DEGREE PROJECT IN TECHNOLOGY,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, *SWEDEN 2021*

How a decentralized peer-to-peer based private contact discovery system performs depending on user base size and network performance

MARTIN FAGERLUND

ADAM MELANDER

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

How a decentralized peer-to-peer based private contact discovery system performs depending on user base size and network performance

Swedish title: Hur en decentraliserad P2P baserad kontaktboksmatchning presterar beroende på användarbasstorlek och nätverksprestanda

Martin Fagerlund & Adam Melander

Supervisor: Daniel Bosk
Examiner: Pawel Herman
DA150X EECS/KTH 2021-06-09
Bachelor in Computer Science

Abstract

Internet privacy and security has become more discussed in public in recent years. Decentralization is a technique that removes trust in central authorities by distributing authority across a network of nodes. But it comes with challenges. This report investigated the advantages and limitations a decentralized contact discovery system for a messaging application may have. The areas of interest in this study concerned performance and scalability of the system. The simulation used a mathematical approach where each task's expected runtime was estimated from experiments or previous conducted studies. Multiple tests were performed to examine how different factors affected the overall performance. The results from the simulations showed that the performance and scalability of the system could be considered viable. However, since the simulation couldn't be compared to a working system or previous research and therefore lacked an estimation of its accuracy together with some limitations in the model, further work needs to be done in order to determine the simulation's predictive power.

Sammanfattning

Säkerhet och personlig integritet på internet har blivit ett flitigt diskuterat ämne under de senaste åren. Decentralisering är en metod för att undvika behovet av att lita på enskilda aktörer genom att distribuera auktoritet över flera noder i ett nätverk. Men decentralisering kommer inte utan utmaningar. I denna rapport undersöks för- och nackdelarna med en decentraliserad kontaktbokssökning för meddelande-applikationer. Huvudsakliga intresseområden för studien var prestanda och skalbarhet för systemet. Simulatorn som skapades använde en matematiskt modell där varje enskild uppgifts körtid uppskattades med hjälp av experiment eller tidigare utförd forskning. Flertalet olika test utfördes för att analysera hur olika faktorer påverkade slutresultatet. Resultaten från simuleringarna visade att systemets prestanda och skalbarhet kan bedömas som adekvata nog för att anses uppnå förväntningarna på systemet. Däremot saknades möjligheten att jämföra simulatorn mot ett utvecklat fungerande system eller tidigare forskning och därav kunde inget fastställande av simulatorns precision presenteras.

Contents

1 Introduction

1.1 Purpose

Today the vast majority of applications are using centralized servers. The servers need to grow along with the usage to match the demand and to maintain its performance. Expanding the hardware support for an application is often costly and requires predicting the future, which isn't always easy. One application that's made headlines recently is Signal [22], a private messaging application with privacy and integrity as its main focus, which has seen a massive increase in users after the most used private messaging application in the world, WhatsApp, changed their privacy policy to include a more aggressive gathering of data. This in combination with an igniting tweet [9] from Tesla CEO Elon Musk encouraging people to "Use Signal" caused a migration of millions of users to Signal in a very short time frame. The unexpected increase got the centralised system overflowed, as the servers got more requests than it could handle, which in turn led to a downtime of one and a half days.

As a centralised system requires dedicated hardware and more importantly, enough dedicated hardware to support the active user base, an issue occurs when the user base outgrows the dedicated resources. One solution utilised by many bigger applications is simply to ensure that they have enough resources, even for the unexpected. But as Signal, and many other growing applications are non-profit applications, resources are finite. Therefore the purpose of this report is to explore the possibilities to utilise a decentralized peer-to-peer system instead of a centralized system for a part of messaging applications, more specifically for private contact discovery, and see how the decentralized system would perform.

1.2 Problem statement

How does a decentralized peer-to-peer based private contact discovery system perform depending on user base size and network performance?

1.3 Scope

This report will cover the possibilities to utilise a decentralized peer-to-peer system for a contact discovery function. As it's hard to create a fair comparison between a centralized and a decentralized system, as the centralized systems performance is heavily based on the amount of resources it's given, this report will only investigate the performance and scalability of the decentralized solution and will then discuss if it could be considered viable.

1.4 Outline

The background chapter starts with a more in-depth explanation of what a contact discovery system is, describing its purpose and the current solution implemented by Signal as the model created for this study takes a lot of inspiration from it. This will be followed by a description of the core concepts of a decentralized peer-to-peer technology, including a description of distributed hash tables and their protocols. Some of the related works that this report has been based upon will also be introduced and reviewed in the background. The method chapter will focus on describing a model that a decentralized peer-to-peer contact discovery system could use, and the simulator which is used to predict the performance of the model. The method chapter will also go through the tests the simulator runs and also which data the tests are based upon. The results of our tests will be presented and we will discuss our findings to later present whether it could be considered viable or not to implement a decentralized peer-to-peer contact discovery system.

2 Background

2.1 Signal

Signal is an open-source messaging application and is funded entirely by grants and donations. Its main focus is to offer a state-of-the-art encrypted alternative with complete privacy to end-users with the guarantee that their messages can't be read by Signal or by anyone else. Signal is open-source and hence peer reviewed to gain credibility for its protocol [5]. The application gained traction last year and came to a point where their servers could not handle the traffic. Today Signal, like the majority of services, handle all their traffic on centralized servers but messages, pictures etc are stored locally on every users device [17].

2.2 Private contact discovery

This section mainly describes how Signal has developed their private contact discovery function. All information below comes from Signal's blog [19] and open source code [15].

A building block in almost all social software is a social graph. Clients need to know which of their friends are on the application and how to contact them. One way to build a social graph is to make use of an already existing one e.g. Facebook. But then Facebook is the owner of the social graph, which could be problematic from a privacy perspective. Signal on the other hand do not want anyone other than their clients to have knowledge about their individual social graph. Not even Signal itself. Signal's solution is to use the address books in client's phones and make the clients ask the server which of these are Signal users. By doing so the client's contacts keep oblivious to the Signal service if the exchange and lookup is done in a privacy-preserving way as the server does not store a copy of the graph as the social graph already exists locally. So, if the service was compromised no information about different social graphs would exist on Signal's servers. To avoid that users have to trust Signal when they say that their open-source code is actually the code running, Signal uses a feature on modern Intel chips called Software Guard Extension (SGX) which provision applications to use a "secure enclave". The enclave is isolated from the kernel and the host operating system and can perform computations on encrypted client data without the OS learning the content or the result of the computation [21]. Another feature of the enclave is the remote attestation which provides a guarantee of the code that is running so the client can confirm that the code it's expecting is the code that's running on the enclave.

The next challenge is to keep the RAM oblivious during the lookup. When comparing contacts from the client's contact book to the database with registered users the OS should not learn anything from monitoring access patterns. For example it's not possible to send a list or a hash table, encrypted or not, to the server to traverse and find which users match. This would make it possible for the server to see which users from the database connects to the client querying its contacts by monitoring memory accesses on the database. One solution is to instead compare every item in the database to the contact list. This makes the enclave touch every item from the database. By reading every entry in the database the OS can't learn anything about the contact book by monitoring memory access patterns. Lastly, a hash table is being created to increase the performance and to avoid that the contact list needs to be traversed for every user in the database. To secure that the OS once again still is oblivious, a special construction of the hash table is required. The basic idea when constructing the hash table is instead of iterating the contact list and adding each contact to its appropriate place in the hash table, iterate each cache line of the hash table and add contacts who should be stored there or write to a "dummy slot" on the cache line if the contact does not belong there. This means that every cache line receives writes during the process and the OS gets no knowledge about the hash table's layout or content.

2.3 Different types of network

2.3.1 Centralized networks

In the more traditional sense, a network consists of clients and a centralized server. The clients submit requests asking for information, and the server audits the request and either denies or approves it. If the server approves the request, it creates a package of the requested information, and sends it back to the client. A centralized network gives great control to a single authority that is trusted by the clients to manage information and resources, which gives great manageability and is usually the standard used by large scale commercial applications. The network grows by more clients joining, and thus the amount of requests submitted to the server increases, which is usually solved by adding more computing power to the server side. Adding more computing power to a server can be an expensive solution, but with the necessary means it can create powerful networks with plenty of computing power.

2.3.2 Decentralized peer-to-peer (P2P) networks

In a decentralized peer-to-peer network, there is no central authority in the form of a server that can audit requests and manage information. Instead, every user, or node as they usually are referred to in peer-to-peer networks, acts both as a client and a server depending on the context. Thus, when a new node joins the network and is generating more requests in the network, there is no need for extra server sided resources to handle said requests as the node also supplies the network with extra resources. This sort of self-sufficiency makes decentralized P2P networks scalable to the amount of users, without the need of adding extra dedicated resources [13].

Since each node can set its own rules, there is no trusted central authority figure that a client can trust. Instead, all computations and the management of information is performed by the client's peers, meaning that no one in the network can be trusted.

The network type has historically been very popular in file sharing applications, where users of the application wish to share resources with each other without the insight nor meddling from central authorities [1]. Further than the historical usage, the peer-to-peer network models usage has recently seen an increase in popularity with the increased interest of cryptocurrencies and their underlying blockchain protocols [11].

2.4 Distributed hash tables (DHT)

A distributed hash table, or a DHT, is similar to a hash table except that it's distributed among multiple nodes. It provides a lookup service with key-value pairs and participating nodes can fetch values associated with the given key. The DHT plays an important role in a decentralized P2P network as it decides how the resources of the network are managed [23]. A DHT does not require a central server, instead each node has an ID which is mapped to a certain slot in the DHT space. Each node maintains the data that is mapped into its own slot.

Today numerous variants of DHT:s exists with different advantages and different purposes. In this study Chord is the DHT-protocol discussed and used but one should be aware that Chord is one of many protocols that aim to solve the same problem.

2.5 Chord

Chord is a distributed lookup protocol and provides a solution to the fundamental problem in P2P networking i.e. how to efficiently locate the node which stores the requested data. Chord is also designed to handle aspects in P2P networks such as load balancing and scalability [23]. The DHT space is arranged in a circle where both the nodes and data are mapped into the space according to the result from a predefined consistent hash function. The circle has at most 2^m nodes (m is the

number of bits in the hash) with identifiers (IDs) ranging from 0 to 2^{m-1}. The nodes are placed in incremental order so the node with the highest ID is the closest neighbour to the node with the lowest ID. Nodes have a successor and a predecessor which points to the closest active neighbor node. If the DHT space is entirely filled the successor points to the node directly following in the ring, and vice versa for the predecessor. But it's uncommon that all positions in the ring are used so usually there is a jump between the two closest nodes. For example Node 123 may have its successor in Node 135 and predecessor in Node 117. To improve the lookup time complexity from $O(n)$ which the structure with only successors and predecessors offers, Chord also implements finger tables on each node. The finger table contains as many or fewer nodes as the number of bits in the hash (m), with the purpose of making the connectivity in the graph more effective. The ith entry in the finger table for a node N contains the first node clockwise from $N + 2^{i-1}$. The finger table reduces the lookup time complexity from $O(n)$ to $O(log\ n)$ by enabling bigger jumps in the graph which reduces the path length [23]. Figure 1 shows a simple Chord ring with what possibilities a specific node has in terms of further connection.

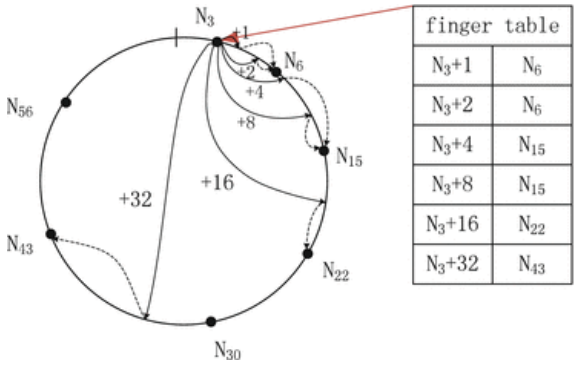


Figure 1: Model over a chord ring and a nodes finger table. Figure from DHT Theory by Zhang et al. [23]

Today, many improved and modified Chord protocols exist with the purpose of improving the average path length [6], reduce the maintenance cost [12] or achieving them both [10].

2.6 Related work

There is a lot of research conducted in parts of this study's scope. When it comes to resource discovery in distributed systems many techniques and systems have been presented. Lazaro et al. [14] did a comparison between 31 different previous presented resource discovery systems and graded them according to six different requirements: search flexibility, scalability, fault-tolerance, completeness, accuracy and security. They concluded that different topologies or systems performed better on different aspects and that all implementations need to make trade-offs to suit its purpose.

Similarly within decentralization, trade-offs are required to make the system meet its purpose. Troncoso et al. [20] revised 15 years of previous research on decentralization and privacy. They show that decentralization can enhance privacy, integrity and availability but that there is not a solution that fits every situation — "A good design for one can be an unsafe design pattern for another" [20, p.420]. Furthermore, they find that key-features in decentralized systems often fall back to being centralized because of scalability and security difficulties. The conclusion ends with them stating that "decentralization is a result from a breakdown in trust in centralised institutions" [20, p.421], but that we today have not found a way to total decentralization.

Regarding private contact discovery, Demmler et al. [7] presented a protocol where the server in an exchange with a client only learns the size of the client's contact list and the client only learns the intersection of its own contacts and the server's user database. The protocol is similar to the one Signal currently is using in their private contact discovery. However no research specifically on private contact discovery on a P2P network could be found which makes it hard to compare our results to previous presented as they would not be based on the same premises.

2.7 Other technical parameters

2.7.1 Latency

Latency is a measurement for the time delay between the moment where a signal is sent, and the moment when the signal arrives, usually measured in milliseconds. Some common causes for latency are the distance the signal has to travel and the amount of traffic the internet service providers (ISPs) are facing [4].

2.7.2 Bandwidth

When transfering data bandwidth is the capacity of a computer network in terms of how many bits can be transferred per second (bps). Internet service providers typically set the capacity limits they offer separately for upload speed and download speed, as it's more common for regular users to have a higher demand for download speed. Generally speaking, the more bandwidth available, the quicker it is to transfer data.

2.7.3 TLS

TLS is a cryptographic protocol designed to provide secure communications between nodes in a computer network. It is designed to offer privacy and data integrity after a handshake protocol has taken place between a sender and a receiver. Figure 2 displays the flow of the handshake and what each step in the handshake contains in terms of tasks. TLS is often running on top of TCP since it by itself does not offer a transport layer protocol. A complete TCP and TLS handshake requires 6 one-way trips between the sender and the receiver before a secure connection is established.

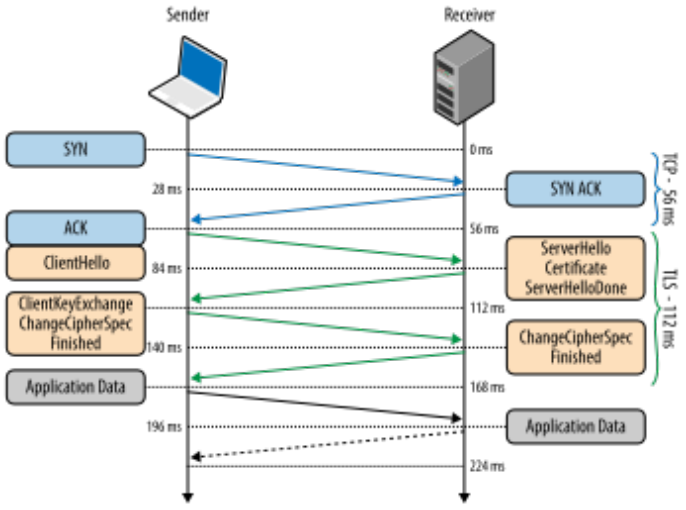


Figure 2: Model over a TLS handshake showing the six one way trips necessary.

3 Method

To be able to determine if a decentralized P2P private contact discovery function would be viable, we constructed a test environment which enabled testing a wide variety of database sizes, database distributions and network parameters. The following chapter further explains the testing environment, optimizations and finally defines the different tests used to determine the performance and scalability of a decentralized P2P private contact discovery system.

3.1 Model overview

To perform the tests a model has been designed for performing a contact discovery over a peer to peer network. The model is using a database containing the contact information for each user of the network. The database is divided in multiple packages which are distributed over the nodes in the network. Each node is found by following the Chord protocol. The node sends a request to the client to establish a TCP connection and when the connection has been established the client sends over the contact book table to the node. Subsequently the node is now able to carry out the comparison between the client's contact book table and the nodes partition of the database. The results are then sent back to the client which later can process the results to conclude the contact discovery.

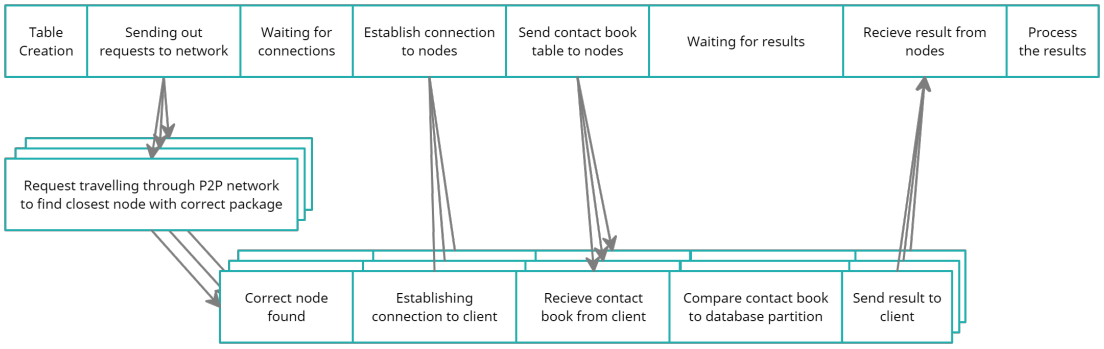


Figure 3: A flow chart describing the model.

3.2 Simulator overview

As creating a real network with millions of nodes would require unfeasible resources, the tests will be based on a simulation estimating the time required for each step in the model. As the client sends out requests to the network, the simulator creates different tracks which are able to parallelize the tasks that in a real life scenario would happen in parallel, i.e. requests travelling throughout the peer to peer network. To further make the simulations similar to a real life implementation, different limitations such as network quality and node processing speed were accounted for. They will be further explained in the sub chapters stated below.

3.2.1 Network quality

To give a fair representation of network qualities in terms of latency and upload/download speed to nearby nodes the tests are based on Ookla Open Datasets. The data set is based on data provided by Speedtest.net, which is one of the biggest fixed broadband and mobile network testers [2]. The data gathered comes from tests performed globally to the closest of 8000 different servers. Which

creates a reasonably similar distance between the user testing their internet quality and the distance between the hypothetical nodes in the model. The data gathered from speedtest.net is presented in Ookla Open Datasets as statistics in zoom level 16 web mercator tiles (approximately 610.8 meters by 610.8 meters at the equator). In each chunk the amount of unique devices tested varied. An average upload speed, an average download speed and an average latency were presented for each chunk. This is not optimal as the preferred data would be each device's raw data and not a chunk of information, however to combat this each chunk's average was weighted by the amount of devices in said chunk to give a more fair representation of the data.

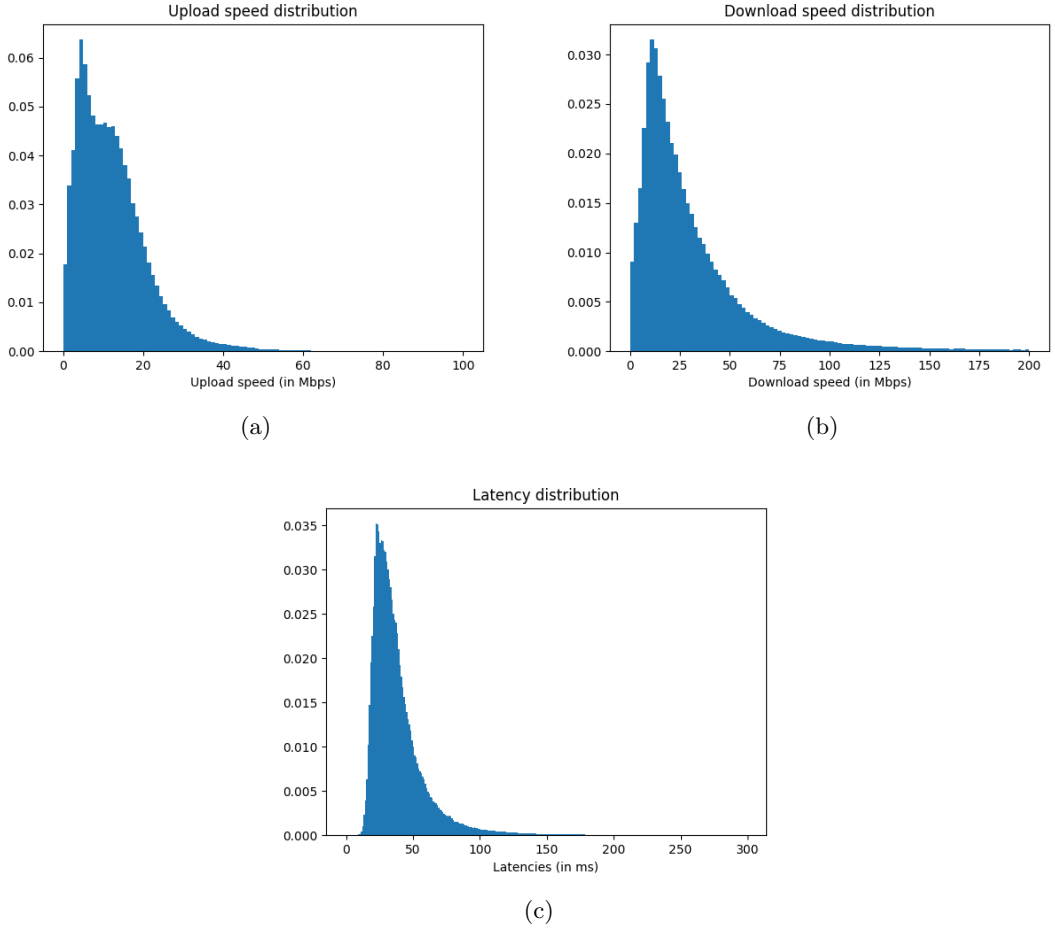


Figure 4: Upload speed (a), download speed (b) and latency (c) distributions from the first quarter of 2020 from Ookla Open Datasets [2]

It should also be said that since the Chord protocol utilises a stabilization protocol to ensure that lookups execute correctly the choice was made to exclude what could be considered "dead nodes", i.e. nodes which upload- or download speed was too low or had a latency that was considered too high for the node to be considered online. The values chosen for this was as following:

Minimum upload speed	0.1 Mbps
Minimum download speed	0.1 Mbps
Maximum latency	300 ms

3.2.2 Table creation

The creation of the oblivious hash table was moved compared to Signal from the server to instead be created by the client querying its contacts. This reduces the workload on each node during the lookup process as they don't need to transform the contact list into a hash table. We did a small experiment to examine the time it took to create said hash table. A relationship between the size of the hash table and the time it took to create it could be observed and the values were used in the simulation. Though, the time it took to create the hash table was extremely small and had little to no impact on the final performance of the contact discovery simulation.

3.2.3 Finding nodes in the network

The Chord protocol was used as the DHT protocol as it is a well-suited protocol for a contact discovery application. Chord spreads keys evenly over the nodes which create a form of natural load balancing. It is fully decentralized and every node is equally important. The cost of lookup grows with the logarithm of the number of nodes so even large systems are feasible. Chord also efficiently handles joining and leaving nodes. All these attributes are wanted in our system. Load balancing since we do not want a few nodes to perform all the work. Decentralization as it improves robustness. Scalability to avoid computational performance bottlenecks in the system and still be reasonably fast. Lastly, the ability to handle nodes joining and leaving efficiently is also wanted since it's going to be normal in the system as phones regularly get turned off, loses connection etc.

To determine how long a path between two nodes is in the simulation, an approximated distribution was created based on the experimented path length distribution given from the Chord protocol paper [18]. When deciding how long a specific path is, a random sample from the distribution was taken.

3.2.4 Connection between nodes

In the simulation, the connection between nodes used the TLS protocol as secure communication layer and TCP as the transportation layer. The time it took to establish a TLS connection between two nodes was estimated by multiplying the node latency to match a TLS handshake procedure. This simulated the whole TCP and TLS handshake procedure described in Figure 2.

To take into consideration the TCP maximum throughput, described in the following equation:

$$\frac{TCP\ window\ size\ in\ bits}{Latency\ in\ seconds} = Bits\ per\ second\ throughput \tag{1}$$

A comparison between the nodes upload speed and maximum TCP throughput was performed and set the final upload speed to the one which was lower [16]. The TCP window size remained static at the standard window size of 65536 Bytes, or 524288 bits [8].

3.2.5 Contact comparison algorithm

The time it takes for a node in the network to compare its user database to a set of contacts depends on the database size and available processing power. A simple test in Android studio was designed inspired by Signal's algorithm. The test performed a linear search across the database to see which registered users corresponds to the requested contact hashset. The database was stored as a file and, to avoid using too much primary memory during lookup, one line at a time was read and then compared that registered user to the contact set to see if the registered user also existed

in the contact set. If a contact existed in the database that contact was placed in a list which later was going to be sent back to the client.

Two different phones and one Android emulator were used to perform this experiment. Eleven different database sizes were tested, and each phone and emulator performed 7 tests on each database size. An average value for the search time on each database size was calculated and then used to display the search time as a function of database size. A second function which defined the search time variance as a function of database size was also calculated.

3.3 Optimal package distribution

The model faces a balancing issue between the time it takes for one node to compare the contact book to the local part of the database, versus the amount of time it takes for the client to send the contact book to multiple nodes. In a scenario where latency and upload time were negligible, the optimal distribution would be to distribute the computing work to as many nodes as possible to reduce the time it would take for each node to compare the contact book to the local part of the database. Meanwhile, in a scenario were the time required to compare the contact book to the database were negligible the optimal distribution would be to upload the clients contact book to only one node and let that node do all of the calculations. In reality, neither scenario is realistic and a balance between network delays and calculation times must be found and to find this balance an optimization function was developed for the model.

The optimal distribution should in theory vary depending on the size of the user base, the size of the contact book and the quality of the network as they affect the search time for each node and upload times respectively. This would lead to a different optimization being used depending on the different states of those variables. However, the size of contact books varies depending on the active user rather than the model as a whole so the optimization function is therefore based on the average contact book which includes 308 entries [3]. To give a fair representation of different network qualities, samples were drawn from the entire distribution which would mean that the optimal distribution used is based on a network with very varied qualities.

As the model is based on a variation of random sampling, race conditions and overall unpredictable methods the optimization would benefit the most from running real scenarios with different distributions to be able to compare the effect of different package distributions depending on the database size. A negative effect that should be directly descended from the models use of many random factors is the considerable amount of noise the results would be affected by. To parry the noise the optimization function takes a brute force approach and runs each scenario for multiple iterations to get a more reliable average result.

3.4 Testing

3.4.1 Optimal distribution test

To find the optimal package size for each database size an exhaustive testing algorithm was used. Two tests were executed with different database sizes. In Test 1 the database size varied from 5–100 million users and the amount of packages varied from 10–1000. In Test 2 database size varied from 50–1000 million users and the number of packages varied from 50–2000. In both tests each combination of database size and package size were tested multiple times and the average from the iterations was used as the result. In Test 1, 200 iterations were performed and in Test 2 only 30 iterations. The simulation time for larger database sizes were significantly higher and reached a point where the simulation required more computational power than we had at our disposal hence the reason why Test 2 used fewer iterations. To simulate a correct representation of network

quality the test used the data from Ookla, described in Section 3.2.1. Each node sampled a test result from their database containing real measured data.

3.4.2 Network case test

To examine how different networks' performance affected the search time in the contact discovery system the database from Speedtest was sorted and then divided into 6 partitions. Partition 1 contained the fastest 1/5 part of the data, meaning the fastest upload speeds, fastest download speeds and lowest latencies, partition 2 the second fastest and so on. This created 5 different cases with various network performances which were then used to compare the contact discovery simulation on five different networks. By using the data from Speedtest the different cases are comparative to real world network capabilities and situations that likely would occur in real life. For reference, a 6th case was added which samples from the whole distributions, to represent a scenario with a wider variety than the other cases.

Table 1: Average latency, upload speed and download speed for the six network cases.

Percentiles	Avg Latency (ms)	Avg Upload Speed (Mbps)	Avg Download Speed (Mbps)
$P_{0\%} - P_{20\%}$	20	25	92
$P_{20\%} - P_{40\%}$	26	15	35
$P_{40\%} - P_{60\%}$	33	10	22
$P_{60\%} - P_{80\%}$	42	6	14
$P_{80\%} - P_{100\%}$	74	2	6
$P_{0\%} - P_{100\%}$	39	12	34

A set with predetermined premises containing database size and package size was chosen with the information given from the optimal distribution test described in Section 3.4.1. Every network case was tested with each premise multiple times to reduce the chance of abnormal results. The premises used were the same database sizes tested in Section 3.4.1. So for the Network test 1 databases with 5–100 million entries were tested. For Network test 2 databases with 50–1000 million entries were tested.

3.4.3 Contact book size test

To study the impact the size of the contact book has on the total contact discovery search time a test was created. The test, similar to the network tests in Section 3.4.2, used the results from the distribution optimization from Section 3.4.1 to determine the optimal package size for each database size. And used the same set of database sizes as the previous tests. A set of different contact book sizes to test were determined around the average number of contacts in a phone book (308) [3]. Then each contact book was tested with different database sizes and the results were presented as a graph.

3.4.4 Network bottleneck test

To investigate which network parameter limits the model a test was created that simulates the contact discovery system over a database size of 50 million users and 500 million users while sampling from different partitions of the bandwidth distributions and the latency distribution.

This test could be interesting to determine which parameter that needs further improvement to enhance the performance of the model.

4 Results

4.1 Search time

In Figure 5 the search time can be seen as a function of the number of entries in the list. Each red data point represents an average result from a test on three devices. As shown from the trendline in the graph the search time grows linearly with the amount of users in the database. The trendline represents the function $y = 0.0009x + 112.44$.

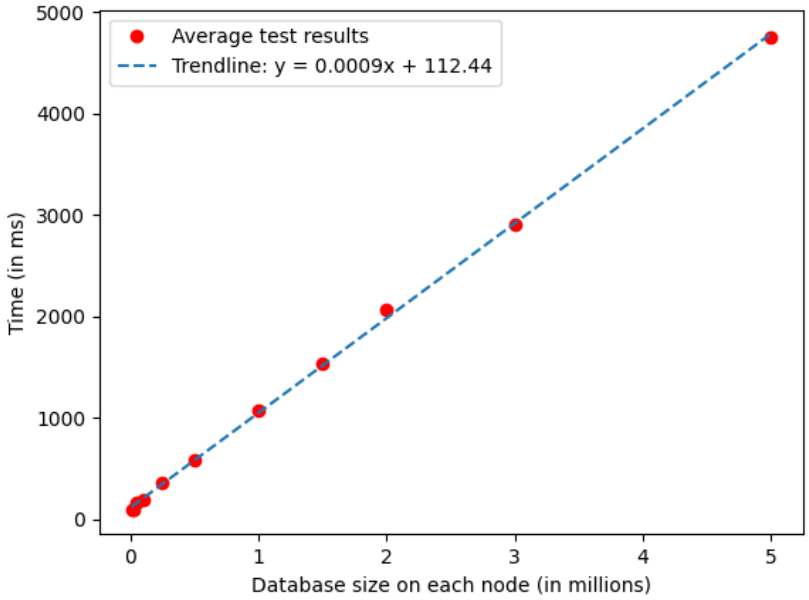
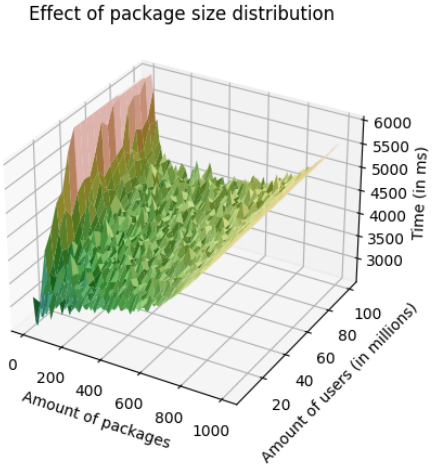


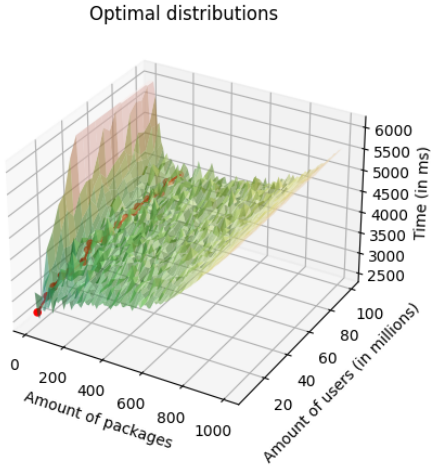
Figure 5: The average time to compare a list of SHA-256 hash entries from a file to an oblivious created hash table on a mobile device

4.2 Optimal distribution

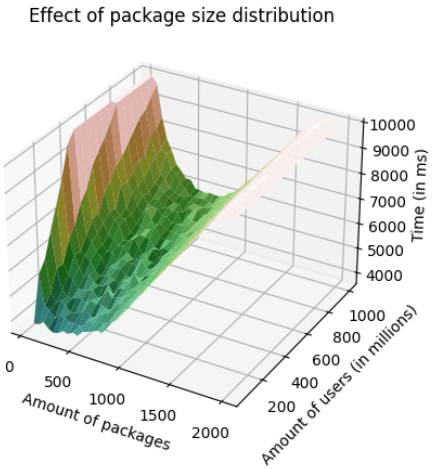
The optimal distribution of the database changes as the network grows. The optimal amount of contacts stored on each node varied between 250 000 and 570 000 entries in the first test displayed in Figure 6.a and Figure 6.b. The optimal number of packages ranged between 20-170. In the second test displayed in Figure 6.c and Figure 6.d the optimal amount of contacts stored on each node varied between 280 000 and 1.1 million. The optimal number of packages ranged between 150-850.



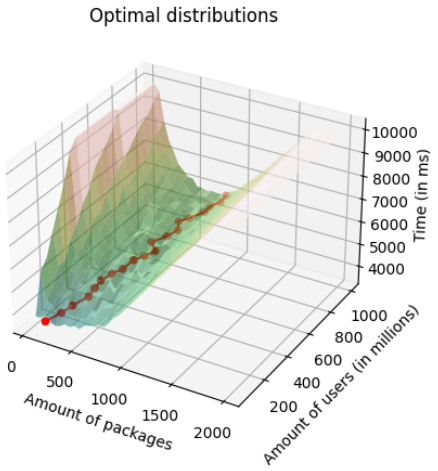
(a) Effect of package size distribution on user bases ranging between 5-100M users and 10-1000 packages. Graph is capped at 6000 ms.



(b) Optimal amount of packages from simulation displayed in fig. 6.a, optimal path is marked in red. Graph is capped at 6000 ms.



(c) Effect of package size distribution on user bases ranging between 50-1000M users and 50-2000 packages. Graph is capped at 10000 ms.



(d) Optimal amount of packages from simulation displayed in fig. 6.c, optimal path is marked in red. Graph is capped at 10000 ms.

Figure 6: Graph with simulation results with variable database distribution and user base size.

To easier understand how the runtime in Figure 6 grows. Figure 7 displays the best result for each database size from simulations presented in Figure 6.b and Figure 6.d. Each result from the red line in Figure 6.b and 6.d is displayed as a red dot. The plot shows that the runtime of the system is logarithmically related to the network/database size.

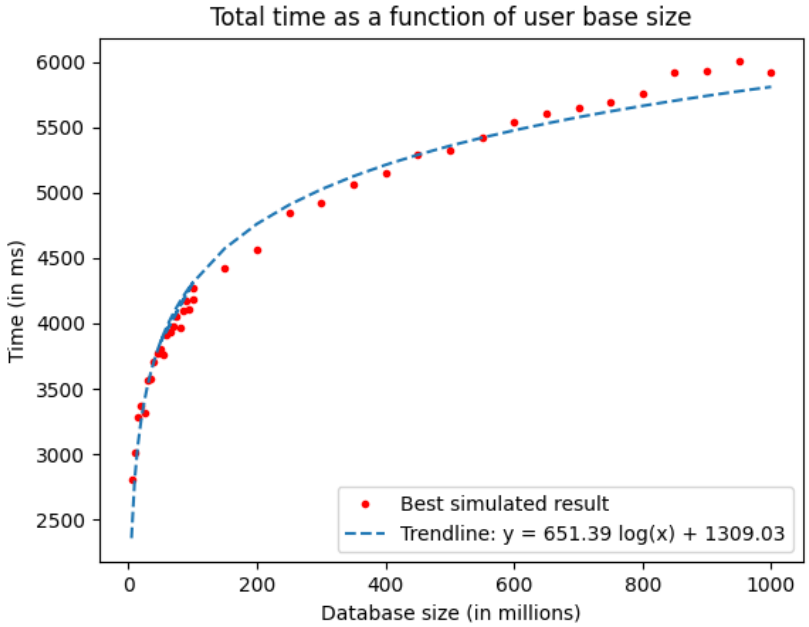
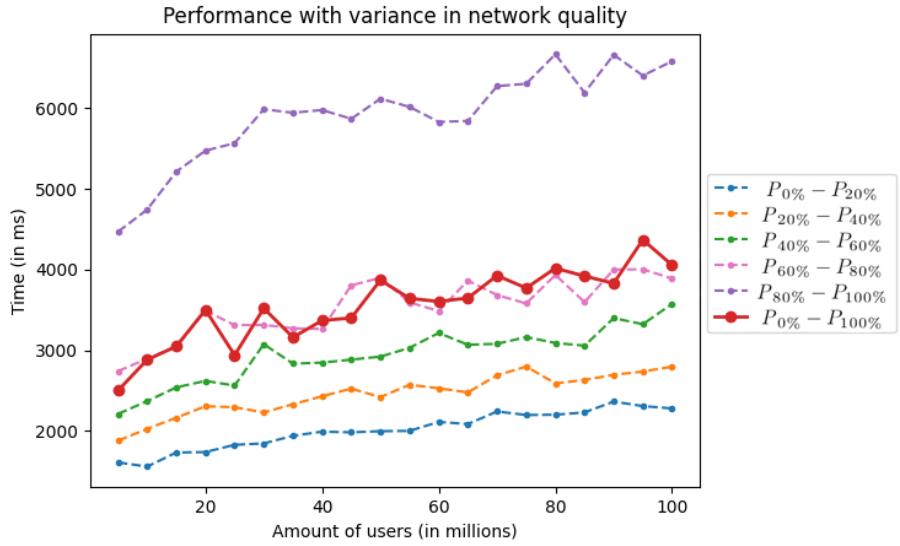


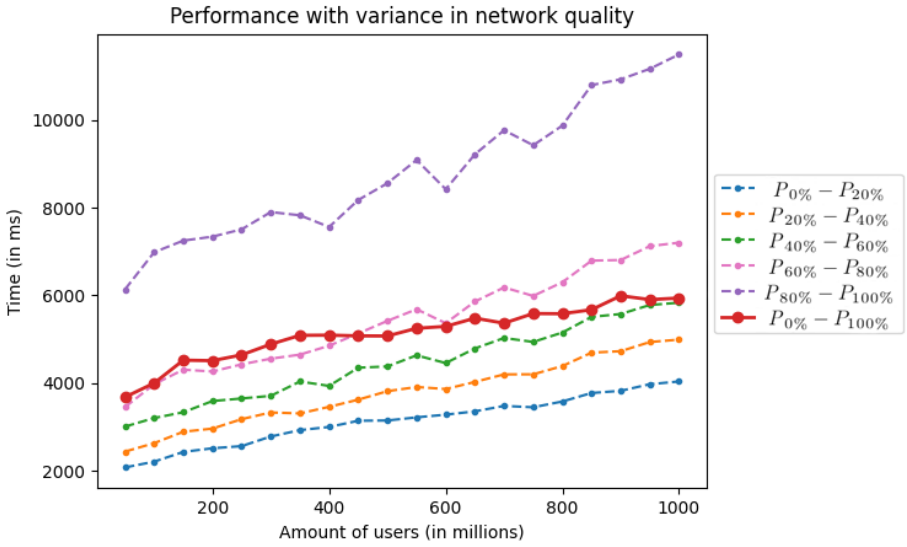
Figure 7: Total time as a function of user base size. Data points are collected from simulations displayed in Figure 6.b and Figure 6.d

4.3 Network case result

In Figure 8, simulations with different network qualities can be observed. Other than that better network quality results in better performance, the simulations showed that the group with the worst network qualities, $P_{80\%} - P_{100\%}$, gets penalized more comparing to the other groups.



(a) Networks with 5-100M user base size.

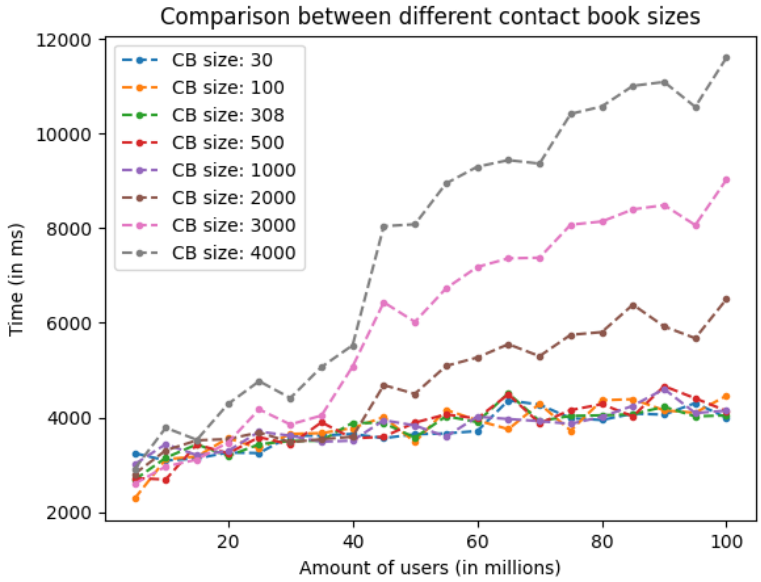


(b) Networks with 100-1000M user base size.

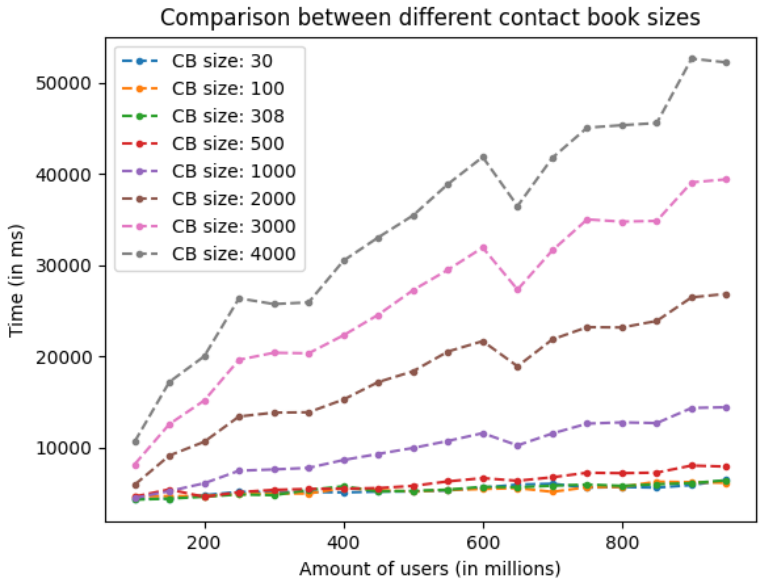
Figure 8: Simulation with different network quality partitions and varying user base size with a set contact book size of 308 entries.

4.4 Contact book size result

In Figure 9 simulations with different contact book sizes are displayed. All simulations used the same network qualities. The bigger the user base size is, the greater impact has the contact book size on the overall performance.



(a) Networks with 5-100 million users.

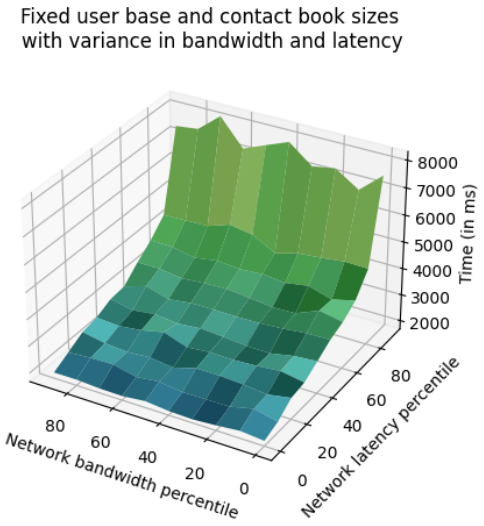


(b) Networks with 100-1000 million users.

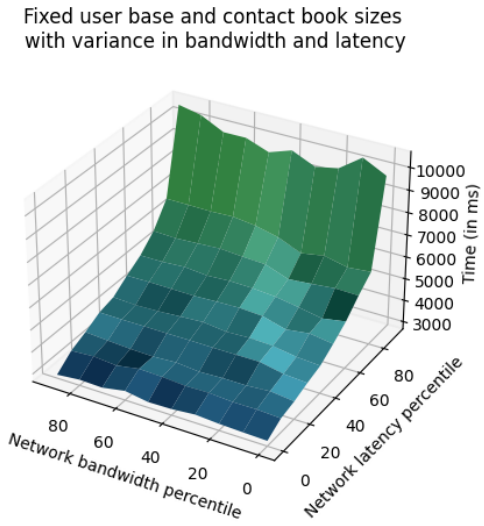
Figure 9: Simulation with different contact book sizes (CB size:s)

4.5 Network bottleneck result

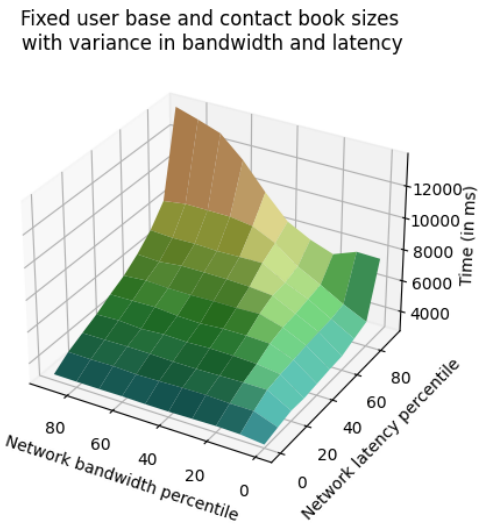
Figure 10 shows 4 different simulations with different contact book and user base sizes and where the latency and bandwidth performances changes. The x- and y-axis represent latency and bandwidth performances where lower on the axis represent better performance. For small contact book sizes, Figure 10.a and 10.b, mostly latency affect the runtime. For larger contact book sizes, Figure 10.c and 10.d, both latency and bandwidth affect the runtime.



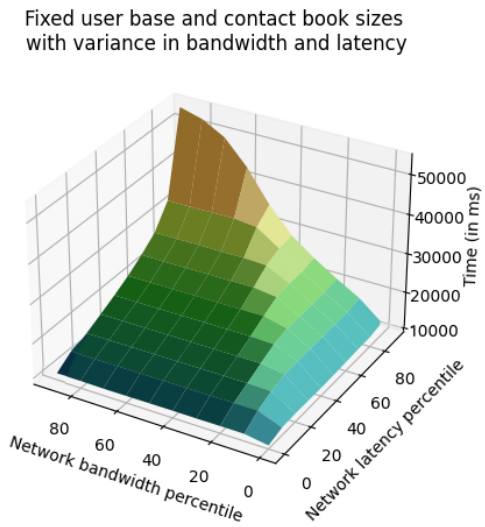
(a) User base: 50M Contact book: 308 entries



(b) User base: 500M Contact book: 308 entries



(c) User base: 50M Contact book: 2000 entries



(d) User base: 500M Contact book: 2000 entries

Figure 10: Simulation over fixed user base sizes and fixed contact book sizes while varying the network bandwidth and latency.

5 Discussion

5.1 Discussing our results

We would like to start the discussion by commenting on a few things on our results. Throughout the testing we came across results that were somewhat illogical. A run with a bigger database with the same network and distribution premise as a smaller database could result in a lower total running time for the bigger database. We believe that the reason behind this is that in the optimal distribution test too few iterations were used to cancel all possible noise created by the random sampling of network qualities. This could possibly be solved by increasing the iterations but since the simulation already took a very long time to run this was not possible in the simulator's current state. However, the difference between the simulated results and the logical results in these cases were not that substantial so the precision of the results can be considered adequate enough to be used to reach a conclusion.

5.1.1 Optimization of the distribution

By studying the results from the optimization test of the database distribution, see Figure 6, it can be concluded that too few or too many packages results in poor performance. The optimal amount of contacts stored on each node varied between 250 000 and 1.1 million entries in the two tests. For small networks (up to 20 million nodes) the database could be distributed on very few nodes and still remain reasonably good performance. But as the size of the database grows, the time it takes for a node to compare the contact book to its database becomes the bottleneck and considerably worsens the total performance. Considering that the comparison between the contact book and the database is a background process on stand-by users, it could be preferable to choose a model with too many packages, compared to the optimal, to avoid these kinds of bottlenecks but also to reduce the workload on single stand-by users in the network.

On the other end of the spectrum, a too-distributed database also impairs the system's performance. Though, as can be seen in Figure 6 the decrease in performance is not as dramatic when splitting the database into too many partitions as if using too few. The reasons behind the decrease in performance when using a too-distributed database are partly the process of setting up TLS connections to all the required nodes but also the process of uploading and sending the contact book to a high number of nodes.

5.1.2 Performance of the decentralized solution

When examining the time it takes to perform the contact discovery in the simulation, Figure 7, the results fit a logarithmic function very well. This is expected as a large part of the total runtime is the node lookup procedure. Since the lookup is done via the Chord protocol we can expect to see a time complexity of $O(log\, n)$. Further on, looking closer on the times the simulator predict it would take to perform the contact discovery it's done in 2.8 seconds for a network containing 5 million nodes, 3.7 seconds for a network with the same amount of users as Signal currently has (40 million) and around 6 seconds for a network with 1 billion nodes. These times are, in our opinion, quick enough to make the system considered viable. Though, it should be noted that these numbers were measured with the assumption that nodes in the network were able to perform its work with full computational power so the times are more than likely lower than what could be expected in a real working system. But as an indication to what magnitude of times could be expected, the results are in a reasonable window to still be acceptable even if they turn out to be a bit higher in a real working implementation.

From the network tests presented in Figure 8 in Section 4.3, it's clear that network quality has a big impact on the final performance of the contact discovery system. The difference in performance between the partitions with the best network quality and the partitions with the worst was significant. For example with a user base of 100 million, as can be seen in Figure 8.a, the partition with the best network quality took just above 2 seconds to finish while the partition with the worst network quality took more than 6 seconds to finish. We can also determine, by looking at the red line, that the overall average case tends to perform worse than the average split-up case. This could be explained in the simulation only being as fast as the slowest path. Which could be very slow if either an unfortunately slow path has been taken, a node with a capped bandwidth has been found or a combination of them both. It should be said however that in the case of a slow path in the varied case the performance would still, in most scenarios, outperform an equally slow path in a bad network case. This would be as plenty of the paths in the varied case are far enough ahead in their process to leave the slow path with less competition for resources. While in a bad network case, where the paths are more equally slow, the need for resources are more likely to occur simultaneously.

When comparing the effect the contact book size has on total performance, one can read from the contact book size test in Figure 9.a that sizes below 1000 contacts have very low impact on networks containing up to 100 million users. In Figure 9.b contact book sizes above 500 entries starts to separate from the others and makes a difference on the total performance on networks containing 50–1000 million users. A bigger contact book primarily causes a bottleneck when the client finds nodes in the network faster than it can upload its contact book to each found node. This leads to nodes having to wait to receive the contact book from the client leading to a slower overall performance. And with a bigger network size more nodes needs to receive the contact book from the client hence does the size of the contact book affect the performance even more as the network grows. There is also a slightly slower hash table creation when the contact book is larger but since this only happens once the difference isn't noticeable in the final result.

In Figure 10.a and 10.b we can see that even though the bandwidth has an impact on the performance at a regular contact book size, it's mainly the latency that can improve or deteriorate the total time of the contact discovery. The latency between nodes in the network greatly affects the node lookup time since a TCP connection between each node in the lookup needs to be established. The TCP-handshake consists of multiple messages between the sender and the receiver so a high latency results in a slow handshake. The latency has the same effect when the client finally should establish a TLS connection with a correct node to send the contact book to. A high upload speed could be crucial if the client's contact book is very large but otherwise the time it takes to upload the contact book is not that substantial. However, with the information gained from the network bottleneck test presented in Figure 10. We can conclude that as the contact book size grows, bandwidth becomes more and more important as the bottleneck potentially moves from the time it takes to find nodes in the network, to the time it takes for the client to upload its contact book to all found nodes.

5.1.3 Scalability of the decentralized solution

As shown in Figure 7 the decentralized solution scales during normal circumstances in a logarithmic way. This means that expanding the user base will leave a noticeable, but acceptable, impact until the user base consist of about one hundred million users. After that however the performance will remain very consistent with only a slight increase in the time required to perform the contact discovery even when massively expanding the user base.

5.2 Limitations

The method in the conducted study contains a series of simplifications and limitations because of its complex nature. These simplifications likely contributed to a variation in the results and a possibility of results that differentiates from ones one could expect from a fully working system.

As the method of choice in this study was a mathematical model it's hard to predict how different processes would work when later put together. Since the workflow of an operating system is very hard to simulate it's possible that processes that would run parallel, in our case instead runs sequentially or vice versa. The model also expects to always be prioritized by the OS even though many of the processes should be considered background processes or processes with low priority. How this affects final performance is something that is not discussed in this study.

The way network configurations are sampled may entail some flaws to the method. One possible flaw is that the data is collected from a website that offers internet speed testing. There is a possibility that the reason people test their internet for example could be because they experience that it's running slow at the moment which would result in an incorrect picture of peoples actual internet connection. Another potential flaw with the data is how it is presented by Ookla. The data doesn't contain individual tests from a device but instead contains average results from multiple tests from a region in the world. This removes important outliers who could have had an impact on the performance.

A factor that the simulator doesn't account for is the relation between the amount of nodes in the network and the latency between them. Assuming the application is used in a large region, there should in theory be a more or less direct relation between the amount of nodes in the network and the distance between them. This should lead to either slower latencies if the distance between the nodes is large, or faster latencies if the distance between the nodes is small. However this is a complex factor depending on a lot of regional factors and distribution of the nodes within the user region which was to complex to take into account in the simulator, but could be taken into consideration if applied in real life.

Another limitation of the simulator is the non-existing node joining and leaving. It's not possible that the simulation during the node lookup can reach a dead node i.e. a node that is no longer connected to the network. As this is something that would occur in a real network as phones lose connection, get turned off etc it is likely that it would have a negative impact on the performance. How well maintained a network remains during frequent node joining and leaving also affects the scalability aspects since if a network needs a lot of reworking when a node leaves or joins it will probably not scale very well when the network grows.

5.3 Future work

There are many possibilities for future research within the area of this thesis. The simulation used in this study has, as stated in Section 5.2, limitations in some areas that could be improved on in order to get more accurate results. Another area of interest and a topic that was not discussed in the study, which would be a vital part of the system, is the implementation of ARM TrustZone and its impact on the system. ARM TrustZone is on ARM chips the equivalent to the SGX on Intel chips. An implementation of TrustZone could possibly bring performance limitations to the system that could be crucial. ARM Trustzone is an important feature to make the application secure and private. A working prototype however would probably be the most rewarding expansion of this study as it would contribute a lot to see the model when used in a real life system. This would also be rewarding as it would become possible to draw further conclusions regarding the precision of our simulator but most important enabling to more accurately predict a real system's performance.

6 Conclusion

The limitations of this study should be taken into account and therefore no absolute conclusions can be given. However, the results from the simulation allowed for some insights on the problem. As the results showed, many factors take part in the performance of a decentralized peer-to-peer contact discovery system. Where factors such as the network quality in terms of latency and bandwidth are the most dominant. The performance of the decentralized system could be considered to be viable for the purpose of a contact discovery system during normal circumstances. The performance of the decentralized solution varies greatly given the circumstances but even with a large user base, poor network quality and a large contact book size, it still manages to perform the work in a reasonable amount of time. As this feature only is executed when someone is joining the platform, a few cases with extreme prerequisites resulting in longer runtimes would still be acceptable.

When it comes to scalability, the decentralized solution is very promising. The logarithmic complexity leads to, while still being quite good, a more noticeable growth in search time for user base sizes below 200 million. After that however, the search time increases with less than a second when doubling the amount of users. This is a great accomplishment given that no resources, besides the new users joining, needs to be added to the system to keep expanding it.

References

[1] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. "A survey of peer-to-peer content distribution technologies". In: 36.4 (2004). ISSN: 0360-0300.

[2] *Announcing Ookla Open Datasets*. 2020. URL: https://www.speedtest.net/insights/blog/announcing-ookla-open-datasets/.

[3] Frank Bentley and Ying-Yu Chen. "The Composition and Use of Modern Mobile Phonebooks". In: (2015). DOI: http://dx.doi.org/10.1145/2702123.2702182.

[4] Ilker Nadi Bozkurt et al. "Why Is the Internet so Slow?" eng. In: *Passive and Active Measurement*. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017. ISBN: 9783319543277.

[5] Katriel Cohn-Gordon et al. "A Formal Security Analysis of the Signal Messaging Protocol". eng. In: *Journal of cryptology* 33.4 (2020), pp. 1914–1983. ISSN: 0933-2790.

[6] G Cordasco et al. "F-Chord: Improved uniform routing on Chord". eng. In: *Networks* 52.4 (2008), pp. 325–332. ISSN: 0028-3045.

[7] Daniel Demmler et al. "PIR-PSI: Scaling Private Contact Discovery". In: *Proceedings on Privacy Enhancing Technologies* 2018.4 (2018), pp. 159–178. DOI: doi:10.1515/popets-2018-0037. URL: https://doi.org/10.1515/popets-2018-0037.

[8] *Description of Windows TCP features*. 2020. URL: https://docs.microsoft.com/en-us/troubleshoot/windows-server/networking/description-tcp-features.

[9] *Elon Musk Tweet*. 2021. URL: https://twitter.com/elonmusk/status/1347165127036977153.

[10] Paola Flocchini, Amiya Nayak, and Ming Xie. "Enhancing peer-to-peer systems through redundancy". In: *IEEE Journal on Selected Areas in Communications* 25.1 (2007), pp. 15–24. DOI: 10.1109/JSAC.2007.070103.

[11] Cristina Frank Bentley et al. "Cryptocurrency Networks: A New P2P Paradigm". In: *Mobile information systems* 2018 (2018). ISSN: 1574-017X.

[12] Yuh-Jzer Joung and Jiaw-Chang Wang. "Chord2: A two-layer Chord for reducing maintenance overhead via heterogeneity". eng. In: *Computer networks (Amsterdam, Netherlands : 1999)* 51.3 (2007), pp. 712–731. ISSN: 1389-1286.

[13] Yu-Kwong Ricky Kwok. *Peer-to-peer computing applications, architecture, protocols, and challenges*. 1st edition. Chapman Hall/CRC computational science series. Boca Raton, FL: CRC Press, 2012. ISBN: 0-429-09237-7.

[14] Daniel Lazaro et al. "Decentralized Resource Discovery Mechanisms for Distributed Computing in Peer-to-Peer Environments". In: (2013). DOI: http://dx.doi.org/10.1145/2501654.2501668.

[15] *Private Contact Discovery Service (Beta)*. URL: https://github.com/signalapp/ContactDiscoveryService/.

[16] Reinhard Schrage et al. *Framework for TCP Throughput Testing*. Aug. 2011. URL: https://rfc-editor.org/rfc/rfc6349.txt.

[17] *Signal and the General Data Protection Regulation (GDPR)*. URL: https://support.signal.org/hc/en-us/articles/360007059412-Signal-and-the-General-Data-Protection-Regulation-GDPR-.

[18] I Stoica et al. "Chord: a scalable peer-to-peer lookup protocol for Internet applications". eng. In: *IEEE/ACM transactions on networking* 11.1 (2003), pp. 17–32. ISSN: 1063-6692.

[19] *Technology preview: Private contact discovery for Signal*. 2017. URL: https://signal.org/blog/private-contact-discovery/.

[20] Carmela Troncoso et al. "Systematizing Decentralization and Privacy: Lessons from 15 Years of Research and Deployments". In: (2017). URL: https://sciendo.com/downloadpdf/journals/popets/2017/4/article-p404.pdf.

[21] *What is Intel SGX?* URL: https://www.intel.com/content/www/us/en/architecture-and-technology/software-guard-extensions.html.

[22] *Why messaging app Signal is surging in popularity right now*. 2021. URL: https://edition.cnn.com/2021/01/12/tech/signal-growth-whatsapp-confusion/index.html.

[23] Hao Zhang et al. *DHT Theory*. Springer, New York, NY, 2013. URL: https://doi-org.focus.lib.kth.se/10.1007/978-1-4614-9008-1_2.