



THE UNIVERSITY OF HONG KONG

FINAL YEAR PROJECT

**Augmenting Blockchain System Efficiency and
Security From Peer-to-Peer Network Layer**

INTERIM REPORT

by

Haoran Qiu, Tao Ji

Department of Computer Science

Supervised by

Dr. Heming Cui

Department of Computer Science

Jan 20th, 2019

Abstract

Blockchain records the transactions among parties by leveraging the decentralized nature of a distributed ledger. The verifiability and immutability of records led to the emergence of cryptocurrencies in finance industry. Starting from Bitcoin, the first blockchain application, there have been various blockchains using different consensus protocols to achieve faster and more efficient transaction process. However, they are still inefficient because there are redundant messages flooding in the underlying Peer-to-Peer (P2P) Network. Experiments and studies have shown that the network layer became the bottleneck of further increasing the transaction rate. In this project, we make use of Intel Software Guard eXtensions (SGX) to facilitate the design of a new P2P Network Protocol. It makes use of geographical locality of nodes and achieves $O(N)$ message complexity and $O(\log N)$ time complexity on broadcast. It can also be proved safe against various attack models, relying on the guarantee of code and execution integrity provided by SGX. The protocol can be utilized in blockchain systems to achieve fast message broadcasting and improve the throughput of the whole blockchain system. Currently, the project is at the final stage of implementation. Project details and current deliverable can be find through: i.cs.hku.hk/fyp/2018/fyp18006/.

Acknowledgments

Conducted by Haoran Qiu and Tao Ji, this research project is supported by Department of Computer Science, The University of Hong Kong and the Dr. Heming Cui's System Lab. We thank Dr. Heming Cui for supervising us and providing valuable suggestions. We would also like to show our gratitude to Shixiong Zhao, Xusheng Chen, and Zijian Bao for assistance and giving many suggestions all the time. Last but not least, we want to thank Miss Mable for giving us a lot of helpful suggestions in the project progress reports. The project would not be on the right track without the help of the above mentioned people.

Table of Contents

Abstract	I
Acknowledgement	II
List of Figures	V
List of Tables	V
List of Abbreviations	VI
1 Introduction	1
1.1 Background	3
1.1.1 Consensus Protocols	3
1.1.2 P2P Network	3
1.1.3 Trusted Execution Environment	4
1.2 Motivation	5
1.3 Objective, Scope and Deliverables	5
1.4 Report Outline	6
2 Literature Review	8
2.1 Peer-to-Peer Overlay Networks	8
2.1.1 Data Look-up	9
2.1.2 Message Dissemination	10
2.2 P2P Networks in Blockchain Systems	11
2.2.1 Requirements from Blockchain Systems	11
2.2.2 State-of-Art P2P Networks in Blockchain Systems	11
3 Methodology	13
3.1 Protocol Overview	13
3.1.1 Within-ring Broadcast Mechanism	15

3.2	System Implementation	17
3.3	Evaluation	18
4	Project Status	19
4.1	Schedule and Progress	19
4.2	Challenges and Mitigation	21
4.3	Next Step	21
5	Conclusion	23
6	References	24

List of Figures

3.1	Network Topology	14
3.2	Illustration of a broadcast process from a node to the whole network in a three-layer network.	17
3.3	Broadcast by K-ary Distributed Spanning Tree	17

List of Tables

1.1	Consensus Algorithms and Their Tolerated Percentages	3
4.1	Project Schedule	20

List of Abbreviations

DHT Distributed Hash Table.

HLF HyperLedger Fabric.

NFV Network Function Virtualization.

P2P Peer-to-Peer.

PoET Proof-of-Elapsed Time.

PoL Proof-of-Luck.

PoM Proof-of-Membership.

PoS Proof-of-Stake.

PoW Proof-of-Work.

PoX Proof-of-X.

SGX Software Guard eXtentions.

TPM Trusted Platform Module.

1 Introduction

A blockchain is essentially a distributed ledger that permanently records the transactions among parties [1]. The transactions recorded are verifiable and resistant to modification. This feature of security has led to the emergence of cryptocurrencies that leverage the blockchain as their cornerstone, the most salient example being Bitcoin [2]. Despite the popularity of cryptocurrencies, general and all-purpose blockchains that accommodate various applications, such as Ethereum [3] have been proposed. However, although Ethereum is a Turing-complete system [3], it is in essence designed for the cryptocurrency based on it. Hence the Proof-of-Work (PoW) consensus has been utilized used to support the valuation of the cryptocurrency in the socioeconomic sense, which results in poor efficiency. To facilitate general-purpose applications in a more efficient manner, many other consensus protocols have been proposed to improve the performance of the blockchain in different scenarios (See Section 1.1.1).

With the invention of new consensus protocols, the transaction rates of blockchains are growing, which has resulted in an increasing frequency of broadcast operation in the networks. Unfortunately, studies [4, 5, 6] have shown that the network layer became the bottleneck of further growing the transaction rate. For example, EOS report [5] shows that EOS is sensitive to network latency. Higher network latency caused by other bandwidth-intensive applications in the same network or high client transaction input rate can both lead to significant drop of the throughput. HyperLedger Fabric (HLF) [4] also shows that its throughput is capped by the low efficiency of the underlying P2P network. This has attracted our research focus to the P2P network. While much effort has been devoted to the development of new consensus protocols, little effort has been made to improve those P2P networks dedicated for blockchains, even though the P2P network itself is no novel field of research.

One of the techniques that may enable further improvements of the existing P2P protocols under blockchains is Network Function Virtualization (NFV). With NFV, upper-layer

applications are allowed to control the lower-layer functionalities of the network such as routing. One problem of the existing P2P protocols is the bandwidth consumption caused by redundant messages, which is a waste of resource. Our key insight is that the Gossip algorithm [7] used to broadcast a message does not fit in the demand for P2P networks from blockchain systems. Although many improvements have been made on Gossip algorithm such as adding unique message ID, using Time-to-Live field to control flooding, using pull-version sending mechanism to reduce repeated messages, our evaluations show that they are not efficient enough, and still suffer from traffic congestion problem. It is not an optimal model for blockchain system. Unstructured networks are robust but lead to inefficiency in both node discovery and broadcast. Structured networks based on DHT are efficient in terms of node discovery and data look up (they are designed for this use [8]), however, Gossiping on such a network is inefficient. Since blockchain systems emphasize broadcast more than node discovery, a new P2P network protocol is needed.

Blockchain systems require two main functions from the underlying P2P network: peer discovery and message dissemination. For peer discovery, DHT-based protocols such as Chord [8], Pastry [9], Tapestry [10], and Kademlia [11] are used to achieve efficiency. For message dissemination, Gossip algorithms are used due to its robustness and simplicity. Under 50% failure, Gossip can send twice amount of messages to cover the remaining nodes. However, the robustness of Gossip exceeds too much of the requirement from the consensus protocols in most blockchain systems (see Table 1.1). As a side effect, Gossip generates redundant messages in the network which lead to traffic congestion. To tackle the problem, our key idea is that broadcasting using the DHT-based structure in a hidden and secure way can improve the broadcast efficiency in terms of both time complexity and message complexity.

The remaining part of introduction section includes detailed background, the motivation, the objective of this project, followed by an outline of this progress report.

Consensus Protocols	Tolerated Percentage	Examples
<i>Proof-of-Work</i>	$< 25\%$	Bitcoin
<i>Proof-of-Stake</i>	$< 51\%$	PeerCoin
<i>Practical BFT</i>	$< 33.3\%$	HyperLedger Fabric
<i>Distributed PoS</i>	$< 51\%$	Bitshares, EOS
<i>Ripple</i>	$< 20\%$	Ripple
<i>Tendermint</i>	$< 33.3\%$	Tendermint

Table 1.1: Consensus Algorithms and Their Tolerated Percentages

1.1 Background

1.1.1 Consensus Protocols

Despite the popularity of cryptocurrencies, general and all-purpose blockchain systems that accommodate various applications, such as Ethereum [3] have been proposed. However, although Ethereum is a Turing-complete system [3], it is in essence designed for the cryptocurrency based on it. Hence the Proof-of-Work (PoW) consensus has been utilized used to support the valuation of the cryptocurrency in the socioeconomic sense, which results in poor efficiency. To facilitate general-purpose applications in a more efficient manner, other consensus protocols have been proposed to improve the performance of the blockchain systems in different scenarios such as Proof-of-Stake (PoS) [12], Proof-of-Luck (PoL) [13], and Proof-of-Membership (PoM) [14]. Hyperledger Sawtooth [15] utilizes Intel Software Guard eXtensions (SGX, introduced below) to trust nodes and proposes Proof-of-Elapsed Time (PoET) believed to be highly efficient. Additionally, EOS [16] based on DPoS, NEO [17] based on DBFT, Conflux [6], Omniledger [18], and Hyperledger Fabric [4] are all examples of new blockchain systems which are claimed to achieve more than 10k transaction rate [19].

1.1.2 P2P Network

A blockchain is built based on a decentralized P2P network which is mainly utilized for propagating system information such as transactions, blocks or chain member updates

[20]. Each node in the network are equally treated, fully connected and their behaviour should be the same. Other than its decentralization feature, the strength of P2P network also includes self-organization, load-balancing, adaptation, and fault-tolerance. Though P2P network has many advantages as an underlying network layer of its atop applications like blockchains and Bittorrent, the communication on a P2P network suffers from message redundancy [21, 22].

There are two approaches to perform the broadcast operation: the flooding approach and the structured tree-based approach [23]. To reduce as many messages as possible, there should be less repeated messages sent to the same node. Meanwhile, the property required by the blockchain application on top of the network should not be affected. With Trusted Execution Environment (TEE) on Intel SGX, some key protocol components like routing algorithm and network information can be put in the hardware enclave [24, 25]. Relying on the guarantee of integrity of each node, the network protocol can be modified to achieve efficiency without sacrificing any security.

1.1.3 Trusted Execution Environment

Trusted computing has been defined to help systems to achieve secure computation, privacy and data protection. Originally, the Trusted Platform Module (TPM) allows a system to provide evidence of its integrity in a separate hardware module. In recent years, a new approach to address trusted computing has emerged, which allow the execution of arbitrary code within a confined environment that provides tamper-resistant execution to its applications - trusted execution environment (TEE) [26]. TEE is a secure, integrity-protected processing environment, consisting memory and storage capabilities [27].

Intel SGX is one popular instance of TEE which is a set of extensions to the Intel architecture that aims to provide integrity and confidentiality guarantees to security sensitive computation performed on a computer where all the privileged software (kernel, hypervisor, etc.) is potentially malicious [24]. Intel SGX provides two kinds of attestations (local

and remote) to prove that particular piece of code is running in a genuine SGX-enabled CPU [28] and also provides a trustworthy source of random number [29]. Currently there is one related work which uses Intel SGX to provide reliable broadcast for P2P network [30]. However, there is no related work on using SGX to improve asynchronous P2P network performance, which is the main focus of this project.

1.2 Motivation

The first blockchain system - Bitcoin suffers from exhaustion of computing power because each node needs to solve a computation puzzle before committing a block to the blockchain [2]. However, this mechanism is set to achieve consensus among peers in the blockchain. Though various improvements like GEEC, PoET, and other Proof-of-X (PoX)s are made to achieve consensus in a different way which satisfies both efficiency and consistency (security), currently there is no work been doing on the P2P layer. Repeated messages consume a lot of bandwidth in the network. Especially when the transaction rate of blockchain systems increases, high bandwidth usually leads to congestion of the network and data loss. Reducing redundant messages can improve the broadcast efficiency and save bandwidth for the network.

Another motivation for this project is Intel SGX. GEEC [28] and one work done in P2P field [30] make use of Intel SGX to provide guarantee on code and execution integrity. Combining SGX and P2P network could lead to a more efficient P2P protocol because routing table and other information or code can be stored in SGX.

1.3 Objective, Scope and Deliverables

Therefore, in this project, we propose the Hierarchically Geographically Fractal Random Rings (HGFRFRR), a new P2P network designed specifically for blockchains that features a geographically fractal circular DHT and a hybrid broadcast algorithm based on it. We

argue it is faster, sufficiently robust, more efficient and secure.

The scope of this project includes:

- i) the design of HGFRR, including the algorithms of DHT maintenance and the hybrid broadcast
- ii) the theoretical proof of the performance of the new design in terms of robustness, efficiency, and security.
- iii) implementing the model as a P2P network library for blockchains that can be leveraged by current blockchain systems with minimal adaptation; and
- iv) the evaluation of the implemented network in convergence speed, robustness, and efficiency.

The design and implementation of upper layers of the blockchain systems will not be included.

By the end of this project, we will deliver:

- i) the HGFRR design model with DHT maintenance and hybrid broadcast algorithms;
- ii) a P2P network library for blockchain systems implemented with the proposed network;
- iii) (if time allows) a blockchain system adapted to use our network library; and
- iv) a paper detailing the contributions of this project.

The project progress can be checked on the website: <https://i.cs.hku.hk/~fyp18006>

1.4 Report Outline

The rest of this report is structured in this way: Section 2 provides insights into related works. Current works on data look up and broadcast in P2P networks will be introduced. Section 3 presents the methodology applied in this project, which covers the protocol

design, system implementation and evaluation. Section 4 presents the current status of the project including what we have completed, what challenges we are facing and future plans. We also prepared solutions to mitigate those challenges. Section 5 concludes this project progress report.

2 Literature Review

This section presents the literature related to P2P network field. There are two important functions provided by the P2P network, the first one being data lookup (Section 2.1.1) and the second one being message dissemination (Section 2.1.2). Works on P2P networks under blockchain systems are studied and summarized in Section 2.2.

2.1 Peer-to-Peer Overlay Networks

There have been tremendous efforts and many technical innovations in the Internet broadcasting in the past three decades [31]. Internet protocol (IP) multicast represented an earlier attempt to tackle this problem but failed largely due to concerns regarding scalability, deployment, and support for higher level functionality. In contrast, Peer-to-peer based broadcast has been shown to be cost effective and easy to deploy. This new paradigm brings a number of unique advantages such as scalability, resilience, and effectiveness in coping with dynamics and heterogeneity. With Network Function Virtualization (NFV), upper-layer applications are allowed to control the lower-layer functionalities of the network such as routing.

The most important feature of the P2P architecture is that individuals in their network are equal in role and function. Although each individual may handle different requests, the actual resources provided may differ after specific quantification, but they can all simultaneously provide and consume resources. If the resources in the entire network, including but not limited to computing power, storage space, network bandwidth, etc., are regarded as a total amount, the resource distribution in the p2p network is dispersed among individuals (maybe not necessarily evenly distributed). Therefore, the P2P network architecture is naturally decentralized and distributed.

2.1.1 Data Look-up

Not every individual communicates with other peers in the network. This is actually a very important feature of the p2p network: an individual only needs to connect with a small part of nodes in the network. How many neighbors and how to connect vary from one to another. Basically, P2P networks are divided into unstructured and structured networks [32]. Unstructured P2P networks are simple and easy to deploy, and the individuals in the local area of the network can be arbitrarily distributed. When dealing with a large number of new individuals joining the network and the old individuals leaving the network (churn), unstructured P2P networks are very stable [33]. The disadvantage is that the efficiency of finding data in the network is too low. Because there is no foreseeable information, it is often necessary to send query requests throughout the network (at least most individuals) or use flooding, which will occupy a large part of the network resources and greatly slow down other businesses in the same network.

The individual distribution of structured P2P networks has been carefully designed, and the main purpose is to improve the efficiency of querying data and reduce the resource consumption caused by query data. The basic means to improve query efficiency is to index data [34]. The most common implementation of structured p2p networks is Distributed Hash Table (DHT) [35], which assigns a key to each data (value) to form $(key, value)$ pairs. Hashing can be used to uniquely identify a particular object from a group of similar objects by assigning each object a hash value. Nodes in the system are responsible for managing the mapping from keys to values in a way that minimizing the disruption caused to the participants. In this way, when looking for a certain item of data, the search area can be continuously reduced according to the key, thereby greatly reducing resource consumption. Although structured P2P network is efficient in data look-up, the robustness is a concern. Since each individual needs to maintain a large number of neighboring individuals, when the churn events in which a large number of new and old individuals frequently join and leave occur in the network, the performance of the entire network will be greatly deteriorated. Part of the resources are consumed when updating

the neighbor list (including the update of the neighbor list, and the stored list is updated between each other), and the keys of many peers also need to be redefined. Most used DHTs are Chord [8], Pastry [9], and Tapestry [10].

2.1.2 Message Dissemination

Gossip based protocols are developed for providing high reliability and scalability of message delivery [36]. Gossip protocols are highly used for reducing control message overhead [37]. Gossip protocols are scalable because they do not require as much synchronization as traditional reliable multicast protocols. In Gossip-based protocols, each node contacts one or a few nodes in each round usually chosen at random, and exchanges information with these nodes. The dynamics of information spread algorithm behavior stems from the work in epidemiology, and leads to high fault tolerance. Gossip-based protocols usually do not require error recovery mechanisms, and thus enjoy a large advantage in simplicity, while often incurring only moderate overhead compared to optimal deterministic protocols.

Unfortunately, Gossip algorithms suffer from repeated messages which may lead to traffic congestion when broadcast frequency grows. There are several improvements made on Gossip algorithms. Directional Gossip uses a Gossip server to construct spanning tree but it is not scalable. Intelligent select node selects directional children to build a tree. Some other improvements add TTL, use UID to reduce redundancy but still has overhead.

In addition, tree-based and data-driven broadcast/multicast algorithms in video streaming or file sharing does not fit in blockchain system context [31]. Tree-based approaches like SplitStream [38] and CoopNet [39] is efficient but need to be maintained. It works poorly when dealing with node failures. For DHT, the cost of correcting the routing table of each node is also high. In addition to Gossip, other data-driven approaches like ChainSaw [40], Bullet [41], and CoolStream [42] have reduced the redundancy of Gossip a lot. However, pull-based broadcast and single-source broadcast/multicast do not fit into the context of blockchain systems.

2.2 P2P Networks in Blockchain Systems

2.2.1 Requirements from Blockchain Systems

Blockchain systems' requirements are different from other Peer-to-Peer applications: (i) on-demand streaming allows users to look up data in the P2P network and download stream data from the source, e.g. BitTorrent-based streaming systems like BASS [43], Peer-Assisted [44], LiveBT [45], and Give-To-Get [46]; (ii) audio/video conferencing applications deal with small scale point-to-point connected networks which requires low latency, e.g. Skype [47]; (iii) peer-to-peer file sharing makes efficient indexing and searching possible, e.g. Napster [48], Gnutella [49], and KaZaA [50]; (iv) video streaming applications enables single-source broadcasting efficient, e.g. SplitStream [38], Bullet [41], and ChainSaw [40]. (i) and (ii) are not relevant to the context of blockchain systems since nodes in a blockchain system network should be in a large scale and broadcasting a message is an active operation instead of searching and downloading data. (iii) and (iv) are more similar to blockchain systems' use case. However, P2P file sharing is not real-time and the broadcast model in a blockchain system is not indexing and searching. In video streaming, time is stringent and the network size can be large-scale. However, it is a data or bandwidth-intensive communication which means control messages in a broadcast operation are relatively small compared to the data to transmit.

2.2.2 State-of-Art P2P Networks in Blockchain Systems

Blockchain systems are either based random unstructured network or DHT-based structured network. Ethereum is implemented based on Kademlia [11], which is also a distributed hash table for decentralized P2P networks. Kademlia uses UDP for communication among peers and specifies the structure of the network and the exchange of information through node lookups. Similar to Pastry, each node is identified by a Node ID. Kademlia has many ideal features that previous DTHs could not provide at the same time. By incorporating broadcast configuration information into the loop-up messages, it minim-

izes the configuration messages that nodes must send in order to understand each other. Nodes have enough knowledge and flexibility to route queries through low latency paths. Kademlia uses concurrent asynchronous queries to avoid timeouts caused by node failures. Nodes record each other's existence against certain basic denial of service attacks.

While searching for n nodes in a system, Kademlia only contacts $O(\log(n))$ nodes, which is very efficient. Unlike first or second generation P2P file sharing networks such as Napster[51] or the Gnutella[49], Kademlia uses DHTs to look up files in the network. Many of the advantages stem from the use of novel XOR metrics to define the distance between two points in the primary key space. XOR is symmetric, which allows Kademlia participants to receive query requests from the exact same node distribution contained in their routing tables. Without this feature, systems like Chord cannot learn useful routing information from the queries they receive. Worse, asymmetry can make routing tables less flexible. For example, in Chord, each finger table must store the exact nodes before an interval. In fact, any node within the interval and those nodes before the same interval may be physically far apart. In contrast, Kademlia can send queries to any node within an interval, which allows it to select the optimal route based on the delay, and even asynchronously query several equally suitable nodes in parallel.

Most existing blockchain systems use some form of Gossiping to disseminate transactions, blocks, and membership information. By utilizing Gossip, participants will eventually receive all transactions and blocks with high probability. For example, participants in Bitcoin [2] Gossip with neighboring peers about recent transactions, blocks, and advertise membership of other participants. Hyperledger Fabric [52] is a platform for deploying and operating permissioned blockchains. In Fabric, participants Gossip about blocks, transactions, and membership information. Fabric divides Gossiping into two modes: pull and push, where participants request state from other peers during pulling, and sends their state while pushing. Algorand [53] uses a similar Gossip approach as Bitcoin [2], where participants select a small subset of peers to Gossip with.

3 Methodology

In order to make sure the project is delivered successfully, we divided the project into three main phases and the methodology have been using or plan to use in each phase are stated in the following three subsections.

3.1 Protocol Overview

This section presents an overview of the system protocol design. See the complete detail of the protocol through the GitHub page link: <https://github.com/James-QiuHaoran/hgfr>.

Before presenting the topology and the protocol, several key concepts should be defined clearly:

- Node: One instance of a server/virtual machine in the network;
- Ring: A group of nodes connected in a ring-like structure.
- Contact Node: the node on the ring who is in charge of adding new nodes, contacting the nodes in the upper level of the network, and broadcasting the message.

The network topology (see Figure 3.1) is basically a recursive ring-fractal structure. At the upper level resides a large ring where several sub-rings reside on. The figure shows the structure of the network in a recursive way. There are three sub-rings in the level number n . There are three contact nodes selected from each ring, which are going to be the normal nodes in the level number $n + 1$. Recursively, there will be contact nodes selected from each ring in the level number $n + 1$. The whole network structure is formed in this recursive way. That is also the reason why this network is called a fractal ring.

The protocol consists of mainly four parts: Node-Join, Node-Leave, Contact Node(s) Election and Broadcast.

Node-Join is the process of a new node joining the network. When a new node wants to

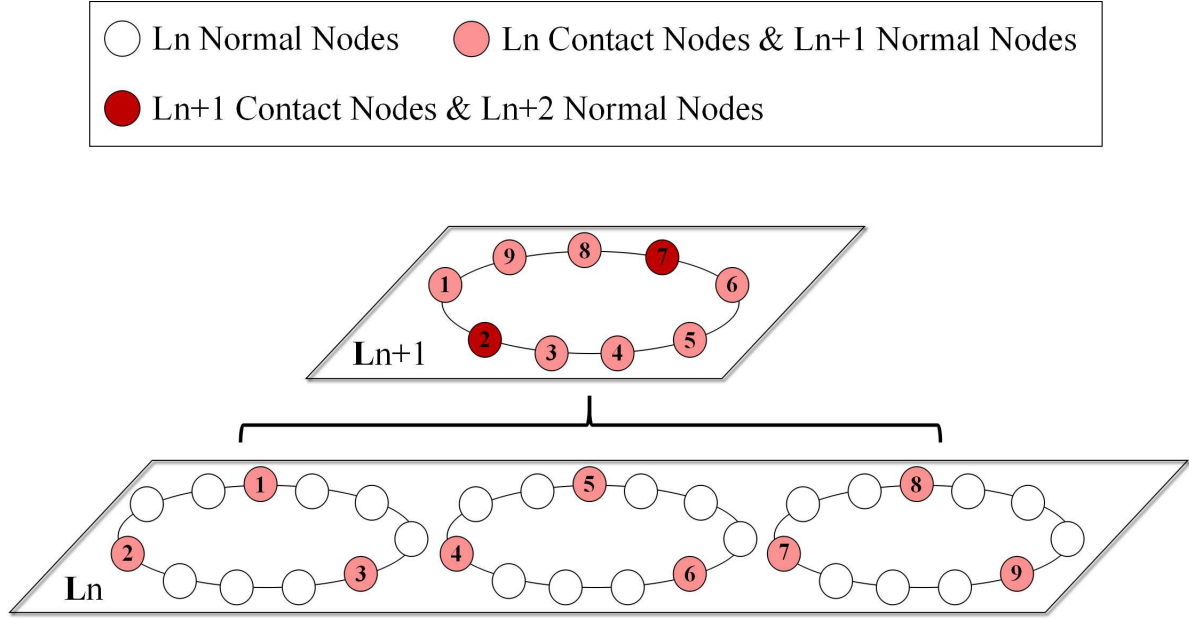


Figure 3.1: Network Topology

An illustration of the network topology in a recursive way. The structure is basically a ring-like recursive fractal. It facilitates the dissemination of messages due to its structure and geographical distribution.

join the network, it will send message to the contact nodes of the largest ring. The contact node will judge which sub-ring this new node should be added to, based on some metrics related to locality. Recursively, the contact node of the sub-ring will then introduce the new node to the sub-sub-ring. In the end, the contact node of a ring in the lowest level will then add the new node to the ring. If the number of node on a ring exceeds a threshold, then this ring will be split into two rings.

Node-Leave is the process of the network reacting to node-leave. Each node on the ring will send heart-beat messages to its successor and predecessor to check the aliveness of them. Once a node are not responding to the heart-beat message, the node will double check this with the neighbor of the dead node. If they agree that this node left the network (intentionally or accidentally), the information will be disseminated to the ring and this node will be officially removed from the network. If the missing node is the contact node, then the next generation of contact node(s) will be elected. If the number of nodes on the ring is smaller than the lower limit, the ring will be merged with a neighbor ring via the

contact nodes in the same upper level ring.

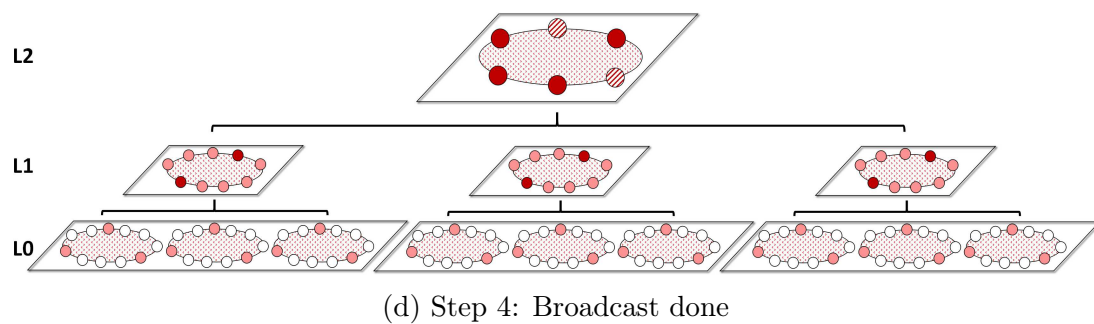
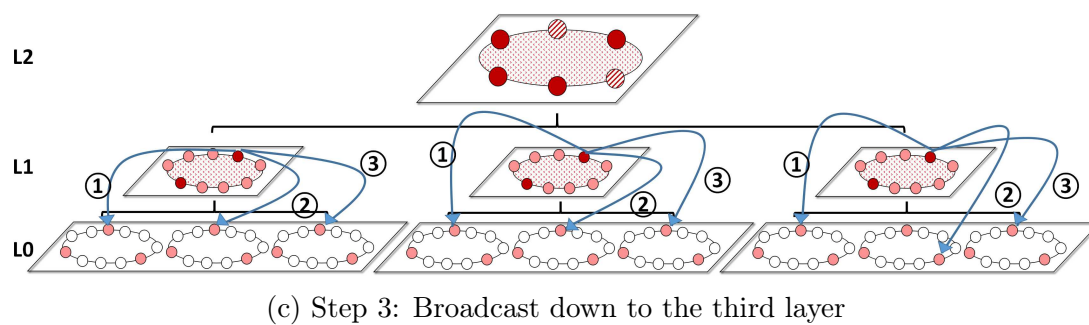
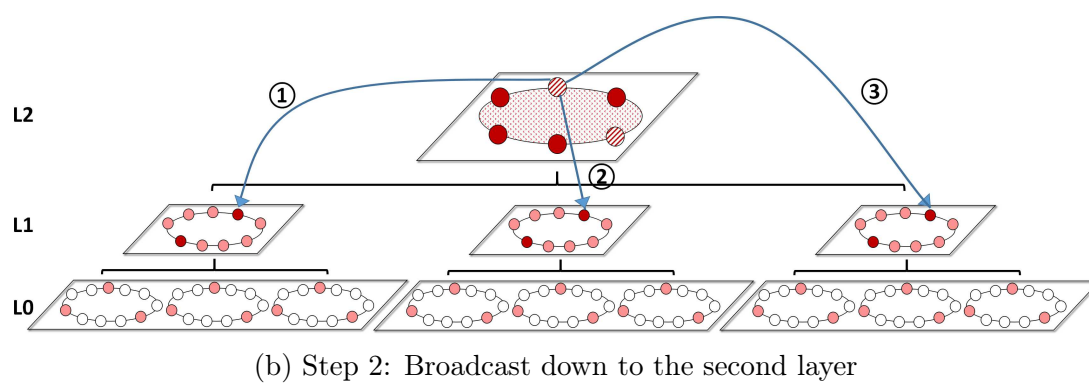
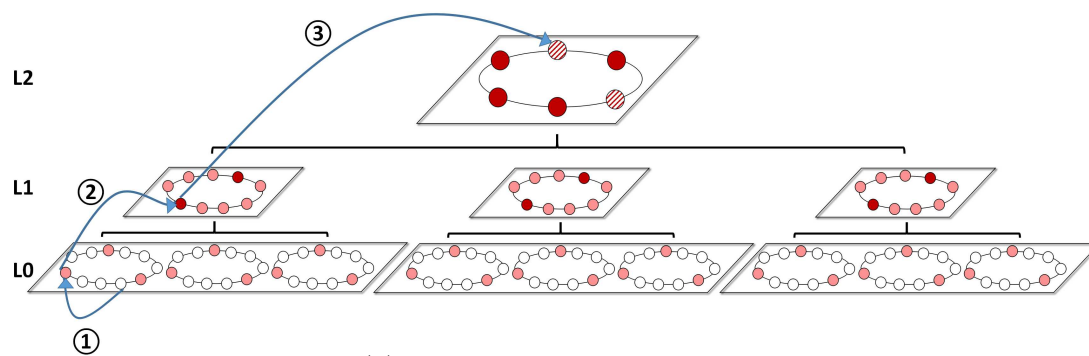
Contact Node(s) Election is the process of selecting contact nodes of a ring. In order to avoid some hackers who intentionally attack the contact nodes, contact nodes will be periodically changed and the election process is done randomly (we use Intel SGX to generate a set of random numbers which are the IDs of the nodes). The election results will be disseminated to nodes on the ring and the contact nodes in the upper level.

Broadcast is the process of disseminating a message from one node to the whole network (see Figure 3.2). When a node wants to send a message to the whole network, it will first contact the contact node of the ring it resides on. Then the message will be routed to two directions: one direction is downwards, i.e. the contact node will broadcast the message in the ring and recursively in the sub-rings; the other direction is upwards, i.e. the contact node will find the contact node of the largest ring by recursively contacting the contact node in the upper level. Once the contact node in the largest ring is found, it will broadcast recursively in the rings and sub-rings.

We devised a k-ary distributed spanning tree method to broadcast message in a ring. Details will be presented in the subsection. Based on this method, the time complexity of broadcast will be $O(\log N)$ and message complexity will be $(O(N))$, which are currently the best among related works.

3.1.1 Within-ring Broadcast Mechanism

We devised a k-ary distributed spanning tree method to facilitate the in-ring broadcast in the protocol. The basic idea is to generate a random number k , form a k-ary spanning tree, and broadcast from the root to every node in the tree. The reason we choose to randomize the parameter k is that the network should be hidden from the attacker. If it keeps using the same k , the routing pattern will be known easily by watching the network activities for a long time. The spanning tree are formed by using the broadcaster (which is numbered 0) as the root. Node 0 will connect to node $0 + k^0$, $0 + k^1$, $0 + k^2$, and so



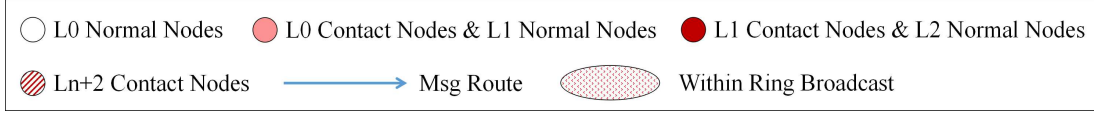


Figure 3.2: Illustration of a broadcast process from a node to the whole network in a three-layer network.

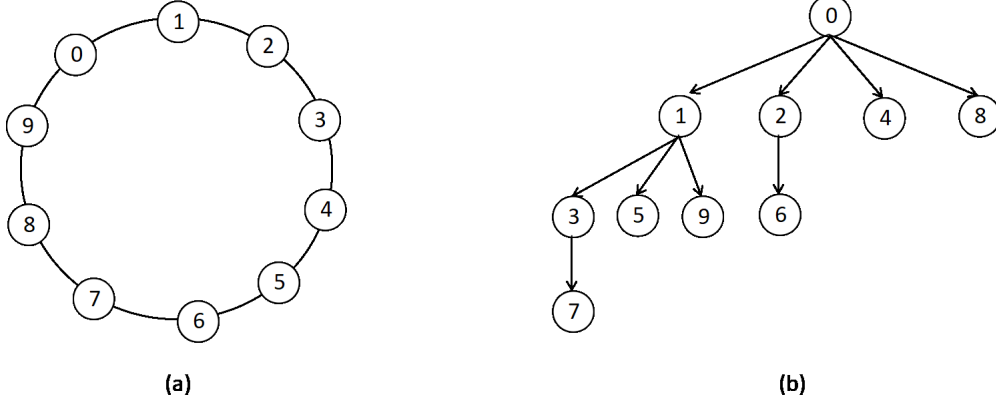


Figure 3.3: Broadcast by K-ary Distributed Spanning Tree

(a) shows a ring of 10 nodes numbered from 0 to 9. When node 0 wants to broadcast a message to other nodes in the ring, a spanning tree will be formed as shown in (b).

on. Similarly, node 1 will connect to $1 + k^0$, $1 + k^1$, $1 + k^2$, and so on. The pattern is: node i will connect to $i + k^0$, $i + k^1$, $i + k^2$, and so on. The overall time complexity of this method will be $O(\log N)$, where N is the number of nodes in the ring.

For example, Figure 3.3(a) shows a ring of 10 nodes numbered from 0 to 9. As the k-ary spanning tree algorithm states, when $k = 2$: node 0 will connect to node 1, 2, 4, 8; node 1 will connect to node 3, 5, 9; node 2 will connect to node 6; node 3 will connect to node 7. After a spanning tree is formed (see Figure 3.3(b)), the message will be disseminated from node 0 down to every node in the spanning tree.

3.2 System Implementation

The implementation of the network library has been completed. It is programmed in C++ as a standalone library and can be leveraged by current blockchains with proper engineering of adaptation. We adopted the Agile development model to guide the implementation

of the system. With frequent meetings and discussion on modification of methodologies, the risk of the development are kept to the lowest. In the future, we will continue to finish the implementation and do some code level optimization.

3.3 Evaluation



After making the system work, we will evaluate our system compared to other existing blockchain systems running the same application. The metrics for performance tentatively include but are not limited to: i) message throughput of the same operation; ii) operation throughput on the same network; and iii) fault-tolerance analysis.

The methodology of each phase may be changed as the project proceeds.

4 Project Status

This section presents the updated project schedule and the current status of this project. Project progress will be reported in Section 4.1.; Section 4.2. will propose our new P2P network protocol and the broadcast mechanism; Section 4.3. will list several challenges we met or expected, and their corresponding mitigation.

4.1 Schedule and Progress

The project is in progress as planned. Project schedule can be found in Table 4.1. It is easily to see that we have complete those items marked by , and we are currently working on the item marked by .

In September 2018, we set up a meeting with our supervisor Dr. Heming Cui. We presented our project scope and project plan to him and got some valuable suggestions from him. We also met with his PhD students Shixiong Zhao and Xusheng Chen and discussed our project plan and feasibility of this project in detail. We improved our project plan and delivered the project proposal and the project website successfully. We also did some literature review on related works and background in September.

In October, we continued to do literature review on related works such as Chord, CAN, Pastry, Kademlia, etc. and recorded what we learned in the wiki page of HKU System Lab. In order to design the system protocol, we did research on plenty of possible attack models or threats so that our system can address those problem. We also designed the first version of the system protocol.

In November, we discussed the feasibility, vulnerability, and robustness of our protocol with Dr. Cui and his PhD students. We completed our protocol design and started to implement the protocol architecture-wise.

From December 2018 to Januray 2019, we completed most of the implementation work

Time Period	Tasks (☑: completed, ⌚: in progress)
September	<ul style="list-style-type: none"> • Meet with the supervisor ☑ • Write a detailed project plan ☑ • Develop the project website ☑ • Do literature review on blockchains ☑
October	<ul style="list-style-type: none"> • Do literature review on P2P network and SGX ☑ • Design and discuss the first version of the protocol ☑
November - December	<ul style="list-style-type: none"> • Complete the protocol design ☑ • Implement the system architecture-wise ☑ • Implement the node-join/bootstrap policy ☑
January	<ul style="list-style-type: none"> • Implement the node-leave policy ☑ • Implement the contact node election mechanism ☑ • Write the interim report ☑
February	<ul style="list-style-type: none"> • Implement the broadcast mechanism ⌚ • Implement other miscellaneous • Optimization
March - April	<ul style="list-style-type: none"> • Design the evaluation and evaluate the system • Write the final report • Prepare for the final presentation • Design the poster and prepare for the exhibition

Table 4.1: Project Schedule

Detailed schedule of the project. The left column is a list of time periods and the right column is a list of tasks that are assigned in each time periods. We completed the design of the protocol and implemented most of the system. We are currently implementing the broadcast mechanism.

and hopefully our P2P network system will be working in early February.

4.2 Challenges and Mitigation

We have met two main challenges in this project:

- To improve the efficiency of the P2P protocol without inducing too much latency of maintaining the network topology

We first implemented a light version of the system and perform an evaluation. If it does not perform well, we can then quickly modify the implementation or protocol design without wasting too much time. After the light evaluation, it actually didn't introduce too much overhead when using common block size.

- To implement the system with Intel SGX

We have done some research on securing NFV with SGX like in a work [54] and how GEEC make use of SGX. We also learned and practiced programming skills on SGX before doing this project.

Currently the progress of this project is as planned. Though we have found some challenges now and expected some difficulties we might meet in the following stages, we came up with corresponding solutions to mitigate them. We will adhere to the plan in the future.

4.3 Next Step

In February 2019, we will continue to implement the P2P network system and finish it by the mid February. Then we will do code level optimization by the end of February.

In March and later stages, we will come up with a full evaluation plan and evaluate the system thoroughly. We will also write the final project report and give a presentation to the faculty.

The following schedule will also adhere to the plan in Table 4.1, which may also be modified during the implementation stage to make sure a smooth progress of this project.

5 Conclusion

Bitcoin and other cryptocurrency applications have thrived because of their decentralization nature and trust model. While much effort has been made in the consensus protocols, little research is done in augmenting the underlying P2P protocols using state-of-the-art techniques such as TEE and NFV. Therefore, in this project, we aim to design such a protocol and implement it in an existing blockchain system. We will also evaluate and discuss our protocol and the system. It is expected that message redundancy and network latency will be reduced compared to the current blockchain systems. The performance will be improved due to the optimized efficiency of peer-to-peer communication. In the future, the transaction rate of the blockchain systems can be improved further in a low bandwidth-consumption network.

Project details, current deliverables, and project schedule can be found on the project website: i.cs.hku.hk/fyp/2018/fyp18006/.

6 References

- [1] Marco Iansiti and Karim R. Lakhani. *The Truth About Blockchain*. URL: <https://hbr.org/2017/01/the-truth-about-blockchain> (visited on 30/09/2018).
- [2] Satoshi Nakamoto. ‘Bitcoin: A peer-to-peer electronic cash system’. In: (2008).
- [3] Gavin Wood. ‘Ethereum: A secure decentralised generalised transaction ledger’. In: *Ethereum project yellow paper* 151 (2014), pp. 1–32.
- [4] Christian Cachin. ‘Architecture of the hyperledger blockchain fabric’. In: *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*. Vol. 310. 2016.
- [5] Brent Xu et al. ‘EOS: An Architectural, Performance, and Economic Analysis’. In: ().
- [6] Chenxing Li et al. ‘Scaling Nakamoto Consensus to Thousands of Transactions per Second’. In: *arXiv preprint arXiv:1805.03870* (2018).
- [7] Patrick T Eugster et al. ‘Epidemic information dissemination in distributed systems’. In: *Computer* 37.5 (2004), pp. 60–67.
- [8] Ion Stoica et al. ‘Chord: A scalable peer-to-peer lookup service for internet applications’. In: *ACM SIGCOMM Computer Communication Review* 31.4 (2001), pp. 149–160.
- [9] Antony Rowstron and Peter Druschel. ‘Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems’. In: *IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing*. Springer. 2001, pp. 329–350.
- [10] Ben Y Zhao et al. ‘Tapestry: A resilient global-scale overlay for service deployment’. In: *IEEE Journal on selected areas in communications* 22.1 (2004), pp. 41–53.
- [11] Petar Maymounkov and David Mazieres. ‘Kademlia: A peer-to-peer information system based on the xor metric’. In: *International Workshop on Peer-to-Peer Systems*. Springer. 2002, pp. 53–65.
- [12] Aggelos Kiayias et al. ‘Ouroboros: A provably secure proof-of-stake blockchain protocol’. In: *Annual International Cryptology Conference*. Springer. 2017, pp. 357–388.

- [13] Mitar Milutinovic et al. ‘Proof of luck: An efficient blockchain consensus protocol’. In: *Proceedings of the 1st Workshop on System Software for Trusted Execution*. ACM. 2016, p. 2.
- [14] Eleftherios Kokoris Kogias et al. ‘Enhancing bitcoin security and performance with strong consistency via collective signing’. In: *25th USENIX Security Symposium (USENIX Security 16)*. 2016, pp. 279–296.
- [15] Intel Corporation. *Introduction*. URL: <https://sawtooth.hyperledger.org/docs/core/releases/latest/introduction.html> (visited on 30/09/2018).
- [16] EOSIO. *Eos.io technical white paper v2*. <https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md>. (Visited on 30/09/2018).
- [17] Ledina Hoxha. ‘Hashgraph the Future of Decentralized Technology and the End of Blockchain’. In: *European Journal of Formal Sciences and Engineering* 1.2 (2018), pp. 29–32.
- [18] Eleftherios Kokoris-Kogias et al. ‘Omniledger: A secure, scale-out, decentralized ledger via sharding’. In: *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2018, pp. 583–598.
- [19] LM Bach, B Mihaljevic and M Zagar. ‘Comparative analysis of blockchain consensus algorithms’. In: *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE. 2018, pp. 1545–1550.
- [20] Joan Antoni Donet Donet, Cristina Pérez-Sola and Jordi Herrera-Joancomartí. ‘The bitcoin P2P network’. In: *International Conference on Financial Cryptography and Data Security*. Springer. 2014, pp. 87–102.
- [21] Hassan Barjini and Mohamed Othman. ‘SmoothFlood: Decreasing redundant messages and increasing search quality of service in peer-to-peer networks’. In: *Information Retrieval & Knowledge Management, (CAMP), 2010 International Conference on*. IEEE. 2010, pp. 138–142.
- [22] L. Guo et al. ‘LightFlood: Minimizing Redundant Messages and Maximizing Scope of Peer-to-Peer Search’. In: *IEEE Transactions on Parallel & Distributed Systems*

- 19 (Sept. 2007), pp. 601–614. ISSN: 1045-9219. DOI: 10.1109/TPDS.2007.70772. URL: doi.ieeecomputersociety.org/10.1109/TPDS.2007.70772.
- [23] Sameh El-Ansary et al. ‘Efficient broadcast in structured P2P networks’. In: *International workshop on Peer-to-Peer systems*. Springer. 2003, pp. 304–314.
 - [24] Victor Costan and Srinivas Devadas. ‘Intel SGX Explained.’ In: *IACR Cryptology ePrint Archive* 2016.086 (2016), pp. 1–118.
 - [25] Frank McKeen et al. ‘Intel® software guard extensions (intel® sgx) support for dynamic memory management inside an enclave’. In: *Proceedings of the Hardware and Architectural Support for Security and Privacy 2016*. ACM. 2016, p. 10.
 - [26] Mohamed Sabt, Mohammed Achemlal and Abdelmadjid Bouabdallah. ‘Trusted execution environment: what it is, and what it is not’. In: *14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*. 2015.
 - [27] N Asokan et al. ‘Mobile trusted computing’. In: *Proceedings of the IEEE* 102.8 (2014), pp. 1189–1206.
 - [28] X. Chen et al. ‘GEEC: Scalable, Efficient, and Consistent Consensus for Blockchains’. In: *ArXiv e-prints* (Aug. 2018). arXiv: 1808.02252 [cs.DC].
 - [29] *Software Guard Extensions Programming Reference*. <http://kib.kiev.ua/x86docs/SDMs/329298-001.pdf>. [Online; accessed 30-September-2018].
 - [30] Yaoqi Jia et al. ‘Robust Synchronous P2P Primitives Using SGX Enclaves.’ In: *IACR Cryptology ePrint Archive* 2017 (2017), p. 180.
 - [31] Jiangchuan Liu et al. ‘Opportunities and challenges of peer-to-peer internet video broadcast’. In: *Proceedings of the IEEE* 96.1 (2008), pp. 11–24.
 - [32] Tongqing Qiu et al. ‘Towards location-aware topology in both unstructured and structured P2P systems’. In: *Parallel Processing, 2007. ICPP 2007. International Conference on*. IEEE. 2007, pp. 30–30.
 - [33] Daniel Stutzbach and Reza Rejaie. ‘Understanding churn in peer-to-peer networks’. In: *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. ACM. 2006, pp. 189–202.

- [34] John Risson and Tim Moors. ‘Survey of research towards robust peer-to-peer networks: Search methods’. In: *Computer networks* 50.17 (2006), pp. 3485–3521.
- [35] Wojciech Galuba and Sarunas Girdzijauskas. ‘Distributed hash table’. In: *Encyclopedia of Database Systems*. Springer, 2009, pp. 903–904.
- [36] Muhammad Hasan Islam, Sanaa Waheed and Izza Zubair. ‘An efficient gossip based overlay network for peer-to-peer Networks’. In: *Ubiquitous and Future Networks, 2009. ICUFN 2009. First International Conference on*. IEEE. 2009, pp. 62–67.
- [37] Indranil Gupta, Kenneth P Birman and Robbert van Renesse. ‘Fighting fire with fire: using randomized gossip to combat stochastic scalability limits’. In: *Quality and Reliability Engineering International* 18.3 (2002), pp. 165–184.
- [38] Miguel Castro et al. ‘SplitStream: high-bandwidth multicast in cooperative environments’. In: *ACM SIGOPS Operating Systems Review*. Vol. 37. 5. ACM. 2003, pp. 298–313.
- [39] Venkata N Padmanabhan et al. ‘Distributing streaming media content using cooperative networking’. In: *Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video*. ACM. 2002, pp. 177–186.
- [40] Vinay Pai et al. ‘Chainsaw: Eliminating trees from overlay multicast’. In: *International Workshop on Peer-to-Peer Systems*. Springer. 2005, pp. 127–140.
- [41] Dejan Kostić et al. ‘Bullet: High bandwidth data dissemination using an overlay mesh’. In: *ACM SIGOPS Operating Systems Review*. Vol. 37. 5. ACM. 2003, pp. 282–297.
- [42] Xinyan Zhang et al. ‘CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming’. In: *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*. Vol. 3. IEEE. 2005, pp. 2102–2111.
- [43] Chris Dana et al. ‘BASS: BitTorrent Assisted Streaming System for Video-on-Demand.’ In: *MMSP*. Vol. 5. 2005, pp. 1–4.

- [44] Niklas Carlsson and Derek L Eager. ‘Peer-assisted on-demand streaming of stored media using BitTorrent-like protocols’. In: *International Conference on Research in Networking*. Springer. 2007, pp. 570–581.
- [45] Jianming Lv et al. ‘LiveBT: Providing video-on-demand streaming service over bit-torrent systems’. In: *Parallel and Distributed Computing, Applications and Technologies, 2007. PDCAT’07. Eighth International Conference on*. IEEE. 2007, pp. 501–508.
- [46] Jacob Jan-David Mol et al. ‘Give-to-get: free-riding resilient video-on-demand in p2p systems’. In: *Multimedia Computing and Networking 2008*. Vol. 6818. International Society for Optics and Photonics. 2008, p. 681804.
- [47] Salman A Baset and Henning Schulzrinne. ‘An analysis of the skype peer-to-peer internet telephony protocol’. In: *arXiv preprint cs/0412017* (2004).
- [48] Stefan Saroiu, Krishna P Gummadi and Steven D Gribble. ‘Measuring and analyzing the characteristics of Napster and Gnutella hosts’. In: *Multimedia systems* 9.2 (2003), pp. 170–184.
- [49] Matei Ripeanu. ‘Peer-to-peer architecture case study: Gnutella network’. In: *Peer-to-Peer Computing, 2001. Proceedings. First International Conference on*. IEEE. 2001, pp. 99–100.
- [50] Nathaniel S Good and Aaron Krekelberg. ‘Usability and privacy: a study of Kazaa P2P file-sharing’. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM. 2003, pp. 137–144.
- [51] Stefan Saroiu, P Krishna Gummadi and Steven D Gribble. ‘Measurement study of peer-to-peer file sharing systems’. In: *Multimedia Computing and Networking 2002*. Vol. 4673. International Society for Optics and Photonics. 2001, pp. 156–171.
- [52] Elli Androulaki et al. ‘Hyperledger fabric: a distributed operating system for permissioned blockchains’. In: *Proceedings of the Thirteenth EuroSys Conference*. ACM. 2018, p. 30.

- [53] Yossi Gilad et al. ‘Algorand: Scaling byzantine agreements for cryptocurrencies’. In: *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM. 2017, pp. 51–68.
- [54] Ming-WeiShih MohanKumar TaesooKim AdaGavrilovska. ‘S-NFV: Securing NFV states by using SGX’. In: (2016).