DDR: A Deadline-Driven Routing Protocol for Delay Guaranteed Service

Pu Yang, Tianfang Chang, Lin Cai
Department of Electrical & Computer Engineering, University of Victoria, Victoria, Canada
puyang@uvic.ca, tfchang@uvic.ca, cai@ece.uvic.ca

***Abstract*— Time-sensitive applications have become increasingly prevalent in modern networks, necessitating the development of Delay-Guaranteed Routing (DGR) solutions. Finding an optimal DGR solution remains a challenging task due to the NP-hard nature of the problem and the dynamic nature of network traffic. In this paper, we propose Deadline-Driven Routing (DDR), a distributed traffic-aware adaptive routing protocol that addresses the DGR problem. Inspired by online navigation techniques, DDR leverages real-time traffic conditions to optimize routing decisions and ensure on-time packet delivery. By combining network topology-based path generation with real-time traffic knowledge, each router can adjust packet forwarding directions to meet its heterogeneous latency requirements. Comprehensive simulations on real-world network topologies demonstrate that DDR can consistently provide delay-guaranteed service in different network topologies with varying traffic conditions. In addition, DDR ensures backward compatibility with legacy devices and existing routing protocols, making it a viable solution for supporting delay-guaranteed service.**

***Index Terms*—Delay guaranteed routing, Adaptive routing.**

I. Introduction

Delay-Guaranteed Routing (DGR) is a fundamental and crucial problem in computer networking. From advanced virtual reality (VR) and digital twin technologies to traditional financial transactions and online meetings, the demand for DGR keeps growing [1, 2]. Existing traffic engineering [3, 4] and software-defined networking (SDN) methods [5, 6] can improve network performance in terms of delay, jitter, and throughput. However, they cannot effectively solve the DGR problem due to limited flexibility and scalability.

To provide delay-guaranteed services, both network capacity and traffic dynamics should be considered for routing protocols. In this context, the DGR problem can be formulated as an integer programming problem, with the objective of minimizing the percentage of packets delivered that exceeds their designed budgets. However, how to develop effective routing solutions for delay-guaranteed services remains a challenging problem.

Existing solutions to integer programming problems are limited by their scalability, which is particularly challenging in large-scale dynamic network environments. To address this limitation, researchers introduce constraints to reduce the search space. For example, Yiyang et al. [7] developed an optimization framework that can account for variations in traffic demand and potential link failures by manually introducing constraints and multiple relaxations. Bogle et al. [8] proposed TEAVAR, which takes a risk management approach to traffic engineering, leveraged by pruning linear programming problems while adding artificial constraints. However, the above solutions require manual intervention by experienced experts and require lengthy iterations, which is not feasible when routers require real-time results.

To address the aforementioned limitations, we propose a distributed traffic-aware adaptive routing protocol named *Deadline-Driven Routing (DDR)*. To ensure delay-guaranteed services, each packet carries its delay limit in the network-layer packet header, making it accessible to all routers. Inspired by online navigation [9], which recommends routes and alternatives based on real-time traffic conditions, DDR optimizes routing solutions for punctual delivery by leveraging the inherent synergy between network capacity and local traffic information.

To achieve efficient path selection, the proposed DDR utilizes both the network topology information and the traffic dynamics information. In particular, DDR benefits from a local complete Forwarding Information Base (FIB), which provides a wealth of routing information. This FIB data can be used by the proposed approach to make intelligent and informed route selections. Finally, the path selection process is optimized based on real-time network conditions and packet-specific requirements.

To gain information about traffic dynamics, DDR uses a locally adaptive forwarding strategy. As a result, routers can dynamically adjust their forwarding decisions based on the immediate network status and traffic conditions in their local vicinity. Using locally adaptive forwarding, the proposed approach can efficiently and effectively handle traffic variations and congestion at a fine-grained level, ensuring timely packet delivery while efficiently utilizing network resources. This feature fits well with the needs to make fast and real-time routing decisions for the DGR problem.

From the above analysis, an accurate estimation of the neighborhood traffic state is important for routing decisions. Ideally, the router can promptly identify alternative routes in response to congestion. However, in real-world networks, traffic volume may change within tens of milliseconds, while the neighbor information delivery delay ranges from a few milliseconds to tens of milliseconds. To minimize the negative impact caused by traffic information delay, DDR adopts a route selection mechanism based on queue status estimation.

The main contributions of this work are summarized below:

- First, we present a formulation of the DGR problem in the context of an autonomous system (AS) network. The problem is formulated to address the challenges of delay guarantee for time-sensitive applications.

- Second, to effectively tackle the DGR problem, we introduce DDR, a novel distributed traffic-aware dynamic routing protocol. DDR explores a large set of feasible paths while leveraging local traffic conditions and network topology knowledge to ensure packet-level delay guarantees, and optimizes the routing decisions in real time. Through careful design, we have ensured its compatibility with legacy devices in AS networks. This feature makes DDR a practical and viable solution that can be deployed without significant disruption to the existing infrastructure.
- Third, we have developed and implemented a working prototype of the DDR protocol. Extensive evaluations have been conducted on real network topologies and traffic data extracted from production networks. The results show that DDR consistently provides delay-guaranteed services in a variety of network and traffic scenarios. This performance underscores the adaptability and reliability of DDR.

The rest of the paper is organized as follows: Sec. II provides a primer on the DGR solution, addressing its key concepts and challenges. In Sec. III, we discuss the pain points in the existing approaches to DGR. The system design is illustrated in Sec. IV. The detailed implementation of the DDR protocol is presented in Sec. V. Evaluation results, including performance comparisons and simulations, are discussed in Sec. VI. We discuss future research directions in Sec. VII, followed by conclusions Sec. VIII.

Open-source. The code of DDR is publicly available at http://tinyurl.com/yowaq7th.

II. DGR Primer

Large Internet Service Providers (ISPs) operate AS networks consisting of tens of servers. These AS networks must continuously adapt to meet evolving user demands, services, and traffic patterns, requiring optimization for efficient and reliable communications. In this context, routing protocol plays a crucial role in achieving efficient and reliable packet delivery for latency-sensitive applications. The main objective of DGR is to identify paths that can route packets within a given latency budget.

The high-level goal of DGR is to provide efficient and reliable routing mechanisms that ensure timely packet delivery for latency-sensitive applications.

First, DGR represents a challenging *combinatorial optimization* problem, with the primary objective of identifying an efficient path for packet routing within a given latency budget. In practical networks, fluctuating traffic patterns can cause variations in edge delay, making static routing protocols unable to achieve optimal performance. The decision-making process must consider both static link properties and real-time traffic conditions to handle latency requirements.

Therefore, the DGR solution requires a *cross-layer* approach that involves both network and link layers. At the network layer, the DGR solution determines appropriate paths to route packets while meeting the specified latency requirements. At the link layer, it implements priority queues to accommodate packets based on their latency budgets and prevailing network conditions. This cross-layer approach enables DGR to make well-informed and coordinated routing decisions to achieve efficient and punctual packet delivery.

Furthermore, the DGR solution in an AS network should be addressed using *distributed* methods. Instead of relying on a centralized controller to make routing decisions for the entire network, the DGR solution allows individual routers to make localized and adaptive decisions based on real-time observations of traffic conditions and queue status. This decentralized approach makes it possible to deploy in traditional networks. It also streamlines the routing process, reducing overhead and complexity, resulting in faster and more efficient decision-making. In addition, local optimization allows each router to make routing decisions independently without extensive communication with other routers or controllers, improving scalability and robustness in large networks.

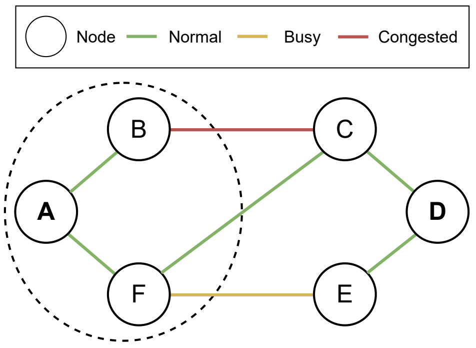


Fig. 1: An illustrative example

We illustrate a delay-guaranteed routing service example, assuming that a packet is to be delivered from A to D within the given latency limit, as shown in Fig. 1. There are four alternative routes from A to D, Route 1 (A-B-C-D), Route 2 (A-F-E-D), Route 3 (A-F-C-D), and Route 4 (A-B-C-F-E-D), but different routes may suffer from different levels of congestion. To satisfy the DGR requirement, Node A needs to know the topology of the subnet as well as the delay upper bound of each link along the paths and select an appropriate direction to route the packet considering the neighborhood link traffic status.

III. Existing Approach and Challenges

In this section, we formulate the DGR problem as an optimization problem and analyze the challenges it poses. We provide an overview of the pain points of the existing techniques in traffic engineering, resource reservation, and deterministic scheduling used to tackle this problem. These techniques, while valuable, have certain limitations in effectively solving the DGR problem, which motivates our exploration of more intelligent and systematic solutions in the subsequent sections.

A. Problem Formulation

Symbol	Description
G	A directed graph with m edges and n nodes
V	Node set
E	Edge set
τ_i	Latency of packet i
L_i	Latency upper limit of packet i
P_i	Set of edges that generates a path to deliver packet i
d_e	The delay of edge $e \in P_i$, d_e is dynamic
z_i	Binary vector edge indicator of path P_i
$I(\cdot)$	Indicator function of the on-time delivery

TABLE I: Key notations in problem formulation

The AS network topology is abstracted as a directed graph, where each router is represented as a node, and the links between them form the edges. Traffic is represented as packets between different nodes with various latency budgets based on the service requirements. The objective of DGR is to maximize the number of packets delivered on time while satisfying all constraints imposed by hardware and operational requirements. The problem can be formulated as an optimization problem under a set of constraints. The mathematical formulation is given below. Table I summarizes the key notations of the problem formulation.

Consider N packets are scheduled to be delivered on a directed graph $G = (V, E)$ with m edges and n nodes. Let P_i be the path that delivers packet i from its source to its destination. Let d_e be the delay of edge $e \in P_i$, which changes dynamically over time. The cumulative latency τ_i of packet i over the path P_i is given by:

$$\tau_i = \sum_{e \in P_i} d_e. \tag{1}$$

Therefore, the objective of the DGR is to maximize the number of packets that can be delivered within their respective delay requirements (called delay budgets in this paper), L_i, i.e.,

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{N} I(\tau_i < L_i), \\ \text{subject to:} \quad & z_i \in \{0,1\}^m, \end{aligned} \tag{2}$$

where $P_i = z_i^T E$. The decision vector z_i of length m represents the selection of edges. Using matrix-vector multiplication, path P_i can be denoted by the binary vector z_i. The inequality constraint $I(\tau_i < L_i)$ ensures that the cumulative delay of each packet τ_i is less than its delay budget L_i. The indicator function $I(\cdot)$ returns 1 if the condition inside the parentheses is true, and 0 otherwise.

This problem is an integer programming problem and is an NP-hard problem, making it challenging to find an optimal solution using traditional iterative optimization methods. Furthermore, in real networks, traffic variability causes fluctuations in edge delay, making the shortest path chosen by a static routing table potentially suboptimal. On the other hand, capturing the traffic state of the entire network is prohibitively expensive and will suffer from long delays. Therefore, how to consider both network topology and traffic dynamics to solve the DGR problem remains a challenging open issue.

B. Pain Points in Today's Approach

We formulate the DGR problem above as an integer programming problem above, but it could not be solved directly with an off-the-shelf integer programming solver. The main problem lies in the computational complexity of Integer Programming solvers, making it challenging to directly apply off-the-shelf solvers such as Gurobi [10] and MOSEK [11]. In addition, the iterative computation of the integer programming approach results in high computational resource consumption and time overhead, which becomes even more problematic under dynamic traffic conditions.

Constrained by hardware processing capabilities, the early networks were designed to provide best-effort services. Over the past few decades, an increasing number of concepts and methods for optimizing network performance have been proposed to meet various Quality of Service (QoS) requirements.

The Internet deployed simple and efficient distributed routing algorithms based on static routing tables OSPF [12], BGP [13], and IS-IS [14]. They simplify the routing problem by ignoring traffic conditions and transforming it into a shortest path problem that can be solved efficiently using algorithms such as Dijkstra's algorithm and Bellman-Ford algorithm. However, table-based routing schemes cannot effectively handle traffic dynamics in the network.

Multipath routing protocols like Equal-cost Multi-path Routing (ECMP) [15], and traffic engineering systems [16, 17] can respond to regular network dynamics. However, when congestion occurs along a path, the routers are unable to change the forwarding direction and search for a new route. This limitation can affect network reliability as well as delay and jitter, and make the Internet incapable of guaranteeing latency for time-sensitive applications.

The inadequacy of traditional Internet interior gateway routing systems led the interest in traffic engineering. Multiprotocol Label Switching (MPLS) [4] employs reactive traffic engineering, which allows routers to make forwarding decisions based on labels rather than destination addresses, resulting in more efficient packet delivery. NetPlumber [18] introduces a policy-checking tool that allows routers to evaluate network policies without delay and make adaptive routing decisions. However, current traffic engineering methods can only optimize a constant scenario but lack robustness. Some even rely on a manual intervention based on analysis of historical experience. We need more automated and intelligent solutions that adaptively adjust routing decisions based on real-time network conditions to ensure efficient and reliable packet delivery for time-sensitive applications.

More recently, SDN has applied centralized control to network management, where the separation of control and data planes allows for more flexible control of network management and configuration. B4 [5] employs hierarchical routing and centralized control based on global network state information to achieve dynamic routing optimization and load balancing. Microsoft proposed SWAN [19], a system that increases network utilization by reconfiguring traffic via the central controller, while LiveNet [20] applies the centralized approach to global routing computation and path assignment, improving performance for large-scale and low-latency streaming. In addition, Kumar et al. [6] formulated the end-to-end delay in SDN as a multi-constraint optimization problem and used heuristic algorithms to solve it. Although the centralized approach offers advantages in SDN due to the logically centralized control plane and regular topologies, it faces scalability and complexity limitations when applied to large-scale networks. Lin proposes the SET [21], which aims to meet the diverse requirements of network applications in various environments through intelligent configuration. However, it has only been proposed as an architecture and urgently awaits implementation.

Advances in hardware performance have led to the exploration and extension of the capabilities of routing devices. One of the approaches is the programming slick network functions [22], which enables dynamic and flexible network control by defining and deploying network functions on-the-fly. In addition, Kinetic et al. [23] used machine learning to optimize network routing and improve performance in dynamic environments.

In addition to routing, scheduling is critical for service quality. Priority Queues [24] can classify packets into different priorities based on their importance or time sensitivity to ensure that high-priority packets can access high-priority queues to be forwarded ahead of lower-priority packets to reduce their queuing delay. The scheduling scheme could improve the performance of time-sensitive packets, but cannot thoroughly solve the DGR problem in the network. Yang [25] proposed a delay-guaranteed routing protocol (DGRP) that uses real-time neighbor congestion information to adjust forwarding directions. However, the study only considers ideal neighbor traffic conditions obtained from a simulator and does not consider the impact of delay to obtain such traffic conditions. To fully satisfy the DGR service, we need cross-layer cooperation between the network layer and the link layer.

To address these challenges, we adopt a greedy routing strategy to ensure timely packet delivery in the presence of traffic fluctuations. In the next section, we show why the central idea of DDR - local traffic condition-based adaptive routing - can be a powerful tool to improve the on-time arrival rate.

IV. DDR Design

We propose DDR, a hybrid approach that leverages the advantages of table-based routing and a local traffic state adaptive method to solve the DGR problem. By exploiting the strengths of both methods, DDR achieves a millisecond-level latency guarantee, enabling efficient and reliable packet delivery under practical network conditions.

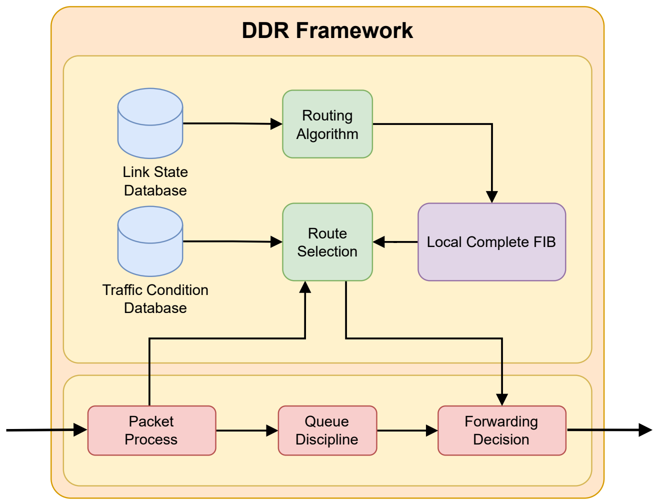


Fig. 2: Framework of DDR

The DDR framework, as depicted in Fig. 2, requires the cooperation of the link layer and network layer. A queuing discipline is used to quantify queuing delay, while a routing algorithm identifies possible forwarding directions. In addition, a route selection algorithm dynamically makes forwarding decisions based on real-time traffic conditions. This collaborative approach ensures that DDR can efficiently find high-quality routing solutions while adapting to the dynamics of network traffic.

A. Traffic Condition Database

The Traffic Condition Database (TCDB) is designed to assist the router in estimating the delay of routes under different traffic conditions, which consists of two main functions: delay quantization and delay estimation.

Delay quantization: To meet the latency requirements of delay-sensitive applications, we introduce a priority queue discipline, as shown in Fig. 3. This priority queue consists of two virtual queues: a high-priority queue for delay-sensitive packets and a low-priority queue for normal best-effort packets. The length of the high-priority queue is configured in milliseconds, representing the amount of time it takes to drain the queue. By measuring the traffic condition reflected in the length of the high-priority queue, we can assign a congestion state to the queue. The local router periodically broadcasts this congestion state to neighboring nodes. Subsequently, neighboring routers can estimate the queueing delay and intelligently choose an appropriate forwarding direction for packets. This approach enables routers to make informed and adaptive routing decisions based on real-time traffic conditions.

Delay estimation: In real-world networks, the delays for the control packet which carries the traffic state to arrive at neighboring routers can cause discrepancies between the observed traffic conditions and the actual conditions at the time of arrival. To address this issue, the TCDB implements a latency estimation process that allows routers to estimate the current traffic conditions of their neighbors based on historical

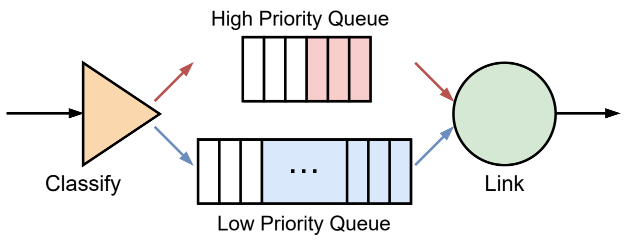


Fig. 3: DDR Queue Discipline

records, while taking into account potential delays between neighbors.

The delay estimation problem can be formulated as a Time Series Estimation (TSE) problem, which is commonly encountered in various domains such as signal processing, control engineering, and traffic engineering. Techniques such as Markov models [26], Hidden Markov models [27], and Reinforcement Learning [28] can be applied to solve TSE problems. In the context of DGR, the goal is to estimate the neighbor's future traffic conditions in a few milliseconds, which is closely associated with the link delay. While machine learning and reinforcement learning methods can achieve accurate estimations, they often demand substantial computing resources and extensive training datasets. We employ a straightforward yet effective Markov model for the latency estimation process.

	Actual level		
Predicted level	S_1	S_2	S_3
S_1	p_{11}	p_{12}	p_{13}
S_2	p_{21}	p_{22}	p_{23}
S_3	p_{31}	p_{32}	p_{33}

TABLE II: Transition Matrix for Markov Model

To represent the traffic state, we utilize the high-priority queue to define the state. The Markov model captures the transition probabilities between different traffic condition states. By observing the current state, the routers can estimate the probabilities of transitioning to different states in the future. This information enables the routers to make better decisions about the most appropriate forwarding directions, taking into account the expected queuing delays and avoiding congestion. The transition matrix, as presented in Table II, is a crucial tool used for estimating and evaluating the Markov model. The values in the matrix represent the probabilities of state transitions.

The estimated queueing delay (D_q) in each neighbor can be calculated using (3), where S represents the total number of states in the Markov model. p_{ij} denotes the transition probability from the ith state to the jth state, while d_j represents the expected queuing delay associated with the jth state. The delay expectation (D_q) is a crucial metric that enables routers to make intelligent decisions regarding forwarding directions. This information plays a significant role in the adaptive and dynamic behavior of the DDR protocol, ultimately contributing to efficient and reliable packet delivery under real-time network conditions.

$$D_q = \sum_{j=1}^{S} p_{ij} \cdot d_j \tag{3}$$

By leveraging the Markov model, the latency estimation process in TCDB provides routers with a reliable and efficient means to estimate their neighbor's future traffic conditions. This estimation contributes to better adaptive behavior of the DDR protocol, enabling timely and efficient packet delivery while considering real-time network conditions.

B. *Local Complete FIB*

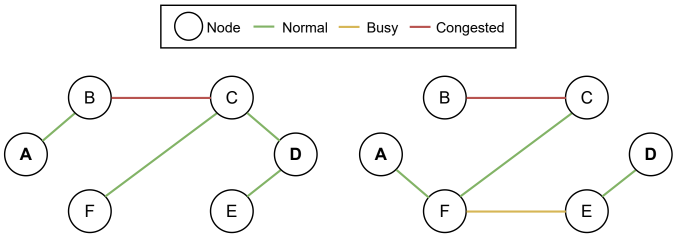


Fig. 4: Feasible Path Candidate Set

OSPF uses the Dijkstra's algorithm to compute the shortest path tree, also known as the Shortest-Path First (SPF) algorithm. However, relying on a single path may lead to network under-utilization and degraded performance with unbalanced traffic. Therefore, for DDR, we aim to identify all potential paths that can meet the delay requirement. To achieve it, a DDR router recursively uses the SPF algorithm to generate a recursive feasible path candidate set. While the RecursiveSPF algorithm efficiently generates the time-oriented routing table based on the static delay component, there are opportunities to optimize its performance. By leveraging parallel computing techniques, such as parallelization across multiple processing units or distributed computing across a network of routers, the algorithm can be executed efficiently. This reduces the computational complexity and enables faster generation of the deadline-driven routing table.

Algorithm 1: Feasible Path Candidate Set

```
Data: A graph $G(V, E)$, a root node $root$
Result: A feasible path candidate set $F$ for the node
        $root$
1 Let $depth$ be the depth of the forest.
2 Function RecursiveSPF($G, root, depth, F$):
3     if $depth == 0$ then
4         $F.push(Dijkstra(G, root))$
5     end
6     for each neighbor router $i$ of $root$ do
7         RecursiveSPF($G, i, depth - 1, F$)
8     end
9 return
```

To explore more alternative forwarding paths, we generate a feasible path candidate set for each node. The pseudo-code is given in Algorithm 1. For a directed graph $G(V, E)$, we assign any node as the root and initialize an empty tree set F to store the feasible path candidate set. Here we introduce $depth$ to represent the number of hops from the root node to its neighbors, which also determines the number of shortest-path trees in the current forest. Then we design a recursive function `RecursiveSPF` with a core of the Dijkstra's algorithm to calculate the feasible path candidate set. Recursive depth represents the range of traffic conditions observed by the router when making routing choices. For instance, as shown in Fig. 1, we set $depth$ to 1. At node A, there are two neighbors in 1 hop, B and F, so the feasible path candidate set includes two shortest-path trees rooted at node B and node F. The recursive depth can be flexibly set according to the research needs, which can gather more neighbor information across multiple hops and optimize the algorithm.

By exploring alternative paths beyond the shortest route, routers can dynamically adapt to changing network conditions, spread traffic load more evenly, and optimize resource utilization. This approach ensures efficient data transmission, minimizes congestion and enhances the overall performance of the time-oriented routing system.

C. Adaptive Route Selection

The adaptive route selection function aims to find a suitable forwarding direction for a packet within its budget time. Algorithm 2 presents the pseudo-code for adaptive route selection. It uses two functions related to the previous section. Function $D(r)$ returns the cost of route r, which is described in Sec. IV-B. Function $D_i(r)$ returns the estimated queueing delay based on local traffic condition, which is described in Sec. IV-A.

Algorithm 2: Adaptive Route Selection

```
Data: Forwarding Table $T$, Destination Address $d$,
      Latency budget $L$
Result: A forwarding direction $R$
1  Let $D(r)$ return the cost of route $r$.
2  Let $D_l(r)$ estimate the local traffic delay of route $r$.
3  Let $MinimumDelay(\cdot)$ return the index of the
     estimated fastest path of Set $(\cdot)$.
4  Let $S$ be an empty set of routes.
5  $S \leftarrow \varnothing$
6  for each route $r \in T$ to $d$ do
7      if $LoopAvoidCheck(r)$
8      && $D(r) + D_l(r) <= L$ then
9          S.Push (r)
10     end
11 end
12 index = $MinimumDelay(S)$
13 return S.Get (index)
```

Since DDR explores multiple paths, a key issue is how to avoid loops. For each packet, the algorithm evaluates all potential routes to the destination, and a loop avoidance check function ensures that the forwarding direction is closer to the destination. Each route is then analyzed based on its static latency and the estimated local queueing delay. Routes that can deliver the packet within its budget time are selected to the candidate route set S. The $MinimumDelay(\cdot)$ function then selects the route with the shortest estimated delay from this set.

The adaptive route selection function plays a critical role in ensuring timely packet delivery while considering both the dynamic traffic conditions and the link capacity. By intelligently selecting the most appropriate route, the algorithm helps achieve millisecond-level delay guarantees in the DDR system.

V. Implementation

In this section, we describe our implementation of DDR using the ns-3 [29] simulator and discuss some of the implementation issues that arose. The packet-level simulation used to evaluate the performance of DDR is illustrated in the next section.

The implementation of DDR is based on an OPSF link-state routing protocol that generates the LSDB by link-state advertisement (LSA). The route computation function for generating the local complete FIB and the route selection function has been modified according to Sec. IV. However, to ensure the optimal performance of DDR, two configuration issues need to be addressed: determining how to set the traffic condition scale and selecting the appropriate sampling rate.

In the proposed priority queue discipline for latency use delay before quantization, we define the maximum queueing delay for the priority queue as $Q_{\max}$. The queue is divided into K states, from 1 to K, and each state k represents a range of queueing delays. Specifically, the queueing delay in state K is between $\frac{(k-1)Q_{\max}}{K}$ and $\frac{kQ_{\max}}{K}$.

The scale traffic condition state setting is crucial and significantly impacts the protocol's performance. If the range for each state is too small, the Markov Model may necessitate predicting more steps to estimate the current state, potentially decreasing the accuracy due to the increased prediction complexity. Conversely, using a large granularity and fewer states might result in higher estimation errors for certain states. Thus, finding the right balance between granularity and prediction steps is essential to achieve accurate and efficient traffic condition estimations in the Markov Model for the priority queue. Properly selecting the traffic condition scale allows DDR to make informed and adaptive routing decisions based on real-time traffic conditions, ensuring reliable and low-latency packet delivery.

Eq. 4 provides a method to calculate the error in the Markov Model. The scale used in this equation is in terms of milliseconds, and the size of the error matrix is denoted as S. The error (err) is calculated as the sum of the products of the error matrix elements $C(i, j)$ and the corresponding differences between i and j, multiplied by half of the scale and then divided by S. The error matrix represents the relationship between the predicted traffic condition and the actual traffic condition. The error calculation takes into account the discrepancies between predicted values and actual values, and the scale factor ensures that the error is appropriately weighted based on the time scale used in the prediction.

$$err = \sum_{i=1}^{S}\sum_{j=1}^{S} C(i,j) \cdot (abs(i-j)+0.5) \cdot scale/S \quad (4)$$

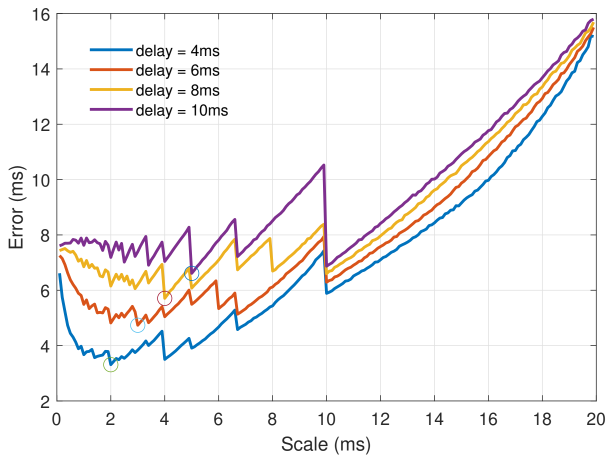


Fig. 5: The error in delay prediction varies different links with diverse delays at various scales.

In our simulation, we assumed a delay ranging from 4 ms to 10 ms for neighboring routers, and the high-priority queue length was set to accommodate data that can be drained in 20 ms. We conducted the simulation to find the optimal scale and sampling rate for the Markov Model. The simulation results that when the scale is configured as half of the delay of the neighboring link, the estimation error in traffic conditions tends to be minimized, as marked in Fig. 5. Thus, setting the scale to be close to the half actual link delay of the neighbor results in more accurate traffic condition predictions.

VI. Evaluation

A. Experimental setup

To evaluate the performance of DDR, we implement the protocol and the comparative benchmark by simulation. The results of the comparison are presented in (§VI-B). In addition, we have investigated the feasibility of deploying DDR in real-world networks in (§VI-C).

The DDR protocol is fully implemented in ns-3, and the evaluation is performed on a consumer-grade computer with Ubuntu 22.04 operating system, Intel i7-12700 CPU, 32GB of RAM, and SSD storage. To investigate its performance in a realistic network, we select four well-known network topologies as shown in Table III, namely Abilene, AT&T, CERNET, and GEANT, obtained from the Internet Zoo [30] dataset.

Topology Name	#Nodes	#Edges
Abilene	11	14
AT&T	25	26
CERNET	36	53
GEANT	22	37

TABLE III: Topology Information

We compare DDR to three protocols, ECMP, LFID [31], and DGRP. ECMP can choose the shortest paths with equal costs for packet delivery, and it is a typical table-based protocol that does not consider traffic conditions. LFID is a multi-path routing protocol that considers traffic engineering to reduce the load on congested links by distributing packets onto different paths. DGRP is a recent work that targets the same problem assuming that the neighbor status information is known without delay. For a fair comparison, we use the same topologies and background traffic conditions to test all routing protocols. Note that we use control packet exchanges to implement the exchange of DGRP neighbor queue information instead of reading the information directly from the simulator's memory. Thus our results are slightly different but more realistic than those reported in DGRP.

B. Performance evaluation

We first compare on-time delivery rates under various scenarios. Including an ideal network, traffic interference caused by elephant flows, and burst traffic. For each scenario, we select specific target transmissions that traverse the network. In the Abilene network, the target transmission is from Seattle to New York; in the AT&T network, it is from San Francisco to Washington D.C.; in the CERNET network, it is from Guangdong to Beijing; and in the GEANT network, it is from Ireland to Hungary. By evaluating DDR's on-time rate in these diverse scenarios, we can effectively evaluate its performance in handling real-world network conditions and interference.

Ideal network. Fig. 6 illustrates the comparison of the on-time delivery ratio between different protocols in ideal network scenarios with no competing traffic. The experiment was repeated 50 times for each protocol in each topology, varying the latency budget from half of the basic round-trip time (RTT) of the shortest path to 1.5 times the basic RTT. DDR achieved a 100% on-time arrival ratio when the latency budget is slightly larger than half of the basic RTT of the shortest path. DGRP exhibited some fluctuations due to fast changes in neighbor's queue lengths when selecting feasible paths based on the known neighbor state information. ECMP exhibited a vertical line, indicating that it forwarded all packets through the shortest paths with the same cost only and do not consider other choices. LFID, considering the k-shortest paths, introduced significant variance in packet arrival times due to separating packets on multiple paths without adaptive adjustment to packet latency requirements.

Elephant flow competing. Fig. 8 illustrates the performance of the protocols in the presence of elephant flow competing. The elephant traffic flow consists of TCP traffic that is not delay-sensitive and occurs on the main path of the target trans-

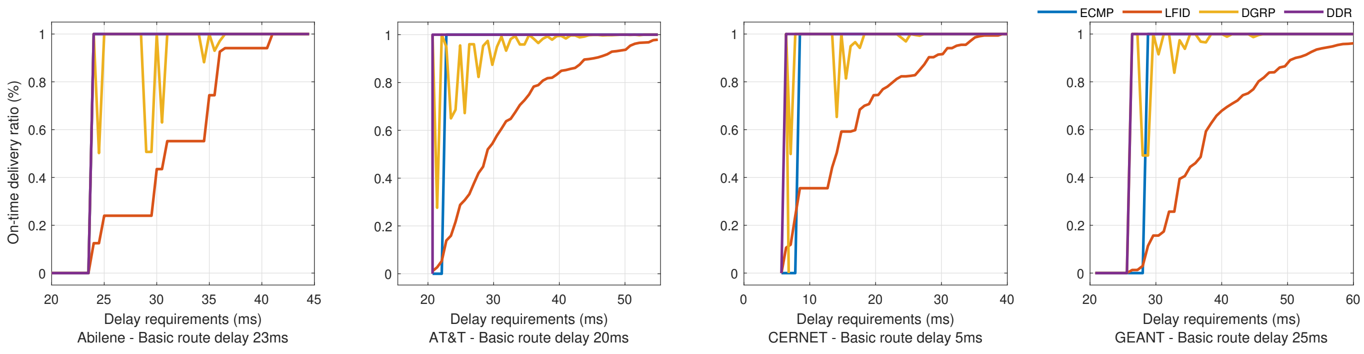
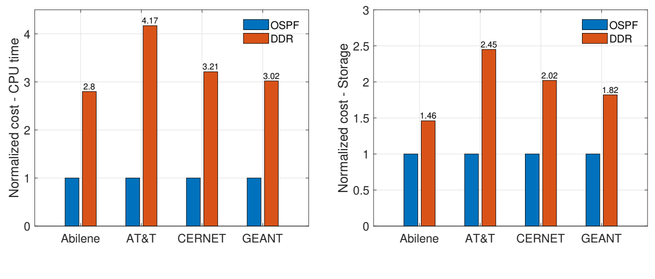


Fig. 6: On-time delivery ratio in ideal network



(a) CPU cost (b) Storage cost

Fig. 7: The CPU cost for protocol initialization and routing table storage cost on each topology.

mission. DGRP and DDR demonstrate superior performance compared to the other two algorithms by effectively utilizing neighbor traffic conditions and network topology information. DDR exhibits greater stability than DGRP, benefiting from the more accurate estimation provided by the Markov-based database. ECMP cannot adapt routing decisions based on traffic conditions, so it fails to deliver packets punctually in the presence of elephant flow. LFID's attempt to separate packets onto multiple paths for load balancing also leads to challenges in meeting latency requirements in this scenario.

Burst traffic competing. In Fig. 9, we evaluate the on-time delivery rate of packets in the presence of burst traffic occurring along the main route to the destination. The burst traffic used in our study is derived from Virtual Reality (VR) applications [32]. In heavy congestion conditions, the ECMP protocol's on-time arrival rate decreases to almost 0, mainly due to its limited route selection options, as ECMP cannot find a correct alternate direction for packets. LFID's performance decreases as some packets are separated onto congested routes, experiencing severe delays, and leading to missed deadlines. In contrast, DDR and DGRP perform better and eventually achieve a 100 $\%$ on-time arrival rate with a larger delay budget, thanks to priority management, which prioritizes delay-sensitive packets for forwarding. However, DGRP's performance is highly unstable, with packets experiencing significant delays even with sufficient budget. The heavy traffic congestion significantly impacts the estimated delay of queue discipline in DGRP, and its random route selection based on estimated information leads to erroneous forwarding decisions. The delay budget that DDR can support is increased compared to the case of elephant best-effort service flow competition. However, DDR can still consistently provide DDR service within a budget of 1.5 times the basic link delay. This success is attributed to accurate traffic condition estimation and a greedy selection of the best forwarding directions, ensuring reliable and low-latency packet delivery.

C. Deployment

The deployability of DDR can be assessed from two perspectives: compatibility with existing routing protocols and compatibility with legacy network devices.

Compatibility with existing routing protocols. DDR's compatibility with current Interior Routing Protocol (IGP) and Border Gateway Protocol (BGP) is twofold: Firstly, as a link-state-based adaptive routing protocol, DDR operates in a distributed manner without relying on a centralized controller, enabling seamless integration into traditional network infrastructures. Secondly, DDR can be deployed as a table-based routing protocol, sharing the Link-State Database (LSDB) with conventional protocols like OSPF, ensuring efficient coexistence with established routing mechanisms in existing networks.

Compatibility with legacy network devices. DDR is designed to be lightweight and resource-efficient, making it suitable for deployment on legacy network devices. The CPU time and storage space costs for protocol initialization, as depicted in Fig. 7(a), are normalized based on the OSPF protocol cost. The results show that DDR's CPU time utilization is approximately three times that of OSPF, even with increasing network size. Similarly, Fig. 7(b) illustrates that DDR's storage requirements are approximately two times that of OSPF with network size growth, normalized to the OSPF protocol cost. These findings demonstrate DDR's ability to operate effectively on legacy infrastructure without costly hardware upgrades.

VII. Discussion

As a new routing protocol, there are many interesting problems that beckon for further investigation.

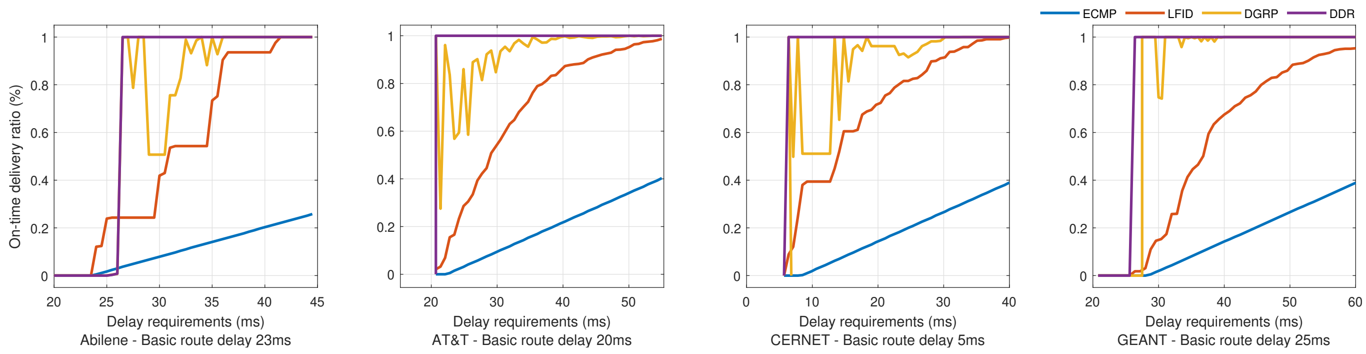


Fig. 8: On-time delivery ratio in elephant flow competing

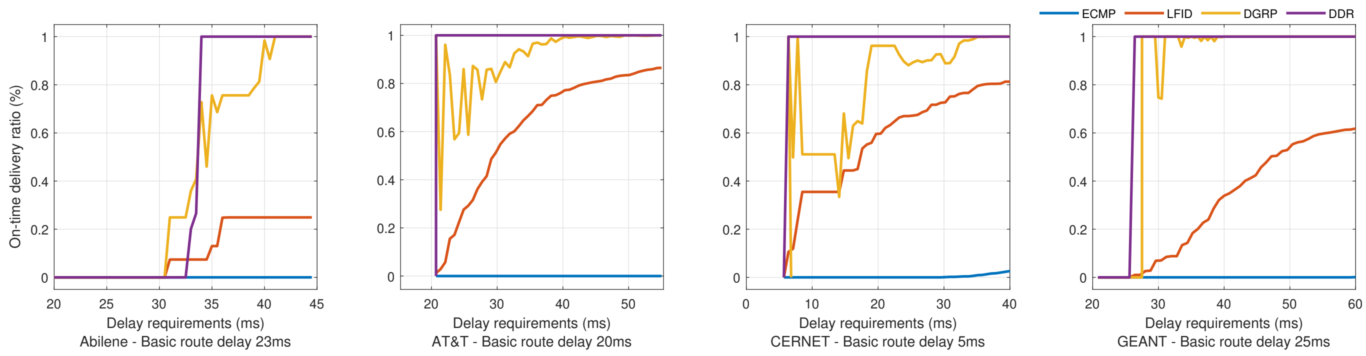


Fig. 9: On-time delivery ratio in burst traffic competing

Accurate estimation of neighbor's queue information. Accurate estimation of neighbor's future queue information is crucial for making informed and adaptive routing decisions in DDR. Improving the accuracy of real-time traffic state estimation, especially considering the dynamic nature of network traffic, is an important area for further research. Techniques such as machine learning or advanced statistical methods could be explored to improve the accuracy of queue information prediction.

Path selection for network utility and stability. Selecting paths that maximize network utility while ensuring system stability is a challenging optimization problem. The trade-off between efficient resource utilization and maintaining stable network operation must be carefully considered. The development of intelligent path selection algorithms that take into account factors such as link capacity, traffic load, and latency constraints could lead to more efficient and reliable DDR solutions.

Large-scale network test bed. Building a large-scale network testbed is crucial to evaluating the performance and scalability of DDR in real-world scenarios. Large-scale experiments can help researchers identify potential bottlenecks and challenges when deploying DDR in production networks. Creating a testbed that accurately replicates real-world network conditions and includes various traffic patterns and topologies will provide valuable insights into the practicality and effectiveness of DDR.

VIII. Conclusions

The increasing demand for new applications has necessitated the need for DGR solutions. In this paper, we formulated the DGR problem and proposed DDR, a heuristic routing prototype for Intra-domain networks. Unlike centralized approaches used in SDN, DDR operates in a distributed manner, making it suitable for networks without a centralized controller. By leveraging both local neighbor traffic conditions and network topology information, DDR adaptively adjusts forwarding directions to provide packet-level delay-guaranteed service. Additionally, DDR ensures backward compatibility with legacy devices and can work in conjunction with existing protocols. Through detailed simulations, we demonstrated that DDR outperforms other approaches in terms of packet-level on-time delivery. These results underscore the effectiveness and practicality of DDR in achieving efficient and reliable communication in modern networks.

References

[1] "Global state of the WAN Report, 2020," https://info.aryaka.com/state-of-the-wan-report-2020.html.

[2] R. Govindan, I. Minei, M. Kallahalla, B. Koley, and A. Vahdat, "Evolve or die: High-availability design principles drawn from googles network infrastructure," in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016, pp. 58–72.

[3] P. Kumar, Y. Yuan, C. Yu, N. Foster, R. Kleinberg, P. Lapukhov, C. L. Lim, and R. Soulé, "Semi-oblivious traffic engineering: The road not taken," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, 2018, pp. 157–170.

[4] X. Xiao, A. Hannan, B. Bailey, and L. M. Ni, "Traffic engineering with MPLS in the internet," *IEEE network*, vol. 14, no. 2, pp. 28–33, 2000.

[5] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, "B4: Experience with a globally-deployed software-defined wan," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 3–14, 2013.

[6] R. Kumar, M. Hasan, S. Padhy, K. Evchenko, L. Piramanayagam, S. Mohan, and R. B. Bobba, "End-to-end network delay guarantees for real-time systems using sdn," in *2017 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2017, pp. 231–242.

[7] Y. Chang, S. Rao, and M. Tawarmalani, "Robust validation of network designs under uncertain demands and failures," in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, 2017, pp. 347–362.

[8] J. Bogle, N. Bhatia, M. Ghobadi, I. Menache, N. Bjørner, A. Valadarsky, and M. Schapira, "TEAVAR: striking the right utilization-availability balance in wan traffic engineering," in *Proceedings of the ACM Special Interest Group on Data Communication*, 2019, pp. 29–43.

[9] R. Geisberger, P. Sanders, D. Schultes, and D. Delling, "Contraction hierarchies: Faster and simpler hierarchical routing in road networks," in *Experimental Algorithms: 7th International Workshop, WEA 2008 Provincetown, MA, USA, May 30-June 1, 2008 Proceedings 7*. Springer, 2008, pp. 319–333.

[10] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2023. [Online]. Available: https://www.gurobi.com

[11] M. ApS, "Mosek optimization toolbox for matlab," *User's Guide and Reference Manual, Version*, vol. 4, p. 1, 2019.

[12] J. T. Moy, *OSPF: anatomy of an Internet routing protocol*. Addison-Wesley Professional, 1998.

[13] Y. Rekhter, T. Li, and S. Hares, "A Border Gateway Protocol 4 (BGP-4)," Internet Requests for Comments, RFC Editor, RFC 4271, January 2006. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4271.txt

[14] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS weights in a changing world," *IEEE journal on selected areas in communications*, vol. 20, no. 4, pp. 756–767, 2002.

[15] C. Hopps, "Analysis of an Equal-Cost Multi-Path Algorithm," Internet Requests for Comments, RFC Editor, RFC 2992, November 2000. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2992.txt

[16] M. A. Qureshi, Y. Cheng, Q. Yin, Q. Fu, G. Kumar, M. Moshref, J. Yan, V. Jacobson, D. Wetherall, and A. Kabbani, "PLB: congestion signals are simple and effective for network load balancing," in *Proceedings of the ACM SIGCOMM 2022 Conference*, 2022, pp. 207–218.

[17] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, V. T. Lam, F. Matus, R. Pan, N. Yadav *et al.*, "CONGA: distributed congestion-aware load balancing for datacenters," in *Proceedings of the 2014 ACM conference on SIGCOMM*, 2014, pp. 503–514.

[18] P. Kazemian, M. Chang, H. Zeng, G. Varghese, N. McKeown, and S. Whyte, "Real-time network policy checking using header space analysis," in *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, 2013, pp. 99–111.

[19] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven wan," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, 2013, pp. 15–26.

[20] J. Li, Z. Li, R. Lu, K. Xiao, S. Li, J. Chen, J. Yang, C. Zong, A. Chen, Q. Wu *et al.*, "Livenet: a low-latency video transport network for large-scale live streaming," in *Proceedings of the ACM SIGCOMM 2022 Conference*, 2022, pp. 812–825.

[21] L. Cai, J. Pan, W. Yang, X. Ren, and X. Shen, "Self-evolving and transformative (SET) protocol architecture for 6g," *IEEE Wireless Communications*, 2022.

[22] B. Anwer, T. Benson, N. Feamster, and D. Levin, "Programming slick network functions," in *Proceedings of the 1st ACM SIGCOMM symposium on software-defined networking research*, 2015, pp. 1–13.

[23] H. Kim, J. Reich, A. Gupta, M. Shahbaz, N. Feamster, and R. Clark, "Kinetic: Verifiable dynamic network control," in *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, 2015, pp. 59–72.

[24] M. P. Grosvenor, M. Schwarzkopf, I. Gog, R. N. Watson, A. W. Moore, S. Hand, and J. Crowcroft, "Queues don't matter when you can JUMP them!" in *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, 2015, pp. 1–14.

[25] P. Yang, L. Cai, and T. Chang, "Dgr: Delay-guaranteed routing protocol," in *2023 IEEE 20th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*, 2023, pp. 19–27.

[26] T. Truong-Huu, P. M. Mohan, and M. Gurusamy, "Virtual network embedding in ring optical data centers using markov chain probability model," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1724–1738, 2019.

[27] Z. Chen, J. Wen, and Y. Geng, "Predicting future traffic using hidden markov models," in *2016 IEEE 24th international conference on network protocols (ICNP)*. IEEE, 2016, pp. 1–6.

[28] Z. Xia, Y. Zhou, F. Y. Yan, and J. Jiang, "Genet: automatic curriculum generation for learning adaptation in networking," in *Proceedings of the ACM SIGCOMM 2022 Conference*, 2022, pp. 397–413.

[29] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena, "Network simulations with the ns-3 simulator," *SIGCOMM demonstration*, vol. 14, no. 14, p. 527, 2008.

[30] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *Selected Areas in Communications, IEEE Journal on*, vol. 29, no. 9, pp. 1765–1775, October 2011.

[31] K. Schneider, B. Zhang, and L. Benmohamed, "Hop-by-hop multipath routing: Choosing the right next-hop set," in *IEEE INFOCOM*, 2020, pp. 2273–2282.

[32] M. Lecci, M. Drago, A. Zanella, and M. Zorzi, "An open framework for analyzing and modeling XR network traffic," *IEEE Access*, vol. 9, pp. 129 782–129 795, 2021.