Adaptive On-Chain and Off-Chain Communication Management Based on SDN and Blockchain

Daming Huang, Jincheng Xiang, Yu Jin, and Chen Tian(✉)

School of Computer Science, Nanjing University, Nanjing, China
tianchen@nju.edu.cn

Abstract. This paper proposes an adaptive communication management method based on Software-Defined Networking (SDN) and blockchain for the Trustworthy On-chain and Off-chain Interoperability System. By leveraging the security advantages of blockchain, this approach constructs a secure, decentralized SDN control system for managing the communication network. The system facilitates adaptive management modules that can autonomously detect failed gateways, select candidate gateways with low load for data transmission, and ensure the repair of data transmission paths when the off-chain network state undergoes changes. Furthermore, each SDN controller serves as a redundant component; in the event of a controller failure, others seamlessly assume its operations, thereby ensuring sustained communication reliability.

Keywords: blockchain · SDN · Trustworthy On-chain and Off-chain Interoperability System · Adaptive Communication Management

1 Introduction

As blockchain applications continue to evolve, the volume of on-chain data is increasing significantly, presenting a substantial challenge to traditional blockchain technologies that rely on a "chain-native" approach to development.

Off-chain and on-chain collaboration is a critical area for the future development of blockchain technology, necessitating research into the key technical challenges across diverse application scenarios that require trustworthy interaction between on-chain and off-chain components. A key element of the Trustworthy On-chain and Off-chain Interoperability System is a hierarchical communication network with a focus on the edge, designed to effectively optimize communication resources and enable collaboration across dynamic networks. This paper investigates the adaptive and reliable aspects of the on-chain and off-chain communication network, and proposes an adaptive management method based on blockchain and Software-Defined Networking (SDN), which offers the following benefits:

This work was supported by the National Key Research and Development Program of China under Grant 2022YFB2702800.

W. Liang et al. (Eds.): SecureComm 2025, LNICST 687, pp. 41–59, 2026.
https://doi.org/10.1007/978-3-032-23447-6_3

(1) By leveraging blockchain technology to establish a distributed SDN management network, SDN controllers utilize a secure supervision chain to exchange network status and management information, facilitating the integration of resources across multiple management domains.
(2) Through distributed SDN management, when a smart gateway fails, the distributed SDN management module can autonomously repair the system and select the most efficient data transmission path.
(3) In the event of an SDN controller failure, other SDN controllers can seamlessly assume its management tasks, ensuring the reliability of the entire distributed SDN management system.

The remainder of the paper is organized as follows: Sect. 2 reviews related work. Section 3 presents the system design and architecture. Section 4 discusses the adaptive communication management strategy. The implementation details and experimental results are provided in Sect. 5. Section 6 offers a discussion, and Sect. 7 concludes the paper.

2 Related Work

2.1 Blockchain

The concept of blockchain technology initially emerged as a supporting technology for Bitcoin [1], leveraging cryptographic techniques and consensus mechanisms to ensure the consistency of distributed ledgers without relying on a trusted third party. Moreover, it guarantees the immutability of the ledger, thereby enabling decentralized trust. Ethereum [2] has realized the full potential of blockchain technology by introducing smart contracts, marking the advent of the blockchain 2.0 phase. The integration of blockchain and smart contracts has expanded the application of blockchain technology beyond cryptocurrencies to various domains such as supply chain management, education, healthcare, energy, and the industrial Internet of Things. Blockchain can be categorized into three types based on its management approach and consensus mechanism: Public blockchains, such as Bitcoin and Ethereum, allow nodes to freely join and leave, and typically have large network sizes. These blockchains, using consensus mechanisms like Proof of Work (PoW) and Proof of Stake (PoS), generally exhibit lower performance due to their non-deterministic nature. Consortium blockchains, formed through the collaboration of multiple entities, require specific management conditions for adding or removing nodes and often employ algorithms like Practical Byzantine Fault Tolerance (PBFT) to achieve consensus. Examples of consortium blockchains include Hyperledger [3], FISCO BCOS [4], and Dyno Chain [5]. Private blockchains, typically built by a single entity, involve nodes that are exclusively managed by that entity and use consensus algorithms such as PAXOS or RAFT to ensure agreement.

The need for trustworthy interaction between on-chain and off-chain systems exists across various sectors such as healthcare, energy, transportation, agriculture, and the industrial Internet of Things (IIoT). To enable reliable and efficient

data transfer of large volumes of terminal data on-chain in these sectors, it is essential to leverage Consortium blockchain technology. Consortium blockchain technology has already been applied in numerous industries.

However, there is currently a lack of dedicated research specifically focused on developing network communication technologies for on-chain and off-chain interaction. The primary focus of this paper is to address the reliability and adaptability of on-chain and off-chain communication networks.

2.2 SDN

Software-Defined Networking (SDN) [6] can be managed through programming, enabling network device control by separating the control plane from the data plane. The SDN architecture is typically divided into three layers: the Data Plane, the Control Plane, and the Application Plane. The Data Plane includes communication devices such as switches, while the Control Plane consists of controllers that implement network control logic. The Application Plane is responsible for implementing specific network applications. The Data Plane and Control Plane communicate through a southbound interface (SBI), commonly referred to as the SDN Data Control Plane Interface (DCPI). Currently, the southbound interface is primarily utilized for OpenFlow communication protocols [7]. The Application Plane communicates with the Control Plane via a Northbound Interface (NBI), utilizing various network resources to rapidly configure and deploy network applications.

The controller is the core component of an SDN network, responsible for implementing the network control and management logic. Based on the OpenFlow protocol, several controllers have been proposed and implemented, including NOX [9], RYU [10], OpenDaylight [11], and Floodlight [12].

The integration of SDN with cloud computing, edge computing, and data center networks has been remarkably successful [13,14]. Currently, numerous research efforts are exploring the convergence of SDN with blockchain technology.

2.3 Blockchain and SDN Based Communication Management

QoSChain [15] is an innovative blockchain-aided framework for inter-Autonomous System Quality of Service (QoS) routing in Software-Defined Networks (SDNs). QoSChain integrates blockchain and SDN technologies to eliminate centralized intermediaries, reduce QoS signaling overhead, and safeguard sensitive information through decentralized coordination. The primary contributions include the introduction of a conflict-resolving coordination framework, the reduction of path setup time and message exchanges, and the enhancement of security and privacy. However, challenges persist in block generation time, consensus protocol scalability, and the long-term storage limitations of the blockchain ledger, which require further refinement for practical deployment.

TRAQR [16] enables trust-aware end-to-end QoS routing in multi-domain SDN environments using blockchain. The framework leverages blockchain's tamper-proof properties to store trust information, verify domain identities, and

ensure QoS compliance. It introduces a trust score metric to quantify domain trustworthiness and a secure log exchange mechanism to audit QoS implementation. The main contributions include developing a scalable trust-aware routing algorithm, enabling decentralized QoS coordination, and providing a proof-of-concept implementation. However, the framework faces challenges in scalability for large-scale networks and balancing trust with resource availability for load balancing. Future work should focus on optimizing these aspects.

Paper [17] proposes a Blockchain-based credible routing scheme for SDN-based cloud environments to address trust issues in cross-domain routing. The main contributions include establishing decentralized trust among distributed SDN controllers using Consortium Blockchain, designing an efficient cross-domain routing mechanism, and introducing a sliding window mechanism to reduce storage overhead. The scheme ensures secure and credible routing with limited bandwidth and storage requirements. However, it faces potential scalability limitations due to the PBFT consensus algorithm's complexity and may require further evaluation on handling high network latency or frequent topology changes.

S-HIDRA [18] constructs a horizontally distributed Software-Defined Networking (SDN) architecture to abstract and schedule container and network services. The framework guarantees Quality of Service (QoS) for Internet of Things (IoT) devices, focusing on delay and availability metrics. It leverages local domain blockchain for secure internal information exchange and employs a global blockchain to facilitate inter-domain communication. This is achieved through the deployment of smart contracts and event-driven analysis, which together enable efficient information exchange and workflow protocol implementation.

Paper [19] presents a two-layer network model designed to capture the path-building relationships within a distributed Software-Defined Networking (SDN) environment. This model encompasses intra-domain node connections, inter-domain links, and connections associated with the SDN network controller. The framework employs the Average Path Cost as a performance metric for each link, assigning weights to represent the network state—where a smaller weight indicates a more favorable network condition. The End-to-End path construction problem is subsequently optimized to identify the path with the minimum total weight.

Paper [20] introduces an intelligent, secure routing method for Internet of Things (IoT) networks in multi-domain Software-Defined Networking (SDN) environments. The approach leverages blockchain technology to ensure the trustworthiness of topological information exchanges between controllers. Furthermore, a local and global reputation system is designed to enhance the reliability of the routing process. A testbed was developed based on ONOS, Mininet, and Hyperledger Fabric to validate the proposed method.

2.4 Load Balance of SDN Network

Paper [21] presents experimental research that demonstrates the advantages of machine learning algorithms for load balancing in large-scale Software-Defined Networking (SDN)-enabled Internet of Things (IoT) networks. The study evaluates four load balancing algorithms: random scheduling, round-robin scheduling, graph coloring, and depth-deterministic policy gradient (DDPG). By comparing these machine learning-based algorithms with traditional static methods, the paper shows that machine learning techniques effectively reduce latency and minimize data packet loss, while also offering better adaptability to dynamic network conditions.

Paper [22] compares four load balancing strategies for Software-Defined Networking (SDN) environments: Round Robin, Random Selection, Dynamic Weighted Round Robin (DWRS), and Least Time-Based Weighted Round Robin, through experimental evaluations. The study finds that dynamic load balancing algorithms, such as DWRS, outperform static strategies in SDN environments. However, as the request volume increases, the packet loss rate also rises, primarily due to a bottleneck caused by a single centralized controller handling a large number of requests. To address this issue, the paper proposes the adoption of a multi-controller architecture to enhance both scalability and performance.

2.5 Load Balance of SDN Controllers of Distrubuted SDN Network

Paper [23] proposes a dynamic decision controller load balancing mechanism (DDCLB) that addresses the load balancing and fault recovery challenges in Software-Defined Networking (SDN) networks. By utilizing load monitoring, fault detection, and optimized switch migration strategies, the proposed mechanism enhances the reliability and performance of distributed controllers.

Paper [24] introduces a load balancing method tailored for SDN multi-controller networks. The approach employs clustering techniques to mitigate the issue of unequal load distribution across controllers. This method not only improves the network's fault tolerance but also optimizes overall network performance.

3 System Desgin

3.1 End-Edge-Chain Network Architecture

To address the challenge of managing large volumes of heterogeneous terminal data on-chain and enabling efficient data retrieval from the blockchain, we have developed a communication network architecture centered around edge smart gateways, as illustrated in Fig. 1.

Each management domain consists of an smart edge gateway, a front-end SDN switch, and other communication devices. Terminals connect to the smart edge gateway through the front-end switch, which in turn is connected to the

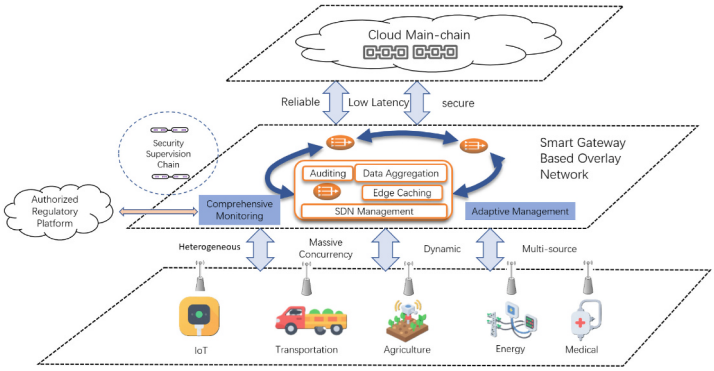

Fig. 1. End-Edge-Chain Network Architecture.

cloud blockchain. The SDN controller within each domain manages the front-end switch and other communication devices, while switches across domains can be interconnected via the Internet.

The smart edge gateway in each management domain aggregates data from its terminal devices to facilitate efficient on-chain data transfer. Both the smart gateway and its front-end switch are managed by the local SDN controller, with the Security Supervision Chain ensuring the reliability of the distributed SDN system. In the event of a smart gateway malfunction, the system can automatically select a candidate gateway node, restoring communication paths to transfer terminal device data to the Cloud Chain. If a local SDN controller fails, other controllers can seamlessly assume control of the network tasks, ensuring uninterrupted operation.

3.2 Data Structure Representation

Each terminal device within a management domain must establish a connection to the smart gateway through the gateway's front-end SDN switch. The front-end SDN switches are interconnected via the Internet to enable communication between gateways. As a result, the global network status is determined by the status of each management domain's smart gateway. The status of each gateway is composed of its IP address, heartbeat timestamp, operational state, and workload, as illustrated in Fig. 2.

When a gateway is operating normally, its operational state is set to "True." The heartbeat timestamp represents the moment when the management contract receives the gateway's heartbeat signal. The management contract calculates the time difference between the current time and the gateway's last received timestamp. If the time difference exceeds the predefined timeout threshold, the gateway's operational state is set to "False," indicating a fault. The gateway's workload serves as a metric to measure its current processing load.

Each domain's SDN controller determines the operational status of the managed gateway based on the global gateway status provided by the management smart contract. In the event of a gateway failure, the responsible SDN controllers must construct a restoration path for the terminal devices. This restoration path is implemented by the SDN controller through the front-end SDN switch.

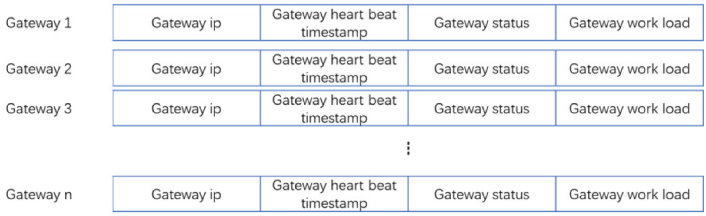

Fig. 2. Gloabl network status.

The status of each domain's SDN controller is composed of the controller ID, workload, heartbeat timestamp, operational state, and the ID of the guardian controller, as shown in Fig. 3. The controller ID uniquely identifies each controller, while the workload serves as a metric for measuring the controller's current processing load. The purpose and functionality of the controller heartbeat and operational state are analogous to those in the gateway status. By processing the received controller heartbeats, the management smart contract determines the controller's operational status. The guardian controller ID refers to the ID of another SDN controller that assumes management responsibilities when the current controller enters a failure state. It is essential that each controller's management tasks are assigned to a unique guardian controller.

The status of all SDN controllers collectively forms the global status of the distributed SDN controller system. In the event of an SDN controller failure, the management smart contract selects the guardian controller. Upon receiving the latest global status of the controllers from the management smart contract, the guardian controller assumes management responsibilities for the faulty SDN controller.

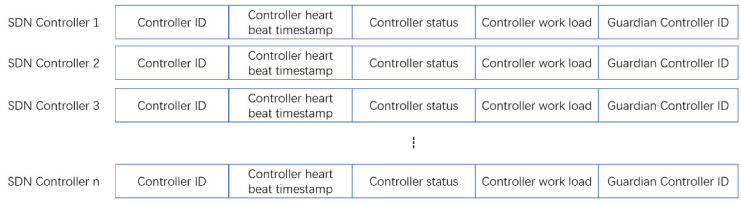

Fig. 3. Gloabl SDN Controller status.

3.3 Essential Components Structure Design

The essential components of the on-chain and off-chain communication network system include domain SDN controllers, smart edge gateways, and the security supervision chain, as illustrated in Fig. 4.

The structure of the management domain SDN controller consists of several key modules: identity verification, global network status maintenance, controller status reporting, and heartbeat modules; data transmission path restore strategy; SDN management and maintenance strategy; and SDN operation audit modules. The communication functions of the smart edge gateway are facilitated by the data-on-chain-related module, while management-related functions are handled by the gateway status reporting and heartbeat module. The management functions of the security supervision chain are implemented through a smart contract, which primarily includes the identity management module, global network status maintenance module, and global controller status maintenance module.

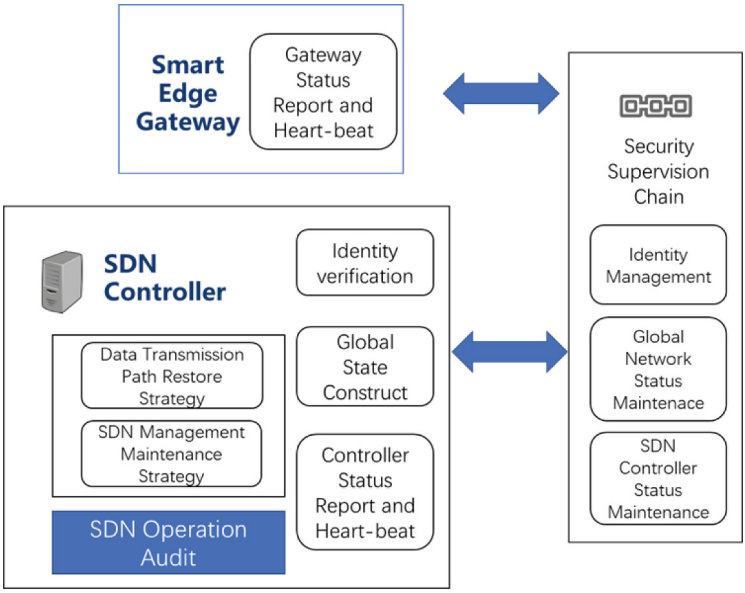

Fig. 4. SDN Controller Structure and Gloabl Status Maintenance.

The integration of the management domain SDN controller with the Security Supervision Chain enables secure and reliable information exchange between controller instances, thereby constructing a decentralized distributed SDN network.

The Security Supervision Chain stores the global network status. Each management domain SDN controller submits its identity and verification information

to the Security Supervision Chain via a decentralized identity management smart contract module. When SDN controllers submit their status and topology information on-chain and communicate with one another, they utilize chain-based identity information for authentication, ensuring the security and integrity of the SDN management system.

3.4 Workflow of Essential Components

Restoration of Data Transmission Path for Faulty Gateway. When a smart gateway fails, the data path through the gateway is disrupted. To restore the data transfer path, the workflow involving the gateway, SDN controller, and security supervision chain is divided into six steps, as illustrated in Fig. 5.

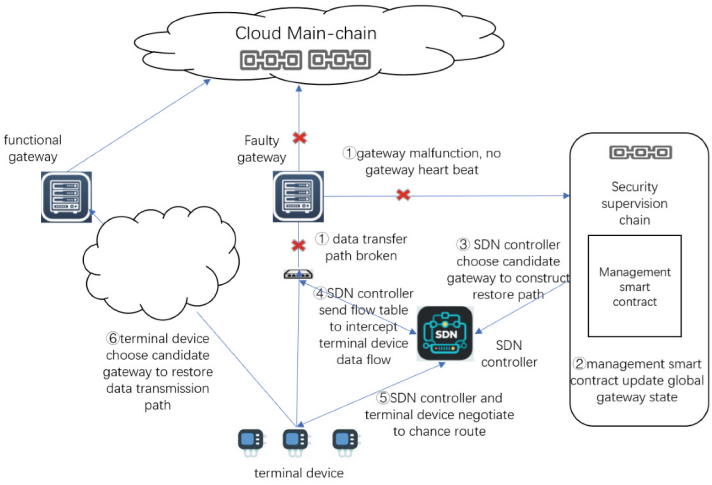

Fig. 5. Restoration of data transmission path for faulty gateway.

① When the gateway fails, it no longer submits heartbeat or status information to the security supervision chain. At the same time, the data path from the terminal device to the faulty gateway is disrupted.

② The management smart contract updates the global gateway status, enabling the identification of faulty gateways.

③ The guardian SDN controller of the faulty gateway, utilizing the global gateway status provided by the management smart contract, selects a candidate gateway to establish the restoration path.

④ The SDN controller transmits flow table data, which captures the traffic traversing the faulty gateway, to the SDN switch at the gateway's front-end.

⑤ The SDN controller engages in negotiation with the corresponding terminal device to facilitate the establishment of a data transfer recovery path.

⑥ The terminal device selects the candidate gateway to restore data transmission path.

Workflow for Taking over a Failed Controller. When a domain SDN controller fails, the management of the gateway and front-end SDN switch within that domain becomes ineffective. In such cases, a guardian controller must be selected to assume the management responsibilities. The workflow illustrated in Fig. 6 consists of four distinct steps.

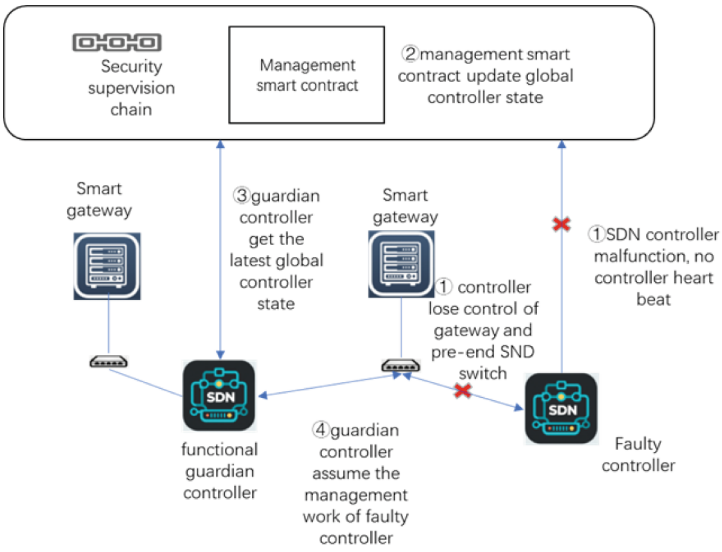

Fig. 6. Assume the Management work of a Faulty Controller.

① The faulty domain SDN controller ceases to send heartbeat and status information to the security supervision chain, rendering it unable to manage the gateway and front-end SDN switch within the domain.

② The management smart contract updates the global controller status, identifies the faulty SDN controller, and selects the guardian controller based on the predefined strategy.

③ The guardian controller retrieves the most recent global controller status from the management smart contract.

④ The guardian controller assumes the management responsibilities of the faulty SDN controller within the domain, ensuring effective management of the gateway and front-end SDN switch.

4 Adapative Comunnication Management Strategy

4.1 Data Transmition Path Restore Strategy

When an edge smart gateway fails, the guardian SDN controller must implement a strategy to identify an appropriate candidate gateway based on the global status of all gateways.

To optimize overall system efficiency, the controller should select the gateway with the lightest load as the candidate, thereby ensuring improved load balancing across the entire blockchain network. As introduced in Sect. 2, several SDN network load balancing algorithms are already in existence. Leveraging blockchain technology, we can access a global gateway status. This paper proposes an optimization algorithm for determining the most suitable candidate gateway by considering both the current load and anticipated future load trends of each gateway, aiming to maximize the overall throughput of the system.

To accurately assess the load status of each gateway, the SDN controller must gather relevant technical metrics. A commonly used set of metrics consists of 30 distinct load balancing indicators, as outlined in the work of Alrammahi et al. [25]. These metrics are categorized and ranked according to various factors, including throughput, overhead, load balancing efficiency, latency, response time, packet loss rate, resource utilization, and transaction time.

The primary function of the on-chain off-chain communication network system is to facilitate the submission of large volumes of data from numerous heterogeneous terminal devices to the cloud-end blockchain. As a result, the smart gateway's capacity to process device data is a more critical metric than other network parameters. This paper proposes a strategy that takes into account the length of data item queues in the gateway server, the gateway's current throughput(TH), and their temporal trends. Based on this, we introduce the Gateway Workload Metric (QLTTMT), which integrates Queue Length and Throughput Trend Metrics over Time to evaluate gateway performance more effectively.

$$Norm(TH_t) = \frac{TH_t}{TH_{max}} \tag{1}$$

$$Norm(\Delta TH) = \frac{TH_t - TH_{t-1}}{TH_{max}} \tag{2}$$

$$Norm(QL_t) = \frac{QL_t}{QL_{max}} \tag{3}$$

$$Norm(\Delta QL) = \frac{QL_t - QL_{t-1}}{QL_{max}} \tag{4}$$

$$QPF = \frac{1}{max(0.01, 1 - Norm(QL_t)} \tag{5}$$

$$RTP = \frac{MAX(TH)}{TH_t} \tag{6}$$

$$QLTTMT = \frac{max(0, Norm(QL_t) + Norm(\Delta QL))}{max(0.1, Norm(TH_t) + Norm(\Delta TH))} \times QPF \times RTP \tag{7}$$

Let QL_t denote the queue length at any given time t, and QL_{max} represents the maximum queue length of the current smart gateway, which is determined by

the gateway's memory and software configuration. TH_t refers to the throughput of the smart gateway at time t, while TH_{max} is the known maximum throughput of the gateway, with its initial value typically set by an administrator. If a new measurement exceeds the current maximum throughput, the value is updated accordingly.

Since the maximum queue length and maximum throughput of each smart gateway are determined by its hardware and software environment, these values may vary. We utilize normalized values of queue length, queue length variation, throughput, and throughput variation. QPF denotes the Queue Penalty Factor. The higher the queue utilization, the weaker the capacity to handle load. Here, the value 0.01 indicates that a queue is considered full if its occupancy exceeds 99%. To balance the capabilities of gateways with different throughput, we further incorporate the Relative Throughput Penalty factor(RTP). In this context, MAX(TH) represents the maximum throughput among all gateways, obtained globally through the blockchain. RTP facilitates optimal gateway selection by favoring well-performing nodes retaining throughput headroom against superficially-underutilized but capability-limited alternatives.

A smaller QLTTMT value indicates that the corresponding gateway possesses a higher capacity to handle load.

4.2 Guardian Controller Selection Strategy

Linear Selection Strategy. In the event of a failure in a domain SDN controller, other domain SDN controllers must be capable of taking over the management of the affected switches and monitoring gateways, thereby ensuring that all SDN gateways and communication devices form a fully self-adaptive data chain communication system. The takeover process must be deterministic—meaning that only one guardian domain SDN controller should assume control of the faulty SDN controller. Simultaneous takeover attempts by multiple controllers would lead to management conflicts and decreased system efficiency. Given that this is a distributed SDN system, maintaining a global management view is critical, which can be facilitated through blockchain technology. Each domain SDN controller periodically reports its status to the blockchain supervision chain, which is managed by the Security Supervision Chain.

A straightforward strategy for achieving deterministic guardian controller takeover of a faulty SDN controller is the linear method. In this approach, all SDN controllers are organized into a circular list, with each controller having a unique predecessor and successor. It is specified that each active SDN controller is responsible for overseeing its unique predecessor. When the predecessor SDN controller fails, the guardian SDN controller assumes control of the switches and gateways previously managed by the failed controller.

Upon activation, each SDN controller proactively takes over the management of its assigned switches and gateways, ensuring a seamless recovery of its supervisory tasks in the event that a faulty SDN controller restarts. If the controller responsible for overseeing the takeover of other faulty SDN controller domains

also fails, its surviving successor must assume control of both the faulty controller and any other controllers it currently manages.

Load Balance Selection Strategy. The Linear Takeover Strategy ensures that, with at least one surviving SDN controller, the entire on-chain off-chain communication network can be effectively managed by the distributed SDN system. However, if a single SDN controller is tasked with overseeing too many faulty SDN controllers, it may lead to significant load imbalance, causing high traffic congestion across the system. Therefore, the selection of the guardian controller for a faulty SDN controller should incorporate a load balancing strategy to prevent such issues.

In the on-chain off-chain communication network, each SDN controller monitors the communication traffic sent from terminal devices to the smart edge gateway. When a local edge gateway fails, the SDN controller intercepts and reroutes the traffic intended for the faulty gateway through managed switches, while notifying the relevant clients to redirect their traffic to other operational gateways. Consequently, the SDN controller's primary load arises during the traffic rerouting process, and we use the packet-in message rate processed per second as a load indicator. This metric helps to more accurately reflect the real-time load of the on-chain off-chain communication network system. Based on the SDN controller's actual load, we have implemented a load balancing selection strategy within the smart contract, as detailed in Algorithm 1.

Algorithm 1. Lightest Load Guardian SDN Controller Selection Algorithm in Smart Contract

Input: List of SDN Controller $ConList$ of length n; Failure SDN Controller's index fi

Output: Failure Controller $Conlist[fi]$'s unique Guardian Controller ID, which has lightest load; if there is no alive controller, return -1.

Initialize $gi \leftarrow 0$;
while $gi \leq n-1$ **AND** $Conlist[i].alive =$ **False do**
  $gi \leftarrow gi + 1$;
end while
Initialize $i \leftarrow gi + 1$
while $i \leq n-1$ **do**
  if $Conlist[i].alive$ **AND** $Conlist[i].load < Conlist[gi].load$ **then**
    $gi \leftarrow i$;
  end if
  $i \leftarrow i + 1$;
end while
if $gi < n$ **then**
  return $Conlist[gi].ControllID$;
else
  return -1;
end if

The selection strategy for the guardian controller should be triggered once the smart contract detects a controller transitioning from a normal operational state to a faulty state. Additionally, when a faulty controller recovers, it naturally resumes management of its local domain's operations, necessitating the smart contract to set the guardian controller of the restored controller to NULL. This global status maintenance process should be automatically executed by the smart contract deployed on the security supervision blockchain upon receiving the controller's heartbeat information. In Algorithm 2, we present a global status maintenance algorithm for the SDN controllers.

Algorithm 2. Global SDN Controller status Maintenance Algorithm in Smart Contract

Input: List of SDN Controller $ConList$ of length n; Heart beat of SDN Controller h
Output: Renewed List of SDN Controller $ConList$ in which each failure controller has a unique Guardian Controller and each working controller has its Guardian Controller set to NULL;

$Conlist[h].timestamp \leftarrow currenttimestamp$
if $Conlist[i].alive =$ **False then**
  $Conlist[h].alive \leftarrow$ **True**
  $Conlist[h].guardiancontorller \leftarrow NULL$
end if
Initialize $i \leftarrow 0$
Initialize $curtime \leftarrow currenttimestamp$
while $i \leq n-1$ **do**
  if $Conlist[i].alive$ **AND** $curtime$ - $Conlist[i].timestamp > TIMEOUTLIMIT$ **then**
    $Conlist[i].alive \leftarrow$ **False**
    $Conlist[i].guardiancontroller \leftarrow guardiancontorllerselection(Conlist, i)$
    Initialize $j \leftarrow 0$
    while $j \leq n-1$ **do**
      if $Conlist[j].alive$=**False** **AND** $Conlist[j].guardiancontroller = Conlist[i].ControllID$ **then**
        $Conlist[j].guardiancontroller \leftarrow guardiancontorllerselection(Conlist, j)$
      end if
    end while
  end if
  $i \leftarrow i+1$;
end while
return $Conlist$;

For each operational SDN controller, upon receiving the global controller status from the security supervision chain, it must compare the current global status with the previous one to determine if it is required to assume the management task of any faulty controllers.

For an SDN controller i, it needs to traverse through each controller node j in the global status. If it identifies that the guardian controller of a domain SDN controller j was not i's ID in the previous global status, but has become i's

ID in the current global status, then controller i will assume responsibility for managing controller j's domain.

5 Implementation and Experiments

5.1 Implementation

For testing purposes, we set up a testbed consisting of four servers, each simulating an individual edge smart gateway. Within each server, we installed a device resembling an OVS switch to emulate a network gateway, responsible for routing client data to the blockchain. Each edge smart gateway deployed a smart gateway module that received client data requests, aggregated them, and subsequently submitted them to a cloud-based blockchain. The cloud blockchain operates on the proprietary Dyno consortium chain. Initially, the security supervision chain utilized FISCO-BCOS, eventually transitioning to the Dyno chain.

The hardware configuration of the gateway server consists of an Intel® Xeon® Silver 4110 CPU running at 2.10 GHz, with 257,616 MB of main memory. The development environment is based on the Ubuntu 22 operating system, with the Ryu SDN controller, Python 3.8.10, and ApacheBench, Version 2.3.

We developed and deployed a server program capable of aggregating data from multiple terminal devices into a single data item, which is then bulk-submitted to the cloud-end blockchain.

5.2 Experiments and Results

Adaptive Data Transmition Path Restore Experiments. The experiment uses three smart gateways, each with four request queues, each with a length of 10,000. The maximum throughput of each gateway is around 18,000. Initially, all three gateways are functioning properly, with Gateway 2 having a throughput of approximately 1400 and Gateway 3 having a throughput of approximately 15000. A simulated client continuously sends 300,000 data entries to Gateway 1, each with an index length of 20 and data length of 100. Gateway 1 is configured to fail immediately upon receiving the first data packet, simulating a gateway failure, with a gateway heartbeat timeout set to 30 s.

Once the SDN controller detects the failure of Gateway 1, it intercepts the traffic destined for Gateway 1, selects a candidate gateway, and guides the simulated terminal device to send the data to the chosen candidate gateway, thereby achieving the repair of the data on-chain path.

Table 1 presents the results of six consecutive tests, each involving the transmission of 300,000 data entries to Gateway 1. It shows the time taken for the SDN controller to complete the data on-chain process after selecting the candidate gateway using both the linear selection strategy and the QLTTMT strategy. The time for the first round includes the approximately 30 s required for the gateway timeout. The subsequent five rounds reflect the time taken after the SDN controller has updated the flow table of the pre-failure SDN switch to repair the path. The experimental results demonstrate that the QLTTMT strategy significantly outperforms the linear selection strategy.

Table 1. Comparsion of Linear Strategy and QLTTMT Strategy.

Round	Linear Strategy	QLTTMT Strategy
Round 1[1]	238.63	136.64
Round 2	143.59	63.18
Round 3	144.33	65.85
Round 4	144.90	65.92
Round 5	142.89	65.72
Round 6	144.86	64.63
AVG time of Round 2 6	144.11	65.06
AVG time of Round 1 6	159.87	76.99

[1] Round 1 include The Gateway Timeout time.

Adaptive Assume the Management Work of Faulty Controller. Based on the testbed, we designed an experiment to validate the adaptive management of SDN controllers in the event of failures. Initially, four gateways and their corresponding front-end SDN switches, along with four management domain SDN controllers, were started and operating normally. Subsequently, one, two, or three controllers were randomly stopped, and the functioning controllers seamlessly took over the management responsibilities of the malfunctioning SDN controllers. When the faulty SDN controllers were restored, they correctly resumed management of their respective local domain gateways and front-end SDN switches. Prior to this, the guardian controllers were automatically relinquished, preventing any disputes over management authority.

6 Discussion

This work differs from similar projects in three key aspects:

(1) The primary focus of this research is the adaptive management of communication networks within the Trustworthy On-chain and Off-chain Interoperability System. While prior works such as QoSChain and TRAQR focus on end-to-end communication quality and optimal path selection, our work ensures reliable data onboarding via gateways for terminals. Crucially, our approach does not select communication paths between terminals and fixed gateways, but rather identifies the optimal gateway upon gateway failure.
(2) In a cross-domain communication system where each domain is connected via the internet, the distributed SDN system proposed in this paper effectively integrates local domain networks to create a comprehensive and trustworthy communication network. When a gateway fails, the SDN controller captures traffic intended for the faulty gateway through the gateway's front-end SDN switch. Utilizing the blockchain's verified global status, the SDN controller can establish a recovery path for data transmission, ultimately restoring the data transfer paths through negotiation with the end-points.

(3) This work enables a more accurate assessment of the actual workload of gateways and controllers, moving beyond standard network metrics. Instead, the workload is assessed based on the actual functional logic of the gateway and controller.

From a security perspective, the management contract on the security supervision chain ensures identity verification for smart gateways and SDN controllers. Only authorized smart gateways and SDN controllers are allowed to submit status and heartbeat data to the management smart contract, as well as to retrieve the aggregated global status. This mechanism guarantees the reliability and trustworthiness of the global status information, which is crucial for the adaptive management approach.

7 Conclusions

Trustworthy On-chain and off-chain interoperability is an emerging research field. Building on the End-Edge-Chain Network Architecture, which is specifically designed for trustworthy on-chain and off-chain interoperability communication network, this work presents a specialized solution for self-adaptive communication management across multiple management domains. The proposed approach integrates distributed SDN and blockchain technologies to address the demands of large-scale data transfer from heterogeneous terminal devices to the cloud-chain through smart edge gateways.

This research primarily focuses on the restoration of transmission paths in the event of gateway failures, as well as ensuring overall system load balancing. To effectively manage the entire on-chain off-chain communication network using a distributed SDN controller system, this paper proposes a mechanism in which gateways and controllers submit their local statuses to a security supervision chain. The management smart contract then aggregates these local statuses into a global status, which includes both the network's global status and the controller's global status.

The controller restores the faulted path and ensures load balancing based on the network's global status. The controller's global status is used to maintain the proper functioning of the distributed SDN controller system and to achieve effective load balancing across controllers. To implement efficient load balancing, this work defines custom metrics for gateway load and controller load, which accurately reflect the operational status of the on-chain off-chain communication network. Additionally, we have constructed a testbed to demonstrate the effectiveness of our proposed adaptive management solution.

Future research will explore the integration of AI technologies to generate more accurate forecasts of future workloads, thereby enhancing load balancing between domain controllers and gateways. This improved load balancing will enable more effective control over network communication, ultimately leading to greater efficiency across the entire on-chain off-chain communication network.

References

1. Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System (2008). https://bitcoin.org/
2. Buterin, V.: A Next Generation Smart Contract and Decentralized Application Platform (2014). https://github.com/ethereum/wiki/wiki/White-Paper
3. Cachin, C.: Architecture of the Hyperledger Blockchain Fabric (2016). https://pdfs.semanticscholar.org/
4. Li, H., et al.: FISCO-BCOS: an enterprise-grade permissioned blockchain system with high-performance. In: SC23: International Conference for High Performance Computing, Networking, Storage and Analysis, Denver, CO, USA, pp. 1–17 (2023). https://doi.org/10.1145/3581784.3607053
5. Duan, S., Zhang, H.: Foundations of dynamic BFT. In: IEEE Symposium on Security and Privacy (SP). San Francisco, CA, USA, vol. 2022, pp. 1317–1334 (2022). https://doi.org/10.1109/SP46214.2022.9833787
6. MeKeown N.: Software-defined networking. In: Proceedings of the INFOCOM Key Note (2009). http://infocom2009.ieee-infoeom.org/technicalProgram.htm
7. McKeown, N., et al.: OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Comput. Commun. Rev. **38**(2), 69–74 (2008)
8. Subramanian, S., Voruganti, S.: Software-Defined Networking (SDN) with OpenStack. Packt, Birmingham, U.K. (2016)
9. Gude, N., et al.: NOX: towards an operating system for networks. In: proceedings of ACM SIGCOMM, vol. 38, pp. 105–110 (2008)
10. NTT Corporation. RYU the Network Operating System (NOS) (2017)
11. Open Daylight. Open Daylight: Open Source SDN Platform (2017)
12. Project Floodlight. Floodlight OpenFlow Controller (2017)
13. Khedkar, S.P., AroulCanessane, R.: SDN enabled cloud, IoT and DCNs: a comprehensive survey. In: 5th International Conference On Computing, Communication, Control And Automation (ICCUBEA). Pune, India **2019**, 1–5 (2019). https://doi.org/10.1109/ICCUBEA47591.2019.9129091
14. Espinel Sarmiento, D., Lebre, A., Nussbaum, L., Chari, A.: Decentralized SDN control plane for a distributed cloud-edge infrastructure: a survey. In: IEEE Communications Surveys and Tutorials, vol. 23, no. 1, pp. 256–281, Firstquarter (2021). https://doi.org/10.1109/COMST.2021.3050297
15. Karakus, M., Guler, E., Uludag, S.: QoSChain: provisioning inter-AS QoS in software-defined networks with blockchain. In: IEEE Transactions on Network and Service Management, vol. 18(2), pp. 1706–1717 (2021)
16. Podili, P., Kataoka, K.: TRAQR: trust aware End-to-End QoS routing in multi-domain SDN using Blockchain. In: Journal of Network and Computer Applications, Vol. 182, pp. 103055 (2021). ISSN: 1084–8045. https://doi.org/10.1016/j.jnca.2021.103055
17. Qiao, Q., Li, X., Wang, Y., Luo, B., Ren, Y., Ma, J.: Credible routing scheme of SDN-based cloud using blockchain. In: Proceedings of the 2019 International Conference on Data Science and Intelligent Applications (ICPCSEE), Singapore, Singapore, 2019, pp. 189–206 (2019)
18. Núñez-Gómez, C., Carrión, C., Caminero, B., Delicado, F.M.: S-HIDRA: a blockchain and SDN domain-based architecture to orchestrate fog computing environments. Comput. Netw. **221**, 109512 (2023). https://doi.org/10.1016/j.comnet.2022.109512

19. Zhang, Z., Ma, L., Leung, K.K., Le, F.: More is not always better: an analytical study of controller synchronizations in distributed SDN. IEEE/ACM Trans. Netw. **29**(4), 1580–1590 (2021). https://doi.org/10.1109/TNET.2021.3066580
20. Zeng, Z., Zhang, X., Xia, Z.: Intelligent blockchain-based secure routing for multidomain SDN-enabled IoT networks. Wirel. Commun. Mobile Comput. **2022**(1), 5693962 (2022). https://doi.org/10.1155/2022/5693962
21. Harbin, A., Baldwin, K., Mhatre, J., Lee, A., Lee, H.: Machine learning load balancing algorithms in SDN-enabled massive IoT networks. In: 2023 IEEE International Performance, Computing, and Communications Conference (IPCCC), pp. 202–203 (2023). https://doi.org/10.1109/IPCCC59175.2023.10253834
22. Shona, M., Sharma, R.: Implementation and comparative analysis of static and dynamic load balancing algorithms in SDN. In: 2023 International Conference for Advancement in Technology (ICONAT), pp. 1–7 (2023). https://doi.org/10.1109/ICONAT57137.2023.10080430
23. Li, S., Zhang, Z., Hu, L., Guo, H.: Research on SDN load balancing and failure handling based on dynamic decision. In: 2024 Prognostics and System Health Management Conference (PHM), pp. 266–270 (2024). https://doi.org/10.1109/PHM61473.2024.00055
24. Zhetov, A., Kaijalainen, V., Mogilatov, V., Volkov, A., Muthanna, A.: Load balancing algorithm in SDN multi controller network. In: Proceedings of the 7th International Conference on Future Networks and Distributed Systems, pp. 689–693 (2024). https://doi.org/10.1145/3644713.3644830
25. Alrammahi, M.A.M., Bhaya, W.S.: A state-of-the-art survey and taxonomy for load balancing metrics in SDN networks. In: 2022 2nd International Conference on Advances in Engineering Science and Technology (AEST), Babil, Iraq, pp. 606–611 (2022). https://doi.org/10.1109/AEST55805.2022.10413168