Design of a surveillance and censorship resistant communication mechanism over the internet

- Semester 7 Report -

Pasan Tennakoon
Supipi Karunathilaka
Rishikeshan Lavakumar

Department of Computer Engineering
University of Peradeniya

Final Year Project (courses CO421 & CO425) report submitted as a requirement of the degree of
B.Sc.Eng. in Computer Engineering

October 2020

Supervisors: Dr. Janaka Alawatugoda (University of Peradeniya)

We would like to dedicate this research to all who helped directly and indirectly to accomplish this. ...

Declaration

We hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my/our own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgments.

Pasan Tennakoon
Supipi Karunathilaka
Rishikeshan Lavakumar
October 2020

Acknowledgements

We would like to express our deepest appreciation to all those who helped to complete this research. We give special gratitude to our final year project supervisor, Dr. Janaka Alawatugoda.

Abstract

Surveillance and censorship-resistant communication have been an interesting topic in online communication. In this research, we report shortcomings of the modern centralized architecture based communication. Then we explore design techniques that protect users' privacy. Under design techniques, we explain how distributed and decentralized systems provide the privacy that users envy along with NAT navigation techniques and methods of safeguarding privacy and anonymity in communication. Then we present our design of a hybrid decentralized communication system and utilize this to establish DTLS tunnels to maintain connectivity among devices behind NAT. Design covers explicit details of the communication protocols designed for connection establishment, communication, tunnel establishment and connection termination. Then we present our experiments and results to measure the performance of the system in a real environment. Then we conclude the research along with the results and the shortcomings of the current design.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

AES	Advanced Encryption Standard
BFS	Bredth First Search
CA	Certificate Authority
DTLS	Datagram Transport Layer Protocol
E2EE	End to End Encryption
GRE	Generic Routing Encapsulation
HTL	Hops To Live
IP	Internet Protocol
NAT	Network Address Translation
P2P	Peer to Peer
PPP	Point to Point Protocol
PPTP	Point to Point Tunneling Protocol
PSK	Pre Shared Key
SSL	Secure Sockets Layer
STUN	Session Traversal Utilities for NAT
TCP	Transmission Control Protocol
TLS	Transport Layer Security

TOR	The Onion Routing
TTL	Time To Live
TURN	Traversal Using Relay NAT
UPnP	Universal Plug and Play
VPN	Virtual Private Network

Chapter 1

Introduction

1.0.1 Background

The Internet has expanded around the world faster than any other technology. The proliferation of internet usage has encouraged more and more services to be provided over the internet. People have accustomed to sacrifice their private information to achieve the convenience of these services. This has attracted governments[1], large organization[2], and criminal organizations to monitor people's interactions over the internet. Although governments justify their mass surveillance programs by saying it's to prevent criminal activities, whether it's legal or not remains a question[3] .Recent accusations on companies like Facebook[2] and Huawei for surveilling on their customers has become an eye-opener for people to realize the importance of private and surveillance-free communication over the internet. Even if companies that have access to people's private information ensure that they do not collect data for their own profit, access to invaluable data is still vulnerable. If this data gets compromised by a malicious attacker, can easily be used to commit various fraudulent activities. These activities can range from simple spamming to cyber extortion.

Since people increasingly rely on the internet, rights that people appreciate in day to day life need to be protected on the internet too. Governments that should protect people's freedom of speech, information, and privacy seem to be the main miscreant who violates these rights[3]. Therefore there is a great need for technologies that protect people's privacy over online communication.

1.0.2 The Problem

Modern internet relies heavily on centralized architectures to provide services. This creates a centralized control of information. Although it is preferable for authorities to have centralized control over an application, users need to trust the authorities to protect the privacy of the communication.

1.0.3 The proposed solution

This research explores a way of using a hybrid approach for decentralized communication and utilize this to establish DTLS tunnels to maintain connectivity among devices behind NAT.

1.0.4 Deliverable and milestones

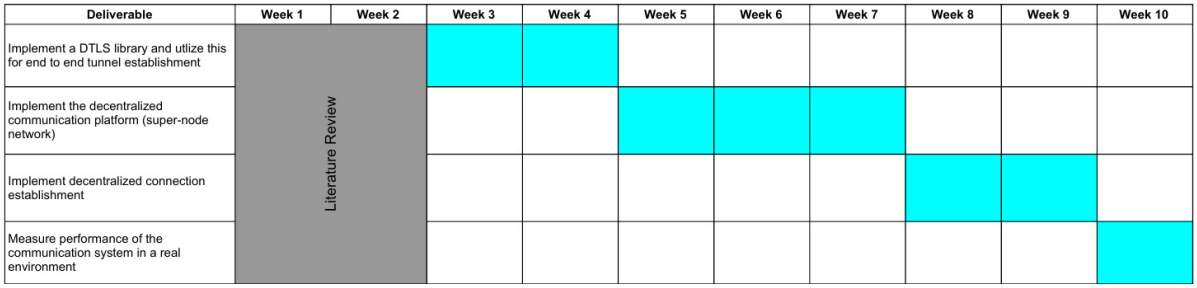

Fig. 1.1 Deliverable and milestones

Chapter 2

Related work

This section explore existing techniques in application designing that hinders the possibility of establishing surveillance over internet communication. We present how different strategies can be utilized from connection establishment to communication, to protect users' privacy. The communication strategies are examined along with real-world applications.

2.1 Design architectures

Centralized communication architecture has been norm for online communication for a very long time. Centralized systems force all network traffic through a single server. Therefore creates a possibility for communication to be monitored. Applications that resist surveillance are designed to be decentralized, distributed or both.

2.1.1 Decentralized Systems

Decentralized systems do not have single centralized management. Instead have multiple decision-making nodes. Access time and performance of these systems are significant to centralized systems. Decentralized systems can adapt two architectures. One is peer to peer architecture[4] where every node is equal and the other one is master-slave architecture[5] where special nodes are selected to have more responsibility of the system. Bitcoin is an example of a peer to peer decentralized application. Skype peer to peer, TOR project are applications that use a master-slave decentralized system. Fig. 2.1 depicts a figure of Skype architecture.

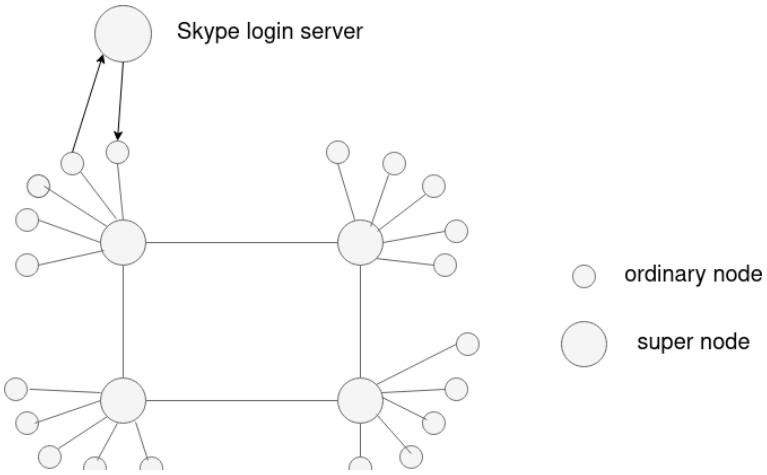

Fig. 2.1 Skype architecture

Although decentralization provides better performance, this creates security and privacy issues.

2.1.2 Distributed Systems

Distributed application has no single point where the decisions are made. Nodes involved in distributed applications can make decisions on its own. Distributed systems provide increased security and reduce privacy concerns. Since the individual failure of nodes does not affect the system, the availability of the system improves. There are three architectures for distributed systems. Peer to peer, Client-server where nodes are selected to act as servers and clients interchangeably and n-tier architecture where the system is partitioned into n tiers. Bitcoin is an example for a peer to peer distributed application. Although distributed systems seem preferable than both decentralized and centralized systems, these systems are difficult to design and develop.

Notice that peer to peer architecture is under both distributed and decentralized systems. This is because there are three types of peer to peer systems[6]. Pure peer to peer systems are distributed. Mediated peer to peer systems use a central server to handle user authentication and manage databases. Hybrid peer to peer systems are decentralized.

2.2 Connection establishment

Connection establishment is a crucial step in communication. Connection establishment techniques differ according to the architecture of the system. We examined different connection establishment techniques utilized in different peer to peer systems[7] which incorporate all previously mentioned architectures.

2.2.1 Connection establishment in Mediated P2P

Mediated peer to peer systems are designed to lessen the complexity of connection establishment. Mediated p2p systems keep a login server which keeps information of client authentication details. All nodes are connected to a central server for authentication. Also, the central server keeps track of the status of its peers. This simplifies the connection establishment but introduce a single point of failure.

2.2.2 Connection establishment in Pure P2P

Pure peer to peer systems are completely distributed systems. No central administrative services to keep information about peers. Therefore connection establishment techniques in pure p2p systems are interesting. There are many types of research focused on improving connection establishment in pure p2p systems[8], [9]. Two most common methods are flooding and random walk. Flooding searching is exhausting and not suitable for large applications. However, there exist many algorithms to optimize the flooding search. TTL restricted flooding, Modified BFS, Directed BFS, Associated search, etc[10]

In random walk rather than broadcasting a node discovery message, node discovery message is routed in random paths. This reduces the bandwidth exhaustion of flooding search. However random walk does not always provide results.

2.2.3 Connection establishment in Hybrid P2P

Hybrid peer to peer systems[11] are a combination of pure and mediated p2p systems. Hybrid p2p systems are designed to overcome the problems of both mediated and pure p2p systems. Hybrid p2p systems provide search efficiency while maintaining reliability. In hybrid peer to peer systems peers are selected to act as super-peers that have more responsibility for the system. These super-peers are in charge of multiple peers.

2.2.4 Applications

Skype peer to peer

Skype peer to peer is an example of a collaboration between mediated p2p architecture and hybrid peer to peer architecture. Skype has three types of nodes in its architecture; super-nodes, ordinary nodes, and a login server. Super-nodes have more responsibility for the network. The login server keeps the register of users and passwords for Skype clients. The initial connection of a Skype client is to the login server. There, the user needs to authenticate himself. After successfully authenticating, the client tries to connect to well-known super-nodes. Among multiple super-nodes, the client selects one super-node and create a TCP connection with that node. To search a user across the Skype network Global Indexing and a variation of a flooding search is used. Skype's node discovery algorithm is told to provide definite results if the users have logged into a Skype server in the last 48 hours. After the user is found, Skype creates a connection between the two peers.
Skype had no centralized server which is perfect for surveillance-free communication. However, there were small issues with privacy. Since the IP address of its peers is accessible[12] this was misused by cybercriminals to perform DOS(Denial of Service) attacks. Also [13] proposes a way of using a traffic analysis attack on Skype to reveal the identity of the clients in 0.33 to 0.55 seconds. After Microsoft bought Skype, they introduced Skype for smartphones. Then the notion of using super nodes became a problem. Since smartphones lacked the processing power and bandwidth to act as super nodes, the number of ordinary nodes increased substantially while the number of super nodes reduced. The architecture of Skype is similar to what we hope to implement, however we need to avoid using a login server(a single point of failure) and therefore authentication complexity increases.

TOR

TOR communication is based on Onion routing. There are three connection establishment procedures in TOR. In this paper, we only focus on version 3. To initiate a TOR connection each relay of the TOR network should report their status to a set of directories. Hosts learn the availability of the relay nodes by periodically calling the directories. Directories respond with a consensus document to the host to determine the relay servers that can be used to establish a route. Directories use a voting procedure to establish the consensus document. According to the received set of relay servers, the host initiates a route chosen randomly at each node to create an anonymous connection.

TOR uses a Diffie-Hellman key exchange to establish a TLS connection between each node. Hosts periodically terminate the connection after a predefined time and establish a new connection using a different set of relay servers. This type of connection provides anonymity for the host since the host is only visible to the initial relay node and the destination is visible only to the exit node.[14] provide a detailed description of TOR's relay selection algorithm. This significantly slows the connection.

2.3 NAT navigation

NAT introduce complications for normal end to end connections. While NAT would work well for typical client server communications since it's always the client that initiates the conversation and normally client doesn't need to maintain the connection for a long time, however installation of NAT would cause major problem for peer-to-peer communication. NAT makes peers behind NAT are not publicly accessible. There exist different ways of traversing NAT. However each have their own limitations.

2.3.1 UPnP

UPnP allow devices behind NAT environment accessible form outside the network. UPnP figures out the external IP address of a client behind NAT and creates a port forwarding mapping in the router. UPnP provides automatic service discovery, automatic addressing and zero configuration[15]. However UPnP is not widely used today due to it's security issues. A malicious program can utilize this UPnP feature to create a port mapping in the router and allow devices to be accessible outside the network. [16] provides different security issues of UPnP.

2.3.2 STUN

STUN allows two devices behind NAT to establish a connection using a STUN server. A device can contact s STUN server by sending a STUN request. STUN server will put the source IP address and the port of the packet(IP and port changes due to NAT) as the content of the packet and send it back to the device. Now any device that need to connect to that device can use this IP address and port information to establish a connection. The main issue with STUN is it does not work with symmetric NAT since NAT mapping changes due to destination IP and port tuple.

2.3.3 TURN

TURN utilize a very simple method to solve the issues with NAT navigation. TURN uses a relay agent to relay a communication between two devices. TURN server is publicly accessible therefore both devices can contact the TURN server. However TURN is really resource exhaustive. TURN works in any different NAT environment unlike STUN protocol.

2.4 Privacy and Anonymity of communication

Protection of online communication depends on two main factors. Privacy[17] and anonymity. Some application designers focus on the privacy of communication while some utilize anonymity. Anonymity and privacy often get mistaken with one another. However, they are different and selecting between two depending on the application and the domain. Application designers need to have a clear understanding of how to utilize these two to protect online communication.

2.4.1 Privacy

Privacy of online communication based on control and trust. Privacy is a question of whether a user can trust the entities that have control over his/her information. That is why most of the privacy-enhancing application rely on distributed systems. Distributed system have less control over the information. Therefore users are able to trust them to some extent. The most common solution to protect the privacy of a user is encryption. SSL (Secure Socket Layer) and TLS (Transport Layer Security) are widely popular in online communication. TLS is the successor of SSL. SSL and TLS establish a tunnel between the two communication endpoints by using public key certificates validated by a certificate authority[18].
Certificate authorities are assumed to be trustworthy. Therefore users can trust the certificates obtained by a certificate authority. In end to end encryption(E2EE), data is encrypted by the sender and can only be decrypted by the receiver. The encryption key needs to be known by the two communication endpoints. This can be achieved by a pre-shared key or by exchanging a key using Diffie-Helman key exchange to produce a session key.
In distributed systems information is distributed. Therefore vulnerability of having all the information in a single place does not exist. However, this means communication is traversing through multiple peers which cannot be trusted. Certificate authority

based encryption schemes are not useful since no centralized control of the system exists[19]. E2EE security models can be utilized in distributed systems to encrypt the communication on end to end basis. A proper way to exchange a key is needed. [20] proposes a way of using a multipath Diffie-Helman key exchange to create an E2EE between two communicating peers. Distributed systems that have a mediated server can use this to initiate secure communication. However, then users need to explicitly trust the mediated server to protect the authenticity of the keys (public keys). In decentralized systems, this becomes a responsibility of the decentralized main servers(super-peers in hybrid p2p).

2.4.2 Anonymity

Anonimity[21] restrict surveillance by hiding the identity of the user rather than concealing the communication. Pseudonymity is closely related to anonymity but not the same. Anonymous applications have no identifiers that could potentially identify a user. Pseudonymous applications conceal the identity of a user by using a fake identity. Using the public key is the most simple method of achieving pseudonymity. Most of the applications mentioned before utilize anonymity or pseudo-anonymity to protect users' privacy. This includes TOR and Bitcoin. VPN also can be an important tool in providing anonymity by using it with TOR. There are three types of anonymity in communication[22].

1. Sender anonymity
2. Receiver anonymity
3. Sender-receiver anonymity

Crowd[23] Hordes[24], Freenet[25] and Tarzan[26] are some anonymous protocols that are used in p2p systems.

2.4.3 Applications

TOR

TOR utilize Onion routing to provide anonymity to the communication. TOR was designed to provide privacy for web surfing. As we discussed there are four types of relays in the TOR network. Entry relay is the entry point to the mesh of relay servers. When a client surfs the web and needs to access a server, TOR encrypts the packets using a

session key which he exchanged with the exit node. TOR adds the IP address of the exit node and adds another layer of encryption using a session key exchanged with the middle relay. Then again the IP address of the middle node is added to the data and gets encrypted with a session key exchanged between him and the entry node. TOR sends these packets to the entry relay where the first layer of encryption gets decrypted. Their entry relay can read the IP address of the middle relay. Then the packets get passed to the middle relay. Similar to that at each relay a layer of encryption gets removed, therefore at the exit node all the layers are removed and exit node can send packets to the corresponding server. This concept of Onion routing is shown in Fig. 2.2. [27] provides a comprehensive report on how TOR works. Onion routing provides true anonymity while surfing the internet.

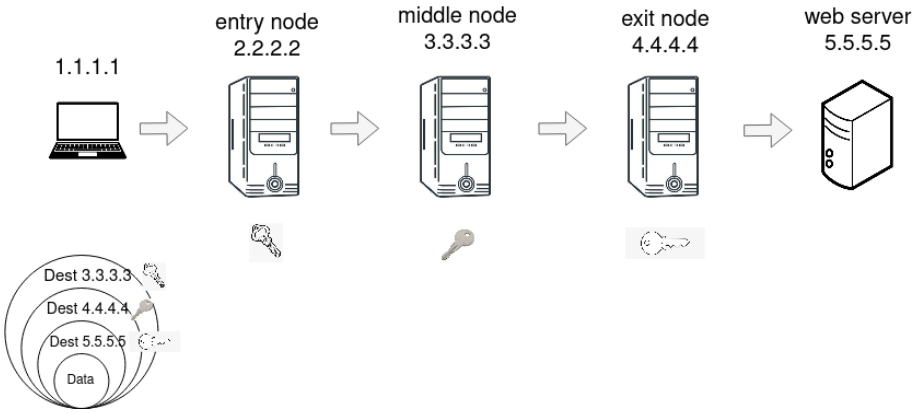

Fig. 2.2 Onion Routing

This concept of onion routing provides true anonymity when surfing the internet. However, users still have to care about the content they share, which can be traced back to you. Still, TOR is not 100% safe. Over the years multiple issues with TOR that could potentially reveal your identity have been published. In 2017 Dr. Neal Krawetz posted that TOR gives away details about the users through the window and screen size of the browser. In 2016 a zero-day vulnerability is found in TOR and Firefox which could potentially allow attackers to target IP and MAC addresses of its users. [28] is a forensic analysis on TOR which has to lead them to find a number of digital artifacts that TOR leaves on the host machines that could compromise the user data. However, the main issue of TOR technology remains as it's misusers. TOR's hidden services have created a pathway for criminals to communicate freely and anonymously. Dark web[29] is a part of the internet that is not indexed by web browsers. In 2016 it has been found that over

60% of the dark web is handled by criminals for criminal activities. TOR which is also known as the dark web browser is used by criminals to stay active on these sites without being detected.

Bitcoin

Bitcoin is an distributed peer to peer system. Bitcoin wallet is used to generate Bitcoin addresses which is a person's identity in the Bitcoin network. Bitcoin uses a shared public ledger to keep the identities of the clients in a secure and anonymous way. This also keeps track of all the transactions. There is no central authority. Therefore the integrity of the public ledger needs to be authenticated by the two clients who are involved in the transaction. Since every participant keeps a copy of the ledger, consensus algorithms and encryption mechanisms are used to keep the consistency and integrity of the ledger. [30] provides an overview of the provided pseudonymity of a Bitcoin transaction. A Bitcoin transaction is signed by the private key of the sender. This prevents the transaction from being altered by anyone. Every transaction gets broadcasted across the Bitcoin network and the confirmation of the transaction happen using a process called mining. Mining is done by Bitcoin miners. Miners need to solve a difficult mathematical puzzle to mine Bitcoins. This is known as proof of work. The miner who first completes the puzzle gets a reward in Bitcoin. This is how new Bitcoins are minted. When a transaction is confirmed it gets added to the blockchain. This ensures Bitcoin users to agree on the state of the Bitcoin system. In the blockchain, the integrity of the transaction is secured since any alteration on a single block invalidate all the subsequent blocks.
Bitcoin is anonymous in the sense that a Bitcoin address is not linked to the identity of the user in any way. Also, a user can have multiple addresses that do not link to each other. However, since every transaction being recorded in the Bitcoin network, users need to be careful not to leave any trace that could potentially reveal your identity[31]. Bitcoin users use a number of clever ways to protect their privacy while using Bitcoin. Bitcoin is vulnerable to Sybil attacks. If an attacker controls a significant portion of the Bitcoin nodes, it is likely that a user is connected to only the attacker nodes. Double spending is another vulnerability in Bitcoin. [32] presents a clever way of using a combination of double spending and Sybil attacks on a Bitcoin blockchain. DOS attacks are another potential risk when using Bitcoin. Although Bitcoin has a number of potential vulnerabilities and privacy issues[33], it is still considered one of the best ways to avoid surveillance when doing payments over the internet.

VPN

In VPN privacy of the hosts are protected using encrypted tunnels. Different protocols of VPN use different methods to protect users' privacy. In this section, we discuss the connection establishment of PPTP(Point to Point Tunneling Protocol) VPN, since this is the most basic VPN protocol. Figuring out PPTP will help in understanding more complex VPN protocols. PPTP[34] is considered the fastest VPN protocol. Therefore it is used in a variety of streaming and gaming applications. PPTP is an extension of PPP(Point to Point Protocol). PPTP encrypt data packets to create an illusion of a network tunnel across the public internet. PPTP uses TCP port 1723 to establish a TCP connection with the ISP. This connection is used to create a GRE (Generic Routing Encapsulation) tunnel. GRE packets are encapsulated into IP packets and sent to port 47. PPTP has two types of tunnelling methods. One is voluntary tunnelling where the tunnel is created by a client. Another one is compulsory tunnelling where the PPTP server initiates the tunnel. In PPTP packets are transferred through an encrypted tunnel. PPTP is a data-link layer protocol. PPTP uses the same negotiation, authentication and encryption schemes used by PPP. PPTP encapsulates PPP packets within an IP envelope inside a GRE tunnel. Therefore any device along the path will treat the packets as IP traffic.

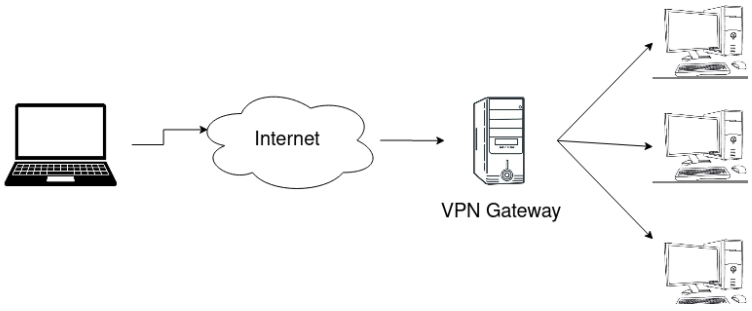

Fig. 2.3 VPN architecture

PPTP is still used in spite of its vulnerabilities. This is because PPTP is considered fast compared to other VPN tunnelling protocols and it's easy to set up.

The issue with VPN is, it is a centralized technology. Every packet a user send using a VPN tunnel goes through a VPN server. It is found that more than 20% of VPN servers contain potential malware. 44% of VPN providers keep usage logs and 6% of VPN servers log browsing activities. Even if the VPN provider does not use these data for malicious activities this makes VPN providers a primary target of cyberattacks. Also

with the proliferation of VPN servers, there are many poorly constructed[8] VPN service providers. Even a good VPN service provider is not secure from government surveillance. [35],[36] are surveys that have provided a comprehensive description of security issues of VPNs. Although the primary goal of using a VPN is to avoid surveillance it's centralized architecture has forced it's users to blindly trust a third-party service provider to provide users with required privacy.

2.5 Summary

These applications are built to avoid surveillance. However each of these have short comings. For example although Skype's architecture provides a good solution against surveillance, Skype peer to peer is a proprietary application and the communication protocols are not properly documented. Therefore cannot be utilized as a platform for generic communication. Also Skype's login server creates a possibility to censor the communication which we are trying to avoid in this research. The technology in VPN cannot be utilized in our design since this force all traffic to go through a server, which can be use by authorities to monitor the communication. TOR is a application designed to provide anonymity in web surfing and not to keep connectivity between two devices. However TOR's utilization in Onion Routing is an interesting method of providing anonymity in communication. Bitcoin also has the same problem, Bitcoin is build for anonymity in online transaction not for generic communication, however the underline block-chain technology is an interesting approach for providing pseudonymity in communication.

Chapter 3

Methodology

3.1 Methodology

As we understood there are three methods to avoid centralized communication.[6]

- Purely distributed
- Mediated
- Hybrid

Purely distributed systems provide the most satisfactory censorship-resistant communication. All participating nodes have equal responsibility for the system. However, connection establishment seems to be an exhaustive and an inefficient task [10], [8], [9]. Mediated systems are designed to lessen the complexity of connection establishment by keeping a login server. This creates a single point of failure and the possibility of censorship and surveillance. Hybrid systems are a combination of purely distributed and mediated systems. Hybrid systems are designed to overcome the problems of the two former systems. Hybrid systems provide search efficiency while maintaining the reliability of decentralization. The goal of this phase of research is to implement a hybrid decentralized system to accomplish censorship-resistant private communication. [37]

3.2 Design

3.2.1 Conceptual design

System is made up of two types of nodes.

- Nodes
- Super-nodes

Nodes are considered to be service requestors. Nodes do not have control over the behaviour of the system. They are connected to the system through super-nodes. Every node is assumed to be behind a NAT environment. Nodes can request services from a super-node to establish an end to end tunnel to another node. Nodes can leave the system at any time without affecting the topology of the network.

Nodes that have publicly available IP addresses, high network bandwidth, enough CPU power and long connectivity are promoted to be super-nodes. Super-nodes have more responsibility for the system. A super-node is connected to one or more super-nodes in the network and responsible for one or more nodes. Super-nodes can communicate among other super-nodes using the super-node network. Super-nodes can join or leave the network at any time. This dynamic behaviour of super-nodes changes the topology of the entire network. Fig. 3.1 illustrates the system architecture consisting of nodes and super-nodes. This is similar to the architecture of Skype P2P. However Skype use a login server for client authentication which creates a single point of failure [38].

Super-nodes are connected to each other using TCP connections secured with SSL[18]. Nodes also use SSL[18] secured TCP connections to connect to a super-node. At the request of any two nodes, a super-node can create a DTLS tunnel to allow nodes to maintain an end to end connectivity.

This phase of the research involves handling the following problems in the system

- Handle dynamic behaviour of super-nodes.
- Decentralized and pseudonymous node discovery.
- NAT navigation of nodes.
- Protect confidentiality of node to node communication.

Existing super-nodes are expected to leave the system at some point and new super-nodes are expected to join. Dynamic behaviour of the super-nodes should not affect

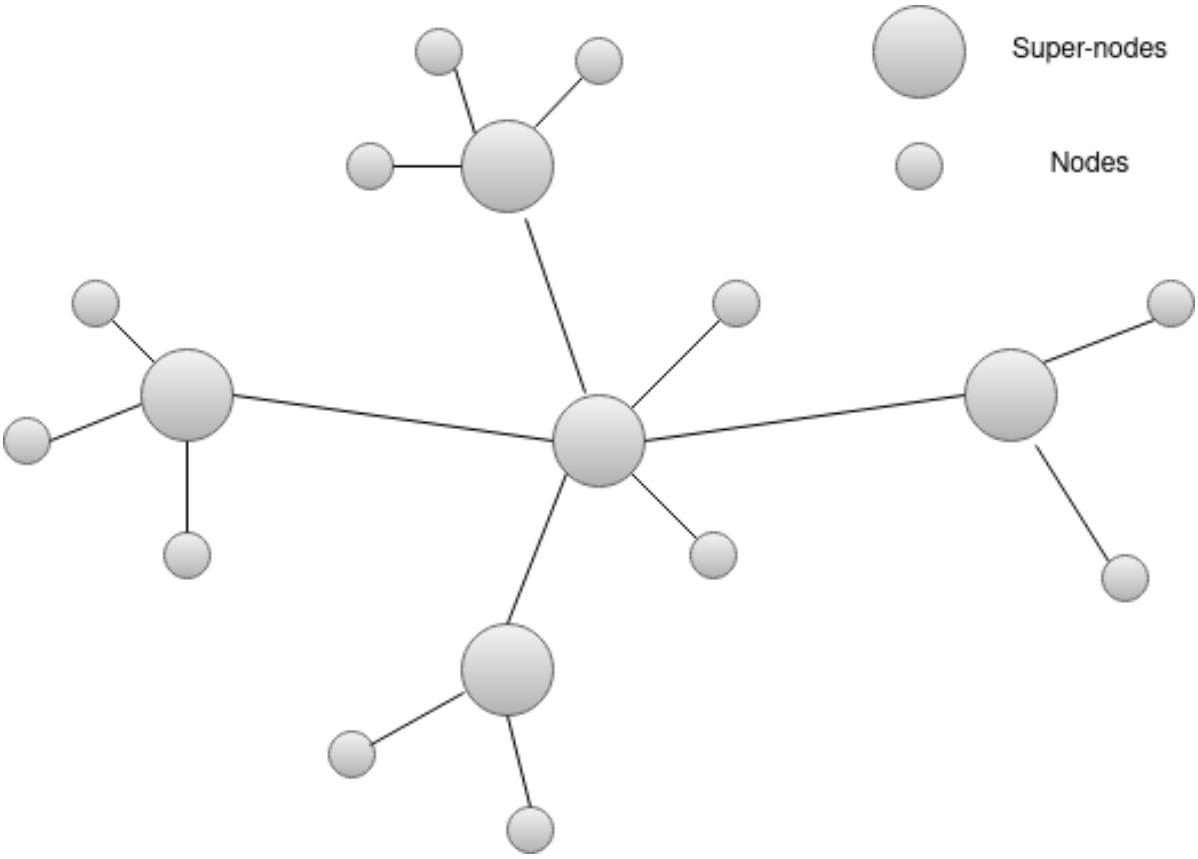

Fig. 3.1 System architecture

the connectivity of the network. The network should be able to change the topology according to this dynamic behaviour of nodes and maintain connectivity among existing super-nodes. A super-node is only responsible for nodes under his scope and does not know any information regarding other nodes of the system. Therefore a node discovery process becomes an exhaustive task. This can be accomplished in two ways.

- Random walk
- Flooding search[10]

We utilize flooding search in this project since the random walk is not guaranteed to produce results[10]. Nodes are expected to be behind a NAT environment. There are different types of NAT environments. Therefore different methods need to be utilized to maintain the connectivity of nodes behind NAT[39], [40]. Confidentiality of the communication needs to be protected to ensure the super-node cannot monitor the communication.

3.2.2 Methodological approach

Handle dynamic behaviour of nodes

A new super-node can join the network as long as it possesses connection details to an active super-node of the network. The first point of connection is to the known super-node. A super-node connected directly to another super-node is known as a neighbour-node. A super-node can connect up to n neighbour-nodes. If the existing number of the neighbour-nodes is less than n, the new super-node can connect to the known super-node directly. If the existing number of neighbour-nodes is equal to n, the topology of the network changes to establish the connection (explained in 4.1.2). After the connection is established, neighbour-nodes distribute the load among the new super-node.
A leaving super-node can partition the network. If there are n neighbour-nodes, one neighbour-node among them is selected and other (n-1) super-nodes connect to this super-node. These connections are handled as the arrival of new super-nodes and therefore the rules mentioned previously are accounted. The load of the leaving super-node get distributed across his neighbours before leaving the network.

Decentralized and pseudonymous node discovery

The nodes are identified by a 32bit hashed value of the node's public key. A connection request by a node in a form of a message is sent to a super-node to establish a connection along with a destination hashed key. Super-node checks the existence of the destination node under his supervision. If the node is not under his supervision a node discovery message is created and sent to the super-node network. Each super-node acts only according to the data that he has locally. If the destination node is not under his local scope super-node forwards the message to his neighbours. When the super-node is found, the details of the destination super-node is sent back to the super-node which originated the request message. Then the node disconnects from the current super-node and connects to the destination super-node. Ones both the origin node and destination node is under the same super-node. DTLS tunnel is established through the super-node to maintain the connectivity among two nodes.
The hashed key is used to identify a node uniquely. Changing the hashed key(changing public, private key pair) for each communication session, makes it difficult to link two communication sessions to a single node. Therefore provide nodes with pseudonymity[22].

NAT navigation of nodes

For nodes that are not behind a symmetric NAT environment, UDP hole punching[39] is a possible way of maintaining connectivity. Nodes that are behind a symmetric NAT environment a super-node need to act as a relay agent to relay traffic among the two communicating nodes[15].

Protect confidentiality of node to node communication.

The communication between two nodes needs to be protected from being monitored by a super-node. This can be accomplished by encrypting the communication with a PSK(Pre Shared Key) at both ends. The problem arises with the method of key exchange. We use Diffie-Hellman key exchange[41] to exchange the keys. However, in this implementation we assume super-nodes to protect the authenticity of the key exchange.

Chapter 4

Experimental Setup and Implementation

4.1 Implementation

4.1.1 Message Types

The system utilizes different types of messages to maintain the connectivity among super-nodes and to provide services to nodes. These messages are used throughout the project and knowing the functionality of each message is important to understand the protocols.

The types of messages and a description of each message are presented in Table 4.1 . Message types SEARCH and FOUND include payloads since super-nodes can execute the corresponding task solely depend on the information in the message and super-node's local knowledge of nodes. The formats of SEARCH and FOUND messages are explained in Fig. 4.1 and Fig. 4.2 .

SEARCH	destination node	source node

Fig. 4.1 Format of a FOUND message.

FOUND	destination super-node

Fig. 4.2 Format of a SEARCH message.

Message	Description
HELLO_S	A connection request from a super-node to another super-node. Sends when a super-node initiates
CONNECT_S	A super-node can directly connect to another super-node
CHANGE_S	A super-node cannot directly connect to another super-node. A change in topology is required
END_S	To end the connection between two super-nodes. Sends when a change of neighbour-nodes is required.
EXIT_S	A super-node which is ready to leave the system sends an EXIT message to all neighbor-nodes.
SEARCH	Search for a node. This is sent when a Node requests a tunnel establishment.
FOUND	The corresponding result of the SEARCH message.
INIT_P	A connection request from a node to a super-node.
FIND_P	A service request to find the location that holds a destination hashed key in a tunnel establishment scenario.
ANONYM_P	Inform a change of public key of a node. This can be utilized by nodes to obtain pseudonymity.
LOCATE_P	A request to find a registered node in the system.
CONNECT_P	Tunnel establishment request sends from a node.
LISTEN_P	A request sent from a node waiting for a tunnel establishment request.
ACCEPT/SUCCESS	Sends when successful completion of a task.
REJECT/FAIL	Sends when an unsuccessful completion of a task.

Table 4.1 Message types.

4.1.2 Connection establishment

Super-node connection establishment

For a node to initiate as a super-node a HELLO_S message is sent to an active super-node of the network. After authenticating the identity of the device, super-nodes checks the number of existing neighbours(n). If n does not exceed the allowed number of neighbours(N), super-node sends a CONNECT_S message. Upon the CONNECT_S message, the new super-node can establish a direct connection to the super-node. After successfully establishing a new connection a SUCCESS message is sent to the new super-node This is shown in Fig. 4.3 .

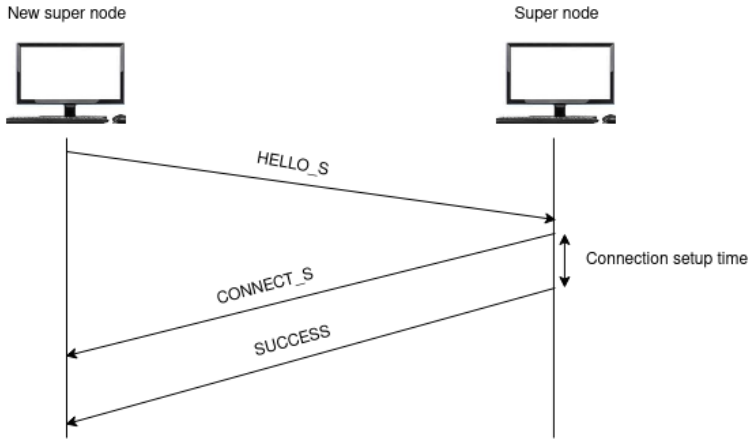

Fig. 4.3 Connection establishment when n < N

If n is equal to N, a CHANGE_S message is sent. Then the super-node disconnects one of his existing neighbour-nodes by sending and END message. When the neighbour-node reply with the ACCEPT message, the connection can be safely terminated. At this point, super-node only have (N-1) neighbour-nodes. Therefore the new super-node can connect directly to the super-node. The communication process completes after receiving a SUCCESS message from the super-node.
After the connection establishment, the previous neighbour remains disconnected from the network. Therefore the new super-node send a HELLO_S message to the disconnected neighbour. Since the number of neighbour-nodes the disconnected neighbour-node has is definitely less than N, the new super-node can establish a direct connection by following the same procedure mentioned in Fig 4.3. The communication process is explained in Fig 4.4 using a scenario where a super-node and two neighbour-nodes exists.

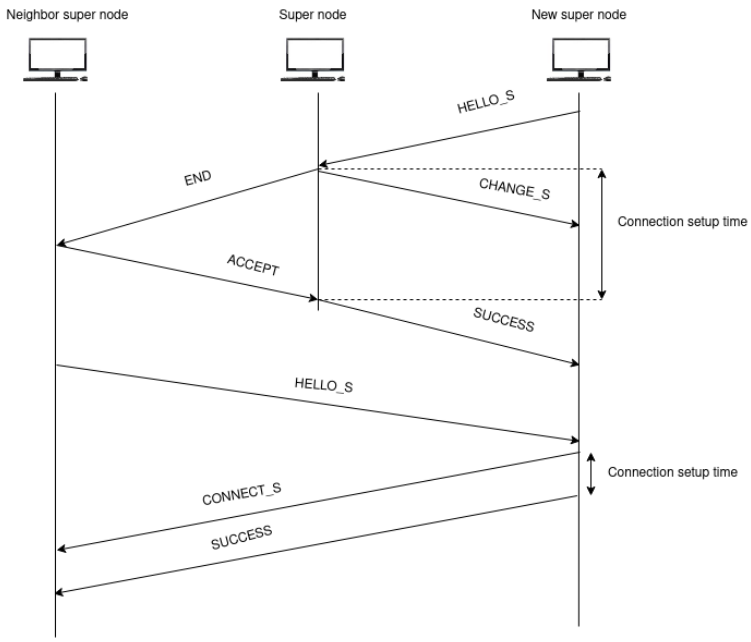

Fig. 4.4 Connection establishment when n = N

Node connection establishment

A node can connect to the network by sending a INIT_P message to one of the active super-nodes. We use a hashed public key to uniquely identify each node. There is a possibility that a node may not be able to connect to a super-node. This is due to the dynamic behaviour of super-nodes and the network.

- Super-node might have left the network.
- New super-nodes might have joined the network.

In both cases the location of the node information changes. The node need to contact an active super-node and sends a LOCATE_P message to find the new location of the client information. The process of handling LOCATE_P is similar to FIND_P message and it is explained in the next section.
A node is identified by a hashed public key. The hashed public key can be utilized to obtain pseudonymity. A user can change the public key using the ANONYM_P message before a communication process. By changing the identity at each communication session, the evidence to link two communication instances to a user vanishes.

4.1.3 Node discovery

A node can send a FIND_P message to request a node discovery from a super-node. Once the information of the destination node is received, super-node checks the availability of the destination node in his local scope. If the destination is found, a FOUND message is sent back to the node. If the destination is not found under the local scope, super-node creates a SEARCH message and assign a HTL(Hops To Live) value to the packet. At each super-node HTL value get reduced by one. Once HTL value get to zero, a super-node drops the packet. Then the SEARCH message is flooded to every neighbour-node. Once a SEARCH message is received by a super-node, the availability of the destination node is checked in his local scope. If the node is not found, the super-node floods the massage to the neighbour-nodes same as before. At every instance of this process, a super-node creates a tunnel which can be utilized to find the path of the SEARCH message. The tunnel holds data about the super-node which the SEARCH message is received from and the super-node which the SEARCH message is forwarded. Once the destination node is found, a FOUND message is initiated by the destination super-node. This is sent back to the origin super-node, along the super-node network utilizing the tunnels created during the SEARCH message. The process of node discovery is explained in Fig 4.5 and Fig 4.6. If the destination node is not found after a specified time, origin super-node creates another SEARCH message and set HTL value to as twice as the previous value and floods again. If the destination node is not found after a specified number of flooding phases, the system assumes that the destination node does not exist in the system. Therefore respond with a FAIL message to origin node.

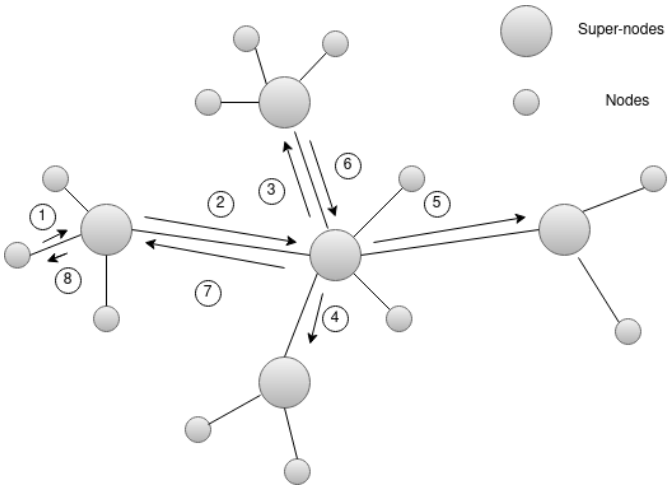

Fig. 4.5 Instance of a node discovery

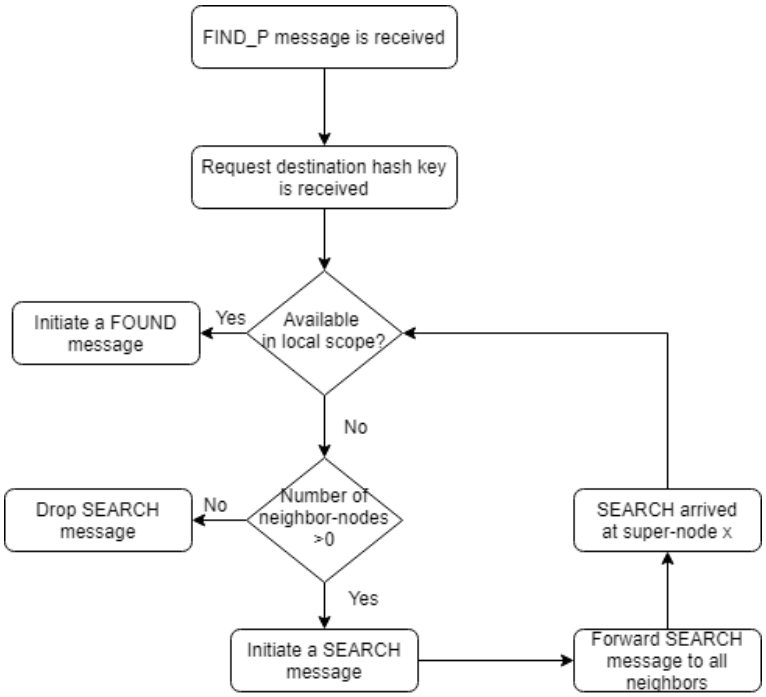

Fig. 4.6 Process of node discovery

4.1.4 Tunnel establishment

To request a tunnel establishment, a CONNECT_P followed by a FIND_P message is sent to the super-node. If the destination super-node is found, the node can expect a FOUND message along with the destination super-node. Then the node disconnects from the current super-node and connects to the destination super-node. Once both the origin node and the destination node is under the same super-node, a DTLS tunnel is established between the super-nodes utilizing the super-node as a relay agent[42].
The tunnel is established using a DTLS library written as a part of the project. The tunnel is encrypted at each end of the node and ends of the super-node. This creates a possibility for the super-nodes to monitor the communication. Therefore the communication is encrypted using a 32bit key at each end of the two nodes. For encryption we used AES. The key used to encrypt the communication is exchanged using a Diffie-Helman key exchange, through the super-node. A malicious super-node can use this to establish a masquerading attack providing a fake key exchange. However, in this part of the research, we assume super-nodes protect the authenticity of the key exchange. Following is the script for relay communication between nodes using the DTLS library.

```
DTLSServer dtls_server0 = new DTLSServer(local_port.ToString(), new byte[] { 0xBA, 0xA0 });
DTLSServer dtls_server1 = new DTLSServer(port.ToString(), new byte[] { 0xBA, 0xA0 });


    if (RuntimeInformation.IsOSPlatform(OSPlatform.Windows))
    {
        dtls_server0.Unbuffer = "winpty.exe";
        dtls_server0.Unbuffer_Args = "-Xplain -Xallow-non-tty";
        dtls_server1.Unbuffer = "winpty.exe";
        dtls_server1.Unbuffer_Args = "-Xplain -Xallow-non-tty";
    }
    else
    {
        dtls_server0.Unbuffer = "stdbuf";
        dtls_server0.Unbuffer_Args = "-i0 -o0";
        dtls_server1.Unbuffer = "stdbuf";
        dtls_server1.Unbuffer_Args = "-i0 -o0";
    }


dtls_server0.Start();
dtls_server1.Start();

new Thread(() => dtls_server0.GetStream().CopyTo(dtls_server1.GetStream(), 16)).Start();
new Thread(() => dtls_server1.GetStream().CopyTo(dtls_server0.GetStream(), 16)).Start();

dtls_server0.WaitForExit();
dtls_server1.WaitForExit();
```

Fig. 4.7 The script for relaying communication between nodes

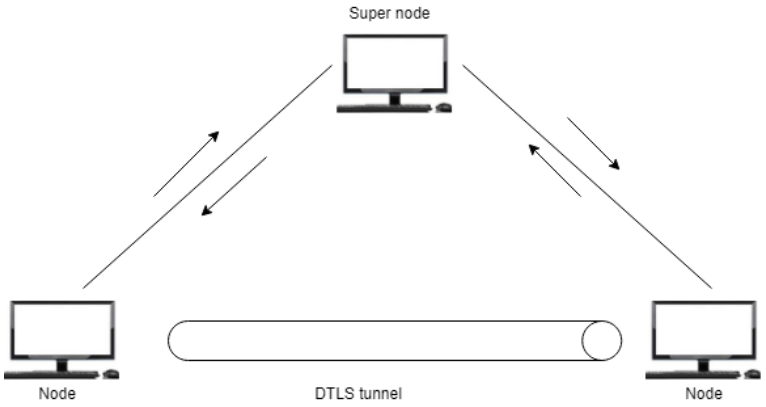

Fig. 4.8 Super-node acts as a relay agent to establish the DTLS tunnel

4.1.5 Connection termination

Super-node connection termination

Prior to leaving the network, a super-node needs to terminate all his existing neighbour-node connections. Once a termination command is received super-node distribute all his node details among the neighbour-nodes. Then to terminate the connections, super-node sends EXIT messages to each of his neighbour-nodes. Upon receiving the EXIT message

neighbour-nodes acknowledge the termination by sending an ACCEPT message. After sending the ACCEPT message neighbour-nodes disconnect the connection with the super-node. Once all ACCEPT messages are received, super-node can safely terminate the connections.
After the Super-node terminates, neighbours need to maintain the connectivity to the network. This is done by sending a HELLO_S message to one of the neighbour-nodes. At this point every neighbour who is requesting a connection act as a new super-node to establish a connection to the selected neighbour-node. The communication process is explained in Fig 4.9 using a scenario where a super-node and two neighbour-nodes exists.

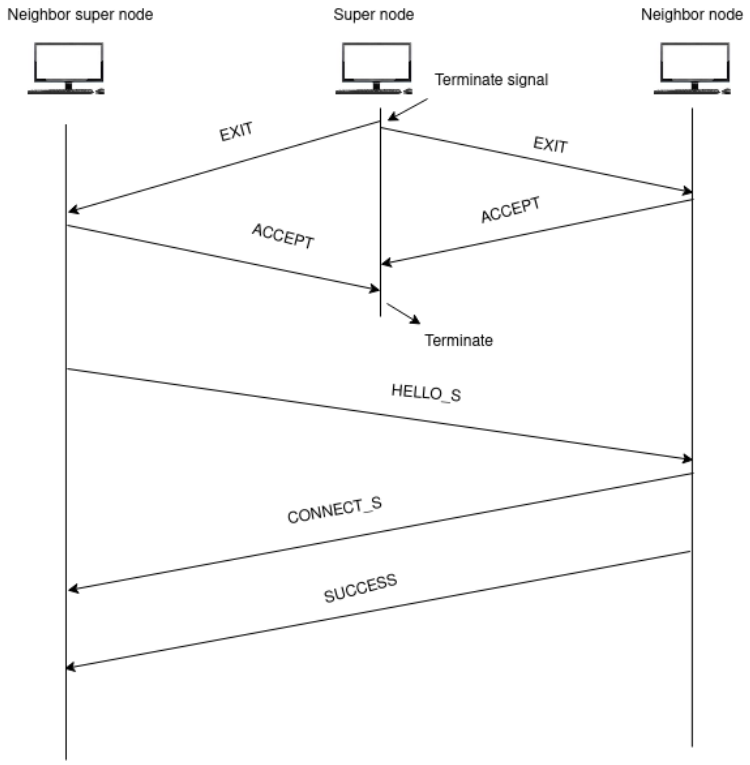

Fig. 4.9 Communication process in super-node termination

4.2 Testing

Testing of the system was done to understand the capabilities of the system. The testing was done using publicly accessible cloud servers. Although a simulation environment would be ideal to do load testing on the system, absence of open-source platforms to simulate network environments which could run c sharp scripts was a problem.

However, a real-world environment helps to understand the system performance in its operating environment. At this phase, our main objective was to build the decentralized communication platform. Therefore tests were done to find the limitations of the system. The first test was done to understand the performance of the node discovery process.

4.2.1 Performance of node discovery

Decentralized systems are designed to improve the scalability. However, decentralized architecture increase the complexity of the node discovery process. This experiment was done to identify how the latency of a successful node discovery varies with increasing network size. The experiment was designed by connecting super-nodes linearly and with each step increasing the number of super-nodes the message had to traverse. Prior to the experiment, a threshold value of five seconds is set for the node discovery. Therefore experiment was carried out until the discovery process exceeds the five second threshold value.
A node connected to one end of the network sends a FIND_P message initiating a node discovery process. The time was calculated from the moment a SEARCH message was initiated, to the time of arrival of the FOUND message at the origin super-node. The HTL values was set to 50, so only one flooding phase exist until 50 super-nodes are connected linearly. The setup was started by having only two super-nodes and at each step we connected an extra super-node. At each step a node discovery process was initiated to a node exist in the other end of the network, therefore the SEARCH message had to traverse every super-node in a linear fashion. The latency of the node discovery processes was calculated at each step to get the average latency.
Due to limited resources, we used only four publicly available servers each in a different country. The selected servers were located in Singapore, India, America and France. Multiple super-node instances were created at each server and super-nodes were connected so that no two neighbour-nodes reside in the same country. This is to intentionally increase the latency of communication.

4.2.2 Performance of a super-node against increasing number of neighbour-nodes

Scalability and topology of the network depend directly on the number of neighbour-nodes a super-nodes is able to maintain connections with. Although theoretically, a server can have 65353 TCP connections, in reality, this number depends on the performance of the server. Not to forget that serving nodes and super-nodes in the network is not

the sole task of the server. Also, each connection is handled using a user-level thread. The Memory and CPU usage need to kept at an acceptable level without slowing down the performance. This test focus on figuring out how the performance of a super-node degrades due to the increasing number of neighbour-node connection. The experiment will help to identify the most suitable threshold value for the maximum number of neighbour-nodes a super-node can connect simultaneously.
The experiment was carried out using the same four servers used in the previous test. Rather than connecting linearly in this test, we connected all super-nodes to a single super-node. Starting by one neighbour-node, the number of neighbour-node was increased by five at each step of the test and measured the increment of CPU Memory usage. However, this depends on the performance of the server. For this test, we used a server with 1GB RAM and a single core. Again multiple super-node instances were created and the test was carried out until the load significantly increased or we ran out of resources (only a limited number of super-node instances can be run on a server without significantly slowing down).

Chapter 5

Results and Analysis

5.1 Results

5.1.1 Performance of node discovery

The test was carried out by increasing the number of linearly connected super-nodes and measuring the latency of the node discovery.

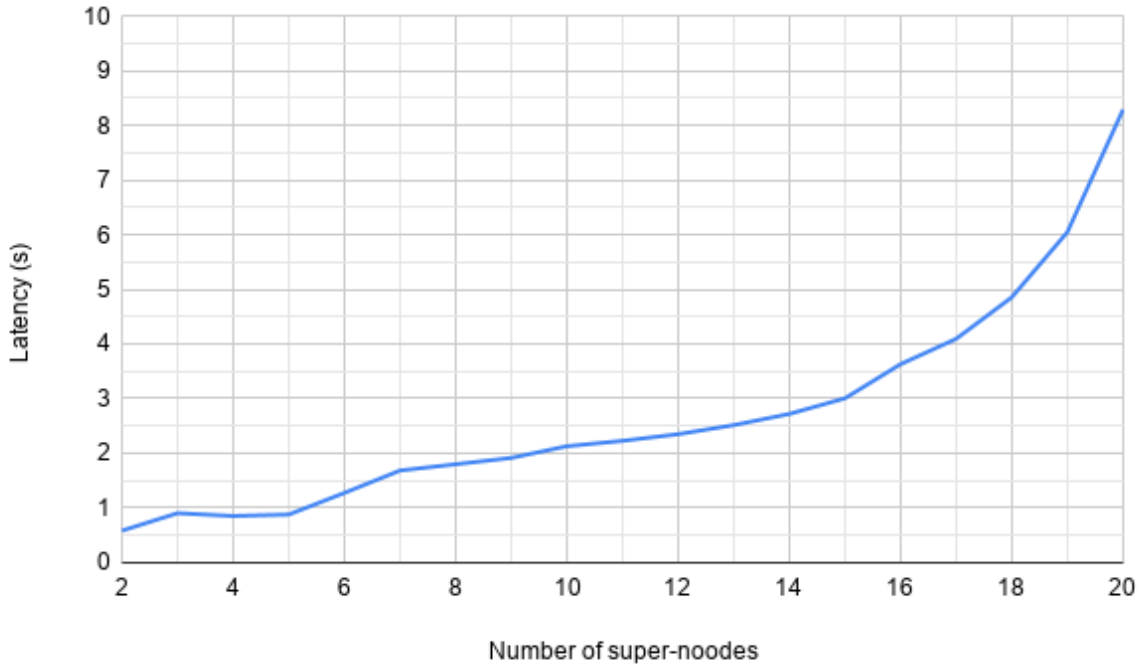

Fig. 5.1 Performance of node discover process

5.1.2 Performance of a super-node against increasing number of neighbour-nodes

The test was carried out by increasing the number of neighbour-nodes in a single super-node and measuring the CPU memory usage.

Number of neighbour-nodes	CPU memory usage (%)
1	24.3095
6	24.5754
11	25.2827
16	25.6848
21	26.1219
26	27.2285
31	27.6023
36	28.2835
41	30.4112
46	32.1423
51	35.7683
55	37.2712
61	39.4051

Table 5.1 Performance of a super-node with increasing number of neighbour-nodes.

5.2 Analysis

Analysing Fig. 5.1 we can see that increasing the linear path of the SEARCH message exponentially increase the latency of the result. The maximum number of possible neighbour-nodes without exceeding the threshold value of five seconds is 18 neighbour-nodes. This concurs that the maximum linear path of the network should be kept less than 18 super-nodes to achieve a node discovery results under five seconds. Therefore the maximum number of super-nodes in the network with the least number of neighbour-nodes per super-node becomes

$$18 + 2\sum_{n=1}^{8} n = 90$$

However, the network congestion at the time of the experiment, the physical location of super-nodes, the performance of the super-nodes can have an effect on the results.

Nonetheless, the experiment is still essential to get an overall idea of the performance of the system.
The number of neighbour-node connections increases the number of possible super-nodes in the network. Therefore understanding the number of possible neighbour-node connections per super-nodes is important to understand the full scalability of the network.

This was the objective of the next test. As it can be seen in Table 5.1 the connection of 60 neighbour-nodes only increased the CPU memory usage only by a 15.0956%. Therefore it can be safely assumed that a super-node can handle more than 60 neighbour-node connections without significantly slowing down the performance. Although continuing the test with more neighbour-node connections was ideal, limited resources forced to stop the test after 61 neighbour-node connections. A server could only create about 20 super-node instances without slowing down. Combining the two test results we can deduce that theoretically, the maximum number of super-nodes the network can handle with a node discovery latency of five seconds is

$$18 + 116 \sum_{n=0}^{7} 59^n$$

However, in practice this number can be significantly lower. Since SEARCH messages are broadcasted across the network, with a large network, the traffic generated by constant SEARCH messages will be higher. Therefore network may get congested and not work properly[37].

Chapter 6

Conclusions and Future Works

The aim of the research was to implement a censorship and surveillance resistant, private communication platform. A network based on common centralized architectures brings surveillance and censorship into communication. The designed super-node network distributes the control of information over multiple nodes. With no single node has all the information about the system and no single node control the behaviour of the system, the network is protected from surveillance and censorship issues. More super-nodes and nodes the network has, more secure the network becomes.

Purely distributed systems suffer from low performance in node discovery. The super-node system improve the node discovery performance while keeping the information distributed to some extent.

The system allows super-nodes to leave and join the network at any time. The project has solved the issue by dynamically changing the topology of the network to keep the connectivity between existing super-nodes. However active communication processes can get interrupted due to this. This could be a problem with a large network where the topology of the network change constantly

. The system allows changing the identity of a user at each communication session. This allows a node to obtain pseudonymity. However, the identity of the destination node needs to remain the same or if changed need to communicate using some other kind of a secured mechanism (email, etc..).

The architecture of the system improves scalability. According to the results drawn from the experiments, a server with 1GB RAM and a single core can handle up to 60 active super-nodes without increasing the CPU usage by 15.0956%. A node discovery can be done under five seconds, along 18 linearly connected super-nodes.

The system assumes all super-nodes are trustworthy. Therefore the authenticity of key exchange is safe. which is not the case in a real environment. Using a bogus key exchange

super-nodes can issue a masquerading attack; impersonate as a node and continue communication. Although a malicious super-node can be removed using revoking the certificate, the identification of a malicious super-node remains a problem.
The research has utilized certificate authorities for authentication. This makes every node and super-node contact a CA prior to the connection setup. A CA can become a single point of failure and a compromised CA (highly unlikely but possible) can monitor the activity of the nodes. Although this cannot be used to monitor the communication, can be utilized to censor the activities of the nodes and super-nodes. Therefore at the next phase of the research, we explore distributed authentication mechanisms[43], [8] to complement the current design.

References

[1] T. Stahl, "Indiscriminate mass surveillance and the public sphere," *Ethics and Information Technology*, vol. 18, no. 1, pp. 33–39, 2016.

[2] J. Semitsu, "From Facebook to Mug Shot: How the Dearth of Social Networking Privacy Rights Revolutionized Online Government Surveillance," *Pace Law Review*, vol. 31, no. 1, pp. 291–381, 2011.

[3] B. C. Newell, "The Massive Metadata Machine: Liberty, Power, and Secret Mass Surveillance in the U.S. and Europe," *Journal of Law and Policy for the Information Society*, vol. 10, no. 2, pp. 481–522, 2013.

[4] E. Pacitti, R. Akbarinia, and M. El-Dick, "P2P Techniques for Decentralized Applications," *Synthesis Lectures on Data Management*, vol. 4, no. 3, pp. 1–104, 2012.

[5] M. Dubreuil, C. Gagné, and M. Parizeau, "Evolutionary Computations," vol. 36, no. 1, pp. 229–235, 2006.

[6] P. Backx, T. Wauters, B. Dhoedt, and P. Demeester, "A comparison of peer-to-peer architectures A comparison of peer-to-peer architectures," no. January, 2002.

[7] J. Åslund, "Authentication in peer-to-peer systems," p. 120, 2002.

[8] A. Takeda, G. Kitagata, T. Kinoshita, K. Hashimoto, S. M. S. Zabir, and N. Shiratori, "A new authentication method with distributed hash table for P2P network," *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, pp. 483–488, 2008.

[9] A. Wierzbicki, A. Zwierko, and Z. Kotulski, "Authentication with controlled anonymity in P2P systems," *Parallel and Distributed Computing, Applications and Technologies, PDCAT Proceedings*, vol. 2005, pp. 871–875, 2005.

[10] R. Ahmed and R. Boutaba, "A survey of distributed search techniques in large scale distributed systems," *IEEE Communications Surveys and Tutorials*, vol. 13, no. 2, pp. 150–167, 2011.

[11] B. Yang and H. Garcia-molina, "Designing a Super-Peer Network," pp. 1–12.

[12] S. Le Blond, C. Zhang, A. Legout, K. Ross, and W. Dabbous, "I know where you are and what you are sharing: Exploiting P2P communications to invade users' privacy," *Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC*, no. November 2014, pp. 45–59, 2011.

[13] Y. Zhu, Y. Lu, A. Vikram, and H. Fu, "On privacy of skype VoIP calls," *GLOBECOM - IEEE Global Telecommunications Conference*, 2009.

[14] C. Wacek, H. Tan, K. Bauer, and M. Sherr, "An Empirical Evaluation of Relay Selection in Tor," *Ndss2013*, 2013.

[15] Z. Hu, "NAT Traversal Techniques and Peer-to-Peer Applications," *Telecommunications Software and Multimedia Laboratory, Helsinki University of Technology*, 2005.

[16] A. A. M. Mahmudul Haque, "UPnP Networking: Architecture and Security Issues," *Security of the End Hosts on the Internet: Seminar on Network Security*, 2007.

[17] L. Reinecke and J. G.-u. Mainz, *Privacy Online*. No. April 2014, 2011.

[18] A. Overview, S. Communications, and P. Simon, *SSL and TLS*. No. April, 2005.

[19] G. Urdaneta, G. Pierre, and M. Van Steen, "A survey of DHT security techniques," *ACM Computing Surveys*, vol. 43, no. 2, 2011.

[20] R. Q. Dqg, D. Ahmat, D. Magoni, and T. F. Bissyandé, "EAI Endorsed Transactions End-to-End Key Exchange through Disjoint Paths in P2P Networks," vol. 2, no. 3, pp. 1–15, 2015.

[21] E. Saboori and S. Mohammadi, "Anonymous Communication in Peer-to-Peer Networks for Providing more Privacy and Security," *International Journal of Modeling and Optimization*, no. August 2012, pp. 217–221, 2012.

[22] R. Jain, "Tools and Protocols for Anonymity on the Internet," pp. 1–9.

[23] M. Reiter and A. Rubin, "Crowds: anonymity for Web transactions," *ACM Transactions on Information and System Security*, vol. 1, no. 1, pp. 66–92, 1998.

[24] B. N. Levine and C. Shields, "Hordes: A multicast based protocol for anonymity," *Journal of Computer Security*, vol. 10, no. 3, pp. 213–240, 2002.

[25] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A distributed anonymous information storage and retrieval system," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2009, pp. 46–66, 2001.

[26] M. J. Freedman and R. Morris, "Tarzan," p. 193, 2002.

[27] D. McCoy, K. Bauer, D. Grunwald, T. Kohno, and D. Sicker, "Shining light in dark places: Understanding the tor network," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5134 LNCS, pp. 63–76, 2008.

[28] A. K. Jadoon, W. Iqbal, M. F. Amjad, H. Afzal, and Y. A. Bangash, "Forensic Analysis of Tor Browser: A Case Study for Privacy and Anonymity on the Web," *Forensic Science International*, vol. 299, no. March, pp. 59–73, 2019.

[29] E. Jardine, "The Dark Web Dilemma: Tor, Anonymity and Online Policing," *SSRN Electronic Journal*, no. 21, 2018.

[30] M. Möser, "Anonymity of Bitcoin transactions. An analysis of mixing services," *Münster Bitcoin Conference, 17-18 July 2013*, pp. 17–18, 2013.

[31] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun, "Evaluating user privacy in Bitcoin," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7859 LNCS, pp. 34–51, 2013.

[32] S. Zhang and J. H. Lee, "Double-Spending with a Sybil Attack in the Bitcoin Decentralized Network," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 10, pp. 5715–5722, 2019.

[33] I. C. Lin and T. C. Liao, "A survey of blockchain security issues and challenges," *International Journal of Network Security*, vol. 19, no. 5, pp. 653–659, 2017.

[34] K. Hamzeh, G. Pall, M. Verthein, W. Taarud, W. Little, and G. Zorn, "Point-to-Point Tunneling Protocol (PPTP)," pp. 1–57, 1999.

[35] M. A. Mohamed, M. E. A. Abou-El-Seoud, and A. M. El-Feki, "A Survey of VPN Security Issues," *International Journal of Computer Science Issues*, vol. 11, no. 4, pp. 106–111, 2014.

[36] A. Mueed, A. Alshehri, H. L. Aljuhani, and A. S. Bajenaid, "SECURITY ISSUES OF VIRTUAL PRIVATE NETWORKS : A SURVEY," vol. 16, no. 2, pp. 63–67, 2018.

[37] B. Beverly Yang and H. Garcia-Molina, "Design a super-peer networks.pdf," *Data Engineering, 2003. Proceedings. 19th International Conference on*, pp. 49–60, 2003.

[38] S. Guha and N. Daswani, "An experimental study of the skype peer-to-peer voip system," *Europe*, vol. 6, no. December 2005, pp. 5–10, 2005.

[39] A. Müller, A. Klenk, and G. Carle, "On the applicability of knowledge based NAT-traversal for home networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 4982 LNCS, pp. 264–275, 2008.

[40] , "No Title ," , vol. 20, no. 4, pp. 99–102, 2002.

[41] M. Mishra and J. Kar, "a Study on Diffie-Hellman Key Exchange Protocols," *International Journal of Pure and Apllied Mathematics*, vol. 114, no. 2, pp. 179–189, 2017.

[42] E. J. MA and Bachelor, "No Title," pp. 1–10, 2010.

[43] S. Gokhale and P. Dasgupta, "Distributed authentication for peer-to-peer networks," *Proceedings - 2003 Symposium on Applications and the Internet Workshops, SAINT 2003*, pp. 347–353, 2003.