloud Computing, 2014, pp. 1–14.

[7] A. Sahoo, S. Chilukuri, DGRAM: a delay guaranteed routing and MAC protocol for wireless sensor networks, IEEE Trans. Mob. Comput. 9 (10) (2010) 1407–1423.

[8] V. Addanki, L. Iannone, Moving a step forward in the quest for deterministic networks (DetNet), in: 2020 IFIP Networking Conference (Networking), IEEE, 2020, pp. 458–466.

[9] N. Cardwell, Y. Cheng, C.S. Gunn, S.H. Yeganeh, V. Jacobson, BBR: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time, Queue 14 (5) (2016) 20–53.

[10] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, J. Sun, DetNet: A backbone network for object detection, 2018, arXiv preprint arXiv:1804.06215.

[11] A. Nasrallah, A.S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, H. ElBakoury, Ultra-low latency (ULL) networks: The IEEE TSN and IETF DetNet standards and related 5G ULL research, IEEE Commun. Surv. Tutor. 21 (1) (2018) 88–145.

[12] M. Gutiérrez, A. Ademaj, W. Steiner, R. Dobrin, S. Punnekkat, Self-configuration of IEEE 802.1 TSN networks, in: 2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation, ETFA, IEEE, 2017, pp. 1–8.

[13] B. Yan, Q. Liu, J. Shen, D. Liang, B. Zhao, L. Ouyang, A survey of low-latency transmission strategies in software defined networking, Comp. Sci. Rev. 40 (2021) 100386.

[14] M.K. Boroujeny, B.L. Mark, Y. Ephraim, Stochastic traffic regulator for end-to-end network delay guarantees, in: ICC 2020-2020 IEEE International Conference on Communications, ICC, IEEE, 2020, pp. 1–6.

[15] M. Garetto, D. Towsley, Modeling, simulation and measurements of queuing delay under long-tail Internet traffic, ACM SIGMETRICS Perform. Eval. Rev. 31 (1) (2003) 47–57.

[16] B.-H. Jang, S. Son, K.-J. Park, Deadline-aware routing with probabilistic delay guarantee in cyber-physical systems, in: 2017 International Conference on Information Networking, ICOIN, IEEE, 2017, pp. 51–53.

[17] K. Agrawal, S. Baruah, Z. Guo, J. Li, S. Vaidhun, Hard-real-time routing in probabilistic graphs to minimize expected delay, in: 2020 IEEE Real-Time Systems Symposium, RTSS, IEEE, 2020, pp. 63–75.

[18] S. Baruah, Rapid routing with guaranteed delay bounds, in: 2018 IEEE Real-Time Systems Symposium, RTSS, IEEE, 2018, pp. 13–22.

[19] W. Liu, H. Zhang, H. Ding, D. Yuan, Delay and energy minimization for adaptive video streaming: A joint edge caching, computing and power allocation approach, IEEE Trans. Veh. Technol. 71 (9) (2022) 9602–9612.

[20] M.J. Neely, E. Modiano, C.-P. Li, Fairness and optimal stochastic control for heterogeneous networks, IEEE/ACM Trans. Netw. 16 (2) (2008) 396–409.

[21] L.B. Le, E. Modiano, N.B. Shroff, Optimal control of wireless networks with finite buffers, in: 2010 Proceedings IEEE INFOCOM, IEEE, 2010, pp. 1–9.

[22] L. Hai, Q. Gao, J. Wang, H. Zhuang, P. Wang, Delay-optimal back-pressure routing algorithm for multihop wireless networks, IEEE Trans. Veh. Technol. 67 (3) (2017) 2617–2630.

[23] A. Clemm, T. Eckert, High-precision latency forwarding over packet-programmable networks, in: NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium, IEEE, 2020, pp. 1–8.

[24] Adaptive real-time routing in polynomial time, in: 2019 IEEE Real-Time Systems Symposium, RTSS, IEEE, 2019, pp. 287–298.

[25] X. Wu, D. Wu, E. Modiano, Overload balancing in single-hop networks with bounded buffers, in: 2022 IFIP Networking Conference (IFIP Networking), IEEE, 2022, pp. 1–9.

[26] Y. Zhong, T.Q. Quek, X. Ge, Heterogeneous cellular networks with spatio-temporal traffic: Delay analysis and scheduling, IEEE J. Sel. Areas Commun. 35 (6) (2017) 1373–1386.

[27] N. Gvozdiev, S. Vissicchio, B. Karp, M. Handley, On low-latency-capable topologies, and their impact on the design of intra-domain routing, in: Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, 2018, pp. 88–102.

[28] L. Cai, J. Pan, W. Yang, X. Ren, X. Shen, Self-evolving and transformative (SET) protocol architecture for 6G, IEEE Wireless Commun. (2022) 1–12, http://dx.doi.org/10.1109/MWC.003.2200022.

[29] M.E.M. Dafalla, R.A. Mokhtar, R.A. Saeed, H. Alhumyani, S. Abdel-Khalek, M. Khayyat, An optimized link state routing protocol for real-time application over vehicular Ad-hoc network, Alex. Eng. J. 61 (6) (2022) 4541–4556.

[30] S.-H. Chang, H. Chen, B.-C. Cheng, Time-predictable routing algorithm for time-sensitive networking: Schedulable guarantee of time-triggered streams, Comput. Commun. 172 (2021) 183–195.

[31] D. Medhi, K. Ramasamy, Chapter 19 circuit-switching: Hierarchical and dynamic call routing, in: D. Medhi, K. Ramasamy (Eds.), Network Routing (Second Edition), second ed., in: The Morgan Kaufmann Series in Networking, Morgan Kaufmann, Boston, 2018, pp. 646–672, [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128007372000235.

[32] M.J. Neely, Dynamic optimization and learning for renewal systems, IEEE Trans. Automat. Control 58 (1) (2012) 32–46.

[33] M.J. Neely, Fast learning for renewal optimization in online task scheduling, J. Mach. Learn. Res. 22 (1) (2021).

[34] X. Ren, J. Ji, L. Cai, Delay laxity-based scheduling with double-deep Q-learning for time-critical applications, in: 2022 IEEE 30th International Conference on Network Protocols, ICNP, IEEE, 2022, pp. 1–6.
[35] J. Ji, X. Ren, L. Cai, K. Zhu, Downlink scheduler for delay guaranteed services using deep reinforcement learning, IEEE Trans. Mobile Comput. (2023) 1–14, http://dx.doi.org/10.1109/TMC.2023.3276697.
[36] G.F. Riley, T.R. Henderson, The ns-3 network simulator, in: Modeling and Tools for Network Simulation, Springer, 2010, pp. 15–34.

Xiangyu Ren (S'19) received his B.Sc. degree from the Department of Automation Engineering, University of Electronic Science and Technology of China, Chengdu, China in 2019. He is currently pursuing his Ph.D. degree in electrical engineering at the Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada. He is the recipient of the Graduate Student Fellowship Award from the University of Victoria in 2021. His research interests include network optimization, wireless communication, deterministic networks, vehicular networks, and the application of machine learning and deep learning in networks.

Lin Cai (S'00-M'06-SM'10-F'20) is a Professor with the Department of Electrical & Computer Engineering at the University of Victoria. She is an NSERC E.W.R. Steacie Memorial Fellow, an Engineering Institute of Canada (EIC) Fellow, and an IEEE Fellow. In 2020, she was elected as a Member of the Royal Society of Canada's College of New Scholars, Artists, and Scientists, and a 2020 "Star in Computer Networking and Communications" by N2Women. Her research interests span several areas in communications and networking, focusing on network protocol and architecture design supporting emerging multimedia traffic and the Internet of Things. She was a recipient of the NSERC Discovery Accelerator Supplement (DAS) Grants in 2010 and 2015, respectively. She has co-founded and chaired the IEEE Victoria Section Vehicular Technology and Communications Joint Societies Chapter. She is an elected member of the IEEE Vehicular Technology Society (VTS) Board of Governors, 2019–2024. She is the Associate Editor-in-Chief for IEEE Transactions on Vehicular Technology and has served as the Distinguished Lecturer of the IEEE VTS Society and IEEE ComSoc Society.

Pu Yang (Student Member, IEEE) received his B.Eng. degree in software engineering from the Software and Microelectronics College at Northwestern Polytechnical University in Xi'an, China, in June 2016. He continued his academic journey by obtaining an M.Eng. degree in Computer Technology from the Computer Science and Technology College, Northwestern Polytechnical University, Xi'an, China, in May 2020. He is currently working toward a Ph.D. degree in Electrical and Computing Engineering at the University of Victoria. His current research interests include network performance analysis and delay-guaranteed routing protocol design.

Jiequ Ji received the Ph. D degree in 2021 from College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China. From October 2018 to January 2020, she was a research assistant at Nanyang Technological University, Singapore, with Prof. Dusit Niyato. She was a research fellow with the Department of Electrical and Computer Engineering, University of Victoria, Canada, from 2021 to 2022. She is currently a Post-Doctoral Research Fellow with the Pillar of Information Systems Technology and Design, Singapore University of Technology and Design, Singapore. Her research interests include UAV-enabled wireless communications, wireless content caching, resource allocation in 5G and beyond, mobile edge computing, and physical layer security.