This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3374213

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2023.0322000

GATE-BC: Genetic Algorithm-Powered QoS-Aware Cross-Network Traffic Engineering in Blockchain-Enabled SDN

MURAT KARAKUS
Department of Software Engineering, Ankara University, Ankara, Türkiye (e-mail: mrtkarakus@ankara.edu.tr)
Corresponding author: MURAT KARAKUS (e-mail: mrtkarakus@ankara.edu.tr).

This work is supported by the Scientific & Technological Research Council of Türkiye (TUBITAK) under Grant No. 120E448.

ABSTRACT As a result of the ubiquitous network applications and services, exacerbated by the overarching digital revolution the need and demand for efficient and dependable connectivity solutions have surged to unprecedented levels. Quality of Service (QoS)-based routing has emerged as a critical solution, enabling service differentiation, efficient resource allocation, and improved network performance. In this study, we introduce a novel Genetic Algorithm-powered QoS-aware Cross-Network Traffic Engineering framework, *GATE-BC*, at the confluence of Software Defined Networking (SDN) and Blockchain (BC) technologies. *GATE-BC* orchestrates end-to-end (E2E) QoS traffic, providing resource-efficient, reliable, and latency-tolerant delivery of intelligent network services in BC-enabled SDNs. Leveraging BC features such as decentralization, transparency, and immutability, *GATE-BC* eliminates the need for centralized entities in QoS-supported cross-network routing models. We compare *GATE-BC* framework with three other traffic management and engineering approaches: QoSChain (*QC*), Hierarchical Routing Approach (*HRA*), and Distributed Routing Approach (*DRA*). The extensive simulations reveal that *GATE-BC* outperforms the other routing strategies in terms of Path Setup Time (PST), Network Message Overhead (NMO), Request Acceptance Ratio (RAR), Network Bandwidth Consumption (NBC), Average Path Length (APL), and Average Network Length (ANL) metrics under various network topologies. Furthermore, *GATE-BC* employs three different feasible path selection strategies based on bandwidth (*GATE-BC_BW*), delay (*GATE-BC_D*), and reliability (*GATE-BC_R*) QoS parameters to satisfy the service levels requested.

INDEX TERMS Blockchain, Genetic Algorithm, Multi Domain, QoS, SDN, Traffic Engineering

I. INTRODUCTION

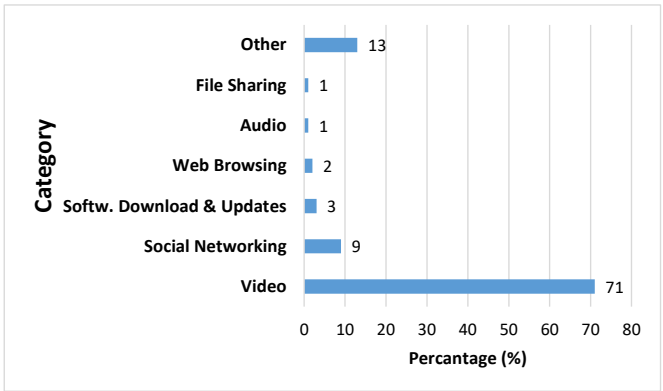
IN the current era of ever-connected world with ubiquitous and digitized applications and services, the expected levels of efficiency and reliability from the networking infrastructure have been reaching unprecedented heights. According to Ericsson's video traffic update study [1], as visualized in Fig. 1, video streaming and sharing are experiencing a significant increase in user engagement. Video is the most prominent and fastest-growing global mobile data traffic segment, especially on social media and video-on-demand platforms. In 2022, video accounted for around 70% of overall traffic and is projected to grow steadily by 30% annually until 2028. It is expected to reach 80% of global mobile data traffic by then.

The goal of QoS-based end-to-end (E2E) inter-network traffic management is to ensure the delivery of network services with predefined performance parameters, such as bandwidth, delay, reliability, etc. Traditional routing protocols often lack the granular control and flexibility required to meet the diverse QoS requirements of different applications and services. Software Defined Networking (SDN) [2], however, offers centralized control and programmability capabilities to implement QoS-based routing effectively. SDN facilitates the separation of the control and data layers, allowing for centralized administration and the programmability of network elements. The separation of control logic from the physical network structure in SDN offers a versatile and dynamic framework for network management, setup, and optimization. The programmability feature enables the flexible (re)allocation of network assets in response to application demands, traffic flow characteristics, and other dynamic variations [3].

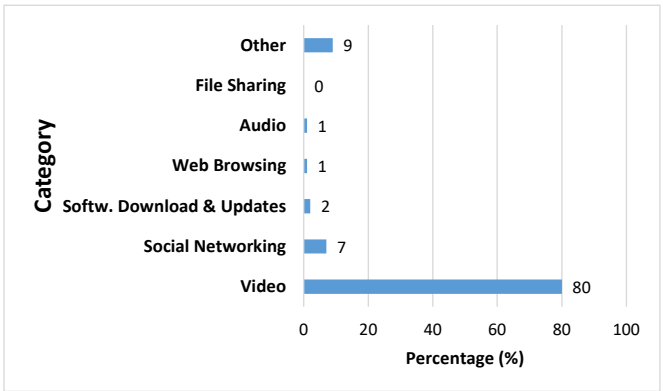
We improve the decision making processes of the QoS-based routing in terms of reliability, security, and transparency by means of integrating the Blockchain (BC) technology. BC, originally developed to support cryptocurrencies

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3374213



(a) 2022: 90 Exabytes (EB) per month



(b) 2028: 324 Exabytes (EB) per month

FIGURE 1: Mobile data traffic by application category per month [1].

like Bitcoin [4], is a distributed and decentralized ledger that records transactions across a network of computers. Its key characteristics, including immutability, transparency, and consensus mechanisms, make it a powerful tool for securing and auditing routing decisions in a decentralized manner [5].

A. MOTIVATION

As alluded earlier, QoS-based routing is a crucial aspect of cross-network traffic management, focusing on bandwidth, latency, and reliability to optimize performance and differentiate services. It dynamically allocates resources to QoS channels, ensuring real-time communication, video streaming, and online gaming. This approach adapts to network updates, dynamics, and failures, improving fault tolerance and resilience of the infrastructure. It can divert traffic to QoS-satisfying paths upon a network connection or node failure, ensuring key services remain running during outages [6]. QoS-based routing also helps network operators satisfy Service-Level Agreements.

While QoS-based provides the aforementioned benefits it is not without several problems and challenges:

- **Scalability:** Computational and spatial complexity of routing increases, especially in larger networks.
- **Heterogeneous QoS Metrics:** Under disparate multiple metrics, QoS needs for various services and applications become more difficult to express and manage.
- **Dynamic Network Conditions:** Traffic patterns, congestion, and connection disruptions can dramatically change network conditions, which in turn require fast adaptation by the QoS-based routing algorithms to maintain acceptable service levels.
- **Interoperability:** With many devices and components involved, such as routers, switches, controllers, and applications, interoperability gets more challenging.
- **Security and Trust:** Malicious actors may alter or fake network measurements, resulting in QoS-violating routing decisions, among many other threats and vulnerabilities. Inter-network routing demands QoS data security and trustworthiness.
- **Overhead and Resource Consumption:** QoS routing involves more computation, signaling overhead, and resource consumption.

We posit that SDN is a powerful tool that can significantly enhance QoS-based cross-network traffic management. It centralizes network administration by physically separating control and data planes, allowing network managers to monitor the entire network and make QoS-based routing decisions. SDN's *programmability* allows for QoS-specific routing decisions, allowing network managers to prioritize traffic and applications for key services, while also enabling the creation and enforcement of QoS-aligned policies, enabling network-wide QoS routing. Its *flexibility* allows for dynamic routing path adjustments based on network conditions, enhancing QoS performance. SDN also allows fine-grained traffic engineering and load balancing for QoS-based routing, improving network performance by distributing traffic across various channels [7]. It provides real-time traffic and QoS analytics, allowing administrators to analyze network data and optimize QoS-based routing. SDN also allows service chaining, ensuring QoS-compliant traffic routing [8].

On the other hand, BC technology has a great deal to offer in QoS-based cross-network traffic management, providing such features as *data sharing*, *immutability*, *transparency*, and *decentralization* [9]. It can create a distributed ledger for sharing routing-related data among network participants, enhancing collaboration and coordination. BC can create the integrity and trustworthiness of routing decisions and QoS parameters, reducing the risk of unauthorized modifications or disputes [10]. Transparency is provided by making routing-related data accessible to all authorized participants, fostering accountability, and identifying potential issues. BC operates decentralized, eliminating the need for a central authority/control aligning with the distributed nature of inter-network routing [11]. This decentralization enhances fault tolerance and resilience, reducing the risk of a single point of failure, and ensuring collective validation of routing decisions through a consensus mechanism. Overall, BC technology is a valuable tool to improve the performance of the QoS-based

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3374213

inter-network routing.

Finally, the combination of SDN and BC technology offers a promising solution to address the aforementioned challenges, particularly in the realm of cross-network QoS-based traffic management. SDN's programmability allows for dynamic routing decisions based on real-time network conditions and QoS requirements, while BC ensures decentralization, integrity, and transparency, providing a *trustless* environment for QoS-supported cross-network traffic management [12]. This convergence holds immense potential for efficient effective resource allocation, enhanced performance, and improved network reliability, paving the way for a more robust and adaptable networking infrastructure.

B. CONTRIBUTION AND ORGANIZATION

The contributions of this research are as follows:

- This study presents a Genetic Algorithm (GA)-powered QoS-aware cross-network traffic management framework, *GATE-BC*, in BC-enabled SDN (BC-SDN) networks to steer E2E QoS traffic for resource-thrifty, reliable, and delay-tolerant delivery of intelligent network services and applications. The proposed *GATE-BC* framework presents a coordination framework that unites parties with conflicting interests concerning SLAs, security, privacy risks, and other factors for network QoS-based traffic management.
- The performance of the proposed *GATE-BC* framework with bandwidth, delay, and reliability QoS parameters is evaluated and compared against three QoS-based routing strategies in SDN networks [7], [13]: QoSChain (*QC*) [14], Hierarchical Routing Approach (*HRA*) [15], and Distributed Routing Approach (*DRA*).
- *GATE-BC* employs three different feasible path selection strategies based on three QoS parameters to assign an E2E path for service requests: *GATE-BC_BW*, *GATE-BC_D*, and *GATE-BC_R*.
- The study uses NSFNET (14-network), US Backbone (24-network), Random-14, and Random-24 topologies with a size ranging from 6 to 10 switches per network to conduct sensitivity analysis.
- The study exploits Path Setup Time (PST), Network Message Overhead (NMO), Request Acceptance Ratio (RAR), Network Bandwidth Consumption (NBC), Average Path Length (APL), and Average Network Length (ANL) metrics to conduct a comprehensive analysis on the performance of the proposed *GATE-BC* framework.

As a result, this study contributes to the understanding of SDN and BC convergence within the context of cross-network QoS-based traffic management leveraging optimization capabilities of the GA to offer potential solutions to long-standing networking challenges.

This paper's organization is as follows: Section II presents state-of-the-art regarding routing/QoS in SDN and BC. In Section III, SDN and BC technologies are briefly explained to make this paper self-inclusive. Section IV presents the *GATE-BC* framework along with basic terms, processes, functionalities, and concepts related to both networking and BC. Section V provides the experiment results. Section VI discusses open problems and future research directions emanating from our vision before the concluding remarks in Section VII.

II. RELATED WORK

This section first presents the survey studies that investigate the intersection of BC and SDN. Then, we review the relevant literature with or without SDN networks. Table 1 shows the classification of studies regarding their use of technologies, target services, and network types.

A. REVIEW ON BLOCKCHAIN AND SDN SURVEYS

Numerous surveys or position papers have explored the integration of BC and SDN technologies with a focus on their synergies. The research conducted by [16] examines the use of SDN and BC technologies to create resilient cybersecurity measures, while [17] investigates current security concerns and measures within the BC-based SDN (BC-SDN) framework. According to [18], the present state of BC-SDN may be classified into many categories, including consensus methods, encryption assurances, data security, and log transparency. Focusing on the use of BC technologies in combination with SDN, [19] analyzes both security-aware and security-agnostic perspectives. Also, the survey conducted by [20] analyzes the BC-SDN setting, highlighting its security and privacy drawbacks. Exploring the use of BC, SDN, and Network Function Virtualization (NFV) in intelligent residential environments, the researchers in [21] introduce a security framework and address security issues. In [22], the authors explore the potential benefits of BC and SDN in the healthcare industry. This includes addressing system failure reduction, facilitating medical data sharing, maintaining personal health records, and enhancing audit conduct. Additionally, [3] delves into the integration of BC with SDN within the IoT ecosystem, known as BC-SDIoT. The focus is on various aspects such as security, computing models, trust management, access control, authentication, privacy, and networking. Lastly, [23] scrutinizes the integration of BC with SDN-based IoT systems utilizing NFV for intelligent applications. The examination provides comprehensive coverage of BC-SDN and NFV-enabled intelligent applications.

B. REVIEW ON BLOCKCHAIN-ONLY STUDIES

Next, BC is applied in diverse research studies related to networking without SDN. The Border Gateway Protocol (BGP) is crucial to modern Internet infrastructure. However, its security risks and inability to accommodate advanced applications make it vulnerable to attacks. [24] presents a new architecture for managing BGP, called BGPChain, that utilizes BC to create a secure, intelligent, and adaptable routing infrastructure for the evolving Internet. The research conducted in [9] also examines the current status of inter-domain routing security and provides an overview of the deficiencies found in the existing PKI-based authentication method. ROAchain [25] is a new BGP security infrastructure that uses BC to mitigate Resource Public Key Infrastructure (RPKI) hazards. It is de-

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3374213

TABLE 1: Classification of the related studies regarding their use of technologies, target services, and network types.

Reference	BC	SDN	Routing	QoS	Multi-Domain	Network	Solution/Method	Summary
[9]	✓		✓		✓	WAN	PKI Authentication	Analyzing inter-domain routing security and identifying PKI authentication flaws
[14]	✓	✓	✓	✓	✓	WAN	Shortest Path Routing	A BC-aided inter-AS routing architecture in SDNs
[24]	✓		✓	✓		WAN	Contract-based route scheduling	An architecture for managing BGP with BC to create a secure Internet routing infrastructure
[25]	✓		✓		✓	WAN	Decentralized ROA repositories	BGP security infrastructure to mitigate RPKI risks using BC
[26]	✓		✓			WSN	Trust evaluation model for ANs/SNs	BC-based secure routing mechanism for nodes in WSNs
[27]	✓		✓		✓	WAN	IP Prefix validation using SC and CA	Blocking IP prefix hijacking in the BGP using a consortium BC
[28]	✓		✓			IoT	BC Security IoT (BSI) protocol	Authentication of mobile sensor nodes with CHs and data integrity using BC
[29]	✓		✓			RAN	PipeNet-like protocol to prevent ledger forgery	BC-based multi-hop routing protocol for self-organizing and non-cooperative MANETs
[30]	✓			✓		WAN	BC-based SC and zk-SNARK algorithm	Enabling users to request on-demand QoS sessions in a traceable and fairly monetized manner
[31]	✓	✓				WAN	SCs to secure IP prefix and BGP4 messages	Immutable Internet routing registry and origin repository for BGP4 security management in SDN
[32]	✓	✓				WAN	BGP attack detection by DA-T2TD3 algorithm	Securing BGP route update validation by GRU against attacks using deep learning and BC
[33]	✓	✓	✓		✓	IoT	SCs to store topologies and reputation model	Securing routing among multi-domain SDN-enabled IoT network against black hole attacks
[34]	✓	✓	✓		✓	IoT	SCs to ensure UAV-IoT network topology in BC	Securing cross-domain routing and building trust in controllers of SDN-based UAV-IoT networks
[35]	✓	✓	✓			IoT	LRA and RCM with SCs for validation in BC	Authentication of RNs, optimized routing, and route validation in SDN-enabled IoT networks
[36]	✓	✓	✓	✓	✓	WAN	Domain trust/reputation scores stored in BC	Enabling trust-aware E2E QoS routing in multi-SDNs
GATE-BC	✓	✓	✓	✓	✓	**WAN**	**Genetic Algorithm-based routing by sharing pathlets in BC**	**E2E QoS-aware traffic management with various path selection strategies in multi BC-SDNs**

centralized with each Autonomous System (AS) maintaining a globally consistent and tamper-proof Route Origin Authorization (ROA) repository. ROAchain proposes a unique consensus method with trust value integration, collaborative signing, sharding, and punishment schemes. [26] presents a BC-based encryption and trust assessment framework, utilizing public and private BCs to secure the identities of Aggregator Nodes (ANs) and Sensor Nodes (SNs). However, this system faces challenges as well due to limited resources and processing capacity. The authors in [37] introduce Ad-hoc On-Demand Distance Vector multi-path QoS (AODV-MQS), a security technique for multi-path QoS routing based on BC, which improves the AODV protocol. It establishes a network chain with intermediate nodes, selectively identifying and separating nodes adhering to QoS requirements using a Smart Contract (SC). BlockJack [27] is a decentralized method that uses a consortium BC along with SC and Certificate Authority (CA) to prevent IP prefix hijacking in the BGP. It synchronizes BC and BGP networks using operational autonomy interfaces with adaptable BC and BGP update periods. Limited computing, memory, energy, and encryption algorithm complexity are concerns for encrypting data in IoT wireless sensors. The BC Security IoT (BSI) protocol proposed in [28] addresses challenges of data encryption in IoT wireless sensors, such as limited computing, memory, energy, and encryption algorithm complexity. Storing keys at the sensor level enables decentralized, transparent, and scalable authentication with Cluster Heads (CHs). It also facilitates linking to multiple networks or clusters without impacting sensor or network performance. The authors of [29] propose a multi-hop routing protocol called Data Broker, designed for self-organizing and non-cooperative Mobile Ad hoc Networks (MANETs). This protocol documents route information and provides incentives to brokers. The BlockQoS framework [30] uses BC to monetize on-demand QoS, utilizing a SC-based system. This decentralized approach eliminates middlemen, establishes SLAs, and reduces BC operating costs through off-chain invoicing confirmed by zero-knowledge proofs (zk-SNARK).

C. REVIEW ON BLOCKCHAIN-ENABLED SDN STUDIES

Finally, BC and SDN integration is a prominent focus in networking research, investigating their potential to improve network security, efficiency, and management. [31] introduces BC-based BGP4 Orchestration (BBO) to enhance BGP4 transaction openness. BBO uses an Internet Number Resource Authority and a trustworthy management body and serves as an advertising origin source. It outperforms RPKI, ensuring origin advertisement security and configuring resource repositories in SDN-based federations. The SDBGPChain model proposed in [32] uses deep learning and BC to protect against BGP attacks using the Dual Agent-

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3374213

based Twin Actor Twin Delayed Deep Deterministic Policy Gradient (DA-T2TD3) algorithm. The authors perform route update validation by Gated Recurrent Unit (GRU) to update message tuples. The GRU validates the updates by identifying misconfigurations and errors in the updated messages. The authors in [33] propose an architecture using BCs for secure routing in SDN-enabled IoT networks. The BC's consensus and immutability provide an authentic network view while the proposed reputation mechanism ensures routing reliability at both local and global levels. Similarly, the study in [34] proposes a BC framework for secure routing and establishing trust across multiple SDN-enabled Unmanned Aerial Vehicle-IoT (UAV-IoT) networks. SCs enable SDN network controllers to post abstract typologies into BC to ensure consensus, immutability, routing reliability, and global credibility on network topologies while routing across multiple domains. [35] proposes a BC-based authentication method for Relay Nodes (RNs) in IoT networks to store minimal credentials for lightweight authentication using the Lightweight Registration and Authentication (LRA) mechanism. The SDN controller uses a GA to find the most energy-efficient routes to send into a public BC. It also uses the Route Correctness Mechanism (RCM) and SCs to ensure accurate IoT network routes. Next, the TRAQR framework [36] aims to establish trust and verify QoS compliance in E2E routing over SDNs across multiple domains. It uses BC for trusted data storage and exchange but does not consider path selection for domain trust scores. Contrary to the TRAQR framework, the proposed *GATE-BC* framework uses a modified Genetic Algorithm for dynamic and performance-driven cross-network QoS-aware routing. Furthermore, *GATE-BC* employs three different feasible path selection strategies based on the bandwidth, delay, and reliability QoS parameters to assign an E2E path for service requests. This flexibility allows for tailoring the framework's performance to specific network requirements and ensures its applicability across diverse network environments. Furthermore, this study conducts a topology-wise sensitivity analysis using diverse network topologies, including NSFNET, US Backbone, Random-14, and Random-24, and a comprehensive set of performance indicators.Finally, our previous work, QoSChain (*QC*) [14], combined SDN and BC for the cross-network traffic management without GA and QoS metric-aware path selection capabilities.

As can be seen from the above explanations about the previous studies, and as summarized in Table 1, *GATE-BC* introduces the following key innovations: First, it leverages a customized Genetic Algorithm (GA) specifically designed for QoS-aware path selection using pathlets stored in BC as transactions from networks. This novel approach beats traditional routing algorithms in terms of optimality, adaptability, and robustness, and ensures efficient and reliable traffic flow across diverse network topologies. Second, *GATE-BC* employs a multi-objective optimization to simultaneously satisfy multiple QoS parameters, including bandwidth, delay, and reliability. Unlike existing frameworks that prioritize single aspects of QoS, *GATE-BC* offers a comprehensive approach that balances various performance metrics. Third, it conducts a comprehensive performance analysis using a diverse set of performance indicators, including PST, NMO, RAR, NBC, APL, and ANL. These diverse metrics provide a holistic understanding of the framework's performance and demonstrate significant improvements compared to existing solutions.

Additionally, *GATE-BC* employs three different feasible path selection strategies (*GATE-BC_BW*, *GATE-BC_D*, and *GATE-BC_R*) based on the QoS parameters to assign an E2E path for service requests. This adaptability stands in contrast to rigid frameworks that offer limited options for path selection. Lastly, the study showcases the robustness and generalizability of *GATE-BC* through sensitivity analysis using diverse network topologies, including NSFNET, US Backbone, Random-14, and Random-24.

III. BACKGROUND

In this section,we provide a brie overview of SDN and BC in order to provide a self-contained narrative to the readers. Those with a prior background in these topics may skip ahead.

A. OVERVIEW OF SOFTWARE DEFINED NETWORKING (SDN)

SDN is a network framework that enhances network programmability and adaptability by segregating the control plane from the data plane. This centralized control layer is distinct from the data layer, allowing for better monitoring and management over the network. The control plane is managed by a software-based controller, responsible for making network-wide decisions like routing, traffic engineering, and policy enforcement. The data plane, consisting of physical network infrastructure, such as switches and routers, forwards packets based on instructions received from the controller through Data-Controller Plane Interfaces (D-CPIs). The data plane provides OpenFlow-enabled interfaces for programming forwarding tables, defining flow rules, and collecting network statistics. Communication between the control plane and data plane occurs through the OpenFlow protocol [38], which defines a standardized set of instructions and messages for managing and controlling the forwarding behavior of network devices. This allows the controller to dynamically adapt network flows, redirect traffic, and enforce QoS policies based on real-time network conditions.

As illustrated in Fig. 2, the SDN architecture is built upon a layered approach consisting of multiple layers with specific functions:

- **Infrastructure Layer:** The SDN controller manages physical network infrastructure, utilizing D-CPIs to forward packets. Transport Layer Security (TLS) is used to secure D-CPI messages, which are transmitted through the OpenFlow protocol. This protocol is widely adopted as the D-CPI for communication between controllers and data layer devices in network connectivity.
- **Control Layer:** The SDN controller is a crucial component in the control layer, responsible for network-wide decision-

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3374213

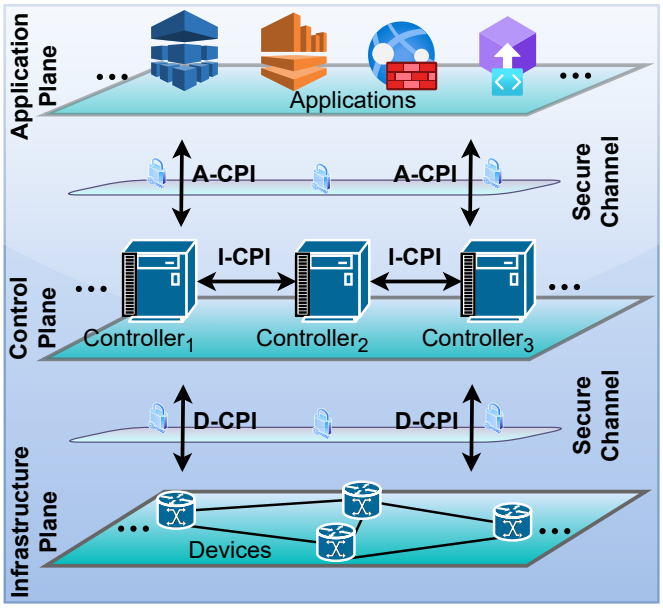


FIGURE 2: SDN architecture with its main layers (infrastructure layer, control layer, and application layer) and components (e.g. A-CPI, D-CPI, and I-CPI).

making and policy enforcement. It interacts with infrastructure layer devices through D-CPIs, exchanging control messages and programming flow rules. The control layer abstracts the underlying physical network, providing a logical view to higher layers. Intermediate-Controller Plane Interfaces (I-CPIs) [39] facilitate data sharing between controllers, while Application-Controller Plane Interfaces (A-CPIs) enable communication among controllers and applications for security, management, and other functionalities.

- **Application Layer:** This layer comprises network applications and services that utilize the control layer's capabilities. These applications interact with the controller via the A-CPI, such as REST API, to request resources, define policies, and receive status updates, enabling dynamic control and customization of network behavior.

SDN simplifies configuration and provisioning tasks, automates network services, and reduces operational burdens. Its programming ability allows for automation and orchestration, while the abstraction layer enhances scalability by dynamically allocating resources to meet varying traffic needs.

B. OVERVIEW OF BLOCKCHAIN TECHNOLOGY

BC is a decentralized ledger that ensures secure and transparent transactions through cryptographic security, consensus procedures, and decentralized governance [40]. Its emergence has the potential to transform sectors by providing a resilient structure for trustworthiness and responsibility in digital transactions.The key features of BC can be summarized as follows:

- **Decentralization:** BC operates through a decentralized network of participants, eliminating the need for a centralized governing entity, ensuring system durability, fault resistance, and resistance to censorship.
- **Transparency and Immutability:** BC ensures transparency and visibility of transactions, making them difficult to alter or delete, preserving data integrity, and enabling audits.
- **Consensus Mechanisms:** Consensus methods in a BC network, such as Proof of Work (PoW) and Proof of Stake (PoS), ensure transaction legitimacy, chronological arrangement, network security, and scalability [41].
- **Cryptographic Security:** BC uses cryptographic methods like hash functions, digital signatures, and encryption to protect data, transactions, and individuals from tampering and fraudulent activities.

BC networks can be categorized into public/permissionless (PU-BC), private/permissioned (PR-BC), and consortium (CO-BC) (hybrid) types [42]. PU-BCs are open and secure, such as Bitcoin and Ethereum, but may face scalability and transaction speed issues. PR-BCs are limited and centralized, providing privacy and control. CO-BCs combine PU-BC and PR-BC characteristics with governance by a consortium, offering scalability, reduced resource usage, and customized consensus procedures.

Fig. 3 is a high-level conceptual illustration of BC. In general, a block is composed of various data fields depending on the use case and consensus mechanism. However, some data fields are immanent, as given below:

- **Block Number:** This field indicates numbers to identify the blocks.
- **Block Version:** The version field indicates the protocol version used in the BC.
- **Hash:** This field holds the cryptographic digest of the block.
- **Previous Block Hash:** This field holds the cryptographic digest of the previous block.
- **Merkle Root:** The Merkle root is the cryptographic hash of all transactions included in the block.
- **Timestamp:** The timestamp field records the time at which the block is created to establish a chronological order of blocks.
- **Transaction Data:** Storage of information that could represent transactions, contracts, assets, identities, or practically anything that can be described in digital form.

BC provides a secure, transparent, and decentralized framework for digital transactions. Its three types (PU-BC, PR-BC, and CO-BC) cater to specific use cases, with block header fields ensuring transaction integrity. As noted earlier, BC has transformative potential for various industries.

IV. GATE-BC: GENETIC ALGORITHM-POWERED QOS-AWARE CROSS-NETWORK TRAFFIC MANAGEMENT FRAMEWORK

We provide the details of *GATE-BC* in this section, including the network model, BC model, and modified GA to compute E2E paths.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3374213

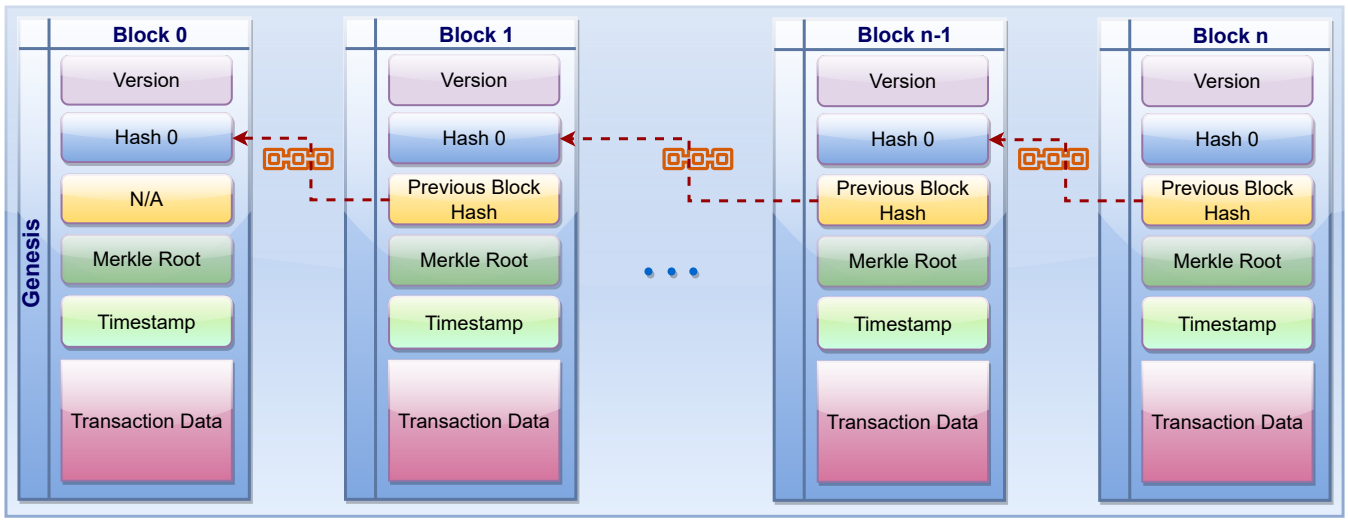


FIGURE 3: High-level representation of BC and block data structures.

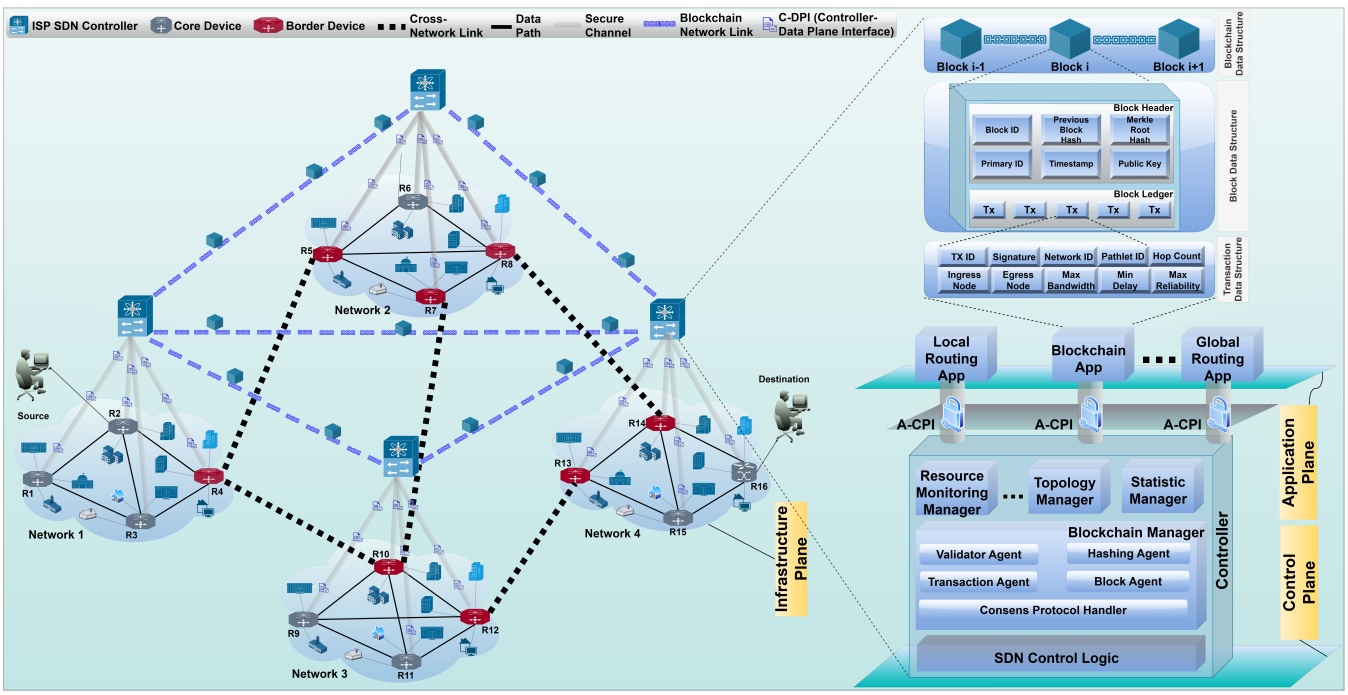


FIGURE 4: A representation of the *GATE-BC* framework with four SDN networks with layers, controllers with respective modules and applications, a BC network among controllers, block and transaction structures.

A. CROSS-NETWORK QOS-BASED E2E PATH PROBLEM (CN-QEBP)

Consider a collection of networks with network topologies denoted as $N_i(V_i, E_i)$, as shown in Fig. 4. Let V_i, E_i, and ${}^*V_i$ represent the sets of nodes (i.e., devices), links, and edge devices, respectively, in network N_i, where ${}^*V_i \subseteq V_i \, (1 \leq i \leq n)$. Let $N(V, E)$ represent an overlay network topology abstracted from the networks N_i where $V = \bigcup_{i=1}^{n} {}^*V_i$ and E are the set of all border nodes of the networks and (logical) links among those border nodes, respectively. Consider a scenario where the quantity of QoS requirements, such as latency, bandwidth, and reliability, is represented by the variable m. In this scenario, each link in the network is associated with a m-dimensional vector of link weights. These link weights consist of m non-negative QoS weights $(w_i(u, v), i = 1, ..., m$ where $u, v \in V$, and $(u, v) \in E)$. A cross-network QoS-based E2E path problem (CN-QEbP) can be formulated as a (multi) constrained path problem as follows:

Definition 1 (***Cross-Network QoS-based E2E Path Problem*** **(*CN-QEbP*)** *Consider a network $N(V, E)$ abstracted from networks $N_i(V_i, E_i)$. Each link $(u, v) \in E$ is specified by a*

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3374213

link weight vector of m additive QoS weights $w_i(u,v) \geq 0$, $i = 1, ..., m$. *Given m constraints* L_i, $i = 1, ..., m$, *the problem is to find a path P from a source node s to a destination node d (where* $s \in {}^*V_i$, $d \in {}^*V_j$ *and* ${}^*V_i \neq {}^*V_j$*) such that* $w_i(P) \stackrel{def}{=} \sum_{(u,v)\in P} w_i(u,v) \leq L_i$ *for* $i = 1, ..., m$.

Any path that can satisfy all requested QoS parameters m is called a *feasible path*. There might be more than one feasible path, which yields a feasible set. Each of such paths is a potential solution to the CN-QEbP

B. NETWORK MODEL

Fig. 4 represents the general structure of BC-powered SDN network architecture scenario in *GATE-BC*. The architecture consists of 4 SDNs with application, control, and infrastructure layers. In such an architecture, devices are either a border/edge device (brown colored) connecting to a different network's border node or a core device (grey colored) with no inter-network links. Black dashed lines represent the cross-network links among networks. The straight lines represent the data paths (links) between network devices. The transparent tunnels represent the secure channels such as TLS for Openflow messages exchanged among network controllers and infrastructure plane devices. A BC among the controllers is overlaid with blue lines representing the logical links with blocks shared between SDN controllers in the BC network.

1) BC-Enabled SDN Controllers

The SDN controllers serve as BC nodes actively participating in the BC network. The SDN controllers within the *GATE-BC* framework have been augmented with additional controller modules and applications to enhance their capabilities for BC functionalities. Controller modules are responsible for implementing essential networking capabilities commonly required by most applications. These capabilities include detecting and notifying network state events, such as topology changes, device availability, and flow updates. Additionally, controller components facilitate communication between the controller and network equipment and the collection of data corresponding to network resources.

Fig. 4 also shows the BC-enabler components and applications of SDN controllers. The capabilities of these components and applications can be summarized as follows: The Blockchain Manager (BM) and its sub-components serve as the BC-enabler primary module. The module is responsible for executing all functions related to BC. The role of the Validator Agent is to verify the integrity of incoming blocks from the other controllers in accordance with the block validation rules established in the BC. The Hashing Agent performs the hashing process on transactions and blocks. The Transaction Agent is responsible for generating transactions, while the Block Agent creates blocks within the controller, depending on the BC's specific use case and consensus protocol. The Consensus Protocol Handler is responsible for implementing the consensus algorithm to facilitate agreement among participants regarding approved transactions and blocks. The Resource Monitoring Manager (RMM) is responsible for the ongoing monitoring of various network resources, including but not limited to bandwidth, delay, and reliability.

The controller design has new applications as well. The Global Routing Agent (GRA) is responsible for executing the cross-network routing operation within the controller when a service request spanning multiple networks is received. The agent utilizes the ingress and egress data of new transactions in the BC ledger as input parameters in E2E path computation. These parameters are used to identify E2E paths for incoming connection requests and subsequently trigger the execution of a Blockchain Application (BA). The BA assumes the responsibility of transmitting and receiving blocks to and from the BC network. It also manages the connection and pathlet demands in collaboration with the BM.

2) Pathlets in Networks

A pathlet represents a path fragment as a distinct route from a starting *ingress* node to an ending *egress* node in an ISP network. It can be defined as follows:

Definition 2 (*Pathlet*) *Given a network,* N_i, *with a topology represented as* $N_i(V_i, E_i)$, *a path* $P =< v_i^1, ..., v_i^t >$ *is called a pathlet if* $(\forall v_i^j \text{ in } P) \in V_i$ *and* $v_i^1, v_i^t \in {}^*V_i$, *and an edge* e_i^l *from* v_i^j *to* v_i^{j+1}, $(\forall v_i^j, v_i^{j+1} \in P, \forall e_i^l \in E_i)$.

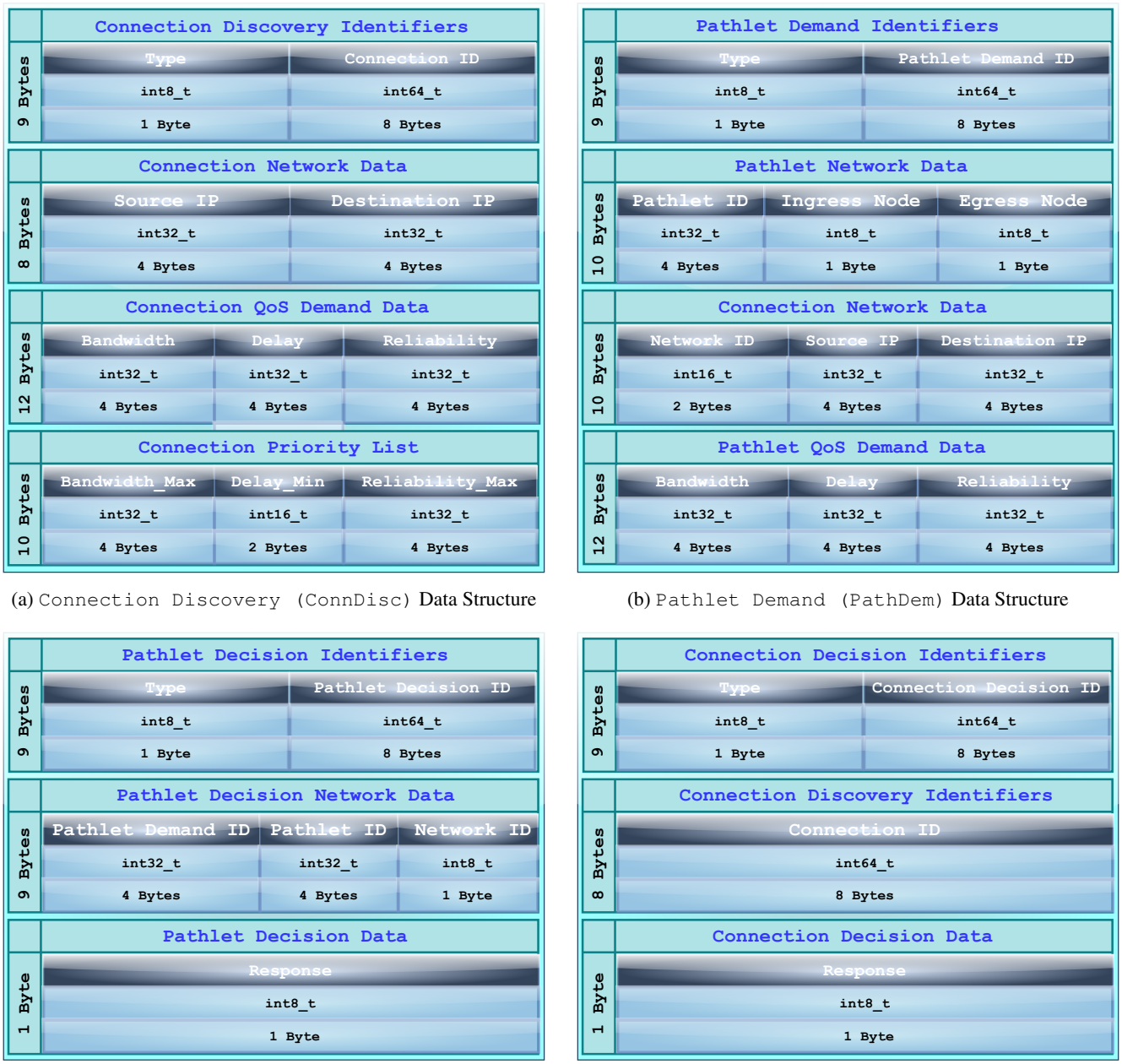
For example, the intra-network routes (i) `R5-R7`, (ii) `R5-R8-R7`, and (iii) `R5-R6-R8-R7` serve as pathlets connecting the edge nodes `R5` and `R7` inside `Network 2` as seen in Fig. 4. The network controllers compute unique pathlets for each pair of border/edge nodes inside their networks. As explained in the next section, BC nodes as SDN controllers generate transactions representing pathlets and their corresponding QoS values.

3) Cross-Network Path Setup

Network services are specific operations resulting from the harmony of multiple functions carried out by network applications within the network. In networking, applications, thus services, have a wide range of selections, such as optimization, security, monitoring, etc., particularly in programmable networks such as SDN, due to the flexibility and ability to program network devices. Traffic steering is a fundamental operational feature used in a network to facilitate intra- and inter-network connections. In the *GATE-BC* architecture, the term **connection demand** refers to the establishment of *connectivity* between users (hosts) in either the same or different networks while adhering to certain QoS factors such as bandwidth, latency, and reliability. However, such cross-network connectivity requires an intelligent and efficient path setup process to provide an E2E cross-network path guaranteeing the QoS demands of users. Fig. 5 exhibits the data structures of the four different messages created and exchanged among networks to set up an E2E QoS-based path in *GATE-BC* framework.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3374213



(a) `Connection Discovery (ConnDisc)` Data Structure

(b) `Pathlet Demand (PathDem)` Data Structure

(c) `Pathlet Decision (PathDec)` Data Structure

(d) `Connection Decision (ConnDec)` Data Structure

FIGURE 5: Data structures of the `Connection Discovery (ConnDisc)`, `Pathlet Demand (PathDem)`, `Pathlet Decision (PathDec)`, and `Connection Decision (ConnDec)` messages created and exchanged among networks to set up an E2E QoS-based path in *GATE-BC* framework.

a: Connection Discovery (ConnDisc) Message

As seen in Fig. 5a, a `Connection Discovery (ConnDisc)` message is characterized by four categories: (i) `Connection Discovery Identifiers (CDI)` to identify the `ConnDisc` message, (ii) `Connection Network Data (CND)` to define the networking-related information of the connection demand, (iii) `Connection QoS Demand Data (CQDD)` to express the QoS requirements of the connection demand, and (iv) `Connection Priority List (CPL)` to state the priority factors of the connection demand in case of availability of multiple E2E paths to choose. The list can be empty or include any number of parameters. If empty, the E2E path is selected randomly. The order of the parameters in the `CPL` states the precedence of the QoS parameters for the connection demand. A `ConnDisc` message consists of the following data fields to characterize a connection demand from a user to a Source-Network-Controller (SNC):

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3374213

- *Type*: The current message type
- *Connection ID*: A persistent connection identifier for the `ConnDisc` message
- *Source/Destination IP*s: IP addresses of the source/destination hosts for the connection
- *Bandwidth/Delay/Reliability* to indicate the requested bandwidth, delay, and reliability values for the connection
- *Bandwidth_Max/Delay_Min/Reliability_Max*: the QoS parameters priority order

`ConnDisc` message is sent by the user to the SNC.

b: Pathlet Demand (PathDem) Message

As seen in Fig. 5b, a `Pathlet Demand (PathDem)` message is characterized in four categories: (i) `Pathlet Demand Identifiers (PDI)` to identify the `PathDem` message, (ii) `Pathlet Network Data (PND)` to define the networking-related information of the pathlet demanded for the connection, (iii) `Connection Network Data (CND)` to identify source and destination points of the connection, and (iv) `Pathlet QoS Demand Data (PQDD)` to express the QoS requirements of the pathlet demanded for the connection in an Intermediary-Network-Controller (INC). The BA module at the SNC creates a `PathDem` message consisting of various data fields to characterize a pathlet demand from an SNC. A `PathDem` message includes the following data fields:

- *Type*: The current message type
- *Pathlet Demand ID*: A persistent pathlet demand identifier for the `PathDem` message
- *Pathlet ID*: The unique ID of the pathlet requested from an INC
- *Ingress/Egress Nodes*: The start/end nodes of the pathlet demanded from an INC
- *Network ID*: The unique network ID of the INC
- *Source/Destination IP*s: IP addresses of the source/destination hosts for the connection
- *Bandwidth/Delay/Reliability*: The requested bandwidth, delay, and reliability values over the pathlet demanded for the connection

The BA module at the SNC sends a `PathDem` message to the INCs over the E2E path computed for the connection.

c: Pathlet Decision (PathDec) Message

As seen in Fig. 5c, a `Pathlet Decision (PathDec)` message is characterized in three categories: (i) `Pathlet Decision Identifiers (P_Dec_I)` to identify the `PathDec` message in response to the respective `PathDem` message, (ii) `Pathlet Decision Network Data (PDND)` to define the networking-related information of the demanded pathlet from the respective INC in `PathDem` message, and (iii) `Pathlet Decision Data (PDD)` to express the decision on providing the demanded pathlet from the respective INC. A `PathDec` message consisting of the following data fields to characterize a decision from an INC for a pathlet demand from an SNC:

- *Type*: The current message type
- *Pathlet Decision ID*: A persistent pathlet decision identifier for the `PathDec` message
- *Pathlet Demand ID*: The persistent pathlet demand identifier from the respective `PathDem` message
- *Pathlet ID*: The unique ID of the demanded pathlet from an INC for the connection in the respective `PathDem` message
- *Network ID*: The unique network ID of the INC
- *Response*: The decision (Accept or Reject) of an INC on providing the demanded pathlet for the connection in response to the respective `PathDem` message

The `PathDec` messages are created by the BA module at all INCs over the E2E path computed for the connection and sent back to the SNC controller.

d: Connection Decision (ConnDec) Message

As seen in Fig. 5d, a `Connection Decision (ConnDec)` message is characterized in three categories: (i) `Connection Decision Identifiers (C_Dec_I)` to identify the `ConnDec` message in response to the respective `ConnDisc` message, (ii) `Connection Discovery Identifiers (C_Disc_I)` to identify the connection from the user to SNC in `ConnDisc` message, and (iii) `Connection Decision Data (CDD)` to express the decision on providing the demanded connection from the respective SNC. A `ConnDec` message includes the following data fields:

- *Type*: The current message type
- *Connection Decision ID*: A persistent connection decision identifier for the `ConnDec` message
- *Connection ID*: The connection identifier in the `ConnDec` message
- *Response*: The final decision (Accept or Reject) by the SNC on providing the demanded connection in response to the respective `ConnDisc` message

The `ConnDec` message is created by the BA module at the SNC for the connection and sent back to the user in response to the respective `ConnDisc` message.

C. BLOCKCHAIN MODEL

This subsection presents the BC-related concepts, such as transactions and blocks, to explain the BC model employed by network controllers in the *GATE-BC* framework.

1) Network Controllers as Blockchain Nodes

In *GATE-BC*, an SDN network controller is a BC node. It operates its BC instance and establishes a peer-to-peer (P2P) full-mesh network with other BC nodes (i.e., SDN controllers) to facilitate communication via the Internet. Therefore, this study uses a BC node and a network controller interchangeably. A BC node possesses an IP address for accessibility and a public-private key pair for secure communication. They share their public keys to establish peer relationships. BC nodes use "keep-alive" messages to control whether their

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3374213

peers are alive. A BC node terminates the peering relationship with another peer BC node if the peer is unresponsive. SDN controller (i.e., BC nodes) maintain their own instances of BC distributed ledger. The *GATE-BC* system utilizes a customized implementation of a permissioned BC, adhering to established principles of BC. In order to participate in BC operations, SDN controllers must undergo specific verification procedures. These authentication procedures aim to establish and authenticate the legitimacy of networks seeking to join the BC network. It is assumed that all networks involved in the experiments undergo a verification process prior to their integration into the BC network.

2) Transaction Model

In *GATE-BC*, transactions are generated by BC nodes based on pathlets from networks. Therefore, controllers compute different pathlets for all distinct pairs of border nodes within their respective networks. Fig. 4 also illustrates the transaction data structure for a transaction generated by an SDN controller within the *GATE-BC* framework. A transaction includes the following data fields:

- `Tx ID`: The unique ID of the transaction
- `Signature`: The digital signature of the transaction
- `Network ID`: The unique ID of the network
- `Pathlet ID`: The unique ID of a distinct pathlet in a network
- `Hop Count`: The number of links over the pathlet
- `Ingress/Egress Nodes`: The start/end nodes of the pathlet in a network
- `Max Bandwidth/Min Delay/Max Reliability`: The max bandwidth/min delay/max reliability available in a pathlet

For example, the tuple ⟨`N2_3, 0x4bc23a..., N2, R5_R7_3, 3, R5, R7, 15, 10, 0.999`⟩ is an example of a BC transaction for the pathlet `R5-R6-R8-R7` from Network 2 in Fig. 4. When a state update (such as an augmentation in bandwidth on a link or a change in topology) occurs within a pathlet, the associated network controller generates new transactions for the pathlets influenced by the updates. This process keeps the BC with up-to-date transactions of the pathlets from networks for E2E path computations by SDN controllers. Transactions are generated for each distinct pathlets between pairs of network border devices in all networks. These transactions are then distributed to network controllers to be included in the BC ledgers by the controllers. If multiple transactions exist for a pathlet in the BC ledger, the controller considers the most current transaction based on the timestamp while computing an E2E path for a connection demand.

In the *GATE-BC* framework, nodes validate a transaction using a predefined set of rules. The following are the rules used for transaction validation:

- Ensuring the transaction is digitally signed.
- Ensuring none of the fields in the transaction data structure is empty.
- Ensuring a pathlet's QoS-related fields of a transaction (min bandwidth and max delay) are positive values in the transaction data structure.
- Ensuring a valid network ID is available in the `Network ID` field of the transaction data structure.
- Ensuring the transaction's ingress and egress node IDs belong to the network creating the transaction. If they do not match with IDs of border nodes previously shared by transaction-creator AS, the transaction will be discarded.

The aforementioned rules authenticate and verify different elements of a transaction.

3) Block Model

Transactions are made up of blocks. Each block is validated once its structure are verified through digital signatures. A hash pointer in a block connects it to the previous block to enhance the integrity and tampering more difficult, if not impossible. Fig. 4 presents the data structure of a block, and BC used in *GATE-BC* framework. The framework is composed of two parts: block header and transactions. The block header consists of the following fields:

- `Block ID`: A unique ID (a sequential number) of the block to distinguish the block(s)
- `Previous Block Hash`: The previous block's hash in the BC to ensure the consistency
- `Merkle Root Hash`: A data structure with a singular root hash
- `Primary ID`: The unique primary BC node ID in a block generation interval (epoch)
- `Timestamp`: The time when the block is published
- `Public Key`: The public key of the SDN controller (i.e., the BC node)

In *GATE-BC*, the leading node within the block of the current epoch assumes responsibility for creating a fresh block. The primary function transitions from one node to the next with each new epoch. The process of generating blocks within a BC system entails several sequential steps. Initially, all recently submitted transactions are collected from the transaction pool. These transactions are subsequently assessed based on the validation rules, where those deemed invalid are rejected. The primary node plays a pivotal role and oversees adherence to block generation constraints, which may include time limits, where applicable. Following the successful completion of validation, a block containing valid transactions is produced. The primary node signs this block using its private key, ensuring the authenticity and integrity of the block. Finally, the newly created block is propagated to the other nodes within the BC network, to trigger the consensus algorithm and synchronization of the distributed ledger, if needed. The primary focus of block validation criteria is on ensuring syntactic and semantic correctness, which are independent of the transaction data's content. The consensus protocol rules are implemented to validate the integrity and correctness of the blocks. Adding a block in the ledger is dependent upon the verification of the following rules:

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3374213

- confirm the authenticity of the block's source node,
- validate the accurate generation and signing of the block,
- evaluate compliance with transaction rules,
- insert the block if all preceding requirements are satisfied; otherwise, reject it.

If a node keeps producing void blocks, it may be banned or removed from the list of peer nodes. It is worth noting, however, that this particular mechanism is beyond our focus this study.

D. A MODIFIED GENETIC ALGORITHM FOR TRAFFIC STEERING

This subsection presents the modified Genetic Algorithm (GA) designed for QoS-aware cross-network traffic engineering within the *GATE-BC* framework. Additionally, we introduce sub-procedures, detailed in Algorithms 1 through 5, to implement the processes involved in cross-network QoS-aware E2E routing for the *CN-QEbP* problem outlined in Definition 1. We leverage the GA to reduce the computational overhead in the *CN-QEbP* problem while steering QoS-aware cross-network traffic. The algorithm computes a cross-network E2E path using pathlet information held in transactions from BC to satisfy a `ConnDisc` message while reducing network resource usage and satisfying QoS constraints.

Algorithm 1 GATE-BC: Genetic Algorithm-powered QoS-aware Cross-Network Traffic Engineering Algorithm

```
1:  procedure GATE-BC(G^O, ConnDisc, BC, iteration, populationSize, elitizmRate, randomRate)
2:      counter ← 0
3:      Population, nextGeneration ← ∅
4:      Population ← createPopulation(populationSize, G^O, ConnDisc)
5:      Population ← findPath(Population, G^O, ConnDisc, BC)
6:      while counter < iteration do
7:          nextGeneration ← ELITISM_SELECTION(Population, elitizmRate)
8:          Population ← CROSSOVER(Population, nextGeneration)
9:          Population ← MUTATION(Population, randomRate)
10:         Population ← FITNESS(Population, G^O, ConnDisc)
11:         counter + +
12:     return Population[0]
```

In Algorithm 1, the initial step involves the utilization of an overlay network (G^O) which corresponds to a graph $N(V,E)$ as described in Subsection IV-A. The algorithm also requires a `Connection Discovery (ConnDisc)` message, BC transactions (*BC*), a specified number of iterations (*iteration*), a population size denoting the total number of individuals in the population to establish the search space (*populationSize*), the number of elite chromosomes (*elitizmRate*), and the mutation rate (*randomRate*). The algorithm commences by initializing the population and generating the next generation of individuals. In line 4, the algorithm generates initial chromosomes randomly using the QoS metrics in `ConnDisc`. These chromosomes are created while obtaining connection information from G^O and are stored in *Population* along with their corresponding fitness values, which are set to ∞. The population in line 5 encompasses pairs comprising chromosomes and their corresponding fitness values, which are obtained through the *findPath*() technique. To determine the fitness value of each chromosome, a random priority value between 0 and 1 is assigned to each gene inside the chromosome. The Source-Network (Src-Net) and Destination-Network (Dst-Net) are both defined as 1 within each chromosome. The method aims to identify a path originating from the source node and terminating at the destination node by assigning the highest random numbers to each gene inside a chromosome. The *Population* consists of two sets, $C = \{c_1, c_2, ..., c_p\}$ and $F = \{f_1, f_2, ..., f_p\}$, to represent the chromosomes c_i and their corresponding fitness values f_i, where $i \leq p$. Here, p denotes the size of the population (i.e., *populationSize*). Next, the algorithm proceeds to calculate the fitness value of each chromosome within the *Population* in order to identify the most optimal chromosome with the lowest fitness value (lines $6-11$).

The Algorithm 1 calls the `ELITISM_SELECTION`() procedure from Algorithm 3 to determine a new cohort of chromosomes, which is a subset of the current population (line 7). Next, the `CROSSOVER`() procedure described in Algorithm 4 is invoked to generate a new population, which consists of chromosomes randomly selected from the new generation (line 8). Subsequently, the algorithm invokes the `MUTATION`() method outlined in Algorithm 5 in order to provide different possibilities for new chromosomes (line 9). After the population is obtained via Algorithm 5, the GA proceeds to compute and arrange the present chromosomes in the population using the `FITNESS`() method (line 10). For the populations obtained, the algorithm eventually proceeds to go through the aforementioned phases per the fixed number of iterations stated at the beginning of the process. At the end of the predetermined number of iterations, Algorithm 1 is finished by picking the first element (i.e., the first chromosome with the lowest fitness value) from the population (line 12). This particular chromosome is chosen because it demonstrates the least network resources utilized for the `ConnDisc` message while achieving the specified QoS parameters.

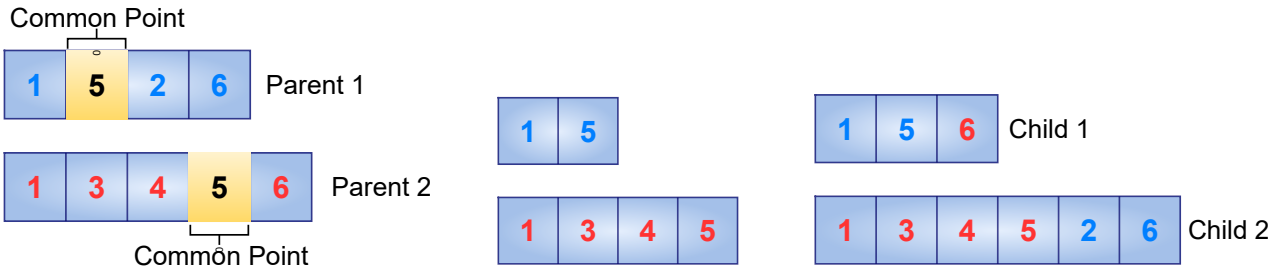
1) Fitness Value Calculation

`FITNESS`() procedure of the Algorithm 2 specifies the complete network resource usage of the request by using the random priority value of each gene to connect the Src-Net with the Dst-Net. In lines $2-14$ of Algorithm 2, the procedure identifies all feasible pathlets that meet the specified QoS metrics. The *findPath*() procedure evaluates the utilization of network resources, referred to as the fitness value, for every chromosome in the population to minimize resource consumption. The `FITNESS`() method assigns the fitness value and substrate linkages employed for accessing the Dst-Net from the Src-Net as F (line 14). Upon evaluating the fitness values of the chromosomes, the algorithm proceeds to arrange all chromosomes within the population in ascending order based on their respective fitness values (line 15). Subsequently, the algorithm returns the entire population (line 16).

As an example based on Fig. 6, where it is desired to go from node number 1 to node number 6, assume that

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3374213



(a) Step 1 (b) Step 2 (c) Step 3

FIGURE 6: Example of Crossover Operation

Algorithm 2 FITNESS(): Calculate Fitness Value of Each Chromosome

```
1:  procedure Fitness(Population, G^O, ConnDisc, BC)
2:      for each chromosome in Population do
3:          pathList ← ∅, fitnessValue ← 0
4:          pathList ← findPath(chromosome, G^O, ConnDisc, BC)
5:          if pathList == ∅ then
6:              fitnessValue ← ∞
7:              break
8:          minFitness ← ∞
9:          for each path ∈ pathList do
10:             fitness ← β * Σ_{∀(i,i+1)∈path} w(i, i+1)
11:             if fitness < minFitness then
12:                 minFitness ← fitness
13:         fitnessValue ← fitnessValue + minFitness
14:         F of chromosome ← fitnessValue
15:     Population ← sort(Population)
16:     return Population
```

the $[1, 0.509, 0.341, 0.294, 0.713, 1]$ numbers are randomly generated as priorities for each node in a chromosome. It is aimed to reach the target by starting from the highest priority from the source to be looked at first. In other words, a path is created by going from node 1 (priority 1) to node 5 (priority 0.713), then from node 5 to node 2 (priority 0.509), and finally from node 2 to node 6 (priority 1).

2) Elite Chromosome Selection

The ELITISM_SELECTION() method of Algorithm 3 begins by accepting the currently selected population (*Population*) and a constant value (*elitismCount*) as input. In lines $4-6$ of Algorithm 3, the ELITISM_SELECTION() procedure is used to choose the initial chromosomes from the population, up to a specified count denoted by *elitismCount*. These selected chromosomes are then utilized to create a new population, referred to as *nextGeneration*, which is then returned as the output in line 7.

Algorithm 3 ELITISM_SELECTION(): Select the Elite Chromosomes

```
1:  procedure Elitism_Selection(Population, elitizmCount)
2:      nextGeneration ← ∅
3:      counter ← 0
4:      while counter < elitizmCount do
5:          nextGeneration ← nextGeneration ∪ Population[counter]
6:          counter + +
7:      return nextGeneration
```

3) Crossover Operation

The CROSSOVER() procedure is used in Algorithm 4 to generate additional chromosomes, as illustrated in Fig. 6. The algorithm identifies a shared spot among the randomly chosen chromosomes to be retained to create a new chromosome, as seen in Fig. 6a. In the example, the first gene denotes the Src-Net, while the last gene corresponds to the Dst-Net. After putting the genes up to the common point in the chromosomes in Fig. 6b, the CROSSOVER() technique proceeds to place genes located after the shared locus of the chosen chromosomes into a newly created offspring, according to the order of the received chromosomes. Furthermore, selected genes of Parent 2 in Fig. 6a (i.e., genes A, C, B) are mapped into Child 1 in Fig. 6c in the order of C, B, D. This is because the gene A is in the selected genes in the first step and the method puts the following genes into the current child after the common point of chromosomes. Algorithm 4 produces two new chromosomes (line 6) and stores them into *crossoverNextGeneration* (line 8).

Algorithm 4 CROSSOVER(): Operators for Genetic Algorithm - Crossover

```
1:  procedure Crossover(Population, nextGeneration)
2:      crossoverNextGeneration ← nextGeneration
3:      while size(crossoverNextGeneration) < size(Population) do
4:          parent1 ← randomly select a chromosome from nextGeneration
5:          parent2 ← randomly select a chromosome from nextGeneration
6:          (child1, child2) ← findCommonPoint(parent1, parent2)
7:          if there is a common point then
8:              crossoverNextGeneration ← crossoverNextGeneration ∪ {(child1, child2)}
9:      return crossoverNextGeneration
```

The algorithm iterates through this procedure until the size of the *crossoverNextGeneration* is equal to the size of the population (line 3). It then returns the latest generation (line 9), where it is assigned to the variable *Population* in line 8 of the proposed Algorithm 1.

4) Gene Mutation Process

The Algorithm 5 algorithm begins by accepting the present *Population* and *randomRate* as input parameters. These parameters are used to decide whether the present chromosome will undergo mutation or modification. The MUTATION()

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3374213

method is responsible for selecting a chromosome in the given algorithm.

Algorithm 5 `MUTATION()`: Operators for Genetic Algorithm - Mutation

1: **procedure** Mutation(*Population*, *randomRate*)
2: *counter* $\leftarrow 0$
3: **while** *counter* $<$ *size*(*Population*) **do**
4: *rnd* $\leftarrow$ *random.range*$(0, 1)$
5: **if** *rnd* $<$ *randomRate* **then**
6: *selectedChromosome* $\leftarrow$ select chromosome in *Population*[*counter*]
7: *firstNode* $\leftarrow$ randomly select an available node from *selectedChromosome*, save the index of the node as *firstIndex* and set its value as empty in the chromosome
8: *secondNode* $\leftarrow$ randomly select an available node from *selectedChromosome*, save the index of node as *secondIndex* and set its value as empty in the chromosome
9: *Population*[*counter*] $\leftarrow$ *swap*(*firstNode*, *secondNode*, *firstIndex*, *secondIndex*)
10: *counter* $++$
11: **return** *Population*

It proceeds by identifying two nodes, namely *firstNode* and *secondNode*, inside the overlay network to be swapped (lines $5-8$). Finally, the changed chromosome, which is the presently chosen chromosome, is saved in the population at line 9. Upon performing these procedures for every chromosome within the population, the resulting population is returned at line 11.

E. OPERATIONAL PROCESSES AND MANAGEMENT IN GATE-BC FRAMEWORK

This subsection provides a concise explanation and a demonstrative example of the operation of the *GATE-BC* architecture, as shown in Fig. 7.

No E2E Path Case: When a user requests a cross-network connection with QoS requirements to another user from a different network, the request is submitted to the SNC in the form of a `ConnDisc` message, as shown in step 1.1 in Fig. 7. Then, the SNC starts path computation using the GA methods from Algorithms 1 through 5 for an E2E path from Src-Net to Dst-Net, as depicted in step 1.2 in Fig. 7. This computation uses transactions from the BC representing the pathlets with the QoS parameters and priority specified in the `ConnDisc` message. The E2E path consists of pathlets given in the Definition 2, which are segments that originate at a border node in the Src-Net and terminate at a border node in the Dst-Net while traversing an overlay network. The SNC uses various transaction data, including `Network ID`, `Ingress Node`, `Egress Node`, `Max Bandwidth`, `Min Delay`, and `Max Reliability` to compute the E2E path. When the SNC cannot find an E2E path that meets the given requirements indicated in the `ConnDisc` message (step 1.3 in Fig. 7), it sends a `ConnDec` message back to the user as shown in step 4 in Fig. 7. This `ConnDec` message includes the Reject response to the user/client to state that the connection demand cannot be provided.

E2E Path Available Case: When an E2E path is found satisfying the QoS requirements of the service request (step 1.4

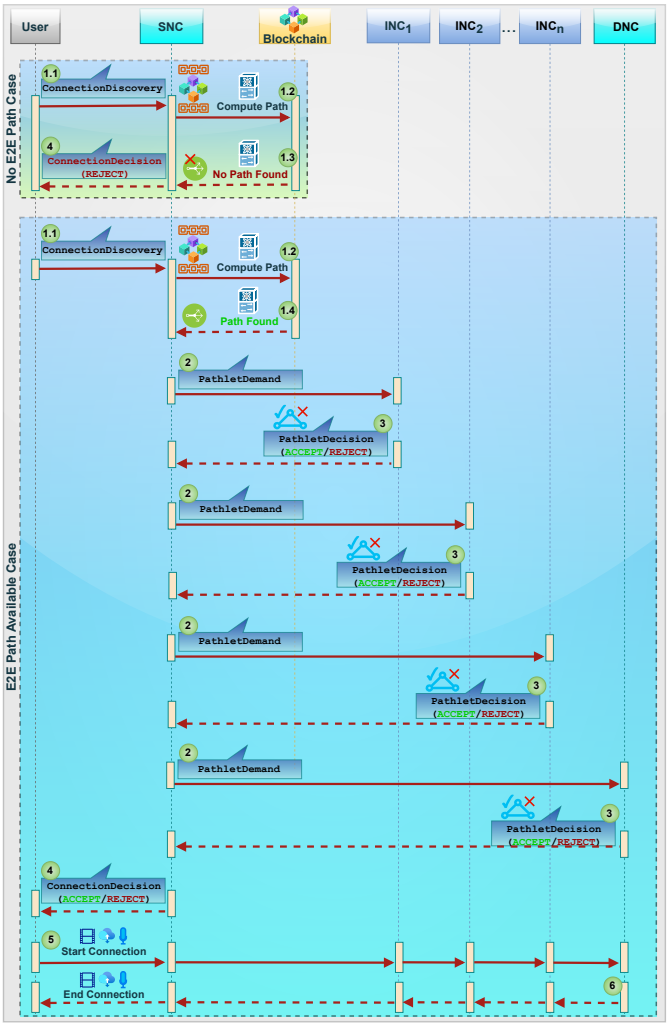


FIGURE 7: The processes and actors involved in the operation of *GATE-BC* framework in handling a connection request scenario.

in Fig. 7), the SNC asks each INC and Destination-Network-Controller (DNC) that has a pathlet over the E2E path (step 2 in Fig. 7) whether they can provide the requested QoS values over the corresponding pathlet by sending `PathDem` messages. This inquiry aims to confirm that the corresponding pathlet can still provide QoS values advertised in the respective transactions. Because transaction updates on the BC are not real-time compared to the state updates on the pathlets of networks, a pathlet's QoS values may be different than the advertised ones due to various reasons such as intra-network traffic and/or traversing inter-network traffic. After receiving the `PathDem` messages from the SNC, the respective INCs and DNC send `PathDec` messages (including Accept or Reject response) back to the SNC to state whether they can provide the corresponding pathlet with requested QoS parameters (step 3 in Fig. 7). If all respective INCs and DNC send an Accept response in `PathDec` messages to the SNC, the SNC sends an `ConnDec` message including an Accept response to the user (step 4 in Fig. 7) to state that connection demand

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3374213

can be satisfied, therefore, can start (step 5 in Fig. 7) in the network. The corresponding network controllers (i.e., SNC, INCs, and DNC) handle the underlying SDN networking-related arrangements. These tasks include adding flow entries in flow tables of the corresponding data plane devices and reserving network resources for the promised pathlet according to the data in the `ConnDisc` and `PathDem` messages if they accept providing the service/pathlet request. Therefore, a service flow is not investigated by any INC and DNC over the E2E path again when it enters the network. Also, the partial paths from the source-user to the start border node of the E2E path in the Src-Net and from a border node to the destination-user in Dst-Net are handled by the SNC and DNC, respectively, because those partial paths are subject to intra-domain routing. Therefore, the E2E path is found from a border node of Src-Net to a border node of Dst-Net. After the connection expires, it ends in all INCs over the E2E path and DNC (step 6in Fig. 7).

On the contrary, if any of the INCs and DNC over the E2E path sends a Reject response in `PathDec` messages (step 3) to the SNC, the SNC starts computing another E2E path with the same properties and conditions using its BC ledger and GA procedures as explained earlier.

V. EVALUATION

This section evaluates and compares the performance of the *GATE-BC* framework against three QoS-based routing approaches in SDNs [7], [13]: (i) QoSChain (*QC*), a BC-enabled E2E multi-network routing framework introduced in our previous work [14], (ii) the Hierarchical Routing Approach (*HRA*), proposed in our earlier research [15], and (iii) the Distributed Routing Approach (*DRA*), a strategy reflecting the present operational manner of BGP-based networks. Also, *GATE-BC* employs three different feasible path selection strategies based on the QoS parameters to assign an E2E path for service requests. The first strategy, *GATE-BC_BW*, selects the E2E path providing the highest bandwidth on the respective QoS parameter specified in the priority field of the request. The second strategy, *GATE-BC_D*, selects the path with minimum delay among all feasible paths. The third strategy, *GATE-BC_R*, selects the path with the highest reliability among all feasible paths computed by the controller.

The study employs E2E Path Setup Time (PST), Network Message Overhead (NMO), Request Acceptance Ratio (RAR), Network Bandwidth Consumption (NBC), Average Path Length (APL), and Average Network Length (ANL) metrics for the performance analysis. The study uses NSFNET (14-Network), US Backbone (24-Network), Random-14 (14-Network), and Random-24 (24-Network) topologies at the network level to conduct a topology-wise sensitivity analysis on the performance of the *GATE-BC* and the other approaches. We use the same randomly generated topologies with 0.5 connectivity degrees in the intra-network settings of all topologies as the number of switches per network varies from 6 to 10 in the simulations. The bandwidth amount in the connection requests varies from 5 to 15 units. The delay and reliability thresholds for the feasible paths are over 100 *ms* and 0.8, respectively. While the intra-network link capacities of the networks in the topologies are given randomly between 5 units and 45 units, the inter-network link capacities are given 1000 units. The total number of connection requests sent is 800 in each simulation scenario. The results in scenarios are the average of the results from accepted connection requests in the simulations.

A. PATH SETUP TIME (PST)

Path Setup Time (PST) metric refers to the time to install corresponding flow rule entries enabling the service/flow request in flow-tables of SDN switches over the E2E path. The metric incorporates several factors, such as propagation delay, processing time, and path computation time, so it serves as a complete measure of network performance.

Fig. 8 shows PST (in *ms*) of *GATE-BC_BW*, *GATE-BC_D*, *GATE-BC_R*, *QC*, *HRA*, and *DRA* in different topology scenarios. The *QC* framework gives the least PST results among all frameworks in all topology settings because it chooses the first available feasible path when computing E2E path. *GATE-BC* strategies outperform the non-BC *HRA* and *DRA* frameworks in all topologies and switch-per-network settings because the *HRA* and *DRA* frameworks handle more intra-network and inter-network communication time among SDN controllers while computing E2E path for connection requests. As the number of switches per network increases in each topology scenario in Figs. 8a through 8d, the PST results also increase due to path computation times, controller communication times, and device flow rule installation times depending on the network scale. Among all frameworks, BC-employed frameworks (*GATE-BC* and *QC*) give better results compared to non-BC frameworks (*HRA* and *DRA*) because BC makes the pathlet information available for controllers use in distributed shared ledgers. Therefore, such availability reduces the PST in BC-employed frameworks.

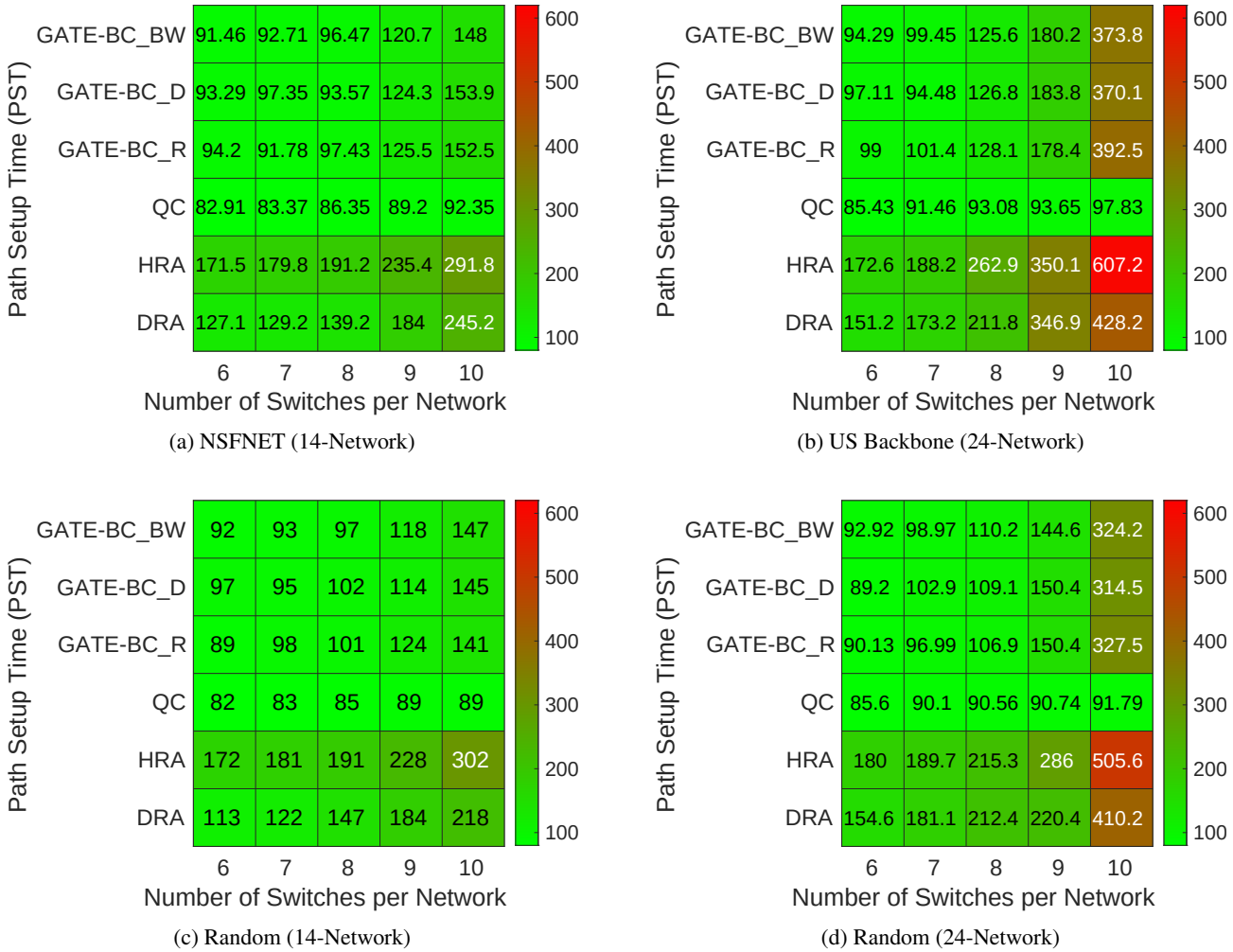
B. NETWORK MESSAGE OVERHEAD (NMO)

As the network grows, SDN controllers must handle more connection requests and messages exchanged among data layer devices and controllers to establish QoS-based E2E paths. Limited computational resources in controllers can create bottlenecks while processing packets. Therefore, minimizing the number of messages exchanged and processed by controllers, referred to as the Network Message Overhead (NMO) metric, is essential while setting up E2E paths because it is a crucial performance indicator for evaluating routing frameworks in SDN networks. Moreover, it directly affects the overall scalability of the network.

Table 2 presents the NMO results of *GATE-BC_BW*, *GATE-BC_D*, *GATE-BC_R*, *QC*, *HRA*, and *DRA* in different topology scenarios. The *HRA* framework reveals 2-2.5 times greater NMO than the others across all topologies, depending on the network size. This is because the *HRA* uses hierarchical functioning and processes among the domain controllers and the Broker for intra- and inter-network state information.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3374213



(a) NSFNET (14-Network)
(b) US Backbone (24-Network)
(c) Random (14-Network)
(d) Random (24-Network)

FIGURE 8: Path Setup Time (PST), in *ms*, of *GATE-BC_BW*, *GATE-BC_D*, *GATE-BC_R*, *QC*, *HRA*, and *DRA* in different topology scenarios.

TABLE 2: Network Message Overhead (NMO) of *GATE-BC_BW*, *GATE-BC_D*, *GATE-BC_R*, *QC*, *HRA*, and *DRA* in different topology scenarios.

	Number of Switches per Network				
	6	**7**	**8**	**9**	**10**
GATE-BC_BW	22.79	22.42	27.34	27.94	30.81
GATE-BC_D	23.24	23.09	25.97	29.33	31.43
GATE-BC_R	23.47	22.86	27.07	29.33	30.51
QC	23.82	23.95	26.78	27.36	29.57
HRA	46.31	49.57	52.16	53.53	57.61
DRA	19.48	19.31	22.17	23.31	26.48

(a) NSFNET (14-Network)

	Number of Switches per Network				
	6	**7**	**8**	**9**	**10**
GATE-BC_BW	29.01	32.14	32.68	33.86	39.53
GATE-BC_D	27.85	31.50	33.01	33.52	41.51
GATE-BC_R	27.56	33.43	31.37	33.18	37.55
QC	28.71	34.11	35.91	37.85	41.53
HRA	65.60	73.50	82.18	86.83	95.58
DRA	26.05	26.10	28.93	29.52	35.18

(b) US Backbone (24-Network)

	Number of Switches per Network				
	6	**7**	**8**	**9**	**10**
GATE-BC_BW	20.90	24.11	24.74	27.10	28.30
GATE-BC_D	20.07	23.63	23.75	28.18	27.73
GATE-BC_R	20.28	23.63	25.23	28.18	29.71
QC	20.69	23.86	24.99	28.20	31.28
HRA	45.02	50.52	51.88	57.77	59.68
DRA	21.84	21.98	26.37	27.18	27.41

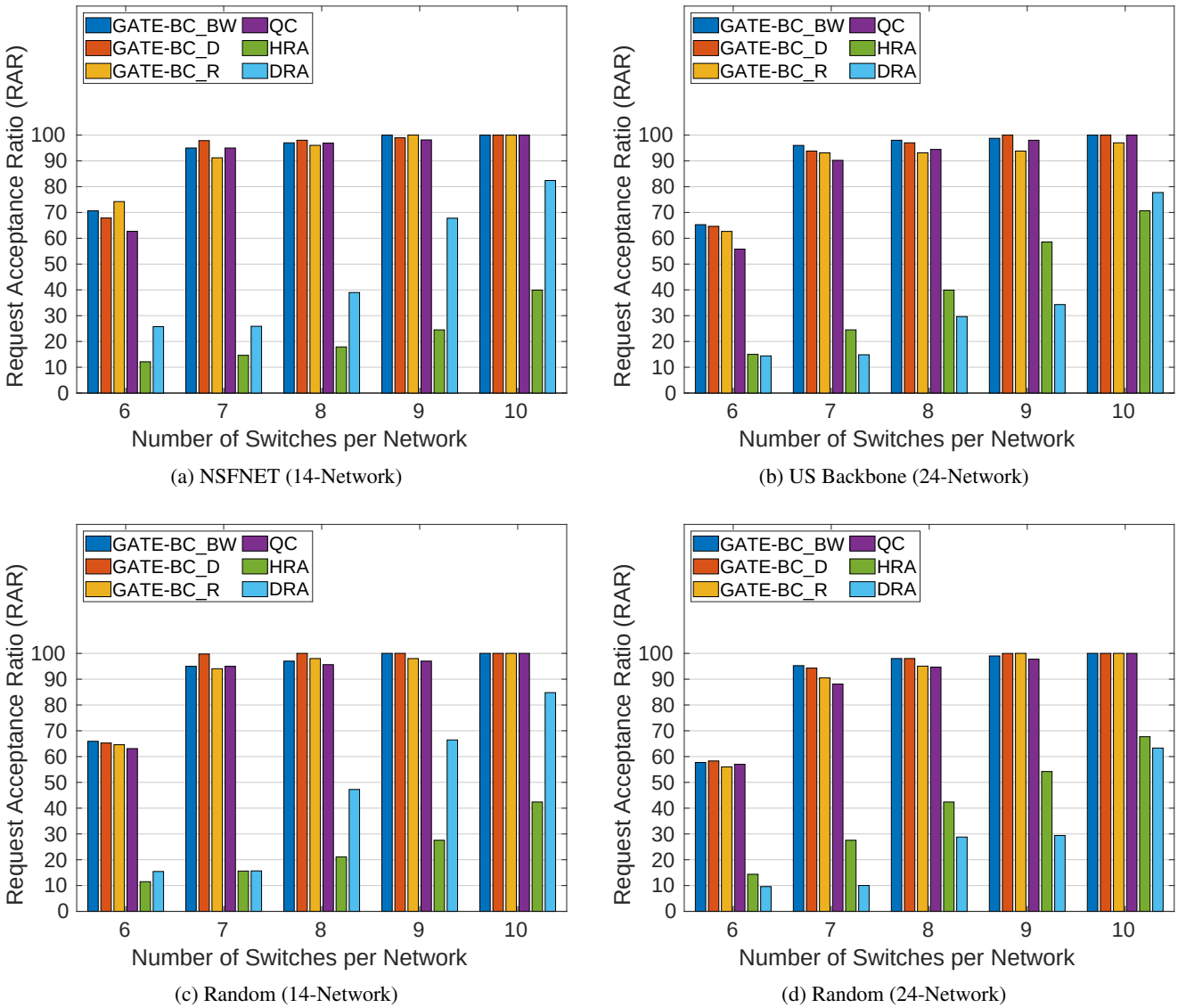
(c) Random (14-Network)

	Number of Switches per Network				
	6	**7**	**8**	**9**	**10**
GATE-BC_BW	26.21	32.44	33.47	35.26	36.28
GATE-BC_D	26.73	31.79	32.46	35.96	35.55
GATE-BC_R	27.52	31.47	32.13	36.67	35.92
QC	26.21	32.78	34.85	36.66	37.44
HRA	56.05	78.82	83.03	95.90	106.54
DRA	24.34	26.17	27.89	28.37	36.17

(d) Random (24-Network)

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3374213



(a) NSFNET (14-Network)

(b) US Backbone (24-Network)

(c) Random (14-Network)

(d) Random (24-Network)

FIGURE 9: Request Acceptance Ratio (RAR) of *GATE-BC_BW*, *GATE-BC_D*, *GATE-BC_R*, *QC*, *HRA*, and *DRA* in different topology scenarios.

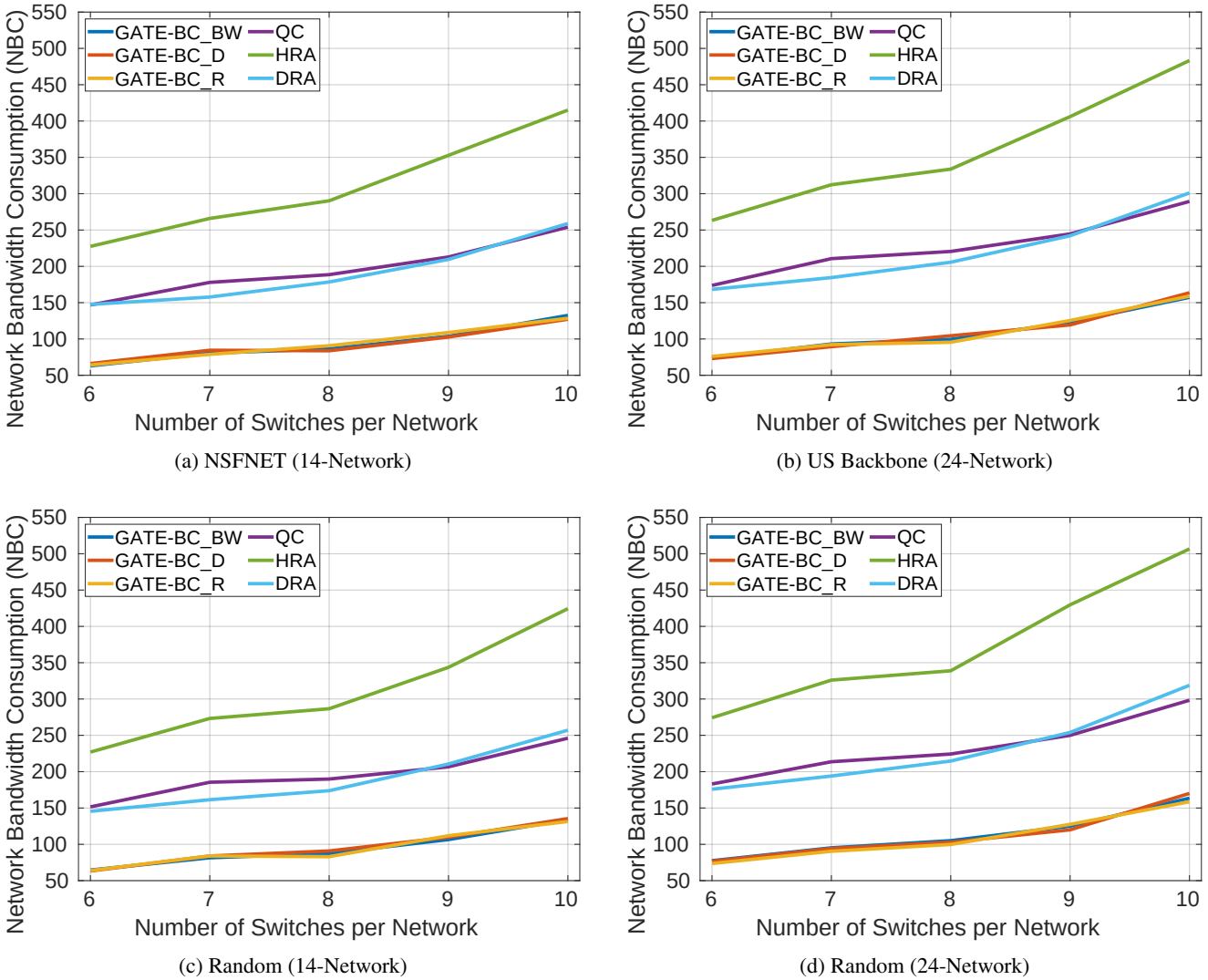
As the number of switches per network increases, the BC-employed *GATE-BC* strategies and *QC* exhibit comparable NMO values to the non-BC-based *DRA* framework. However, it should be noted that the E2E path for all frameworks may not be identical for the same request, resulting in potentially different values for the entire number of switches and controllers over an E2E path. Moreover, a network is able to determine a feasible path by only querying its default BGP next-hop network towards the destination in the *DRA*, namely by using the shortest path at the inter-network level. Hence, it is unnecessary for a network to engage in communication and message exchange with any network other than its default next-hop network in order to reach the intended destination. This results in reduced message communication and exchange in *DRA*. Hence, it is important to highlight that while the NMO of *GATE-BC* is comparable to that of *DRA*, the key distinction lies in the fact that *DRA* works on the shortest BGP route at the inter-network level. Nevertheless, *GATE-BC* demonstrates a substantial reduction in the PST, as seen in Fig. 8. The *GATE-BC* and *QC* frameworks show very similar NMO performances thanks to the employment of the BC. Also, as the number of switches per network and the number of networks increase, the NMO results increase for all frameworks, as shown in Tables 2a through 2d because more network devices and controllers are involved in both E2E path computations and SDN-based flow rule installations for the computed paths.

C. REQUEST ACCEPTANCE RATIO (RAR)

The Request Acceptance Ratio (RAR) metric is a critical performance measure in SDNs. The RAR metric quantifies the proportion of incoming connection requests successfully

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3374213



(a) NSFNET (14-Network)

(b) US Backbone (24-Network)

(c) Random (14-Network)

(d) Random (24-Network)

FIGURE 10: Network Bandwidth Consumption (NBC) of *GATE-BC_BW*, *GATE-BC_D*, *GATE-BC_R*, *QC*, *HRA*, and *DRA* in different topology scenarios.

accommodated by the routing framework in the network. It directly reflects a routing framework's ability to meet service-level agreements (SLAs) and deliver requested services with the desired quality parameters by efficiently managing network resources. A high RAR measure signifies network efficiency and effective handling of service requests, while a low RAR indicates potential service degradation or inefficiency.

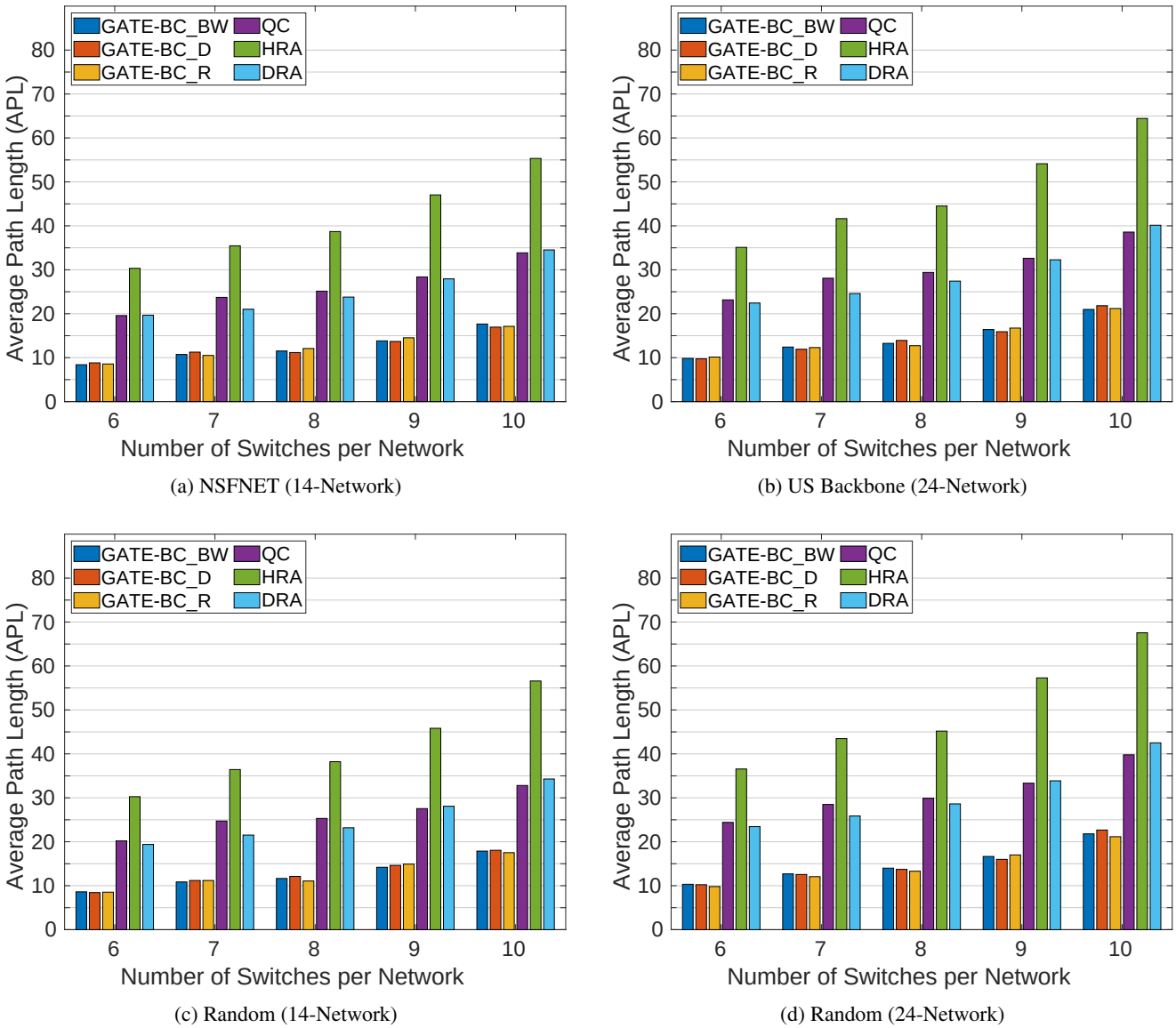
Fig. 9 presents the RAR values of *GATE-BC_BW*, *GATE-BC_D*, *GATE-BC_R*, *QC*, *HRA*, and *DRA* frameworks in different topology scenarios. The results reveal that the choice of network topology substantially impacts the RAR performances of the frameworks. RAR performances of the *GATE-BC* frameworks consistently demonstrate superior performance in all topologies in Figs. 9a through 9d. due to their capabilities to exploit multiple alternative E2E paths. This showcases their robustness in accommodating requests across different topologies and switch configurations. In addition, the BC-employed frameworks (*GATE-BC* and *QC*) present far superior performance than non-BC-employed ones (*HRA* and *DRA*) because they take advantage of BC transactions to find QoS-based E2E paths over networks. Moreover, as the number of switches per network increases, the RAR performances of all frameworks improve, as shown in Figs. 9a through 9d because more network resources and alternative E2E paths become available for the requests. These findings make it clear how important network topology and framework choices are for network RAR performance.

D. NETWORK BANDWIDTH CONSUMPTION (NBC)

The Network Bandwidth Consumption (NBC) metric quantifies the amount of network bandwidth needed by the E2E paths computed for connection requests by the routing frameworks. NBC metric is vital because it measures the efficiency of the frameworks in managing network resources. Lower NBC values indicate efficient bandwidth utilization, while higher values may suggest suboptimal resource allocation.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3374213



(a) NSFNET (14-Network)

(b) US Backbone (24-Network)

(c) Random (14-Network)

(d) Random (24-Network)

FIGURE 11: Average Path Length (APL) of *GATE-BC_BW*, *GATE-BC_D*, *GATE-BC_R*, *QC*, *HRA*, and *DRA* in different topology scenarios.

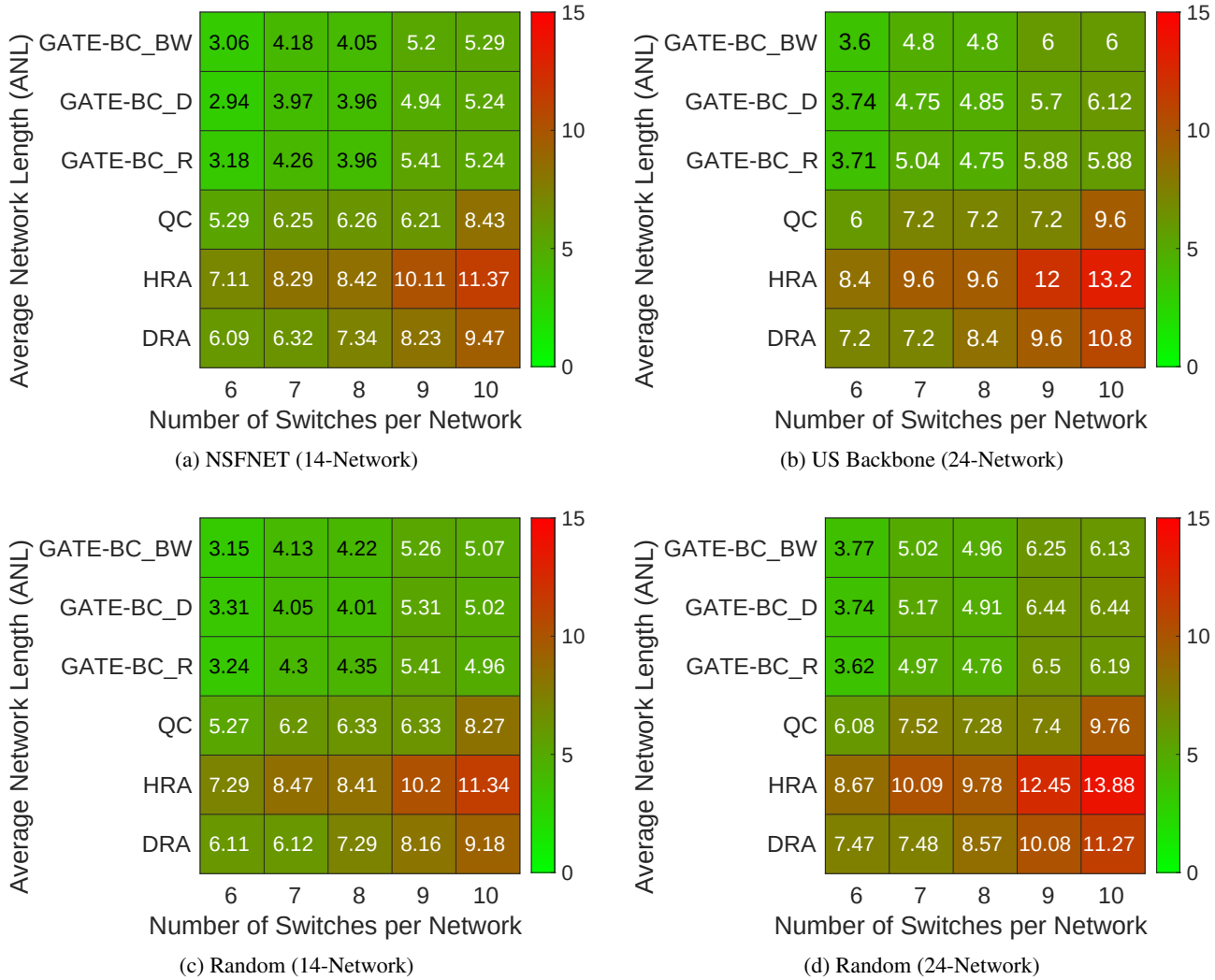
Fig. 10 exhibits the NBC results of *GATE-BC_BW*, *GATE-BC_D*, *GATE-BC_R*, *QC*, *HRA*, and *DRA* in different topology scenarios. The GA and BC-powered *GATE-BC* frameworks outperform the others in terms of NBC. This is primarily due to their ability to utilize a greater number of alternative paths with shorter lengths dispersed across the networks. This, in turn, demonstrates their efficiency in allocating bandwidth effectively. Notably, all *GATE-BC* frameworks consistently demonstrate similar bandwidth consumption due to the nature of the GA. The results concerning the number of switches per network and topology sensitivity reveal consistent patterns across the topologies. With the increasing number of switches, there is a clear escalation in NBC performance of all frameworks, indicating heightened resource utilization, as shown in Figs. 10a through 10d. This sensitivity to topology changes holds across all examined topologies. Furthermore, the results emphasize the significance of incorporating optimization algorithms such as GA and BC into SDN routing frameworks, highlighting their potential to enhance network performance and resource management.

E. AVERAGE PATH LENGTH (APL)

The Average Path Length (APL) metric refers to the average number of data plane devices over the E2E paths from source to destination computed by the frameworks for connection requests. It is an important metric as it quantifies the efficiency of routing frameworks and network topology in terms of minimizing data transmission delays and network resource utilization. A lower APL indicates more efficient routing, reduced latency, and potentially lower network congestion,

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3374213



(a) NSFNET (14-Network)

(b) US Backbone (24-Network)

(c) Random (14-Network)

(d) Random (24-Network)

FIGURE 12: Average Network Length (ANL) of *GATE-BC_BW*, *GATE-BC_D*, *GATE-BC_R*, *QC*, *HRA*, and *DRA* in different topology scenarios.

which are critical factors for ensuring the QoS in SDNs.

Fig. 11 presents APL measures of *GATE-BC_BW*, *GATE-BC_D*, *GATE-BC_R*, *QC*, *HRA*, and *DRA* in different topology and switch-per-network scenarios. In general, *GATE-BC* frameworks exhibit lower APL values compared to the others because they select the min-hop E2E paths and try to find the min-cost chromosome among the chromosomes in the population using GA. This indicates their ability to optimize path selection for reduced latency and efficient data transmission. This trend holds consistently across all topologies and switch-per-network variations in Figs. 11a through 11d. BC notably impacts the APL because network controllers can readily use pathlet state information when computing E2E paths for connection requests from BC's shared ledgers. Additionally, the results reveal that network topology impacts APL with more complex and random topologies often resulting in slightly higher APL values. Overall, these APL findings underscore the effectiveness of GA-powered frameworks in enhancing routing efficiency within SDNs.

F. AVERAGE NETWORK LENGTH (ANL)

The Average Network Length (ANL) metric refers to the average number of networks over the E2E paths from source to destination computed by the frameworks for connection requests. It is an important metric because it quantifies the efficiency of routing algorithms and network topology. A lower ANL value indicates more efficient inter-network routing, which is crucial for optimizing network performance and resource utilization.

Fig. 12 reflects ANL measures of *GATE-BC_BW*, *GATE-BC_D*, *GATE-BC_R*, *QC*, *HRA*, and *DRA* in different topology and switch-per-network scenarios. Analyzing the results, it is observed that there are significant variations in ANL across the frameworks, switch-per-network configurations, and network topologies. Notably, GA-powered frameworks consistently exhibit lower ANL values compared to the others, indicating their ability to optimize routing paths within in-

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3374213

dividual networks more effectively. This trend persists across all network topologies and switch-per-network variations, as shown in Figs. 12a through 12d. Additionally, ANL is sensitive to changes in the number of switches per network with higher switch numbers generally resulting in slightly higher ANL values. Moreover, network topology also influences ANL with more complex topologies tending to yield marginally higher ANL values. Overall, these ANL findings underscore the effectiveness of GA-powered frameworks in enhancing routing efficiency over SDN networks.

VI. LIMITATIONS AND OPEN CHALLENGES

This section examines several limitations, challenges, and potential directions for further research to stimulate more exploration of the proposed *GATE-BC* traffic management framework.

Outdated BC Ledger: The transactions holding pathlet states inside *GATE-BC* experience ongoing alterations due to network and traffic conditions variations. As previously explained, any change in the state of a pathlet leads to the creation of a new transaction. Following this, the transaction undergoes a sequence of phases (such as transmission to block proposer nodes, waiting for inclusion in the next block, diffusion of the new block over the BC network, etc.) prior to being ultimately recorded in the BC ledgers. There is a potential scenario whereby the status update of the pathlet in the transaction might not persist until it is recorded in the BC due to a different update on the state of the relevant pathlet inside the network. Hence, it is possible that certain transactions may not accurately represent the current and most recent conditions of pathlets on the BC. This closely resembles the standard routing updates in conventional networks.

BC Ledger Size: Transactions represent pathlets within networks and contain both topological and QoS state information. Due to the complex characteristics shown by networks, including variations in size, topology, services, and user interactions, pathlets commonly undergo frequent updates during their operational lifespan. The aforementioned modifications are documented in new transactions and disseminated across all relevant networks. Hence, the long-term storage of these transactions can pose challenges and potentially impact the network overhead controllers deal with.

Block Period: This time frame is also known as the "block interval". This feature plays a pivotal part in enhancing the speed at which transactions are handled throughout the entire framework and ensures that the BC ledgers remain updated with the most recent status updates from ISPs. Nevertheless, this time frame could potentially result in increased computing demands for network controllers and the utilization of additional network resources. Hence, it is necessary to achieve a delicate equilibrium among the block interval, the associated computational load, and the obsolescence of the BC ledger.

Consensus Protocol Overhead: Consensus protocols substantially influence the performance of the *GATE-BC* architecture related to network controller overhead in the SDNs. Within the proposed framework, the ISP controllers, which function as BC nodes, are already engaged in a variety of networking tasks and communications. Assigning computationally intensive tasks, such as solving cryptographic challenges for block creation, might impede the efficiency of network operations and require additional processing resources. Hence, employing an efficient and lightweight consensus technique that requires fewer resources will help reduce the burden on these controllers.

Consequently, further exploration of the aforementioned limits and problems would be beneficial for undertaking a more complete network and controller load examinations. BC is considered a disruptive technology with great potential. Application of BC on network traffic management is one such use case if the aforementioned challenges are well addressed by researchers and practitioners from academia and industry.

VII. CONCLUSION AND FUTURE WORK

This study has introduced a framework called *GATE-BC*, which utilizes a modified Genetic Algorithm (GA) to control cross-network traffic in a QoS-aware manner. *GATE-BC* operates within the context of BC-enabled SDNs (BC-SDNs). With the continuous growth of interconnectedness and the dominant presence of video traffic in the digital realm, the integration of SDN and BC emerges as an appealing solution to address the difficulties associated with QoS-based routing across networks. The proposed *GATE-BC* framework has been compared to well-established routing strategies, namely QoSchain (*QC*), Hierarchical Routing Approach (*HRA*), and Distributed Routing Approach (*DRA*). The simulation results have demonstrated the superior performance of *GATE-BC* in minimizing Path Setup Time (PST), mitigating Network Message Overhead (NMO), improving Request Acceptance Ratio (RAR), optimizing Network Bandwidth Consumption (NBC), and streamlining Average Path Length (APL) and Average Network Length (ANL).

While *GATE-BC* demonstrably outperforms existing frameworks in terms of RAR, NBC, APL, and ANL, it is important to acknowledge some of its limitations. While the use of a customized Genetic Algorithm (GA) for path selection contributes significantly to *GATE-BC*'s effectiveness, it also introduces a trade-off in terms of PST. Compared to other frameworks like *QC*, which prioritize the first available path, *GATE-BC*'s GA-driven optimization process requires additional computational resources, leading to slightly higher PST. This trade-off necessitates careful consideration when prioritizing specific performance metrics in network management scenarios. Additionally, compared to the *DRA*, *GATE-BC*'s dynamic path selection and optimization mechanisms introduce increased communication and message exchange. *DRA* reduces the NMO measures, comparable to *GATE-BC*, by operating on the shortest BGP next-hop route at the internetwork level.

However, the implementation of the *GATE-BC* framework is not without challenges and complexities. Key tasks involve addressing BC ledger obsolescence, effectively managing the

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3374213

expanding BC ledger size, optimizing block interval time, and investigating lightweight consensus techniques to alleviate the burden on network controllers. These hurdles highlight potential challenges, demanding careful consideration and strategic solutions for the seamless integration of the *GATE-BC* model. Despite these challenges, adopting *GATE-BC* marks a fundamental step in establishing robust, flexible, and secure networking frameworks.

There are two crucial paths to explore for improving the proposed *GATE-BC* in future research pursuits. Firstly, implementing lightweight blockchain consensus protocols is a critical imperative to mitigate network message overhead, fostering heightened efficiency and scalability within the network infrastructure. The deployment of streamlined protocols addresses communication inefficiencies and contributes to optimizing network resource utilization. Secondly, exploring sharding methodologies holds the promise of distributing the ledger into more manageable partitions, thereby alleviating computational burdens and ensuring the sustained efficacy of *GATE-BC* in the face of an evolving network landscape. These prospective research trajectories underscore the commitment to refining the practical applicability, scalability, and resource efficiency of *GATE-BC* in dynamic BC-SDN environments.

REFERENCES

[1] Ericsson. (2022) Video Traffic Update. https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/traffic-by-application. Last Accessed: 2023-07-15.

[2] R. Venkatasai., U. Prabu., and S. Ch, "A survey and analysis of qos-based routing techniques in software-defined networks," in *2023 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS)*, 2023, pp. 1459–1464.

[3] S. W. Turner, M. Karakus, E. Guler, and S. Uludag, "A promising integration of sdn and blockchain for iot networks: A survey," *IEEE Access*, vol. 11, pp. 29 800–29 822, 2023.

[4] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," http://bitcoin. org/bitcoin. pdf," 2008.

[5] B. Cao, Z. Wang, L. Zhang, D. Feng, M. Peng, L. Zhang, and Z. Han, "Blockchain systems, technologies, and applications: A methodology perspective," *IEEE Communications Surveys Tutorials*, vol. 25, no. 1, pp. 353–385, 2023.

[6] Z. Sihai and L. Wei, "A qos multicast routing protocol for ad hoc network," in *2022 4th International Conference on Communications, Information System and Computer Engineering (CISCE)*, 2022, pp. 243–246.

[7] M. Karakus and A. Durresi, "Quality of Service (QoS) in Software Defined Networking (SDN): A Survey," *Journal of Network and Computer Applications*, vol. 80, pp. 200 – 218, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804516303186

[8] H. Li, L. Wang, Z. Zhu, Y. Chen, Z. Lu, and X. Wen, "Multicast service function chain orchestration in sdn/nfv-enabled networks: Embedding, readjustment, and expanding," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2023.

[9] Y. Su, B. Wang, Q. Xing, P. Li, X. Wang, and C. Li, "Research on blockchain-based inter-domain routing authentication technology," in *2021 IEEE 21st International Conference on Communication Technology (ICCT)*, 2021, pp. 810–816.

[10] E. Guler, M. Karakus, and S. Uludag, "Evaluating path selection strategies with blockchain-based routing in multi-domain sdns," in *2022 International Balkan Conference on Communications and Networking (BalkanCom)*, 2022, pp. 6–10.

[11] M. Karakus and E. Guler, "RoutingChain: A Proof-of-Concept Model for a Blockchain-Enabled QoS-Based Inter-AS Routing in SDN," in *2020 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, 2020, pp. 1–6.

[12] J. Shang, R. Huo, S. Wang, and T. Huang, "An ndn-enabled differentiated routing strategy for blockchain," in *2022 IEEE 2nd International Conference on Computer Communication and Artificial Intelligence (CCAI)*, 2022, pp. 96–101.

[13] M. Karakus and A. Durresi, "A Survey: Control Plane Scalability Issues and Approaches in Software-Defined Networking (SDN)," *Computer Networks*, vol. 112, pp. 279 – 293, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S138912861630411X

[14] M. Karakus, E. Guler, and S. Uludag, "QoSChain: Provisioning Inter-AS QoS in Software-Defined Networks with Blockchain," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2021.

[15] M. Karakus and A. Durresi, "A Scalable Inter-AS QoS Routing Architecture in Software Defined Network (SDN)," in *Advanced Information Networking and Applications (AINA)*, March 2015, pp. 148–154.

[16] T. Alharbi, "Deployment of blockchain technology in software defined networks: A survey," *IEEE Access*, vol. 8, pp. 9146–9156, 2020.

[17] W. Li, W. Meng, Z. Liu, and M.-H. Au, "Towards blockchain-based software-defined networking: Security challenges and solutions," *IEICE Trans. on Information and Systems*, vol. E103.D, no. 2, pp. 196–203, 2020.

[18] C. Xue, N. Xu, and Y. Bo, "Research on key technologies of software-defined network based on blockchain," in *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, 2019, pp. 239–2394.

[19] H. Nam Nguyen, H. Anh Tran, S. Fowler, and S. Souihi, "A survey of blockchain technologies applied to software-defined networking: Research challenges and solutions," *IET Wireless Sensor Systems*, vol. 11, no. 6, pp. 233–247, 2021.

[20] A. Rahman, A. Montieri, D. Kundu, M. Karim, M. Islam, S. Umme, A. Nascita, A. Pescapé *et al.*, "On the integration of blockchain and SDN: Overview, applications, and future perspectives," *Journal of Network and Systems Management*, vol. 30, no. 4, pp. 1–44, 2022.

[21] N. Y.-R. Douha, M. Bhuyan, S. Kashihara, D. Fall, Y. Taenaka, and Y. Kadobayashi, "A survey on blockchain, SDN and NFV for the smart-home security," *Internet of Things*, vol. 20, p. 100588, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660522000750

[22] S. Wadhwa, H. Babbar, and S. Rani, "A survey on emerging software-defined networking and blockchain in smart health care," *IOP Conf. Series: Materials Science and Engineering*, vol. 1022, no. 1, p. 012056, Jan 2021. [Online]. Available: https://doi.org/10.1088/1757-899x/1022/1/012056

[23] A. Rahman, J. Islam, D. Kundu, R. Karim, Z. Rahman, S. S. Band, M. Sookhak, P. Tiwari, and N. Kumar, "Impacts of blockchain in software-defined internet of things ecosystem with network function virtualization for smart applications: Present perspectives and future directions," *International Journal of Communication Systems*, p. e5429, 2023.

[24] Z. Yan and J.-H. Lee, "Bgpchain: Constructing a secure, smart, and agile routing infrastructure based on blockchain," *ICT Express*, 2021.

[25] G. He, W. Su, S. Gao, J. Yue, and S. K. Das, "Roachain: Securing route origin authorization with blockchain for inter-domain routing," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1690–1705, 2021.

[26] S. Awan, N. Javaid, S. Ullah, A. U. Khan, A. M. Qamar, and J.-G. Choi, "Blockchain based secure routing and trust management in wireless sensor networks," *Sensors*, vol. 22, no. 2, p. 411, 2022.

[27] I. W. B. Sentana, M. Ikram, and M. A. Kaafar, "Blockjack: Towards improved prevention of ip prefix hijacking attacks in inter-domain routing via blockchain," 2021.

[28] W. Jerbi, O. Cheikhrouhou, A. Guermazi, M. Baz, and H. Trabelsi, "Bsi: Blockchain to secure routing protocol in internet of things," *Concurrency and Computation: Practice and Experience*, vol. 34, no. 10, p. e6794, 2022.

[29] X. Ling, P. Chen, J. Wang, and Z. Ding, "Data broker: Dynamic multi-hop routing protocol in blockchain radio access network," *IEEE Communications Letters*, vol. 25, no. 12, pp. 4000–4004, 2021.

[30] M. M. Shabir, S. M. Danish, and K. Zhang, "Blockqos: Fair monetization of on-demand quality-of-service using blockchains," *Distributed Ledger Technologies: Research and Practice*, 2023.

[31] M. Hassan, M. Gregory, and S. Li, "Blockchain enhanced bgp4 security for an sdn based federation," in *2022 32nd International Telecommunication Networks and Applications Conference (ITNAC)*, 2022, pp. 1–7.

[32] T. Kayathri, N. Kumaresan, and R. V. Bhasker, "Sdbgpchain: A decentralized low complexity framework to detect and prevent the bgpdendrimer tree blockchain attacks using sdn with smart contract based," *Computer Networks*, p. 109800, 2023.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3374213

[33] Z. Zeng, X. Zhang, and Z. Xia, "Intelligent blockchain-based secure routing for multidomain SDN-enabled IoT networks," *Wireless Communications and Mobile Computing*, vol. 2022, 2022.
[34] A. Aldaej, M. Atiquzzaman, T. A. Ahanger, and P. K. Shukla, "Multidomain blockchain-based intelligent routing in uav-iot networks," *Computer Communications*, vol. 205, pp. 158–169, 2023.
[35] S. Abbas, N. Javaid, A. Almogren, S. M. Gulfam, A. Ahmed, and A. Radwan, "Securing genetic algorithm enabled SDN routing for blockchain based internet of things," *IEEE Access*, vol. 9, pp. 139 739–139 754, 2021.
[36] P. Podili and K. Kataoka, "TRAQR: Trust aware End-to-End QoS routing in multi-domain SDN using Blockchain," *Journal of Network and Computer Applications*, vol. 182, p. 103055, 2021.
[37] C. Ran, S. Yan, L. Huang, and L. Zhang, "An improved aodv routing security algorithm based on blockchain technology in ad hoc network," *EURASIP journal on wireless communications and networking*, vol. 2021, no. 1, pp. 1–16, 2021.
[38] "OpenFlow Switch Specification (1.5.1)." Open Networking Foundation, March 2015. [Online]. Available: https://www.opennetworking.org/software-defined-standards/specifications/
[39] P. Lin, J. Bi, S. Wolff, Y. Wang, A. Xu, Z. Chen, H. Hu, and Y. Lin, "A West-East Bridge Based SDN Inter-Domain Testbed," *Communications Magazine, IEEE*, vol. 53, no. 2, pp. 190–197, Feb 2015.
[40] A. Altaf, F. Iqbal, R. Latif, B. M. Yakubu, S. Latif, and H. Samiullah, "A survey of blockchain technology: Architecture, applied domains, platforms, and security threats," *Social Science Computer Review*, p. 08944393221110148, 2022.
[41] J. Xu, C. Wang, and X. Jia, "A survey of blockchain consensus protocols," *ACM Comput. Surv.*, jan 2023, just Accepted. [Online]. Available: https://doi.org/10.1145/3579845
[42] D. Maldonado-Ruiz, J. Torres, and N. E. Madhoun, "Fundamentals of blockchain technology," in *Principles and Practice of Blockchains*. Springer, 2023, pp. 3–25.



MURAT KARAKUS received the B.S. degree in Mathematics from Suleyman Demirel University, Türkiye, in 2009, and the M.S. degree in Computer Science and Information Systems from the University of Michigan-Flint, USA, in 2013, and the Ph.D. degree in the Computer Science from Purdue School of Science in Indianapolis, USA, in 2018. He is currently working as an Assistant Professor at Ankara University in Türkiye. He has 35+ peer-reviewed publications in and serves as a reviewer for esteemed IEEE, ACM, and Elsevier journals and conferences. His current research interests include next-generation network architectures, such as Software-Defined Networking (SDN), Blockchain technology, network scalability, Quality of Service (QoS), routing, economic analysis of network architectures and designs, and pricing network services.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4