Peer Discovery With Transitive Trust in Distributed System

ChangLiang Luo

Delft University of Technology

Peer Discovery With Transitive Trust in Distributed System

Master's Thesis in Computer Science

Parallel and Distributed Systems group
Faculty of Electrical Engineering, Mathematics, and Computer Science
Delft University of Technology

Changliang Luo

22nd August 2017

Author
Changliang Luo

Title
Peer Discovery With Transitive Trust in Distributed System

MSc presentation
TODO GRADUATION DATE

Graduation Committee
TODO GRADUATION COMMITTEE Delft University of Technology

Abstract

Distributed System is not a new concept, but the success of Internet, Bittorent and other applications proves the potential of such architecture. When it comes to distributed architecture, the bootstrap and peer discovery is an issue that we cannot evade. While there are already many matured peer discovery model, it is still worth to explore potential models. Fortunately Tribler Project provides such a chance. This Article will focus on improving the the peer discovery models of Tribler – Walker. The article will try to explore new strategy to protect the Walker against Sybil Attack, while retaining existing functionality of the current Tribler Walker.

Preface

(do it later)

TODO ACKNOWLEDGEMENTS

Changliang Luo

Delft, The Netherlands
22nd August 2017

Contents

Chapter 1

Introduction

Peer to Peer System is one of the architecture of Distributed System, according to [5], it can be further categorized as ”pure” Peer to Peer or ”hybrid” Peer to Peer. The success of the Internet and Bittorrent shows the potential of hybrid Peer to Peer Architecture. Internet and Bittorrent, while differs in many aspects, share some common features. The nodes in the network are organized with multiple subnetwork, each subnetwork has a centralized node. For Internet, that could be a DNS server or router, while for Bittorent, that is a tracker. While there are some variants in Bittorent network architecture like decentralized tracker proposed by [3], they can still be categorized as hybrid architectures.

Their success can be due to the fact of our real world: resources are scattering around the world and belongs to different parties. It is not realistic to impose a global centralized party to organize the whole network, because people don't trust others and they don't want to obey others. But lacking of a centralized party would lead to many troubles, the first one of them is ”how to find other peers”.In a ”traditional” system with a central node, it is minor issue because everyone registers in the central node, hence one can contact another one through the central node. In peer to peer system, we need to do it in a distributed fashion because there are not a global central node (but there would be some nodes know more than normal node). In Internet, we can find other peers by contacting servers. In Bittorrent, we can find other peers in the same swarm by asking corresponding server. But neither of them is perfect. They all require some ”off-wired” operation – for Internet, you should register the URL with your IP address and more importantly do a advertisement to inform other users of your URL; for Bittorrent, you need to distribute your torrents (or find other torrents) manually, in forum, BBS, or somewhere else. Could we create a more sophisticated system to explore other peers automatically? Maybe we can, but implementing and testing such system is not a single job for a single person. Fortunately, Tribler Provides such opportunity.

In origin, Tribler is a Project aims to help user share videos or hunting videos shared by others. Tribler has a module for peer discovery, called Walker. The Walker now has basic functionalities to detect other peers automatically. However,

there are still spaces for improvement. [7] propose a new exploration strategy using Multichain ,which I will introduce later, and finish a simulation experiment. According to [7], the strategy is not perfect and the simulation has some limitations. This article will implement a independent Walker to address the problems raised by the strategy and simulation.

The article is organized as following. In Chapter 2, I will identify the problems need to be solved.In Chapter 3, I will introduce some related works. In Chapter 4, the design and implementation of the new Walker (I will use old Walker to refer to current Tribler Walker, and new Walker refers to the transitive-trust Walker I implement) will be described. Chapter 5 will focus on the validation of the new Walker, including exploration, load balancing and security issues.

Chapter 2

Problem Description

While the current Tribler Walker already has basic functionalities in place, it is not fully satisfying. First of all, the Walker is embed in a Community class, it has many dependencies on other Tribler components, that makes it hard to maintain and develop. Tribler Team decides to take the Walker out of Community class and forms an independent module. It is a good chance to add additional functionalities by the way.Second, while the Multichain, which is a book keeping system of peers' historic behaviors, has been implemented, the Walker doesn't use it at all. Third, the Walker doesn't has a good defense mechanism for attacks, for example, Sybil attacks.

This chapter will briefly describe the Walker's protocol and identify the weakness of the protocol. Given the weakness, I will introduce some potential attacks

2.1 Tribler Walker Protocol

In bootstrapping aspect Tribler Works in a similar way with BitTorrent. When a fresh peer (Walker) enters network, it does not know where are other peers. To initiate its peer list, it needs to contact to a tracker. Unlike BitTorrent where a peer needs a torrent file to find out a tracker, the address of Tribler trackers are hard coded in Walker hence a Walker does not need anything to locate a tracker. Also, unlike BitTorrent tracker which can reply multiple peers in swarm to the peer who visit it, the Tribler tracker returns an introduction-response message which contains only a single peer in the network. The Walker mainly relays on other Walkers in the network to finish peer discovery task. Just like tracker, once receiving an introduction request, a Walker will respond with an introduction-response containing a single peer it knows.

Besides the introduction mechanism, both trackers and Walkers have NAT puncturing mechanism.NAT is the abbreviation of Network Address Translation, which is a module in network infrastructure that translate a multiple local address to single (or fewer) public IP address. NAT aims to solve the problems that IPV4 address are not enough for all devices in network. While NAT. While there are four basic

types of NAT, I will not describe all of them, I will only talk about the behaviors of typical NATS. A NAT, while well supports applications with Server-Client architecture, it cause big troubles for Peer to Peer system. The nature of NAT can be concluded as following. Generally, it will block all inbound packets in a port unless there are some outbound packets in this port recently, in this case, we say the outbound packet puncture a ”hole” in the NAT. For the Server-Client architecture applications, the NAT of the Server usually maintained by professional, trained people which can make the hole open in certain port (or even better, there is no NAT). But for Peer to Peer applications, require normal, untrained users to configure their NAT is not possible (to be even worse, they have no authority to configure the NAT). In Walker scenario, for example, after Peer B introduces Peer C to Peer A. A still cannot contact with peer C because packets from Peer A is blocked by NAT of Peer C. To solve this, Peer B will send a puncture-request message to Peer C. Then C will send a puncture message to A, which will puncture a hole in C's NAT. This hole will lasts around 60 seconds (depending on configuration of NAT, but it usually will not lasts very long time). Once the duration of hole is over, the inbound packets will be blocked again. As a consequence, the life span of a peer in Walker's peer is less than 60 seconds. The life span of peer is very important, it will be discussed later.

By working in the way above, the current Tribler Walker can finish peer discovery task well. But there are flaws in this protocol. You may notice that there is no authentication mechanism at all: peers trust each other without condition – peer A ask peer B to introduce other peers to it, peer B will do so without condtion; Peer B respond with a peer's address to Peer A, A will store it in peer list without validation, Peer B asks peer C to conduct a NAT puncture, C will do so. If all peers are controlled by nice people, there will be no problem; unfortunately, this assumption does not hold true, when there are evil peers in the network, they can manipulate the network by utilizing the flaws.

2.2 Potential Attacks

There are many ways to perpetrating an attack on Tribler network, includes but not limited in:

First,active attack like DDoS. The most direct way is bombarding the victim with introduction-request message, that will exhaust the bandwidth and CPU resources. To evade filtering like the technique described in [6],attackers can create many fake peer and launching it in different address. Tribler identify a peer according to the self reporting public key from the peer, so creating countless peers are cheap and feasible. While the attackers perpetrate attacks via those fabricated identities, the attack can also be categorized as Sybil Attack, concept of Sybil Attack is described in [2], we will discuss it later.But as I mentioned, the NAT is a big trouble, the trouble is not only for honest peers, but also for attackers. To perpetrate a DDoS attack to a specific victim via evil peers, attackers need other peers to introduce the

victim to all of his peers, it is a hard task, can only be finished by luck rather by skills.In fact, all attempts to actively attack a specific victim will likely fail because of NAT, but attackers can change their target to the whole network rather than a few peers. For example, the attackers can attack whatever peers introduced to them, or take down the trackers directly. Denying such attacks needs to enforce a defense mechanism on whole network, it is not only the effort of Walker, so I will not go further.

Second,passive attack,for example lauching attacks by utilizing introduction-response message. Since introduction-response should only be sent when a peer receives an introduction-request (if a Peer A receive an introduction-response from Peer B without sending any introduction-request to Peer B, the introduction-response will be dropped),attackers need to passively wait for an incoming introduction-request. It is a problem of possibility, but attackers can increase the possibility by using more Sybils (peers with fabricated identities). It is not an easy task, but the effect of this "response" attacks worth the effort: attackers can shifting the traffic of the whole network if they have enough Sybils. By shifting the traffic, attackers can direct honest peers to anywhere they want, for example, they can keep redirecting peers to Sybils which has no contents (videos or other useful files) or malicious contents, they can easily take any peers out of the network by inject fault address into its peer list. In addition, the use of "response attack" is not limited in Tribler scenario, it can be used in any peer discovery system with introduction-request and introduction-response fashion. The response attack may have different variants for different attacking goals, but we don't need to develop a defense mechanism for each variant, we can prevent all those attacks by simply evading visiting evil peers.

2.3 Research Goals

The goals of this research include followings:.

- First, implementing an independent Walker which does not rely on other Tribler modules, this Walker should retain the functionalities of the old Walker; it be compatible with the old Walkers' network in encoding and decoding aspect.
- Second, it should be able to show ability to counter Sybil attacks (to be more accurate, "response-attack") by preventing visiting evil peers.
- Third, validate the new Walker in a Simulated network and evaluate its performance

Chapter 3

Related Work

This chapter will discuss the related works which either directly related to Tribler Walker, or provides inspirations for the new Walker. Articles provides necessary background knowledge will also be included in this chapter

3.1 Multichain

The old Walker has no reputation system which score peers according to their historic behaviors. In fact, the old walker does not aware of other peers historic behavior. It can use a missing-identity message to ask another peer to report its identity (public key),but it is not able to retrieving any other information associated with this identity. It is just like you only know a person by name, you don't know his/her past, his/her background, let alone judging him. The Walker needs a book keeping system storing historic behaviors before any effort on calculating information.

Fortunately, such system has been implemented in [7]. It is a tamper proof book keeping system which keep the records for interaction between peers, the interaction refers to download and upload, which are the most important interaction of Tribler peers. Multichain is a similar to BlockChain of Bitcoin [4], but differ in the point that Multichain does not require user to have the knowledge of the whole transaction history (blocks)

(should add some graphs for illusion and more details of Block Creation and crawling) With the Multichain in place, it is now possible to calculating the reputation of peers hence the Walker can choose to walk or not walk to a peer according to its reputation. In fact, [7] already discuss the walking algorithm basing on peers reputation, though the author does describe a specific way to calculate reputation, he proposes a walking strategy: rank the known peers according to their reputation and the Walker will has higher probability to walk to high-reputation peers. The author uses an experiment to validate this strategy, while it shows greater efficiency in collecting blocks, it causes load balancing problem that the high reputation peers are visited too much.Although this article does not focus on block discovery efficiency, I still need to worry about load balancing issue. If a focus walking strategy

cause load imbalance in blocks crawling, it may also cause load imbalance in sybil evasion, that will undermine the usefulness of reputation system basing on Multichain.But I would say the load balancing will not be a trouble. In the experiment of [7], the author admits he does not consider the life span of peers, so all peers in Walker's peer list will last forever, as a consequence, once the Walker discovers a peer with very high reputation, it will keep visiting that peer in a very high possibility until the end of the experiment or discovering other peers with higher reputation. But in real Walker network, this situation will not happen: all peers, including those have high reputation will time out within 60 seconds, after that they will not be visited again, so a high-reputation will not cause a global load balancing. So,strategies that use reputation system basing on Multichain to evade sybil still worth trying.

3.2 Sybil Attack Defense

While there are many Sybil defense strategies like using Certificate Authority,charging fees for creating new identities, most of them do not fit the Tribler Scenario very well. But we can still find something useful. For example [10][9][1][8]. All those papers can be seen as a series while SybilGuard[10] is the first one. There assumptions are interesting:

- 1.There are two regions in the network, a honest region and a Sybil region The two regions are connected with few edges, called attack edges
- 2.Compared with the number of Sybils, the attack edges are few
- 3.We (the honest users) are born in honest region
- 4.Every node which nodes it is connected to
- 5.Most nodes are always online,that means whenever you send some request to other nodes, they will respond to you on time. And the network is static, the topology will not change.

Similarly, we can model the Walker network as follow: nodes represent identities, edges "knowing address", For example, if node A knows node B, then there will be a link between them, otherwise not. But obviously, the assumption 5 does not hold in Walker network, because the life span of peer is less than 60 seconds, once the peer times out or goes offline,the topology will change. Consider the frequency then the peers time out or go offline, the topology keeps changing frequently. That means it is not possible to apply SybilGuard or its variants directly in Walker.

However, other assumptions holds true: Creating a sybil is cheap, it is only a task of creating an unique public key and deploy with an address (IP, port pair), attackers can easily deploy thousands of peers per second. But creating attack edges is hard, a sybil need to wait for introduction-request from an honest node to establish an

attack edge, it is not realistic to establish thousands of attack edges per second. So, compared with the number of sybils, attack edges are few. Hence the network can be divided into two region – honest region and sybil region, Assumption 1 and 2 hold true. If the trackers are not compromised, a fresh Walker is likely to initiate with a peer list containing only honest peers. If the trackers are compromised, are defense mechanism in Walker side will not works, it needs the effort of whole system rather than Walker itself to defense the attacker,but this article only focus on Walker, so I will assume the trackers are not compromised. Then Assumption 3 holds True.A peer knows the address and identity of all peers it connected to, so Assumption 4 holds true.

With assumption 1 to 4 holds true, we can still learn something valuable from SybilGuard.

Chapter 4

Design of Transitive-Trust Walker

This chapter describe the requirements for the Transitive-Trust.Before explaining the design of the Walker, some necessary knowledge of Tribler will be introduced. Later, the design of the Walker will be described, some critical parts will be explains in details.

4.1 Design Requirements

According to the future plan of Tribler Team, Walker should be taken out of community and forms a independent module.Most functionalities of the Walker should remain. The Walker should retain light-weight feature.

As a independent module, the Walker can't rely on Tribler modules for port listening, packet encoding and decoding, and message handling. The Walker should be able to handle all this tasks by itself.

The Walker should be about to collect Multichain Blocks (I will use the term "crawling" to refer to the task of collecting blocks) from other peers and store blocks in database.Current protocol of Tribler Walker does not include Blocks crawling, the protocol should be redesigned.

The Walker should be able to retrieve blocks from database and analyze the blocks,then calculate reputation of a peer according to its blocks.The Walker should be able to take proper actions basing on reputation.

The Walker should be able to discover peers efficiently, discover as many honest peers as possible while preventing visiting evil peers (attackers' peer). The Walker should show ability to withstand Response-DDoS Attack, at least in some degree.The Walker should concern load balancing issue, because load unbalancing may do harm to other peers.

4.2 Walker Structure

The Walker can be further divided into multiple modules according to functionalities. It is modules include: an UDP packet listener running on a given port, a message handler which can handle incoming messages, an encoder/decoder which can encode the message into binary stream or vice versa,a database used to store and retrieve blocks, a peers manager which determine the next peer to visit and the next peer to introduce to other peers.

While retaining many functionality of current Walker, the new Walker varies in many places, so it is necessary to have thorough introduction.

4.2.1 UDP Packet Listener

4.2.2 Message Handler

The Message Handler handles following messages: introduction-request, introduction-response,puncture-request,puncture,dispersy-identity,missing-identity,crawl-request,crawl-response.

After receive an introduction-request from peer A,Message Handler will response with an introduction-response, containing the address of a random peer C which returned by Peer Manager. At the same time, Message Handler also sends a puncture request to peer C containing the address of peer A. That will cause a puncture message from peer C to peer A, that will punch a hole in C's NAT.

(should have a graph showing the structure here)

4.2.3 Encoder and Decoder

4.2.4 Block Database

4.2.5 Peer Manager

4.2.6 New Walker Protocol With Blocks Crawling

4.2.7 Reputation System and Walking Strategy

4.3 Messages

A Walker need to handle following messages: introduction-request, introduction-response,puncture-request,puncture,dispersy-identity,missing-identity,crawl-request,crawl-response.Before describing the work flow of the Walker, we need to know the content of each message. I will introduce every message in both high level (content) and low level (encoding format), readers who are not Tribler developers can skip the low level part.

4.3.1 Introduction-Request

The structure of introduction-request message is shown in Figure 1 Introduction Request contains following parts:start header,message type id, MemberAuthentication, PublicResolution, DirectDistribution and Introduction Request Payload,Signature

start header: the 22 bytes start header, more details can be found in previous section of this article

message type id: for Introduction Request,the message type id is 246, it takes a space of 1 byte

Member-Authentication: for community with version not greater than 1, the Member-Authentication part is a 20 bytes mid (member id), which is a SHA-1 digest of the member's public key. For community with version greater than 1. The MemberAuthentication part is a (key length,public key) tuple. The key length is a 2 bytes integer indicates the key length of the member. The public key is the public key in string for the member.

Public-Resolution: it is empty, takes 0 byte.

Direct-Distribution:it is a 8 bytes unsigned integer indicates the time represented in logical clock.

Introduction-Request-Payload: it contains: 4 bytes for destination address, 2 bytes for destination port,4 bytes for private (LAN) address of message sender, 2 bytes for private port of message sender. 4 bytes for public (WAN) address of message sender, 2 bytes for public port of message sender. a 1 byte bitmap(the bit map uses its 2 most significant digits representing the NAT type 00 represents unknown, 10 represents public, 11 represents symmetric-NAT. The following 3 bits represents whether the message uses tunnel, 100 represents true, 000 represents false. The following 2 bits represents synchronization option 10 represents true, 00 represents false. The last bit represents advice option, 1 represents true, 0 represents false. Here is an example, a bit map 10100001 means: the NAT type is public, enable tunnel, disable synchronization, enable advice.) and 2 bytes identifier which is a random number

Signature: a signature using the member's private key and M2Crypto policy

4.3.2 Introduction-Response

An Introduction Response almost the same with Introduction request instead of message-type-id and Payload

message type id: for Introduction Response, it is 245

Introduction-Response-Payload:4 bytes for destination IP address,2 bytes for destination port,4 bytes for private (LAN) address of message sender, 2 bytes for private port of message sender. 4 bytes for public (WAN) address of message sender, 2 bytes for public port of message sender.4 byte private address being introduced, 2 bytes private port being introduced. 4 bytes public address being introduced, 2 bytes public port being introduced 1 byte bitmap which is the same as Introduction Request. 2 bytes identifier, must be the same with corresponding

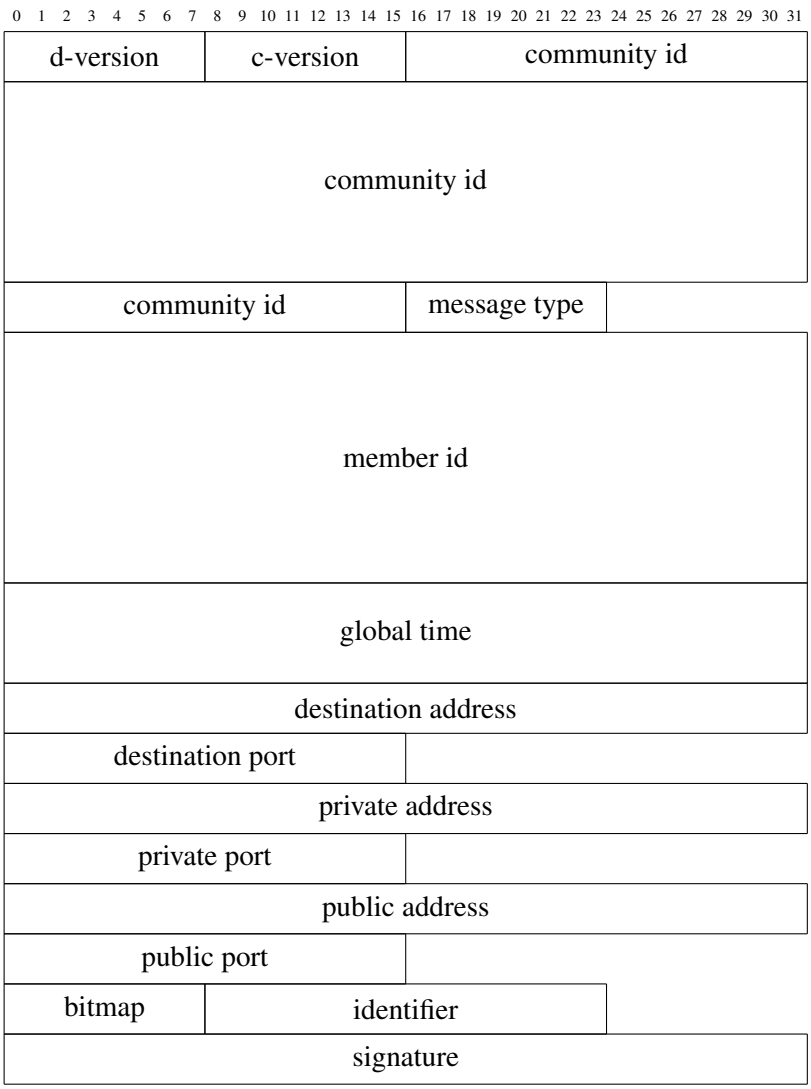

Figure 4.1: Figure 1 bytefield of introduction request,signature length varies over different key type

Introduction Request

4.3.3 Puncture Request

Compared with Introduction Request,Puncture Request differs in message-type-id Authentication and Payload

message type id: for Puncture Request, it is 250

Authentication: Puncture Request uses No-Authentication which is empty, takes 0 byte.

Puncture Request-Payload: 4 bytes private address to puncture, 2 bytes private port to puncture, 4 bytes public address to puncture, 2 bytes port to puncture. 2 bytes identifier

4.3.4 Puncture

compared with Introduction Request, Puncture Message differs in message-type-id and Payload

message type id:for puncture it is 249

Puncture-Payload: 4 bytes for private (LAN) address of message sender, 2 bytes for private port of message sender.4 bytes for public (WAN) address of message sender, 2 bytes for public port of message sender.2 bytes identifier

4.3.5 Missing Identity

Compared with Introduction Request, Missing Identity differs in message-type-id Authentication and Payload

message type id: for Missing Identity, it is 247

Authentication: it uses No-Authentication policy, it is the same with Puncture-Request

Payload: it contains only 20 bytes of member id that you are requesting identity.

4.3.6 Dispersy-Identity

Compared with Introduction Request, Dispersy Identity differs in message type id, Authentication, Distribution and Payload

message-type-id: it is 248

Authentication: it is Member Authentication, but it should always contains (key length,public key) pair (the fashion of community with version greater than 1) regardless of the version of community.

Dispersy Identity Payload: empty, takes 0 byte.

4.3.7 Crawl Request

A crawl-request message consists of followings:

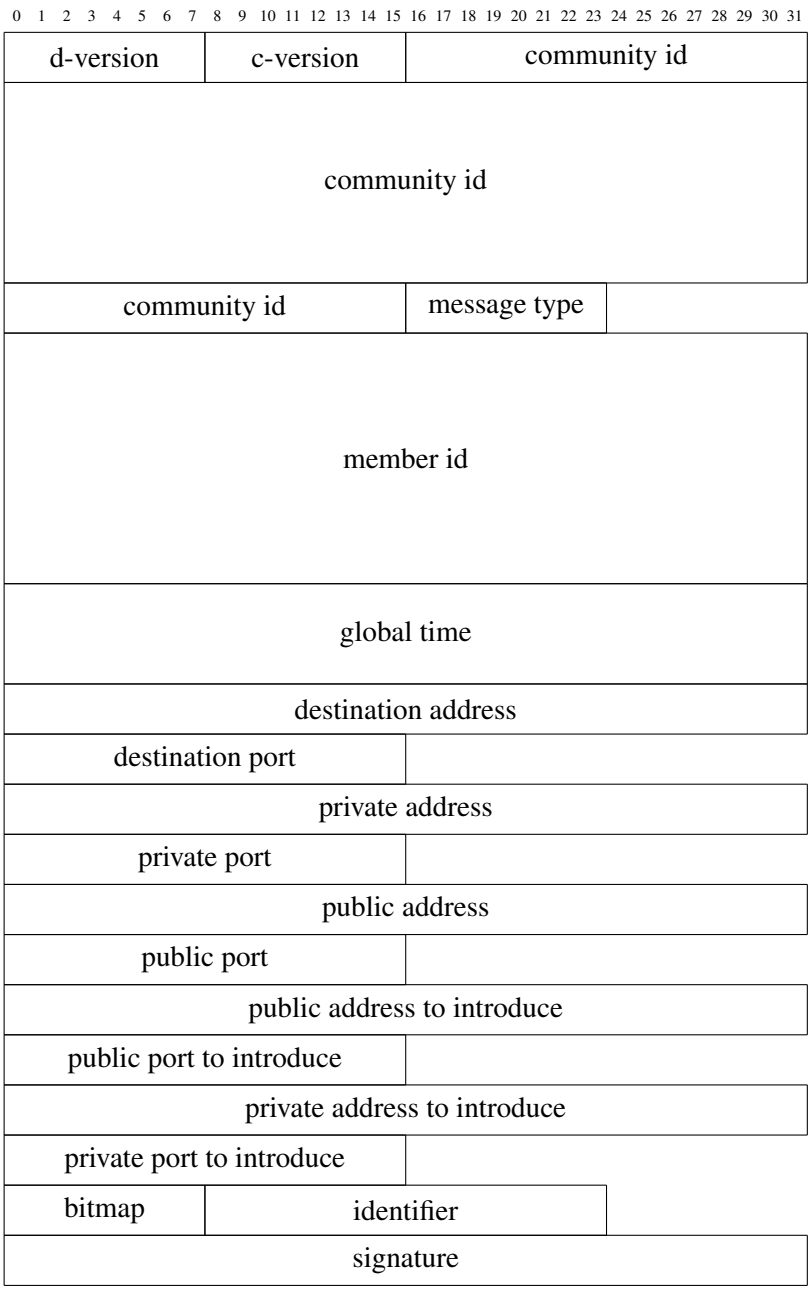

Figure 4.2: Figure 2 bytefield of introduction response,signature length varies over different key type

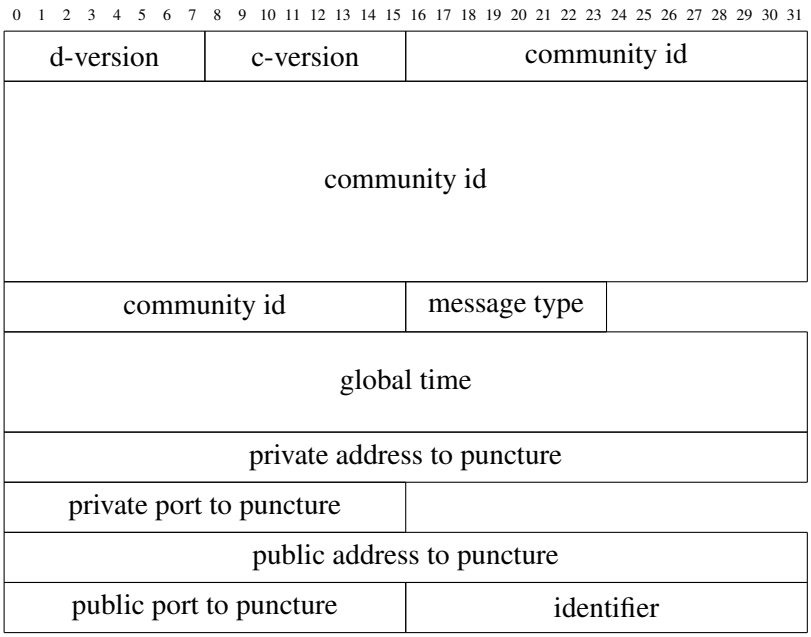

Figure 4.3: Figure 3 bytefield of puncture request,doesn't require signature

A 22 bytes starts header: 1 byte of dispersy version, 1 byte of community version, 20 bytes of community ID.

One byte message type, for crawl-request, it is 2.

20 bytes identity, which is sha1 digest of the public key of the user.

8 bytes of global time, using lamport clock.

74 bytes of public key of the receiver of this message

4 bytes of sequence number, indicates the sequence number of the block we want to crawl

4 bytes of crawl limits, indicating the amount of blocks we want to crawl

A signature, length vary over different key type

4.3.8 Crawl Response

A crawl-response message consists of followings:

A 22 bytes starts header: 1 byte of dispersy version, 1 byte of community version, 20 bytes of community ID.

One byte message type, for crawl-request, it is 1.

20 bytes identity, which is sha1 digest of the public key of the user.

8 bytes of global time, using lamport clock.

a block consisting of:

8 bytes of upload data count (in Mega Bytes)

8 bytes of download data count

8 bytes of total upload data count

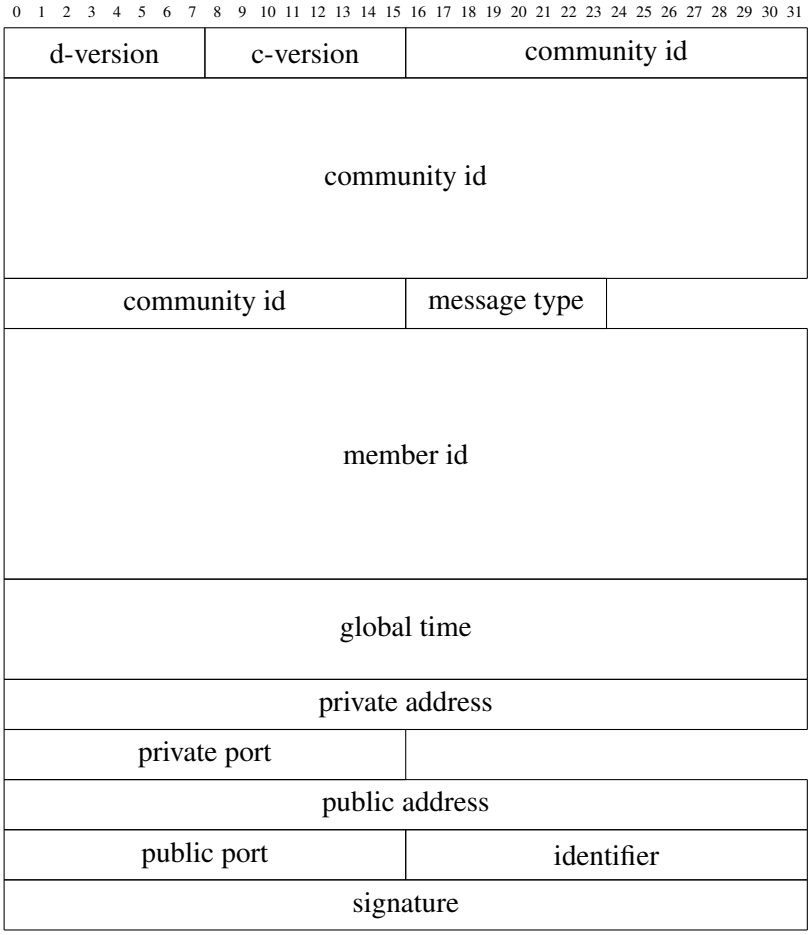

Figure 4.4: Figure 4 bytefield of puncture,signature length varies over different key type

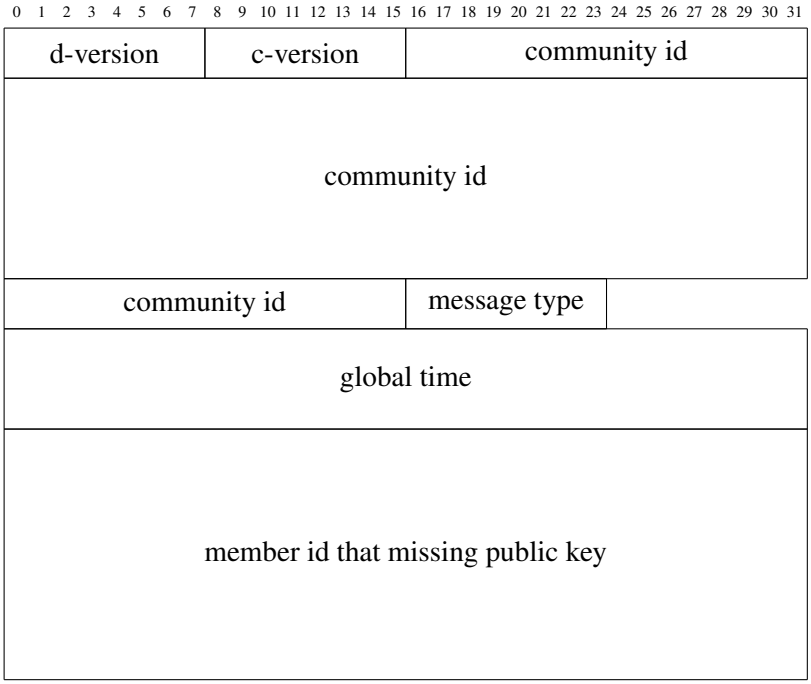

Figure 4.5: Figure 5 bytefield of dispersy-missing-identity

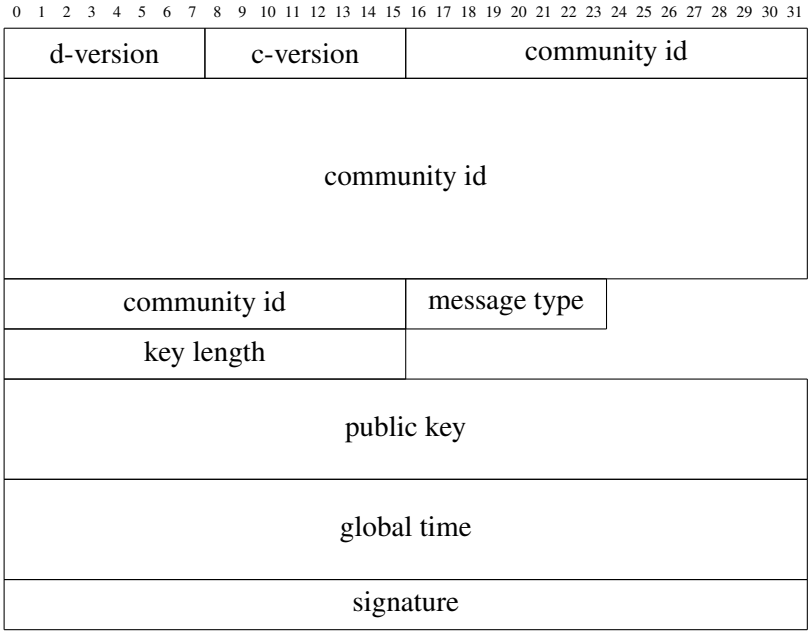

Figure 4.6: Figure 6 bytefield of dispersy-identity,length of key and signature vary over different key type

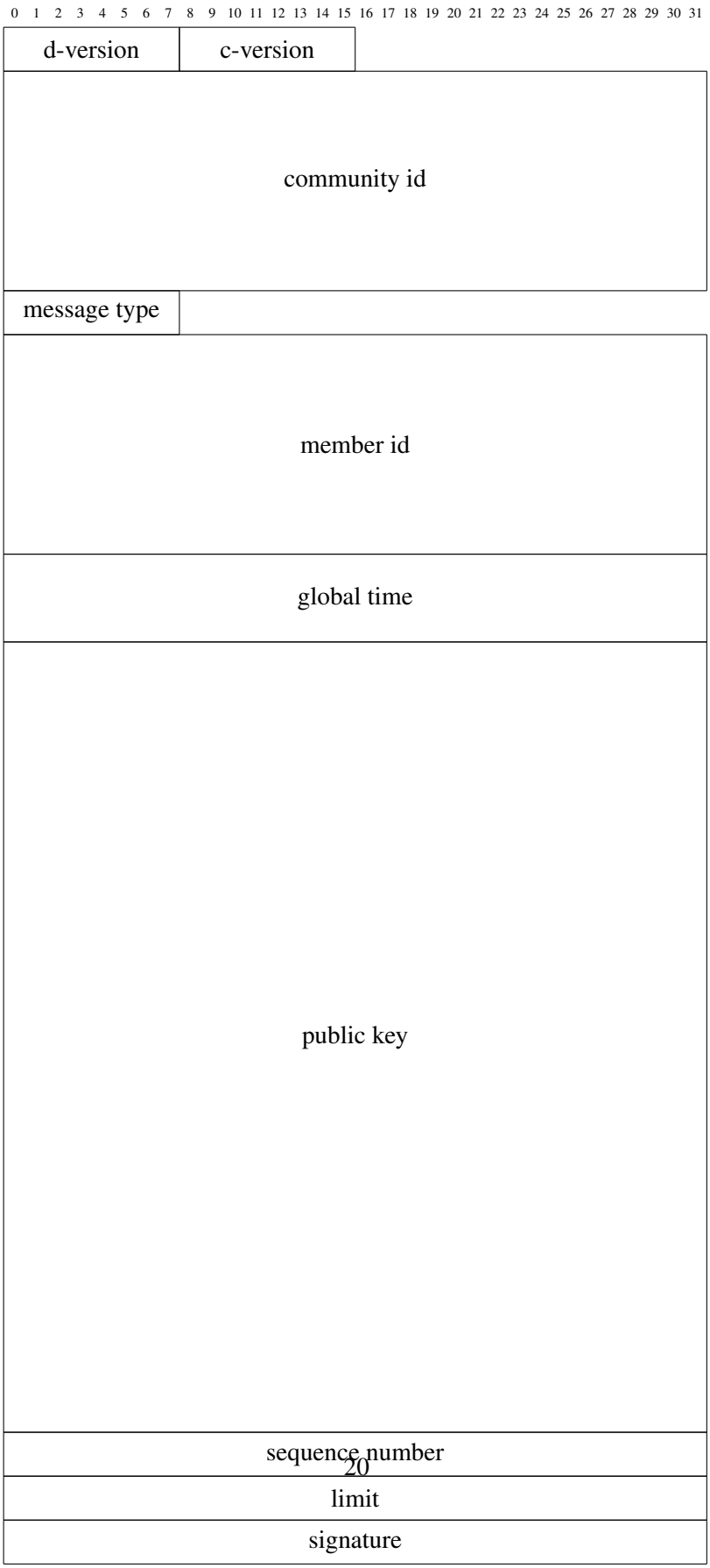

Figure 4.7: bytefield of Crawl Request,length of key and signature vary over different key type

8 bytes of total download data count
74 bytes of public key of requester
8 bytes of sequence number of requester
74 bytes of public key of responser
8 bytes of sequence number of responder
32 byte of previous hash
64 bytes of signature

4.4 Walker Protocol

The new Walker retains some functionality of the current Tribler Walker and it also has functionalities crawling and using Multichain Blocks.

4.5 Basic Functionality

The basic functionality of Walker is discovering peers and introducing peers to other peers.Every 5 seconds, a Walker will randomly pick one of its known peers to visit (send introduction-request). The known peers include trackers, peers it used to visit (outgoing peers), peers used to visit it (incoming peers),peers that being introduced to it (intro peers), and the peers it trusts (trusted peers).

After receiving introduction-request from peer A, a peer will randomly pick one of its known peers (say peer C) and introduce it to peer A, meanwhile, send a puncture-request to peer C, informing address of A.

After peer C receiving puncture-request, it will send a puncture message to peer A, peer A may not be able to see the message due to its NAT. The goal of puncture message is puncturing a hole in the NAT of peer C. Given the fact that the life span of the holes in NATs are usually less than 60 seconds, so for peer A, the life span of peer C is only 57.5 seconds (consider the processing time and network lagging, it is set to be slightly less than 60 seconds), after 57.5 seconds, the hole is likely to be closed and peer A will no longer able to contact peer C. So peer A will remove peer C from its list

4.6 Blocks Crawling

If a peer (say peer A) wants to crawling Multichain Blocks, it can send a crawl-request to any peers. Peer A need to specify the public key of the peer that creates those blocks, as well as the sequence number of blocks. That raises a question: how to know the public key of a peer?

Though the Tribler Team has plan on identity self inclusion feature which enforce a public key in any type of Messages, that feature is not yet implemented. Most of current Messages use 20 bytes hash (called ”Member Id”) of public key to represent the identity of the sender. Due to the irreversible nature of hash function,

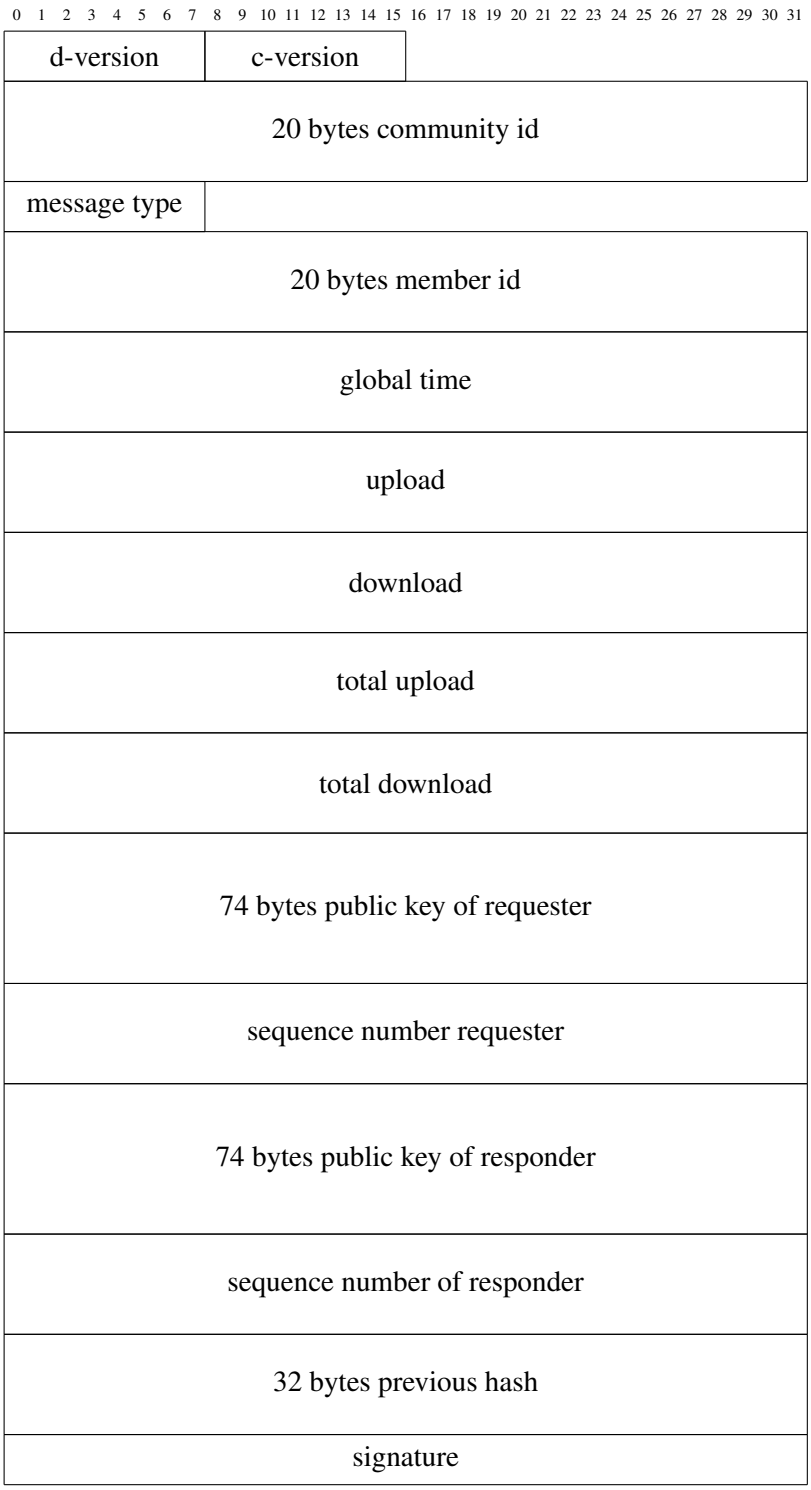

Figure 4.8: Figure 2 bytefield of Crawl Response, payload and signature will be shown in Figure 3,4 and 5,ignore the length of payload and signature and public key here, they are just placeholders

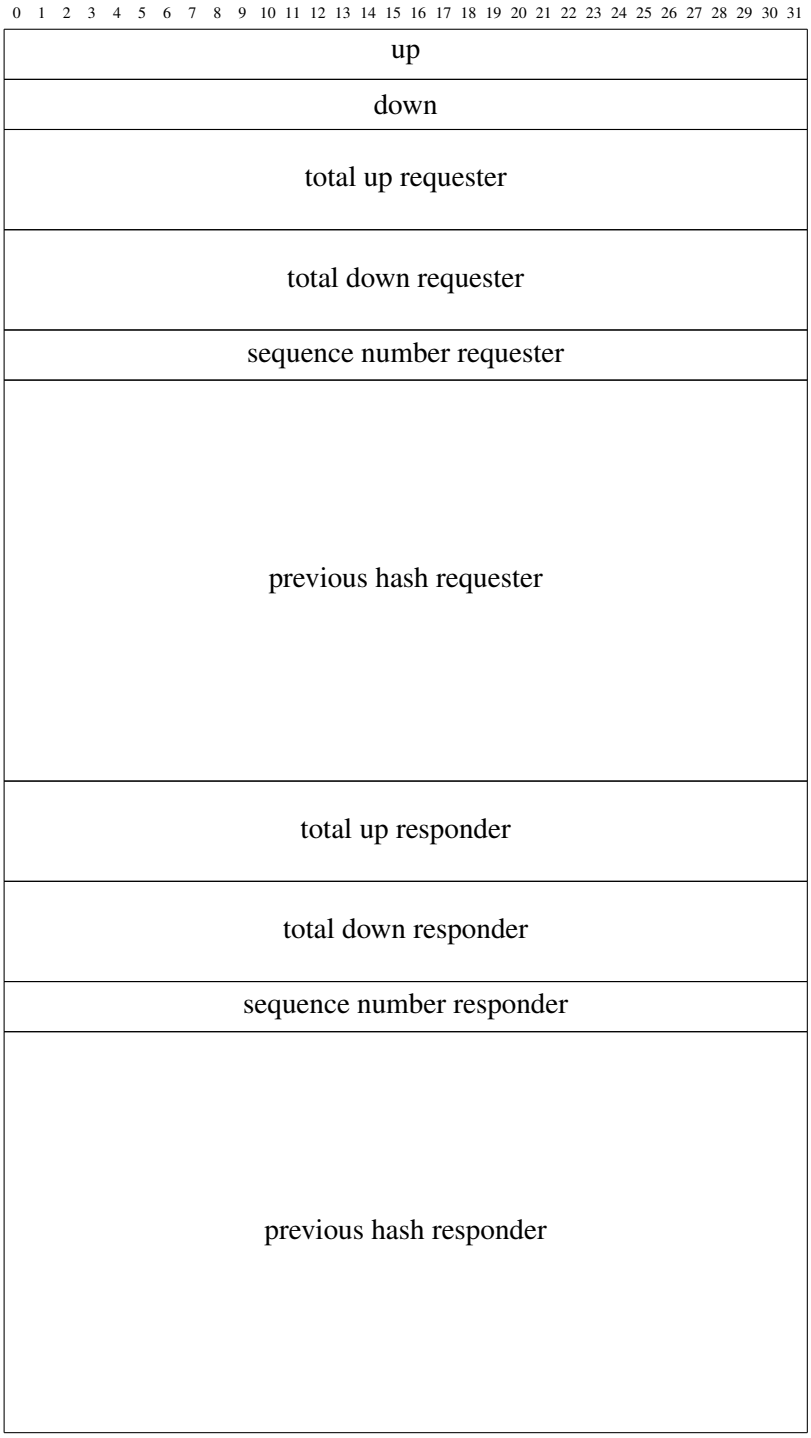

Figure 4.9: Figure 3 bytefield of Crawl Response payload

Walker can not get the public key given that 20 bytes hash. To solve this, Walker uses two Messages:missing-identity and identity.

If peer A knows the address of peer B, it can request B's public key by sending it a missing-identity message, B will rely with a identity message, containing its public key. In the new Walker, after receiving an introduction-request, the Walker will retrieve the database seeking for a peer whose public key fits the 20 bytes Member ID of the message sender (given a public key, one can easily obtain the Member ID by using sha1 hash in the public key), if no public key fits the Member ID, reply the sender of introduction-response with a missing identity.

With the public key, it is now able to send crawl-request. The new Walker will send a crawl-request in to cases. First, upon receiving an introduction-response, if the public key of the sender is known, it will reply with a crawl-request (otherwise a missing-identity).Second, upon receiving an identity Message.

4.7 Reputation System

For now, the Reputation System is simple. For a Walker, the reputation is binary –trusted or untrusted.The Walker maintains a directed graph structure basing on Multichain blocks. Where nodes represent an identity (public key of a peer),and the edges represent download and upload interaction. If Peer A (node A) used to upload some data to peer B (node B), then there will be a directed edge from node A to node B, the weight of the edge is the amount of data uploaded. If there is an edge from Peer A to Peer B, Peer B will label Peer A as trusted. This reputation system allow transitive trust feature. Transitive trust means that a trusted peer can transfer part of his trust to other peers. For example, if A trust B and B trust C, then A will trust C. But the transitive trust should degrade as the number of hops increase, so the number of hops should be limited,for example, 2 hops, that is to say, if A trust B, B trust C and C trust D, then A will trust C (2 hops) but not trust C (3 hops). The Walker assign same "score" to both 1 hop trust and 2 hops trust – labeling both of them as "trusted peers"

4.8 Walking Strategy

The Walking strategy aims to deal with the situation described in SybilGuard algorithm[10]: the network can be divided into a single honest region and a single sybil region. The number of attack edges,which connected two regions, are few. Honest peer should be born in honest region. Because of lacking global views of the network, exploring the network is just like walking at night– the world is broad and dark, and you have no idea where the dangers are. Because the attack edges are few in number, an honest Walker is not likely to cross the "bridge", when it still walks in honest region, the probability that its next destination is an evil node is negligible. The narrow nature of "bridge" serves as a protector to honest peer in this case. However, once the Walker accidentally walks deep into evil region,

the probability to visit an evil peer increase sharply; to be even worse, because the bridge is narrow, the Walker stand limited chance to get back, it is stuck in evil region and keep visiting evil peers. A good walking strategy should prevent this dire situation. To achieve this, I propose two strategies

4.9 Teleport Home Walker

To prevent being stuck in the sybil region, the Walker should have ability to ”teleport home” without the ”bridge”. But that raise two questions.

First,where is the ”home”.We can define home as the starting point of the walking, a point that connects to the peers the Walker knows right after initiation. However, because the life span of peers are short (less than 60 seconds), when the Walker gets stuck in sybil region, it may realize that all those peers it used to knows have been gone, the ”home” no longer exists. To solve this problem, we can define the ”home” as trusted peers, then ”teleport home” means teleport to that trusted peers and visit it, then visit the peers it introduces and so on.

Second,how does the Walker know it is stuck in Sybil region The answer is: it does not need to know that. The Walker does not try to tell whether a peer is honest or evil, it choose ”continue walking” or ”teleport home” by probability.

4.10 Bias Walker

An alternative solution is that: the Walker does not necessarily follow the path towards to latest introduced Peer,after visiting one peer, it randomly teleport to a random chosen peer. But to prevent teleporting to a sybil, we should assign higher probability to trusted peers, and lower probability to untrusted peers.The logic behind this is that trusted peers are less likely to be sybil. Directed trust peers which used to upload data to us is less likely to be sybil compared with a random peer, because few sybils have chance to upload data to honest user, uploading data is much more difficult than creating sybils. But as the hops of trusted chain increase, the transitive trusted peer will be more and more suspicious. So, that raises another problem: we should limit the hops of trust chain by certain number, but what is the optimal number? Due to the complexity of the network, I can not answer this question analytically,I try different number in experiment instead.

Chapter 5

Validation

The goals of this article is implementing a new Walker which can prevent itself walking into sybil region in some degree, in addition, the new Walker, as a Walker should be able to discover peer efficiently and prevent load imbalance. This chapter will validate the newly implemented Walker experimentally. The experiments include: Sybil Evasion experiment, Exploration experiment, Load Balance experiment.

All experiments are done with a single real Walker and a simulated network. All peers in the simulated network are passive, they will not actively walk to any other peers but passively waiting for the real Walker visiting them and then respond to it. For example, a real Walker should send an introduction-request messagge to another peer every 5 seconds, but a simulated peer will not send introduction-request to anyone.Except for introduction-request, a simulated peer use the same logic with real Walker to handle other messages, it can reply you with introduction-response, identity, crawl-reply etc.

Using simulated network is for resources saving consideration.For real Walkers, though they light weight, still consume around one hundred times than a simulated Walker.

5.1 Sybil Evasion Validation

For Sybil Prevention experiments, the Walker will walk in a network consisting of 400 thousand honest peers and 600 thousands sybils. All honest peers lie in the honest region and all sybils lie in the sybil region. The two regions are connected with some attack edges, the number of attack edges range from 50 thousand to 400 thousand with interval 50 thousand,totally 8 configurations, every kind of Walker will be tested with all 8 configurations. The number of visited honest peers and visited sybils will be recorded and used to calculate the evil ratio (number of visited sybils/number of visited honest peers)

There all three kinds of Walker will be tested:

- The first is fully random Walker, which pick a random peer to visit in every step.
- The second is teleport home Walker, which will visit the latest added peer unless it hits the probability to "teleport home", if there is at least one trusted peer, teleport home means teleport to one of the trusted peer; otherwise teleport to a known random peer.
- The third is bias walker, which takes random walk to known peers but assign higher probability to trusted peers.

The second and third Walker are both based on trusted peers. A peer is trusted if it satisfy one of the following:

- It used to upload some data to the verifier (you Walker).
- It is trusted by some of your trusted peers, and the number of hops in trusted chain is not greater than certain number h

The new Walker use $h = 2$

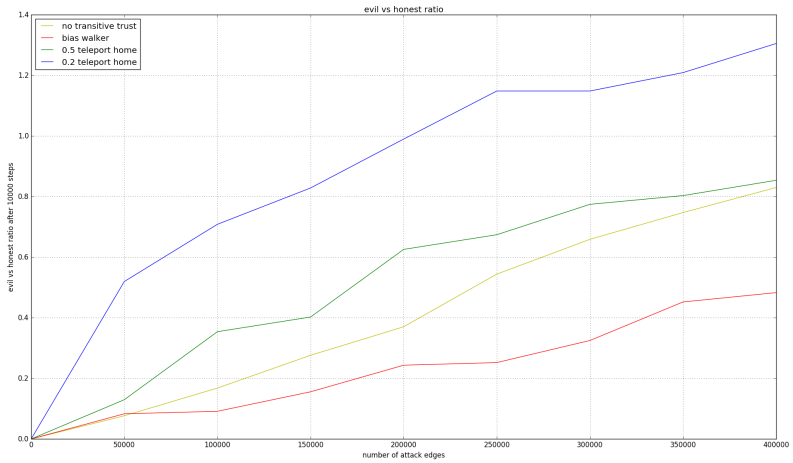

Figure 5.1: evil ratio for different walkers

bias Walker has best performance in sybil evasion, random Walker take the second place, and teleport home Walker with 50% probability to teleport home is the third.

5.2 Exploration Efficiency Validation

random Walker has worst exploration Efficiency, and Walker with 0.2 teleport home probability has the best exploration efficiency

Walker	steps required to exhaust 95% of peers
bias Walker	28758
0.2 teleport home Walker	10349
0.5 teleport home Walker	14989
random Walker	36733

Table 5.1: The steps needed to explore 95% peers in the network

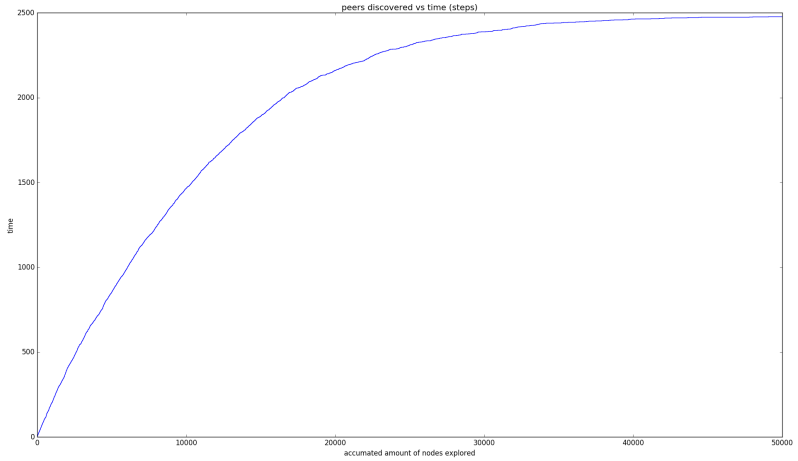

Figure 5.2: peers discovered (accumulated) over time for bias walker

5.3 Load Balancing Validation

References to figures are given with a capital letter for figure, as in “(see Figure 5.9)” or “in Figure 5.9, ...”.

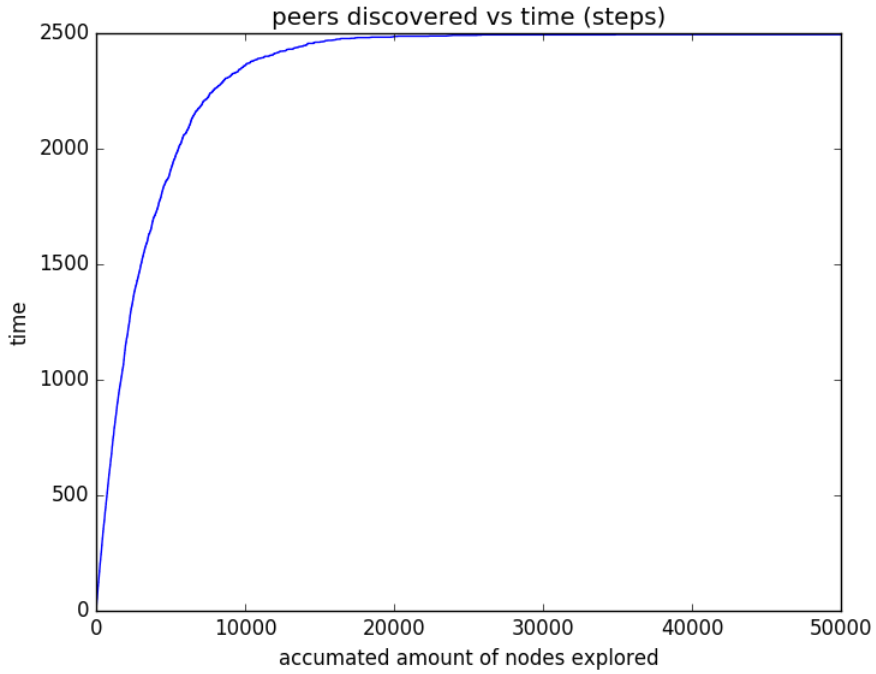

Figure 5.3: peers discovered (accumulated) over time for teleport walker with teleport home probability 0.2

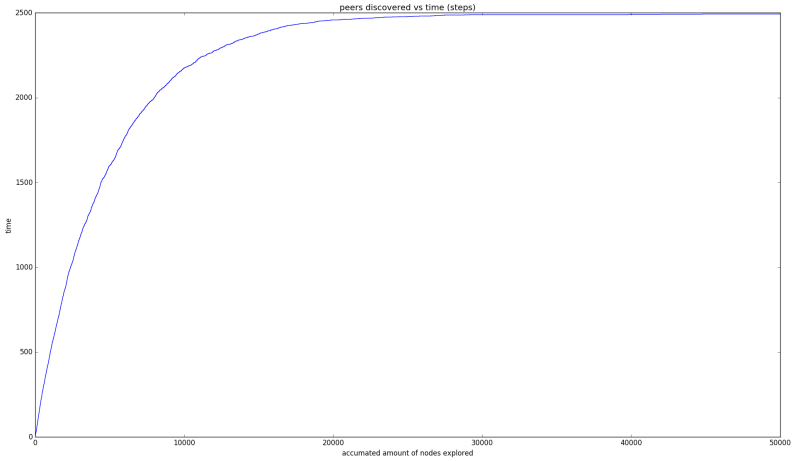

Figure 5.4: peers discovered (accumulated) over time for teleport walker with teleport home probability 0.5

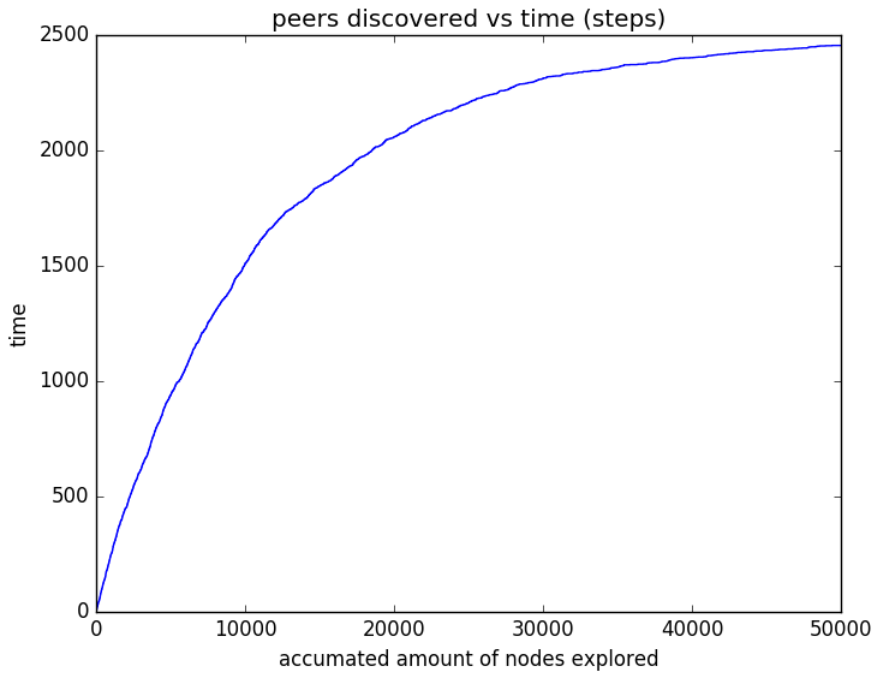

Figure 5.5: peers discovered (accumulated) over time for random walker

Take the fully random Walker as base line, no Walker demonstrate significant load imbalance

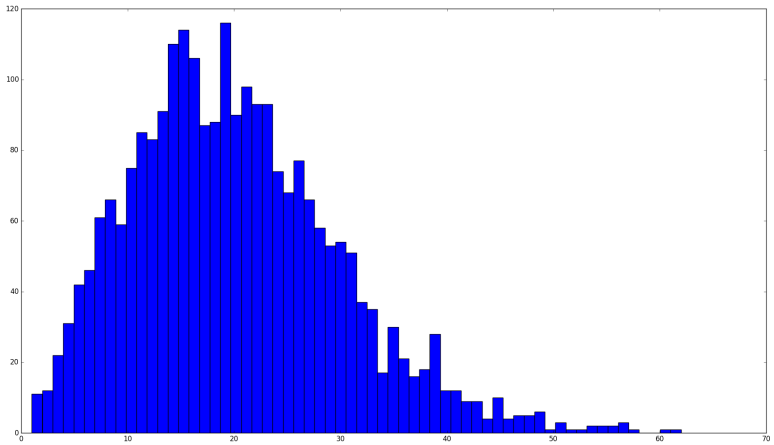

Figure 5.6: histogram of peers visted count with bias Walker

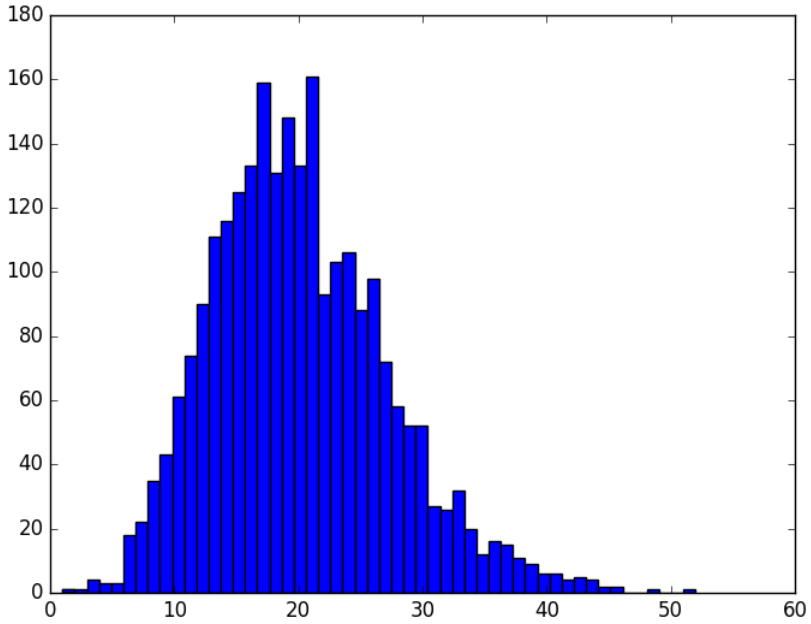

Figure 5.7: histogram of peers visted count with teleport walker with teleport home probability 0.2

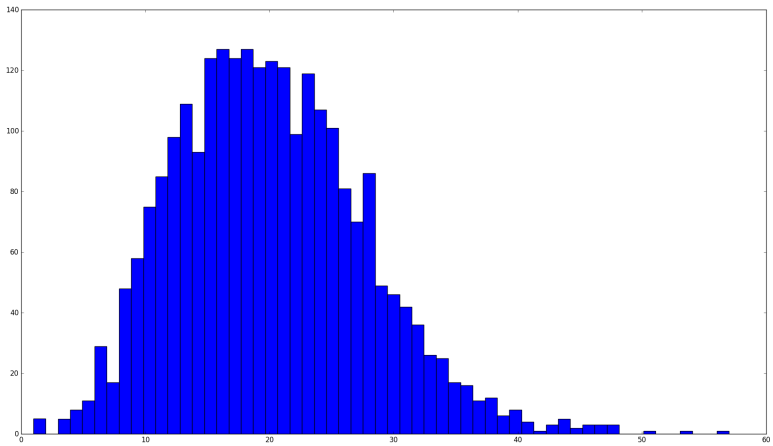

Figure 5.8: histogram of peers visted count with teleport walker with teleport home probability 0.5

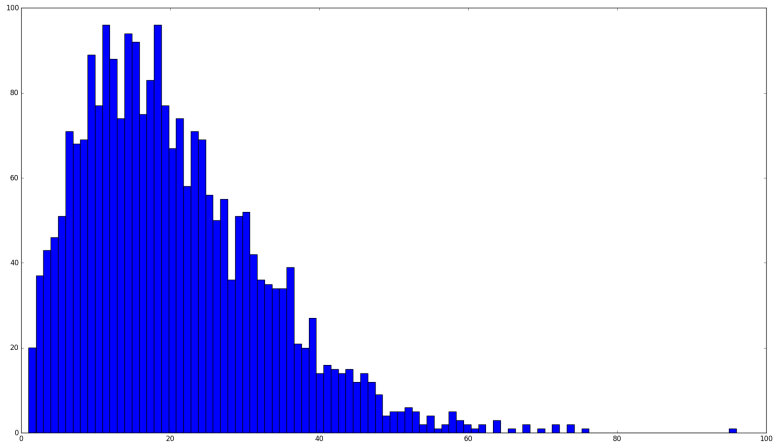

Figure 5.9: histogram of peers visted count with random walker

Chapter 6

Conclusions and Future Work

6.1 Conclusions

TODO CONCLUSIONS

6.2 Future Work

TODO FUTURE WORK

Bibliography

[1] George Danezis and Prateek Mittal. Sybilinfer: Detecting sybil nodes using social networks.

[2] John R Douceur. The sybil attack. In *International Workshop on Peer-to-Peer Systems*, pages 251–260. Springer, 2002.

[3] Jahn Arne Johnsen, Lars Erik Karlsen, and Sebjørn Sæther Birkeland. Peer-to-peer networking with bittorrent. *Department of Telematics, NTNU*, 2005.

[4] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[5] Johan A Pouwelse, Pawel Garbacki, D Epema, and Henk J Sips. A measurement study of the bittorrent peer-to-peer file-sharing system. Technical report, Technical Report PDS-2004-003, Delft University of Technology, The Netherlands, 2004.

[6] D Senie and P Ferguson. Network ingress filtering: Defeating denial of service attacks which employ ip source address spoofing. *Network*, 1998.

[7] Pim Veldhuisen. Leveraging blockchains to establish cooperation. MSc thesis, Delft University of Technology, May 2017.

[8] Wei Wei, Fengyuan Xu, Chiu C Tan, and Qun Li. Sybildefender: Defend against sybil attacks in large social networks.

[9] Haifeng Yu, Phillip B Gibbons, Michael Kaminsky, and Feng Xiao. Sybillimit: A near-optimal social network defense against sybil attacks.

[10] Haifeng Yu, Michael Kaminsky, Phillip B Gibbons, and Abraham Flaxman. Sybilguard: defending against sybil attacks via social networks. In *ACM SIGCOMM Computer Communication Review*, volume 36, pages 267–278. ACM, 2006.