Postprint: The Evolutionary Process of Blockchain P2P Network Protocols

Authors: Wu Yue, Li Junxiang

Date: 2018-09-12T00:00:00+00:00

Abstract

Blockchain is a distributed system that employs peer-to-peer (P2P) networks as the communication protocol for its network layer. The peer-to-peer propagation and verification mechanisms together constitute the cornerstone of the blockchain network layer. In P2P networks, all nodes jointly provide network services, weakening or even eliminating central servers. These unique characteristics enable blockchain systems to break free from the constraints of central servers, achieving true distribution and decentralization. Addressing this characteristic of P2P network protocols, we conducted a detailed study of the P2P protocols of three mainstream blockchain systems—Bitcoin, Ethereum, and Hyperledger Fabric—through source code analysis and official documentation. By examining the evolutionary changes in blockchain P2P protocols, we analyzed the advantages and disadvantages of different protocols, proposed analytical criteria, and performed quantitative evaluations. This comparison aims to provide valuable insights and guidance for future research on blockchain network protocols.

Full Text

Preamble

Title: Evolution Process of Blockchain P2P Network Protocol

Authors: Wu Yue, Li Junxiang† (Business School, University of Shanghai for Science & Technology, Shanghai 200093, China)

Abstract: Blockchain is a distributed system that employs peer-to-peer (P2P) networks as the communication protocol for its network layer. The propagation and verification mechanisms of P2P jointly constitute the cornerstone of the blockchain network layer. In P2P networks, all nodes collectively provide network services, weakening or even eliminating the need for central servers. These

unique features enable blockchain systems to break free from central server dependencies, achieving true distribution and decentralization. Focusing on these characteristics of P2P network protocols, this paper conducts a detailed study of the P2P protocols in three mainstream blockchain systems—Bitcoin, Ethereum, and Hyperledger Fabric—through source code analysis and official documentation. By examining the evolution of blockchain P2P protocols, we analyze the advantages and disadvantages of different protocols, propose evaluation criteria, and provide quantitative assessments. This comparison aims to offer valuable insights and guidance for future research on blockchain network protocols.

Keywords: blockchain; P2P network; Bitcoin; Ethereum; Hyperledger Fabric; Gossip; Kademlia

0 Introduction

Blockchain is a distributed system composed of various technologies including encryption mechanisms, storage mechanisms, and consensus mechanisms, enabling trusted peer-to-peer transactions without central servers. Its most significant characteristics are decentralization and distribution. Participating nodes collectively provide system services through blockchain consensus mechanisms, thereby fulfilling functions typically performed by centralized financial intermediaries. Consensus mechanisms represent the agreement of most network nodes on transaction states within a specified time period, achieved through algorithms such as Proof of Work (POW), Proof of Stake (POS), and Practical Byzantine Fault Tolerance (PBFT). Consistency, meanwhile, requires most nodes to maintain identical data content within a specified timeframe, thereby implementing consensus through consensus algorithms.

Traditional centralized network systems require powerful, stable central servers and sufficient bandwidth to support client nodes. Through P2P networks, blockchain systems can achieve rapid data synchronization and implement consensus mechanisms without central servers. As shown in [Figure 1: see original paper], consensus algorithms and P2P networks respectively undertake tasks of consensus and consistency, jointly forming the consensus mechanism of blockchain and thereby realizing its distributed and decentralized nature.

Roger Wen [1] proposed classification criteria for P2P networks based on centralization and structuralization. Zhou Wenli et al. [2] presented five structural classifications of P2P networks. Wang Xuelong et al. [3] discussed P2P network classification models according to key P2P technology types. Liu Xiaomin et al. [4] provided detailed descriptions of P2P network retrieval mechanisms. Tan Qingfeng et al. [5] proposed covert communication methods based on P2P networks. Shao Qifeng and Liu Aodi et al. [6,7] introduced the blockchain network layer within blockchain systems. Gao Feng et al. [8] proposed a neighbor node identification and traceability model. Ye Congcong et al. [9] developed models for evaluating and inspecting blockchain security. Fukumitsu et al. [10] proposed a blockchain-based confidential sharing P2P storage protocol. Gat-

termayer et al. [11] developed a blockchain-based multi-level scoring system for P2P clusters.

Blockchain products have attracted substantial investment due to their wealth creation effects, driving rapid technological innovation and evolution. However, related academic research has seriously lagged behind and urgently needs to catch up. As of July 2018, searches for "blockchain" and "P2P network" on CNKI and Wanfang yielded no relevant Chinese literature, while Google Scholar searches for "blockchain" and "P2P network" produced only five Bitcoin-related documents [12].

This paper selects Bitcoin, Ethereum [13,14], and Hyperledger Fabric [15] as research objects in chronological order of blockchain product development. Bitcoin and Ethereum are the most influential distributed trading systems, with their token values and overall market capitalizations ranking first and second among virtual currencies. Hyperledger Fabric is the most influential enterprise-level blockchain product, widely applied across various domains. By examining blockchain product whitepapers and source code, we systematically analyze the P2P network technical principles, core advantages, and existing problems of multiple blockchain products, detailing the evolution and technological innovations in blockchain P2P networks.

1.1 P2P Applications in Blockchain

P2P networks offer numerous advantages for blockchain applications:

a) **Decentralization:** Blockchain resources and services are distributed across all participating nodes, maintaining blockchain network consistency through consensus mechanisms without requiring central systems.

b) **Scalability:** Blockchain nodes can freely join and exit, allowing the network system to expand according to node availability.

c) **Robustness:** The absence of central nodes eliminates attack targets. Participating nodes are distributed across the network, so damage to some nodes does not compromise the blockchain system.

d) **Cost-effectiveness:** There is no need for expensive professional equipment to build central servers or pay for costly broadband services, enabling ordinary users to participate in blockchain networks.

e) **Privacy protection:** Block information employs broadcast mechanisms that prevent localization of the broadcasting source node, protecting user communications from surveillance and safeguarding user privacy.

f) **Load balancing:** Blockchain avoids resource overload and network congestion by limiting node connection counts and other configurations.

1.2 Blockchain P2P Network Classification

Based on blockchain application characteristics and considering whether P2P networks are decentralized and whether node addresses are structured, we classify P2P networks into four categories, with network architectures shown in [Figure 2: see original paper].

1) **Centralized P2P Network:** Central servers are crucial for centralized network architectures, storing address information of connected nodes as the network core. Ordinary nodes obtain addresses of other nodes by connecting to central servers, thereby enabling node-to-node communication. The famous MP3 sharing software Napster [16] employed this centralized network structure, associating music files with nodes storing them. When users searched for music, the central server informed them of storage node addresses, enabling peer-to-peer connections to obtain music. The central server also determined overall network stability—once it failed, the entire P2P network would collapse. This resembles traditional centralized networks like financial institutions, except that P2P central servers only provide indexing while traditional central servers provide complete services. P2P network nodes can communicate directly with other nodes, whereas financial institutions do not permit inter-node communication.

2) **Fully Distributed Unstructured P2P Network:** Fully distributed P2P nodes can freely join and exit without central nodes. Node addresses lack structured unified standards, and the entire network structure resembles a random graph without fixed topology. Its successful application is the Gnutella [17] communication protocol, a peer-to-peer search system. Gnutella employs flooding technology to discover other nodes and implement random forwarding mechanisms, using TTL (time to live) decrement to control limited propagation of message communications. However, complete freedom means new nodes cannot learn about P2P network node information and thus cannot join the network. Gnutella also uses directory servers to facilitate nodes in obtaining addresses of other nodes. While fully distributed P2P networks offer greater freedom, they introduce node management problems—frequent node joining and exiting destabilize the overall network structure, and extensive broadcast messages not only waste resources but can even congest the network. Satoshi Nakamoto' s Bitcoin design employs this P2P network structure, where full distribution enables anyone and any node to participate, and unstructured organization allows nodes to both synchronize block data through blockchain P2P protocols and maintain anonymous privacy protection, embedding the concept of distributed decentralization deeply in blockchain.

3) **Fully Distributed Structured P2P Network:** The primary challenge for fully distributed networks lies in node address management—without fixed rules constraining inter-node relationships, node information cannot be precisely located and can only be found through flooding queries, which

consume significant network resources. Structured networks adopt Distributed Hash Tables (DHT) [18], standardizing different node addresses into fixed-length data through cryptographic hash functions. Successful cases include Chord [19] and Pastry [20]. The network structure remains random and unstructured like unstructured networks, but node management follows a fixed structural diagram. Ethereum converts node public keys from elliptic curve encryption algorithms into 64-byte NodeIDs as unique identifiers to distinguish nodes, enabling Ethereum to achieve precise node address lookup without central servers.

4) **Semi-Distributed P2P Network:** Combining advantages of both centralized and distributed models, nodes are classified into ordinary nodes and super nodes, forming a semi-distributed network structure. Each super node maintains partial network node addresses and file indexes, with super nodes collectively implementing central server functions. Super nodes themselves are distributed and can freely expand and exit, possessing distributed network advantages. Kazza [21] is a representative successful application. Hyperledger Fabric adopts this semi-distributed P2P network structure, dividing nodes into ordinary user nodes and super nodes (ordering nodes, endorsement nodes, etc.). Super nodes can be elected from ordinary nodes or self-configured, and the shutdown of a single super node does not affect system operation. All Fabric transactions must be authenticated by super nodes, and blocks are generated by super nodes, while ordinary nodes can synchronize with each other.

Different P2P network structures have their own advantages and disadvantages and adapt to different application scenarios. Bitcoin's original goal was to establish a decentralized trading platform to replace financial institutions like banks for value transfer and exchange. Meanwhile, centralized P2P networks do not conform to blockchain' s decentralized characteristics, and no related products exist. Centralized P2P networks resemble traditional intermediary institutions like banks, which blockchain products aim to replace. Therefore, the centralized systems discussed in this paper refer to bank-like financial intermediaries that lack P2P functions and serve as comparative benchmarks for evaluating blockchain products. Centralized P2P networks, similar to traditional intermediary institutions like banks, represent the systems that blockchain products seek to replace. The centralized systems discussed in this paper are compared with four P2P networks from perspectives of P2P applications, blockchain applications, decentralization, and whether nodes can be precisely located (Table 1).

2 Bitcoin' s P2P Network and Node Discovery

Bitcoin initiated the blockchain era, enabling any node to achieve decentralized, trusted Bitcoin transactions simply by launching a client. However, when a completely new node joins the Bitcoin network, its first task is network access. Due to Bitcoin' s complete decentralization and the free joining and exiting of

nodes, newly joined nodes have no means to obtain addresses of existing network nodes. To address this, Bitcoin designed three node discovery methods.

1) **Seed Nodes:** Although Bitcoin lacks central servers, it adopts Napster' s approach by establishing "seed" nodes. Bitcoin hard-codes a portion of long-term stable nodes into its code. These nodes provide initial entry points for network access during startup. New nodes use these stable nodes as intermediary connections to other nodes and can continuously obtain blockchain network node address lists, hence these nodes are also called seed nodes. As shown in Bitcoin source code [22] in [Figure 3: see original paper], these addresses are the seed addresses loaded during Bitcoin initialization.

2) **Address Broadcasting:** After connecting to a node, that node can serve as an intermediary to obtain addresses of other network nodes—this method is called address broadcasting, which includes two approaches:

   a) **Active Broadcasting (Push):** Node A pushes its own node information to other nodes through the `Net.AdvertiseLocal()` method. As shown in [Figure 4: see original paper], this is a proactive one-way process where Node B receives and saves the information locally without responding. Similarly, Node B can pass its own address to other connected nodes through push methods, enabling connected nodes to obtain Node B' s address information.

   b) **Active Acquisition (Pull) of Other Node Addresses:** Merely pushing its own address only reaches directly connected nodes and is insufficient for widespread broadcasting. Bitcoin implements another address broadcasting mechanism through the `Net.GetAddresses()` method for actively pulling addresses. As shown in [Figure 4: see original paper], Node A requests addresses from Node B and obtains them. Node B returns the address information it has collected. This one-request-one-response approach avoids wasting network resources. Node B can also request addresses from other nodes to expand its own address library. Addresses actively pushed by other nodes are also returned to requesting nodes during pull actions, achieving the goal of address broadcasting.

   Bitcoin' s two address acquisition methods are independent, preventing attackers from forging numerous fake addresses and widely disseminating them, which could prevent normal nodes from accessing the Bitcoin network or connecting to fake networks.

3) **Address Database:** To avoid connection and bandwidth limitations of seed nodes, Bitcoin nodes obtain other node information through broadcasting after network access and save the results for future use. The Bitcoin client stores acquired node information in a local `peers.dat` file using LevelDB format. If Node A has no data communication with an established connection Node B, Node A periodically sends ping messages

containing an 8-bit random number to Node B. Node B replies with a pong message containing the ping random number to maintain the connection. If a node remains unresponsive for 20 minutes, it is considered disconnected from the network. After startup, the Bitcoin client initiates a timed cyclic task to check whether connected nodes are online and saves failed nodes in a local `banlist.dat` file.

Through these three methods, Bitcoin maintains stable network access, implements address list maintenance for frequently joining and exiting nodes, and ensures nodes can freely join the network without central institutions, thereby guaranteeing Bitcoin P2P network stability.

3 Ethereum' s P2P Network and Node Discovery

Ethereum developed from Bitcoin, borrowing many Bitcoin concepts—even half of its whitepaper describes Bitcoin functions. Therefore, Ethereum not only implements a Bitcoin-like trading system but also aims to build a blockchain-based ecosystem, expanding into decentralized applications (DAPPs) dependent on Ethereum. Ethereum developers created Whisper [23], a distributed messaging platform, and Swarm, a distributed storage and content distribution service platform. These applications rely on nodes' ability to precisely locate other node addresses as needed—functionality that unstructured P2P networks cannot satisfy. Consequently, Ethereum adopts structured P2P networks, achieving structuration through DHT technology. DHT hashes P2P network nodes into standard-length data using hash algorithms, forming a giant hash table across the entire network. Each participating node possesses a portion of the hash table and maintains its own data, with the hash table distributed across various P2P network nodes. Any node accessing the P2P network has its own ID located within the hash table, enabling it to find more nodes through DHT and be precisely located by other nodes based on ID values. Although DHT supports free node joining and exiting, its complex maintenance mechanisms make it unsuitable for high-frequency node changes. Ethereum uses the Kademlia [24] (Kad) protocol, a DHT protocol that enables fast and accurate address lookup.

3.1 Kad Introduction

Compared with traditional DHT, Kad offers several advantages:

a) Kad's query requests are parallel and asynchronous, avoiding query failures caused by node exit or faults.

b) Kad simplifies the number of configuration messages nodes must send to understand each other and automatically exchanges configuration information during lookups.

c) Kad uses binary trees to divide nodes into multiple Kad buckets, simplifying query structures. It employs unidirectional XOR algorithms to calculate distances, ensuring all queries for the same TargetID converge

to the same path, reducing inter-node query network consumption and improving query efficiency.

d) The algorithm nodes use to record whether other nodes are available can prevent common denial-of-service (DoS) attacks.

Compared with traditional Kad, Ethereum's Kad differs in convergence methods. Traditional Kad uses selfID as the convergence target, while Ethereum uses not only selfID but also randomly generated TargetIDs as convergence targets to generate a larger-range hash table.

3.2 Ethereum Node Discovery

1) **Seed Nodes:**

   a) **Hard-coded Seed Nodes:** Like Bitcoin and other P2P software, new nodes use hard-coded seed nodes to access the Ethereum P2P network. Ethereum seed node source code [25] structure is shown in [Figure 5: see original paper], consisting of three parts: node information, network address URL, and data protocol. Node information is a 128-bit node ID converted to hexadecimal, the network address URL is the IP address after the separator @, and the data protocol is the TCP listening port (30303) or UDP discovery port (30301).

   Querying the addresses in [Figure 5: see original paper] using IP tools reveals they originate from Ireland, the United States, Brazil, Australia, Singapore, and Germany. High-latency international communications cannot meet domestic node network requirements. To facilitate network access, EthFans initiated the Spark Node Program. EthFans encourages domestic organizations and individuals interested in Ethereum to share their stable node address information, screens high-quality nodes, packages them into a `static-nodes.json` file, and releases it. Ethereum users can connect to more domestic nodes after downloading, accelerating the domestic Ethereum network speed. Similar configuration files include `trusted-nodes.json` for storing trusted node information. These files are not content-restricted but have fixed filenames and formats, hard-coded in configuration files, with configured nodes loaded during Ethereum startup. Static node information is shown in [Figure 6: see original paper].

   b) **Mining Pool Seed Nodes:** Mining pools also provide seed nodes.

2) **Address Database:** Similar to Bitcoin, Ethereum uses an address database to store historically connected nodes. Ethereum employs LevelDB as the history file format, generating multiple `*.ldb` history database files. For nodes connecting to the Ethereum network for the first time, the address database is empty and gradually fills through address broadcasting. For nodes reconnecting to Ethereum, during startup Ethereum uses the `loadSeedNodes()` method to read both seed nodes

and historical data for fast and efficient connection to the Ethereum P2P network.

3) **Address Query:** The greatest feature of Ethereum' s P2P network is its ability to perform address queries. Ethereum Kad parameter configuration is shown in [Figure 7: see original paper]:

- NodeID = Node Identifier (node public key length after elliptic curve encryption / 8) = 64-bit length data standardizing each node's length information.
- NBuckets = Kad buckets, totaling 32*8/15 = 17, representing the number of routing tables maintained by the node itself. More K-buckets enable faster lookup but consume more resources.
- bucketMinDistance: The nearest distance is 256-17 = 239, which is the distance standard between Kad nodes.
- bucketSize: Each Kad bucket contains 16 nodes, representing the number of nodes per routing table. Larger bucketSize enables faster lookup but consumes more resources.
- refreshInterval: Refresh timer = 30 * time.Minute, setting refresh frequency.
- revalidateInterval: Validation timer = 10 * time.Second, setting node validation frequency.
- copyNodesInterval: Persistence timer = 30 * time.Second, setting persistence frequency.

a) **Steps for Kad to Find Random Nodes:** (a) Create a database instance `db` for persistent information storage. (b) Load parameters through the `newTable()` method to generate Kad bucket object instances and initialize K-bucket data. (c) The `loadSeedNodes()` method loads seed nodes and old node data, calculates the ID hash values of seed nodes, selects Kad buckets by comparing with bucketMinDistance, and writes data. (d) Use Go language' s goroutine runtime library, a concept similar to concurrent threads, to make `db` execute the cyclic `loop()` method. (e) The `loop()` method performs three tasks: first, reloads seed nodes; second, finds three randomly generated nodes and searches for nodes around them; finally, saves Kad bucket node information into `db`.

b) **Steps for Kad to Find Fixed Nodes:** (a) First, use the `closest()` method in the current bucket to find nodes nearest to the target node. (b) If not found after one search, use the `Lookup()` method to search again among surrounding nodes. (c) `Lookup()` asks known nodes to find neighbor nodes, which recursively find their nearest alpha (3) nodes through `findnode()`.

4) **Address Broadcasting:** With a fixed Kad table structure, each node has a unique fixed location based on its NodeID. Therefore, Ethereum has no method for actively broadcasting its own address. However, through the

`doRefresh()` method' s `Lookup()` method, it searches for and connects to the 16 nodes nearest to itself, then randomly generates three targetIDs and uses `Lookup()` to find nodes. After finding them, it uses the `pingpong()` method for bidirectional communication. The peer node will compare connected nodes by distance, save them to its own K-bucket, and store node information in the address database, thereby achieving the goal of informing other nodes of its own address.

4 Hyperledger Fabric' s P2P Network and Node Discovery

With the rapid development of Bitcoin and Ethereum, blockchain technology applications have expanded beyond trading systems. For certain enterprise-level problems, blockchain also provides perfect solutions, such as logistics chains and supply chains that require goods tracking and tampering prevention in decentralized systems. Hyperledger Fabric emerged to address these needs. As an enterprise-level blockchain application, Fabric assigns different permissions to different nodes, and transaction processing must pass through super nodes. Although Fabric does not achieve full decentralization, it can optimize network efficiency by dividing workloads among different nodes. To ensure blockchain network security, trustworthiness, and measurability, Fabric adopts Gossip as its P2P network propagation protocol. Gossip supports super node network architectures where super nodes possess stable network services and computing capabilities. Super nodes are responsible for Fabric' s transaction ordering and new block broadcasting functions, maintaining Fabric network information updates and node list management.

4.1 Gossip Introduction

Gossip [26] is based on infectious disease transmission mechanisms and is widely used as an underlying communication protocol in distributed systems. Gossip is not a new P2P network concept. Compared with traditional flooding and routing algorithms, Gossip provides explicit network communication types. Facebook' s Cassandra [27] uses the Gossip protocol and is a popular distributed structured data storage solution [28]. Gossip on the Fabric network is responsible for maintaining new node discovery, cyclically checking nodes, removing offline nodes, and updating node lists. Fabric discovers new blocks or local errors and missing blocks through broadcast communication with nodes in the node list, updating and maintaining local ledger data. The Gossip protocol has three communication types:

a) **Push:** Active information pushing. Assuming Node A randomly selects N nodes from the node list as target nodes, it proactively pushes information containing Node A' s own ID to the N nodes. The N nodes receive and compare the information with their own data, updating themselves. The N nodes can also continue pushing information to other nodes.

b) **Pull:** Active information acquisition. Assuming Node A randomly selects

N nodes from the node list as target nodes, Node A encapsulates its own ID and requested content (such as node lists, block information) into request messages and actively sends them to the N nodes. The N nodes return corresponding information to Node A after receiving requests. Node A compares the returned information with local data and updates its local data.

c) **Push/Pull:** Mixed mode. Although it combines advantages of both push and pull, it consumes more network resources. Assuming Node A pulls information from N nodes and compares it with local data, it then pushes newer data to the N nodes, which update themselves after comparison.

4.2 Fabric Node Discovery

1) **Seed Nodes:** Fabric does not use hard-coded forms but instead uses configuration files through the `core.yaml` configuration file. Ethereum uses Viper [30] as its configuration file loading solution. Viper is an open-source configuration solution written by foreign computer enthusiast spf13. Viper doesn' t concern itself with file formats, can obtain local environment variables, retrieve configuration files from remote sources, and includes buffering mechanisms that enable dynamic loading of new configuration values and real-time effectiveness without restarting services. This is crucial for the Fabric network, as super node administrators can update configuration information and make it effective without affecting system operation or requiring system restarts. In contrast, Bitcoin and Ethereum require full node voting for changes due to decentralization, and failure to reach consensus causes forks, such as Bitcoin forking into Bitcoin (BTC) and Bitcoin Cash (BCC), and Ethereum forking into Ethereum (ETH) and Ethereum Classic (ETC). Seed node-related source code is shown in [Figure 9: see original paper]. The seed node loading process is as follows:

   a) First, start the Gossip service with the `NewGossipService()` method, which calls the `g.connect2BootstrapPeers()` method to load seed nodes during service startup.

   b) Through the `g.conf.BootstrapPeers()` method, read the `core.yaml` configuration file to obtain bootstrap super node values and save them as endpoint address temporary lists.

   c) Then start the connection `g.disc.Connect()` method, using endpoints as parameters.

   d) Use the `d.createMembershipRequest()` method to generate request information, assign it to temporary variable `m`, and encrypt and sign it as request information `req`.

   e) Combine the endpoint and its own PKIid, use the `go d.sendUntilAcked()` method to send the `req` information to the corresponding endpoint.

f) Update the address library after receiving return information from endpoint nodes.

2) **Address Database:** Fabric adopts the super node form, requiring connection to super nodes. Moreover, as an enterprise-level application, there is no need to save historical data because storing data and verifying whether nodes are online consumes substantial resources. Since super nodes exist, one only needs to fetch addresses from super nodes each time. Super nodes also fulfill the work of address databases. After loading seed addresses, the obtained node list is saved in memory rather than in file form as in Bitcoin and Ethereum. Nodes parse received messages, check whether nodes are normal, maintain node lists, and also periodically communicate with connected nodes. Once a connected node fails to respond beyond the configured time, it is removed from the node list and added to the offline list.

3) **Address Broadcasting:** Gossip starts a cyclic method `g.syncDiscovery()` that periodically performs node lookup work every S seconds. By randomly selecting N nodes from the node list, it pushes requests for node list information to synchronize its own node list. In `core.yaml`, the default configuration sets S to 4 seconds and N to 3 nodes, which can be changed according to requirements. While pushing, it also passes its own node information to other nodes and broadcasts it throughout the entire network.

4) **Address Query:** Fabric currently does not support address queries because there are no business scenarios requiring precise queries for addresses of specific IDs. However, due to the existence of super nodes that possess address information of all connected nodes, Fabric could be extended by adding functionality to support lookups.

5 Analysis and Comparison

Each P2P network structure has its own advantages and disadvantages. Different blockchain application scenarios and business requirements employ different P2P network structures. The main application scenario for blockchain is token trading systems. To more intuitively compare the P2P network structures of different blockchain systems, we use bank-like intermediary systems as substitutes for centralized network structure products. Network structure belongs to the architectural design level, differing from functional points, so this paper only describes the network structure composition of different blockchain products without scoring comparisons.

Blockchain, as an innovative product, is characterized mainly by decentralization, anonymity, trustworthiness, and smart contracts—these are functional innovations. This paper analyzes and compares the underlying architecture and technical characteristics of blockchain products by examining their source code implementation, evaluating them across four dimensions: degree of decentral-

ization, privacy protection, security, and application richness. Each dimension represents different functional points with equal weight.

Although P2P networks are not innovative features of blockchain, they serve as the underlying technical support for blockchain, and network quality determines the success or failure of blockchain products. This paper analyzes P2P networks as a functional point of blockchain from a source code technical perspective, evaluating blockchain P2P network quality through the dimension of access efficiency to enable reasonable comparisons.

To quantitatively evaluate the performance of different P2P network structures in blockchain, we use a 1-4 scoring scale for different products, where higher scores indicate better performance. The scoring results are shown in Table 2 .

Table 2: Network Protocol Scoring Table

Dimension	Financial Intermediary System	Bitcoin	Ethereum	Hyperledger Fabric
Degree of Decentralization	1	4	4	2
Node Network Access Efficiency	4	2	3	3
Security	4	3	3	3
Privacy Protection	2	4	3	3
Application Richness	2	1	4	3

1) **Degree of Decentralization:** P2P networks are the cornerstone of blockchain distribution. Since Bitcoin' s success, the concept of blockchain distribution and decentralization has become deeply rooted. Traditional central financial institutions require all transactions to be processed by central systems, and participating nodes cannot communicate directly, representing complete centralization. Both Bitcoin and Ethereum are fully decentralized, with all nodes having equal permissions. Although seed nodes exist, architecturally they are no different from ordinary nodes—only relatively high-performance and high-stability nodes. They perform equally in decentralization. Fabric, however, is equivalent to a distributed centralized system where super nodes possess centralized server functions while also being distributed clusters with distributed characteristics. Therefore,

Fabric' s decentralization performance lies between bank-like systems and Bitcoin/Ethereum.

2) **Node Network Access Efficiency:** Node network access requires two tasks: discovering nodes to access the blockchain network and synchronizing block ledger data and other information to use blockchain services. Bank-like centralized systems have all maintenance work performed by central systems, and nodes can use services immediately upon access, achieving the highest access efficiency. Without central nodes and with ledgers stored locally, Bitcoin and Ethereum must synchronize all blocks before consensus mining can occur. However, Bitcoin's node address management does not require building and maintaining DHT—it can broadcast directly without lookup, making node discovery efficiency higher than Ethereum' s. Fabric has flexibly configurable super nodes, so its access efficiency is higher than Bitcoin' s but weaker than bank-like financial systems with powerful server capabilities.

3) **Security:** Common blockchain attack methods include node forgery (impersonating other nodes for transactions) and DoS attacks (attacking service providers to paralyze systems). Centralized financial systems employ multiple encryption methods and bind personal information, combining facial recognition [31], PIN passwords [32], smart cards bound to personal information [33], and dual protection combined with passwords [34] to ensure transaction security. Strong network and server capabilities also make DoS attacks very difficult. Bitcoin and Ethereum adopt account-node separation designs, do not restrict node joining and exiting, making node forgery meaningless. All nodes provide services through consensus mechanisms without central systems—DoS can only attack the entire network, which is enormously costly. To date, there have been no successful DoS attacks on Bitcoin or Ethereum. Fabric uses CA nodes to assign keys to prevent node forgery and utilizes distributed super nodes to resist attacks. Fabric adds node management on top of Bitcoin and Ethereum' s security, but its authentication requirements are weaker than bank-like systems, so its security lies between the two.

4) **Privacy Protection:** Identity information privacy protection has two levels: the basic level is identity marker protection, and the advanced level is non-traceability of user login behavior. Due to blockchain' s decentralized characteristics, privacy issues in blockchain correspond to whether node IP addresses are anonymous at the basic level and whether identity information is confidential at the advanced level. Node IP addresses contain physical geographic information that can serve as node identity markers. However, multiple users may share one physical node, while identity information can precisely locate users and track user behavior. Ethereum' s DHT stores node IP address information, making it vulnerable to targeted attacks. In terms of accounts, like Bitcoin, it does not contain identity information to protect privacy. Fabric' s node information is stored in

central servers. Although nodes connect randomly, IP addresses, identity information, and other content are fully exposed to super nodes—public to super nodes but anonymous to ordinary nodes. Therefore, its anonymity lies between centralized and distributed systems. For bank-like centralized systems, identity information is part of authentication. ID card information is public to central servers, but nodes cannot communicate with each other. Bank-like centralized systems are responsible for protecting identity information, but once the central system misbehaves, privacy information may be leaked. Bitcoin uses flooding for broadcasting, making it impossible to locate whether a node is the original source or a forwarding node. It protects node address information and uses anonymous accounts to protect user identity information, achieving the highest level of privacy protection.

5) **Application Richness:** P2P networks enable inter-node communication within blockchain systems, forming the underlying network communication foundation. Based on this, blockchain extends a series of applications at the upper layer. Although bank-like systems have rich functions, because they are centralized systems, participants cannot publish smart contracts or distributed extended applications, limiting application expansion. Bitcoin does not support smart contracts, and its main function remains the trading system, making it the least extensible. Ethereum' s DHT supports precise lookup, enabling precise location of nodes or nodes within ranges for functions like peer-to-peer communication and peer-to-peer file transfer, demonstrating the best performance in application richness. Representative applications include CryptoKitties, where the most expensive cat sold for 246.95 Ether (approximately $118,000 or ¥770,000). Hyperledger Fabric, as an enterprise-level blockchain application, adopts consortium chain forms, lying between centralization and decentralization. In [Figure 10: see original paper], it presents a pentagonal shape with relatively balanced performance across dimensions—no obvious advantages but also no obvious weaknesses.

Figure 10: Blockchain Product Comparison Radar Chart (showing the overall capability distribution and focus of each product across the five dimensions)

Bank-like system networks, although not P2P, serve as typical centralized networks and benchmarks for comparison with distributed blockchain products. Their high access efficiency and security guarantee system stability. While sacrificing distribution and scalability, stability and efficiency are paramount for central systems. Since Bitcoin' s 2008 release, its overall architecture has not changed significantly, functionally only meeting trading needs. Its creator, Satoshi Nakamoto, disappeared after Bitcoin' s stable operation. Bitcoin is characterized by decentralization and anonymous privacy protection, laying the foundation for blockchain. Even today, a decade after blockchain' s rapid development, Bitcoin remains remarkable. Although Bitcoin scores only 13 points

overall, it performs best in privacy protection and decentralization. Ethereum attempts to build a blockchain-based ecosystem, creating the concept of DAPPs. Its application richness is the highest among all blockchains. Hyperledger Fabric, as an enterprise-level blockchain application, adopts consortium chain forms and demonstrates balanced performance across dimensions.

6 Conclusion

Blockchain technology has constructed a trusted, decentralized distributed application system, removing completely centralized systems and reducing the cost of social value exchange, with broad application prospects. Blockchain technology is no longer limited to Bitcoin' s pure trading system but extends to Ethereum' s blockchain-based expanded applications. It is no longer limited to Bitcoin' s complete decentralization but extends to Fabric' s semi-distributed networks to solve alliance-based business problems. To adapt to different application scenario requirements, blockchain P2P network protocols continue to evolve. This paper organizes and summarizes the evolution process of blockchain P2P networks from the perspectives of underlying source code and upper-layer application requirements, aiming to provide beneficial inspiration and reference for future research. As blockchain technology develops and application scenarios gradually enrich, blockchain network protocols will continue to evolve—this will be an ongoing evolutionary process.

References

[12] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system [EB/OL]. (2008-10-31) [2018-07-06]. http://www.bitcoin.org/bitcoin.pdf.

[13] Buterin V. Ethereum white paper [EB/OL]. [2018-07-06]. https://github.com/ethereum/wiki/wiki/White-Paper.

[14] Buterin V. Ethereum homestead documentation [EB/OL]. [2018-07-06]. http://ethdocs.org/en/latest/network/connecting-to-the-network.html#the-ethereum-network.

[15] Androulaki E, Manevich Y, Muralidharan S, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains [C]// Proc of the 13th EuroSys Conference. New York: ACM Press, 2018: 1-15.

[16] Garnett N. Digital rights management, copyright, and Napster [J]. ACM SIGecom Exch, 2001, 2(2): 1-5.

[17] Lyu Qin. Search and replication in unstructured peer-to-peer networks [C]// Proc of the 16th International Conference on Supercomputing. New York: ACM Press, 2002: 84-95.

[18] Jain S, Mahajan R, Wetherall D. A study of the performance potential of DHT-based overlays [C]// Proc of Usenix Symposium on Internet Technologies and Systems. Berkeley: USENIX Association Press, 2003.

[19] Stoica I, Morris R, Karger D, et al. Chord: A scalable peer-to-peer lookup service for internet applications [C]// Proc of Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. New York: ACM Press, 2001: 149-160.

[20] Rowstron A I T, Druschel P. Pastry: scalable, decentralized object location, and routing for large-scale peer-to-peer systems [C]// Proc of IFIP/ACM International Conference on Distributed Systems Platforms. Heidelberg: Springer-Verlag Press, 2001: 329-350.

[21] Jiang Cheng. Research on major model of peer-to-peer network [J]. Information Security and Technology, 2013, 4(6): 69-71.

[22] Nakamoto S. Bitcoin source code [EB/OL]. [2018-08-19]. https://github.com/bitcoin/bitcoin.

[23] James Ray. Whisper official document [EB/OL]. [2018-06-27]. https://github.com/ethereum/wiki/wiki/Whisper.

[24] Maymounkov P, Mazieres D. Kademlia: a peer-to-peer information system based on the XOR metric [C]// Proc of the 1st International Workshop on Peer-to-Peer Systems. London: Springer-Verlag, 2002: 53-65.

[25] Buterin V. Ethereum go-ethereum source code [EB/OL]. [2018-08-19]. https://github.com/ethereum/go-ethereum.

[26] Demers A, Greene D, Houser C, et al. Epidemic algorithms for replicated database maintenance [C]// Proc of the 6th Annual ACM Symposium on Principles of Distributed Computing. New York: ACM Press, 1987: 1-12.

[27] Lakshman A, Malik P. Cassandra: a decentralized structured storage system [J]. ACM SIGOPS Operating Systems Review, 2010, 44(2): 35-40.

[28] Li Li. Decentralized structured storage systems—Cassandra [J]. Oriental Enterprise Culture, 2013(15): 142-143.

[29] Foundation L. Hyperledger fabric source code [EB/OL]. [2018-08-19]. https://github.com/hyperledger/fabric.

[30] Spf13. Viper: go configuration with fangs [EB/OL]. [2018-07-06]. https://github.com/spf13/viper.

[31] Liu Xiaoyou. The cmb bank to brush face withdraw money “face recognition” next city [N]. The Securities Times, 2015-10-15(A06).

[32] Wang Ding, Gu Qianchen, Huang Xinyi, et al. Understanding human-chosen PINs: characteristics, distribution and security [C]// Proc of ACM on Asia Conference on Computer and Communications Security. New York: ACM Press, 2017: 372-385.

[33] Wang Ding, Wang Ping. Two birds with one stone: two-factor authentication with security beyond conventional bound [J]. IEEE Trans on Dependable and Secure Computing, 2018, 54(4): 708-722.

[34] Krol K, Philippou E, Cristofaro E D, et al. "They brought in the horrible key ring thing!" analysing the usability of two-factor authentication in UK online banking [EB/OL]. [2018-08-19]. https://www.researchgate.net/publication/271140632_They_brought_in_the Factor_Authentication_in_UK_Online_Banking.

[35] Wang Ding, Wang Ping. On the anonymity of two-factor authentication schemes for wireless sensor networks: attacks, principle and solutions [J]. Computer Networks, 2014, 73(C): 41-57.

Note: Figure translations are in progress. See original paper for figures.