Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

André Gaarder

Routing for a dark net overlay network

Master's thesis in Information Security
Supervisor: Lasse Øverlier
June 2023

NTNU
Norwegian University of
Science and Technology

André Gaarder

Routing for a dark net overlay network

Master's thesis in Information Security
Supervisor: Lasse Øverlier
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

Routing for a dark net overlay network

André Gaarder

2023/06/01

Abstract

Decentralized peer-to-peer networks present challenges in ensuring routing efficiency and network stability. Ensuring efficiency and stability in overlay networks when using Tor for anonymity poses additional challenges. This master thesis presents a routing and connection control protocol for an overlay network built on the Tor anonymity network. The goal of the protocol is to ensure efficiency, stability, and resilience with the Tor network's constraints.

The thesis builds on earlier work that utilizes Tor rendezvous points for connecting peers in an overlay network. It expands on this work to test and evaluate a proposed routing and connection control protocol design. A prototype is built to evaluate the protocol's performance and effectiveness. The prototype is then tested with Tor network relays to simulate a peer-to-peer network. The prototype uses measurements of latency between peers to make routing decisions. A protocol for dynamically connecting peers over Tor rendezvous points is tested to ensure network connectivity. A security feature is having the peers re-establish connections through new routes to deter traffic analysis.

This routing method showed a high degree of success in selecting the most efficient path in the network. The protocol also manages to recover peers from random link failures, providing network resilience against failures or churn. Challenges and possible improvements to the design to deal with scalability and security are discussed. The findings contribute to the field of overlay network design by testing a protocol that ensures the efficiency, stability, and reliable operation of overlay networks in the context of Tor.

Sammendrag

Desentraliserte "peer-to-peer" - nettverk har utfordringer med å rute trafikk effektivt og nettverksstabilitet. Å sikre effektivitet og stabilitet i nettverket når man bruker Tor for anonymitet, utgjør ytterligere utfordringer. Denne masteroppgaven presenterer en ruting- og tilkoblingskontrollprotokoll for et overleggsnettverk bygget på Tor-anonymitetsnettverket. Målet med protokollen er å sikre effektivitet, stabilitet med Tor-nettverkets begrensninger.

Oppgaven bygger på tidligere arbeid som bruker Tor treffpunkt for å koble sammen noder i et overleggsnettverk. Den utvider dette arbeidet for å teste og evaluere en foreslått ruting- og tilkoblingskontrollprotokoll. En prototype er bygget for å evaluere protokollens ytelse og effektivitet. Prototypen testes deretter med Tor-nettverket for å simulere et "peer-to-peer"-nettverk. Prototypen bruker målinger av latens mellom noder for å ta rutingbeslutninger. En protokoll for dynamisk tilkobling mellom noder er testet for å sikre nettverksstabilitet. En sikkerhetsfunksjon er å få noder til å gjenopprette forbindelser gjennom nye ruter for å vanskeliggjøre trafikkanalyse.

Testresultatene viser at protokollen klarer i snitt å velge den mest effektive ruten mellom noder. Protokollen klarer også å gjenopprette noder fra tilfeldig utfall av linker, og gir nettverksresiliens mot feil, eller såkalt *churn*. Utfordringer og mulige forbedringer av designet for å håndtere skalerbarhet og sikkerhet diskuteres. Funnene bidrar til feltet overleggsnettverksdesign ved å teste en protokoll som sikrer effektivitet, stabilitet og pålitelig drift av overleggsnettverk i sammenheng med Tor.

Acknowledgement

This thesis concludes my study programme for an Experience-based master´s degree in Information security. Getting this far has not been easy with full-time work and family, but it has been rewarding.

I would like to thank my supervisor Lasse Øverlier for his guidance and help during this thesis.

A thank you to my employer Tolletaten for giving me the opportunity to do this, both with time off to study and financing.

A special thanks to my family who have supported and encouraged me.

Contents

Figures

Tables

Code Listings

Acronyms

DHT Distributed Hash Table. 7, 12

DSDV Destination Sequenced Distance Vector Routing. 20

MANET Mobile Ad hoc Network. 14

OSPF Open Shortest Path First. 20

TTL Time to Live. 28

Glossary

churn The dynamics of peers joining and leaving the network. 1, 2, 13, 16, 18, 41, 54

clustering coefficient A measure of the degree to which nodes in a graph tend to cluster together. 10, 22, 43

convergence The number of peers that share the same topological information. 22

Dijkstra´s algorithm An algorithm for finding the shortest paths between nodes in a weighted graph. 10, 20

node degree The number of edges connected to a node. 10, 22, 41, 43, 47, 48, 51, 54, 56, 57

onion routing A method of encapsulating messages in layers of encryption. 8

Tor Software for anonymous communication using onion routing. 8

Chapter 1

Introduction

Peer-to-peer (P2P) networks use overlay networks to facilitate decentralized user communication and share resources. Anonymous, or dark net, overlay networks hide the users' identities in the network. These can be combined to form an anonymous peer-to-peer overlay network. A widespread tool for anonymizing communication is the Tor project [1]. This uses onion routing between thousands of relays spread throughout the world. It is one of the most widely used and extensive anonymizing networks. However, it is not designed with peer-to-peer functionality in mind. Fallang [2], in his paper, described a method for using the Tor Rendezvous specification to create a peer-to-peer network using Tor as a substrate. It uses Tor rendezvous point relays (RP) as meeting points for P2P clients to develop a semi-anonymous P2P network.

Decentralized networks do not have a central authority to coordinate participants, and the network needs to be self-organizing to ensure efficiency and stability. This thesis aims to develop further the overlay network that Fallang developed to design, implement and test a routing and connection protocol. This protocol aims to provide a self-organizing capability to this network that ensures efficiency and stability.

Routing in P2P networks can span from simple flooding to more advanced proximity-aware routing. Anonymous networks, like Tor, add extra overhead to communication that regular P2P networks do not have. Latency can be an issue in such networks because of the different layers of encryption, and relay jumps. Therefore, a routing protocol that balances connectivity and efficient routing is essential. Also, in decentralized networks, no central resources can be used to locate nodes in the network. Once a peer is connected to the network, it is crucial that it can stay connected despite the typical churn where peers leave and join the network. It must also handle node and link failures, both deliberate and accidental. The thesis investigates how the network can be made resilient using routing.

A prototype implementing the design protocol is developed and tested on the Tor network. The test results indicate the feasibility of the protocol and its ability to scale to more extensive networks.

1.1 Keywords

Overlay networks, P2P, Tor, anonymity, routing, efficiency, stability, resilience, scalability.

1.2 Problem description

Peer-to-peer networks can provide robust and resilient communication between users. Like in the Tor network, onion routing uses intermediate relays to hide users' identities. Therefore, the overhead of communicating and establishing connections is more significant than regular peer-to-peer networks. In addition, decentralized peer-to-peer networks will have the added challenge of locating users. There are P2P networks that solve this through mechanisms like Distributed Hash Tables (DHT), but this is mainly a resource-locating mechanism and does not necessarily guarantee efficient routing between peers. However, improvements have been made to alleviate this [3]. The thesis delves into the challenge of designing a routing protocol for decentralized peer-to-peer overlay networks utilizing Tor relays and rendezvous points. The main concern is how data can be efficiently routed through the network and how resilience against failures and churn can be ensured.

1.3 Research questions

More specifically, the research questions it tries to answer are:

- RQ1: How can data be routed efficiently in a peer-to-peer overlay network using Tor?
- RQ2: What metric is best suited for this?
- RQ3: How can stability and resilience be ensured in such a network?

1.4 Scope and contributions

This thesis contributes by designing a routing and connection protocol for a peer-to-peer network using Tor relays as rendezvous points. A proof-of-concept prototype is developed, implementing the designed protocol. The prototype is tested and evaluated against the research questions. This thesis limits itself to looking at routing and connection. Therefore, some assumptions and limitations are applied.

This thesis does not assume a particular purpose for this application other than connecting peers. As discussed in section 2.1, peer-to-peer networks serve different purposes. For resource sharing, like file sharing, one would need to associate a file resource with a lookup key and a search protocol to find those resources. This application only looks into routing between peers and structuring the network.

How nodes join and are bootstrapped into the network is not dealt with. In a completely decentralized P2P network, there needs to exist a method for finding other nodes and joining the network. Fallang [2] provided two ways for how two nodes would agree on the rendezvous point and cookie value for splicing the circuit. This thesis assumes that the participants have joined the network by some previously agreed-upon mechanism and that their initial connection point is randomly selected.

The practical problem of implementing the data channel based on the routing mechanism will be left to other development. The *Rtun* application this thesis builds upon uses a SOCKS5-proxy by OpenVPN to transmit data between peers. This provides a simple yet flexible way to carry different types of traffic through the channel. It also delegates the routing to the IP layer, which, in this thesis, is handled in the overlay layer. This could be adapted to use the circuit routing in the application. Further development can implement an end-to-end data transfer mechanism using the router using a layered approach where the data endpoints "plug in" to the router.

Encryption or key management for this protocol is not considered in this thesis. The protocol can be expanded in various ways, including client authentication for trust and encryption for confidentiality and integrity, and this can be researched for future implementations.

1.5 Outline

Chapter 2 gives a background into peer-to-peer overlay networks and Tor, describing the work this thesis builds upon. Resilience and how this relates to peer-to-peer networks is looked into. An overview of some other comparable overlay networks and how routing is handled there is given. Chapter 3 presents the methodology used to develop and test the routing protocol. Chapter 4 details the implementation of the protocol in a prototype application. The experiments, along with the results, are presented. Chapter 5 discusses the results achieved concerning the research questions. Chapter 6 contains the conclusion of the research. Finally, chapter 7 contains recommendations for future work and research.

Chapter 2

Background

This chapter will give some background to the topics that will be researched by looking at some definitions and describing peer-to-peer (P2P) networks. It will describe some of the properties of such networks and relevant features and examples.

2.1 Overlay networks and their characteristics

Overlay networks are virtual network topologies built upon existing networks to provide different services [4]. By creating an extra layer of indirection, the network can route traffic between nodes based on criteria independent of the underlying network technology. Peer-to-peer overlay networks are computer networks built on top of an existing network, like the Internet, where peers are addressed with logical IDs instead of IP addresses or other underlying network addresses [5]. The concept of a *peer* in a network can be described as analogous to its definition in language, which is "*one that is of equal standing with another*" [6].

Peer-to-peer networks differ from the client-server model in that no single point of failure exists. If a node goes down, there should be other nodes in the network that can take over the workload. A client-server model can, in large parts, compensate for this by duplicating content and load-balancing it across many different locations and servers. However, if we look at a definition from Oram [7], it states that "P2P is a class of applications that take advantage of resources storage, cycles, content, human presence available at the edges of the Internet. Because accessing these decentralized resources means operating in an environment of unstable connectivity and unpredictable IP addresses, peer-to-peer nodes must operate outside the DNS and have significant or total autonomy of central servers" [7]. Peers in a P2P network must be even more decoupled from centralized resources, technological or organizational.

A further distinction with the client-server model can be found in Schollmeier's [8] definition of P2P networks. In a client/server model, a client sends a request, and a server responds. In a peer-to-peer network, there is no such separation. Each node in the network can act as a server and a client. Schollmeier calls this

a *Servent*, i.e., a combination of *ser*-ver and cli-*ent*. P2P networks are inherently more scalable as new *servents* can be added to the network and contribute to load sharing.

P2P overlay networks have several different applications and uses, which can be grouped into these categories [5]:

- Data-sharing (data storage and retrieval)
- Bandwidth-sharing (streaming)
- CPU-sharing (distributed computing)

.

There are sub-types under these categories, for example, file sharing, content distribution, IP telephony, shared computing, etc.

2.1.1 Classifications of peer-to-peer overlay networks

To understand how a peer-to-peer network can be structured, it is perhaps helpful to look at the different classifications of such networks.

Centralized or decentralized

The structure of peer-to-peer overlay networks can be grouped into three types [5]:

- Centralized
- Decentralized
- Hybrid

Centralized overlay networks rely on a central authority to manage the peers in the network. This can distribute data and tasks among peers, respond to queries from and to peers, and organize the peers in a topology. The obvious drawback of this is, of course, the single point of failure. The whole network suffers if the central server is attacked or goes down for other reasons, and this negates the resilience which was said was a definition of a P2P network. The overlay network can still benefit from scalability and work-sharing among peers in the network, although the central server also limits this. An example of an early centralized P2P network is Napster [9].

Decentralized networks, conversely, do not have any centrally located service to organize the network and provide an entry point. These are also called "true" peer-to-peer networks because they do not function on a shared service. This type of network can scale to include a large number of participants. They are more resilient against attackers because control of the network is distributed. The downside is that search operations take a longer time. The network needs to be self-organizing to some degree. Methods to overcome this are described further down in the paper.

Hybrid networks combine elements of centralized and decentralized structures by using "Supernodes" or "UltraPeers," i.e., peers that take on the roles of

servers in centralized networks. These peers are used as connection points for other peers to upload directory listings of searchable resources. Hybrid networks share the resilience aspect of decentralized networks since "Supernodes" are geographically dispersed and are not easily disabled.

Structured or unstructured

Unstructured overlay networks typically have arbitrary topology and use flooding-based routing among peers. They are simple and robust, but query speed is poor [5]. In an unstructured network, the peers do not know the route to other peers. Instead, they use *flooding* or *random walks* to find resources. Flooding sends out a message to all the peers. "Random walks" choose a random number of peers to send to.

Structured overlay networks have far more efficient routing capabilities. Instead of a random topology, structured networks are organized to allow for efficient information retrieval. Structures like rings, toruses, hypercubes, de-Bruin networks, or more loose randomized networks exist. Distributed Hash Table (DHT), described below, are a particular kind of structured network enabling efficient lookup.

Routing in structured overlay networks is done using an overlay identifier. It forwards messages only to one neighbor, which minimizes the distance traveled based on a distance metric. Each hop in the network brings the message closer to its destination.

2.1.2 Properties of peer-to-peer overlay networks

The common properties of P2P networks usually include security, anonymity, scalability, resilience, and query efficiency. [7].

Security

A security issue in P2P networks can be malicious nodes. Adversarial nodes can destabilize a network by routing traffic through sub-optimal routes or spoofing the identities of other peers. In anonymizing networks, they can be used to de-anonymize users employing traffic analysis or man-in-the-middle attacks. Timing-correlation attacks, for instance, can be used if an attacker has control of both entry and exit nodes in a network. DDoS attacks are meant to destabilize the network by saturating it with so much traffic that it reduces the speed and usability of the network for other users. In addition to these types of attacks, there are other methods to break the security of a P2P network.

Anonymity

Reiter and Rubin [10] present a taxonomy for anonymous communication properties along three axes. The first axis is the type of anonymity where the concern

is *sender anonymity* and *receiver anonymity*, i.e., the identity of either sender or receiver. The second axis defines against what type of adversary each type of anonymity is provided. In a P2P network, this would be either a peer or an eavesdropper who has access to a node through which traffic flows. The third axis is the degree of anonymity. This can range from *absolute privacy* to *provably exposed*. In between, there are degrees of anonymity, e.g., *beyond suspicion*, *probable innocence*, *possible innocence*, and *exposed*.

A goal of an anonymous peer-to-peer network would be to maximize the anonymity properties along all three axes. However, this might conflict with the performance and scalability of the network as it might require routing along suboptimal paths in the network. For example, to prevent traffic analysis traffic, one would route through paths that are not the shortest distance between peers.

Scalability

A network's ability to scale is the number of nodes that can join and communicate without detrimental effects on the network's performance. Decentralized peer-to-peer networks usually scale better because they do not have a bottleneck in the central component. With increasing size, there also becomes a necessity to structure the network. Unstructured networks usually use *flooding* or *random walks* to locate resources in the network, and this becomes unsustainable when the network scales beyond a certain point. With more extensive peer-to-peer networks, there comes a need to structure the network with more efficient routing algorithms than *flooding* and *random walks* [5]. More on such routing algorithms are covered later in the chapter.

Resilience

Resilience is the network´s ability to function despite nodes failing or transient nodes leaving the network. These nodes leaving or failing can result in separate parts of the network from the overall peer-to-peer network. Depending on the topology, this can leave large network parts without a connection. For instance, if the failing node is the only connection between two parts of the network with multiple nodes. Network resilience is elaborated on in section 2.4.

2.2 Anonymity networks and Tor

Tor is an overlay network of relays using onion routing [1]. The network anonymizes communications between sender and recipient by encapsulating the messages in layers of encryption. Each intermediary along the path between the sender and the final destination peels off a layer of encryption, revealing the next hop in the route. Therefore, each relay along the path only knows its predecessor and successor but no other nodes in the circuit. With enough intermediaries, no single node along the route can know both sender and recipient of the communication.

Tor has between 6000 and 7000 active relays, along with around 2000 bridges that act as unannounced entry points into the Tor network in places where Tor might be blocked.

2.3 Overlay networking on Tor with Rtun

Fallang [2] implemented an application (Rtun) that uses the Tor Hidden services protocol and rendezvous points to create an overlay on top of Tor that facilitates a peer-to-peer network. Peers connect through a Tor rendezvous point (RP) relay. The initiating peer establishes a circuit with a Tor relay through a Tor Guard node and sends a Rendezvous Establish cell with a cookie. The cookie value identifies the RP and is used by the connecting peer. The connecting peer establishes a circuit to the RP and sends a Rendezvous Join cell with the same cookie value. When the RP relay receives the cell, it splices the two circuits and notifies the initiating peer. The two peers can now communicate with each other through the RP. Anonymity can be traded for performance by adjusting the number of hops a peer must traverse before reaching the RP. In the Rtun application, a minimum of two hops is needed between peers due to Tor protocol restrictions.

Rtun creates a SOCKS5 interface using OpenVPN to tunnel traffic through the circuit between the peers. Applications can then use end-to-end encryption and native Linux IP layer routing to route traffic to peers. This has the advantage that routing and end-to-end encryption can be handled by applications like OpenVPN, the Rtun application only provides the communication circuits between peers. The P2P network is "protocol agnostic," meaning it doesn´t limit the type of traffic transported across it. TCP, UDP, and ICMP traffic are supported.

The Tor network currently only supports in-order delivery mechanisms, like TCP and certain DNS queries [11]. Datagram delivery options, like UDP, are not implemented as of yet due to security implications for the network [12].

On the other hand, it is not a pure overlay network where, for instance, routing is abstracted to the overlay layer, like Gnutella. In this layer, each peer has a *node ID*, and traffic is routed through the network based on this *node ID* rather than their IP address and port. The advantage of this is that the P2P network is "aware" of the network and controls routing decisions rather than delegating it to lower network layers. The application would then only need to keep track of which circuits lead to which peers.

It is known that overlay routing enhances IP networks' reliability and performance. This is because they can bypass network congestion and transient outages by forwarding through intermediate overlay nodes [4].

2.4 Resilience in peer-to-peer networks

Resilience in peer-to-peer networks is related to graph theory in mathematics. Graphs are mathematical structures that are made up of *vertices*, or *nodes*, and

edges, which connect the vertices. A basic graph (G) can be described as $G = (V, E)$. V is a set of vertices and E is the set of edges [13]. In computer networks, the vertices can be clients, servers, routers, or other computer hardware or software types. The edges are the network links that connect those vertices. Graphs can be un-directed or directed (digraphs). Directed means every edge has a direction. In computer networks, it is the direction the data must travel through a link. A number can be associated with the edges in un-directed or directed graphs. This can represent the *weight*, or the *cost*, of the path. This can be the bandwidth, latency, or other metrics describing the route's cost for computer networks. When adding this, we have a *weighted* graph. A common problem in routing is finding the minimum spanning tree, i.e., paths with minimal cost and avoiding routing loops. In graph theory and network routing, several methods exist to find the shortest path between a pair of vertices. One such method is called Dijkstra´s algorithm [14]. It is a so-called *greedy* algorithm that starts with the source vertex and iteratively builds a table of distances, based on edge weights, from the source vertex to every other vertex in the graph. An important concept in graphs and network resilience is the *node degree*. This is the number of connections a vertex or node has to other vertices. A network with a high node degree is generally more connected than one with a lower node degree. Related to node degree is known as clustering coefficient. This describes to what degree the nodes of a network tend to cluster together. The *global* clustering coefficient gives an overall indication of the clustering of a network. The *local* clustering coefficient describes clustering around specific nodes. A network with a high average node degree generally has a high global clustering coefficient.

Leonard et al. [15] describe a model and analysis of resilience in P2P networks. They investigate the ability of each user to stay connected to the network in the presence of frequent node departure and partitioning behavior. The metrics used to measure resilience are 1) the time before all neighbors of a node v are in a failed state and 2) the probability of this occurring before v decides to leave the network. These conditions are studied under two models. The passive lifetime model assumes broken neighbor relations are not repaired. Therefore, the node will become isolated when the life span of its neighbors runs out. The other is an active lifetime model where neighbor relations are repaired when one or more becomes broken. For this to happen, the node must know that a neighbor relationship has been broken, and it must know of other nodes in the network with which it can establish new connections. A node failure in a P2P network can be detected through a keep-alive mechanism that includes periodic probing of the neighbor and other mechanisms for detecting lost transmission. Many strategies can be used for non-DHT networks. The paper found that k-regular graphs, i.e., graphs where every vertice has k degrees, are more resilient than expected and that dynamic networks are highly likely to stay connected. Varying node degree was only more resilient when correlated with the user´s life span.

A network´s resilience can be measured by the ratio of which queries reach the correct target or a *query-hit* ratio. Two factors can affect this ratio: 1) The target

node can be down. 2) The path between requester and target can fail because of link failures, either deliberate or accidental [7]. In a peer-to-peer network, one can expect a lot of transient nodes due to peers joining and exiting the network. These transient nodes can either be a destination or intermediate links in the network. Node failures can occur when the node does not behave according to the "rules" of the network, for example, by announcing non-existent routes or not relaying traffic properly. A peer-to-peer network needs to handle these resilience issues in a manner that does not break the network should they occur. For example, nodes should have ways of establishing new connections or redundant routes in case one or more links fail.

2.5 Routing in peer-to-peer overlay networks

Overlay networks need a method for deciding where and how a message, or packet, should be transported through the network. In IP networks, this is handled by the network layer, but a benefit of overlay networks is that routing can be abstracted away from the network layer. Thus, routing decisions can be made independent of the underlying network used to transport data. Overlay networks commonly use Node IDs, or Globally Unique Identifiers (GUID), instead of IP addresses to route traffic. In a DHT network, like Chord for instance, this GUID is generated by a consistent hashing function using the IP/port of the joining peer. In Chord, the address space of the GUID corresponds to the size of the Chord ring. For an m-bit address space, there are 2^m identifiers to choose from.

2.5.1 Types of routing algorithms

Several routing algorithms can be used to transport a message between peers [14] [16] [17]. These are some of the basic types:

Destination-based routing has the participants in the network maintain a routing table of destinations that they use to look up the next hop. Destination-based routing is simple and efficient. The downside is that it doesn´t consider other factors, like congestion, and doesn´t give the sender any control over where the message is sent.

Source-based routing lets the sender of the message influence, wholly or partially, where and how it is routed through the network. This can be useful where there are policies to be enforced or security considerations in which nodes are allowed to see the traffic. This usually needs to be implemented in the application layer of overlay networks to allow control over the routing decisions.

An even more complex routing method is Policy-Based Routing (PBR), where routing decisions are made based on factors like link capabilities, Quality-of-Service, and so forth.

2.5.2 Routing table size and network diameter

In P2P networks, there is a tradeoff between the routing table size and the network diameter. The routing table size and network diameter increase for a peer that maintains a particular network state, and the maintenance cost of keeping connections to neighboring nodes also increases [18]. Therefore, it is often desirable to only maintain a partial state and efficiently traverse the network until the resource is found. Many P2P networks lie between the opposite extremes of maintaining a complete and no state.

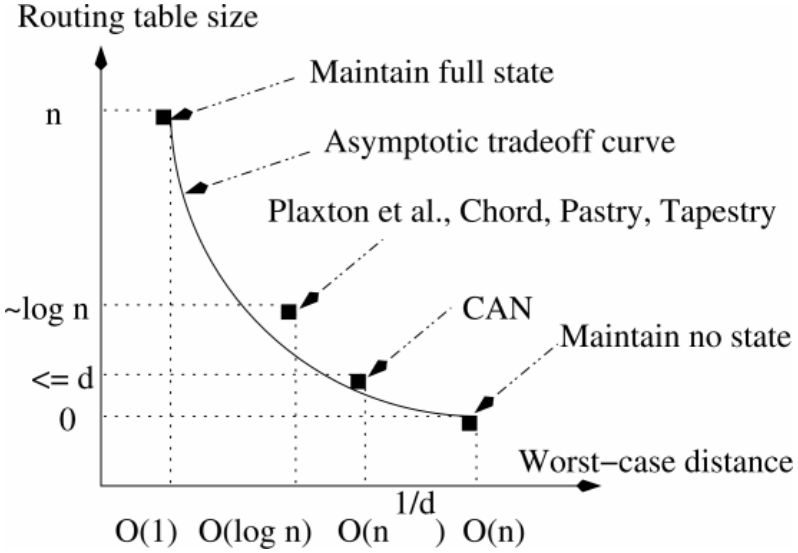


Figure 2.1: Routing table size vs network diameter from [18]

Routing in a peer-to-peer network falls into two categories: 1) where can you find the resource, and 2) how do you reach that resource? The first category can be solved using a technique called DHT. The second must be done by a routing algorithm.

2.5.3 Distributed Hash Tables

A common method for routing in decentralized peer-to-peer networks, like Chord [3], CAN [19], Pastry [20], is called Distributed Hash Tables (DHT). Networks that use DHT construct a virtual key space where each node is responsible for a portion of the key space. The keys are used as destinations in routing traffic. In addition, each node holds a set of nodes as neighbors based on their position in the virtual keyspace. This is a scalable and efficient method of routing in peer-to-peer networks. However, DHT does not consider the underlying physical network the overlay is built upon. This can create "topology mismatches" where close neighbors physically are distant in the overlay [21]. For example, in Chord, when a node wants to join the network, it chooses a random identifier generated by a consistent hashing function. For instance, the hashing function can use its IP and port number

as input or a public key. It then has to send a message to a "bootstrap node" in the network to query for the generated identifier´s successor. When the successor information is returned, it contacts the successor and joins next to the successor in the Chord ring. The Chord ring is a circular topology of 2^m identifiers. After that, the joining node a *finger table* which contains *m*-number of entries to other nodes in the network. When receiving a message, the finger table looks up the nearest node to the identifier. This is done recursively until the message reaches the successor of the destination identifier.

DHT does not ensure locality in routing messages, i.e., the route taken may not be the most efficient in the underlying network. For this, Pastry uses a proximity metric when constructing the routing table. The proximity metric is a scalar value that could approximate the number of IP hops or geographical distance. This would not work for a network that uses Tor as a substrate, as the IP address of peers is hidden from each other.

2.6 Routing security issues in peer-to-peer networks

A common threat to anonymous networks is where an adversary has partial or complete control of the network. If the adversary can monitor the traffic passing through one or more nodes, it can launch many attacks to de-anonymize the participants in the network. The adversary can also compromise the availability of the network by manipulating traffic. Traffic can be routed through non-optimal routes to increase latency and slow traffic. It can prevent traffic from reaching its destination or attack the integrity of the network by fudging the data [22]. In an anonymous network, having the routing information shared by all the network members is not desirable. Preferably, any one member should not have a complete overview of the entire network topology.

2.6.1 Authenticity of routes

A concern with routing protocols is that a malicious insider can inject routes into the network that disrupt the traffic flow [23]. This can result in link overload, long routes, delivery failure, routing loops, or churn. An attacker can also divert traffic to eavesdrop on communication between other nodes in the network, thereby facilitating a man-in-the-middle attack.

2.6.2 Uni-directional or bi-directional tunnels

Tor circuits are bi-directional, meaning the payload data travels back and forth between nodes in the same circuit. With unidirectional tunnels, like those used by I2P, the return traffic from a peer is never observed in the same tunnel as it was sent. Herrmann and Grothoff [24] presented an attack on the I2P network where they compared unidirectional and bi-directional tunnels. Unidirectional tunnels increased the time it took for monitoring peers to be in the correct position to

deanonymize. However, they calculated that with unidirectional tunnels, they had a much higher probability of deanonymizing the correct peer once the monitoring peers were in the right place. Compared with bi-directional tunnels, the false-positive rate was lower. Still, the use of unidirectional data flows could be used for preventing traffic analysis by correlating upstream and downstream data flows. A version of unidirectional tunneling is the "Ferris wheel" proposed by Beitollahi and Deconinck [25]. This ring-based onion circuit forwards clockwise or counter-clockwise traffic between peers connected in a ring. All nodes are homogeneous, and traffic is homogeneous in that dummy packets are not indistinguishable from actual packets.

2.7 Routing in similar peer-to-peer networks

There are P2P networks that can provide anonymity. Some of them are given here, along with an overview of Mobile Ad hoc Network (MANET) which are similar to P2P networks.

2.7.1 Briar

Briar is a peer-to-peer network that uses the Tor Hidden Services protocol to create network endpoints for accepting incoming connections [26]. Each communicating party exchanges public keys that they derive a shared rendezvous key. They derive a stream key from the rendezvous key to create contact details for the peer´s endpoints. From the stream key, the peers create a master identity key pair of a Tor hidden service. The hidden service address is derived from the public key. After the rendezvous is established, the peers keep the connections open and try to connect to the endpoint every minute for 48 hours. If no connection is made after 48 hours, it considers the rendezvous to have failed. The application layer decides how to use the rendezvous connections created.

The protocol's security depends on the confidentiality and integrity of the public keys exchanged. If a Man-in-the-Middle intercepts the keys and exchanges them for its own, it can impersonate one of the parties.

2.7.2 Freenet

Freenet is "an adaptive peer-to-peer network application that permits the publication, replication, and retrieval of data while protecting the anonymity of both authors and readers" [27]. The Freenet project was designed as a distributed storage and retrieval network to address privacy and availability concerns.

Freenet nodes maintain a dynamic routing table containing addresses and information about other nodes in the network. It uses a routing similar to IP (Internet Protocol), where each forwarding node decides where to send the request. When deciding where to send the request, it checks its routing table for the closest

node reference of the hashed key of the file to be retrieved. If a node cannot forward a request further, it sends a backtracking failure message to the preceding node, and the preceding node tries another node. The expected effect is that nodes in the network will, over time, learn paths to keys in the network.

2.7.3 I2P

A similar P2P network to Tor is the Invisible Internet Project (I2P) [28]. It is a decentralized peer-to-peer network that offers anonymity and cryptographically secured end-to-end communication channels between its peers. In contrast to the Tor network, it uses separate inbound and outbound tunnels to prevent traffic analysis. Privacy is achieved similarly to Tor by routing the traffic through intermediary gateways, analogous to Tor relays. I2P supports both TCP and UDP traffic, while Tor only supports TCP. However, there is a proposal to support Tor in the future. I2P uses a network database called NetDB implemented as a DHT to store information about the network [29].

2.7.4 MANET networks

There are similarities in routing challenges between P2P networks and Mobile Ad hoc (MANET) networks [17]. MANETs are self-configuring wireless networks [16]. Nodes, or terminals, within radio communication range connect and carry traffic destined for other nodes. Because they do not rely on any infrastructure, they do not need any central component to organize the network. High terminal mobility often causes breakups in adjacent nodes and frequent topology updates. In addition to low bandwidth between nodes, the result is that this kind of network does not scale well and needs optimized routing algorithms.

MANETs using a proactive routing algorithm have a complete network view from every node at all times, and topology updates are broadcast to all nodes in the network. On the other hand, reactive routing floods the network with routing requests to find nodes that have a route to the destination. The best-known proactive routing scheme is Destination Sequenced Distance Vector Routing (DSDV), consisting of routing tables containing the minimum number of hops and the next hop to the destination. Sequence numbers are used to distinguish between old and new updates. Either complete topology updates or incremental ones can be broadcast. Although the resources for maintaining the routing information are high, it is fast in making routing decisions.

Ad hoc On-Demand Distance Vector Routing (AODV) is an on-demand routing algorithm that sends a route request querying a route to the destination. Intermediate nodes forward the request to their neighbors. When an intermediate node with a route receives the request, it sends a route reply upstream back to the source. The nodes along the reverse path note the address of its next hop, and the route is established. When a node notices a downstream breakup of a route, it sends a reply message with an infinite next hop metric along the reverse path, notifying other nodes of the change.

A hybrid approach to proactive and reactive routing is called Zone Routing Protocol (ZRP). It tries to solve the scalability issue with proactive routing and flooding issues with reactive routing by creating zones. Nodes within a zone use proactive routing to determine routes between nodes in the same zone. Each node has the network's full topology view inside the zone. If there is a need to communicate outside the zone, a route request is made to a border node, checking if a route is in its local area. If not, it forwards the request to its border node. When a border node finds a way to the destination, it sends a reply along the reverse path, similar to AODV.

P2P networks and MANET differ mainly because P2P networks are mostly overlay networks that can span widely geographically, and MANETs are densely distributed in an area. P2P typically uses direct connections, while MANET forward through intermediate nodes. Proactive routing algorithms can be used in MANETs with fewer nodes, and this is not feasible in more extensive P2P networks because of scaling problems where signaling dominates over data traffic.

2.8 Routing areas for peer-to-peer networks

In a peer-to-peer network where active routing decisions are based on distributed information about routes between peers, there is the problem of signaling dominating traffic between peers. As the network grows, this signaling takes up bandwidth and processing resources. Therefore, routing updates should preferably be sent as incremental and not complete topology updates. Still, this signaling traffic will become a problem in a sufficiently large network. A solution to this could be to segment the network into different parts. Nodes in each part of the network only send routing updates between other nodes in its part of the network. This approach resembles hybrid zone routing in MANET networks described in section 2.7.4.

A feature of the OSPF protocol can be used to reduce the problem of churn and continuous routing updates within the network. The P2P network can be divided into routing areas, and these areas exchange routing updates via link state announcements. Nodes inside the area do not send or receive routing updates to nodes in other areas. Other areas can be reachable via so-called *SuperNodes* [4], or *SuperPeers* [30] that route traffic between two areas. These SuperPeers are peers with connections to two or more areas in the network and can provide sufficient bandwidth.

The overlay network is logically layered into two levels, with the "lower" level compromising normal peers and the "top" layer comprising SuperPeers. In the lower layer, each peer can only communicate directly with another peer within its area. To communicate outside its area, it either needs to know the area identifier value or transmit through a SuperPeer. The SuperPeer is a peer with a circuit between two or more areas, and the SuperPeer announces the namespace of the areas it connects but not the area identifier. If a peer wishes to communicate with another outside its area, it routes the message to the SuperPeer connecting the

destination area. If a normal peer knows the area identifier of another peer and connects to it via a rendezvous point, then the two peers automatically become SuperPeers in their area.

Figure 2.2 visualizes two areas connected by SuperPeers represented by a thick box border. The establishing SuperPeer in area A creates a rendezvous point with a cookie value corresponding to the area. B identifier it connects to. The SuperPeer in area B connects to this rendezvous point with the cookie value of its area identifier value.

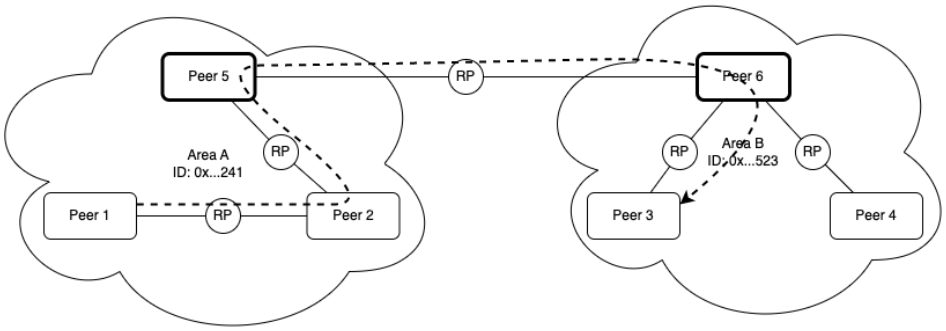


Figure 2.2: Communication between areas

The Tor Rendezvous protocol uses a 20-byte cookie to identify which circuits to splice at a rendezvous point. This cookie value can determine the routing area the peer belongs to by calculating the SHA-1 hash value of a cryptographic key shared between peers in the routing area. The SHA-1 output is a 20-byte value which is the same size as the cookie value used by Tor rendezvous points. Although SHA-1 is considered insecure and deprecated by NIST [31], it is still widely used and could be considered suitable for this application.

The drawback of using one standard cookie value for the entire area is that each peer-to-peer connection will need its exclusive Tor relay as a rendezvous point. Reusing Tor relays as rendezvous points would be beneficial as the number of relays suitable for an area could become a bottleneck as the number of peers grows. For example, there are about 6000 Tor relays, and around 80 are in Norway. A fraction of these are suitable as rendezvous points in our network. The solution could be splitting the area identifier value (i.e., cookie value) into two parts. The first m bytes of the value represent the area identifier, and the last n bytes are a modifier incremented each time a Tor relay is used as a rendezvous point. For example, an SHA-1 hash of a cryptographic key produces the area identifier value:

```
2f20bf1dbec30e4b405d072e172991d602ab1a1f
```

The cookie value used on the rendezvous point for a relay used for the first time becomes:

```
2f20bf1dbec30e4b405d072e172991d602ab1a00
```

if we use the first 19 bytes of the value as the area identifier and the last byte as the modifier (each hexadecimal value represents four bits). When the relay is reused as a rendezvous point between other peers, then the modifier is incremented, e.g.:

```
2f20bf1dbec30e4b405d072e172991d602ab1a01
```

Each rendezvous point can then tunnel 256 connections between peers. The procedure for figuring out the modifier value could be based on the routing table, which contains all the rendezvous points used inside the area. It could also be a try-and-fail logic where the modifier is incremented each time a peer tries to establish a rendezvous point.

In addition to reducing the number of routing updates sent, routing areas also increase the anonymity of the peers in the overall network. Peers that are members of Area A are not visible to members of Area B. Routing areas can also be used to group peers on specific criteria. Geographic proximity can be used to group peers in a joint group, thereby reducing peer latency.

In a P2P network with no centrally controlling entity, there is a possibility for address space collisions, i.e., peers that use the same ID inside the network. In a network divided into zones or areas, the address space encompasses only the size of the zone or area, and peers outside the zone are routed using the zone and local peer identifiers.

The concept of dividing the key space into several sectors to reduce the churn problem has been examined in a paper by Jiunn-Jye Lee et al. [32]. It shows that using a sector-based routing model (SBRM) reduces the maintenance cost of leaves and joins in a structured peer-to-peer network.

These routing areas can divide the network into "communities" as done in the Quiet project [33]. This project uses Tor relays, and each community has its insular network. Each community has an owner who acts as a PKI central authority. For peers to join the community, they need a certificate issued by the community owner´s CA. By establishing trust between community members, malicious actors are less likely to attack the network with fake route announcements or use other attack vectors, as all communication can be authenticated. The challenge is how to select a central authority in a decentralized network. One option is the first-come, first-served approach, where the creator of the community is the de-facto owner and root of trust. Establishing trust with peers joining the community poses another challenge that can be solved with out-of-band methods, for instance.

Chapter 3

Methodology

This chapter will describe the methodology of designing, implementing, and testing the prototype application to help answer the stated research questions.

3.1 Research objectives

The research objective of the thesis is to find a routing protocol suited for a peer-to-peer overlay network using Tor relays as rendezvous points. The routing protocol should perform well and offer resilience for decentralized peer-to-peer networks. A prototype application implementing the protocol will be developed to test the protocol. This prototype will be connected to the live Tor network and obtain results of interest to the research questions. The goal is to see if the peers can communicate and structure themselves according to the protocol design objectives. The main focus of the experiments is to see if a network can be set up to be efficient, stable, and resilient.

3.2 Design of a prototype application

The design will extend the work of Fallang [2] and develop a routing and connection control protocol that utilizes the Tor rendezvous points used in the original application. The overlay network routing protocol in the application will be tested by setting up a peer-to-peer network using Tor network relays with the modifications that have been done. The bootstrapping will be done statically to have the peers join in an initial known topological configuration.

3.2.1 Extending to the overlay layer

Overlay networks are, like the name says, *overlays* that lie on top of the network layer. A part of the development will be to build an overlay addressing scheme into the application. Peer IDs, or Node IDs, will be used instead of IPv4/IPv6 addresses to route communication through the network. It will then use those

Peer IDs to communicate and route traffic inside the network. The peer needs a globally unique identifier (GUID) when joining the P2P network. For testing purposes, a statically defined ID is assigned to the peers instead of calculating it based on a hashing function as DHT networks use.

3.2.2 Routing and connection protocol

The routing protocol will select which paths messages travel through the network. Performance is a critical factor, and therefore *shortest-path* routing seems like the best option for this prototype. Primarily since the path length predominantly affects latency in these kinds of networks, as noted in Fallangs paper [2]. The shortest-path algorithm's efficiency depends, in large part, on the topological view of the network and the path-cost metric used. For the prototype and experiments, an adaptive link-state routing, used in Open Shortest Path First (OSPF) and Destination Sequenced Distance Vector Routing (DSDV), that distributes link-state announcements throughout the network, was opted for [14]. Thereby, peers can use Dijkstra´s algorithm to calculate a route through the network. The path cost metric tested in the experiments is latency, measured in delay in round-trip time between peers. Other metrics are possible and will be discussed later.

The connection protocol, in addition to routing, will structure the peers in the network for resilience and efficiency. When utilizing the Tor network, the connection protocol will be responsible for setting up circuits, ensuring peers have enough redundant paths, and switching high-latency circuits. A method of negotiating rendezvous points and joining peers is tested.

3.3 Implementing the prototype and test environment

The prototype application will be implemented in Python. The NetworkX library is used for graph operations and route calculations. This is because the application this prototype builds upon was developed in Python, making integrating the code generated here easier. Also, the TorPy client library the application utilizes is written in Python. There are simulation environments for the Tor network that can be used for testing, but this takes time to set up, and testing with the live Tor network will give results that are more in line with real-world performance. The prototype will therefore be tested on a virtual machine connected to the Internet that can communicate with Tor relays.

3.4 Testing the protocols

To test the overlay addressing, routing, and connection protocols, data from a series of test runs are collected and analyzed according to performance, stability, and resilience metrics. The experiments will seek to establish the validity of the protocols by testing their functionality, starting with a minimal topology and creating a baseline to compare against more complex setups. The routing and path

selection will be tested for their ability to choose the lowest-cost route. Resilience and stability will be tested by introducing random link failures.

3.4.1 Performance tests

Message routing

The ability to route messages is measured in this test. The application must be able to route messages over multiple hops using the peer ID as an address. The protocol can either use source routing based on the shortest path or flooding by sending messages out on all links until the destination is reached.

Path updates

This will test how well the link cost is measured, distributed, and applied to the network paths. The link cost is calculated by using the round-trip times to neighboring peers. This is distributed with route update messages to all peers in the network, and those peers then use it to calculate new shortest paths.

Rendezvous Point and circuit switching

Rendezvous point and circuit switching alternate routes through the Tor network between peers. Tor uses the same circuit for new TCP streams for 10 minutes as long as the circuit is working [34]. For long-lived TCP streams, the circuit is kept indefinitely. This is so adversaries with a partial view of the network do not get many chances to link to your destination. Staying on a secure circuit will prevent traffic analysis by adversaries but could impact performance if the circuit has high latency. This experiment will test the switching of rendezvous points and circuit rotation of paths between peers.

3.4.2 Stability and resilience tests

Detecting dead routes

The routing protocol must detect when a route has gone down. The prototype will test whether dead paths are removed from the topology and if the rest of the network is updated.

Establishing routes

A resilient network needs redundant paths in case one or more of them is broken, either because of dead links or nodes. The prototype will test how well the protocol maintains minimum connections to peers.

3.4.3 Measurements and metrics

The experiments will measure the reliability of the network. Reliability is measured by the *query hit-ratio*, which is the ratio of query messages sent that reach their destinations [7]. A peer will send a query message to a particular destination peer, and if the message reaches its destination peer, it will be counted as a *hit*. We will simulate path failures to test whether the routing algorithm can route the message along healthy links. The number of query messages sent compared to those that reach their target will result in a *query hit-ratio*. The other variables measured in the experiments will be node degree, clustering coefficient, network convergence, path length and cost, and signaling traffic generated by the network.

Dealing with node (peer) and link failures is vital for a resilient network. A variable that can be used to measure the resilience of a network is average node degree [35]. The average node degree is the average number of edges per node in a graph, and the higher the number, the more alternative paths there are in the network.

Another metric that can be used to explain the network's connectivity is the clustering coefficient [35]. The clustering coefficient measures how connected a node's neighbors are to one another. It is calculated by dividing the number of edges connecting peer *i*'s neighbors by the total number of possible edges between *i*'s neighbors. For instance, a spread-out network will have a lower clustering coefficient than a dense network.

Network convergence is where all the participants in the network have a common view of the network's topology [14]. When a topology changes, for instance, a node leaves or joins, there will be a period where some nodes have an incomplete view of the topology. A topology change will be announced to the network by the neighboring peers of the node joining or leaving. The delay in announcing the change can be broken into two parts: 1) The time it takes for the neighbor to detect that a link has gone down, and 2) the time it takes to propagate the change to the network. The time it takes for the neighbor to discover that a link is down depends on whether the peer becomes unresponsive or there is a link failure or shutdown. If a link becomes unresponsive, there will be a delay before the other peer discovers that it is no longer receiving traffic from the peer via a keep-alive message or other data traffic. The delay can be reduced by increasing the keep-alive frequency, increasing signal traffic in the network. When a link failure is detected, an announcement will be sent from the neighboring peer to all the other participants in the routing area. The delay in propagating the topology change will depend mainly on the size of the network that needs to be notified and the latency of the paths used in the network.

Non-convergent networks can produce routing loops where data is routed around the network without reaching the destination. For example, in a 4-peer network, as shown in figure 3.1 where P1 tries to send to P4 over P2, but P2 has a dead route to P4 that P1 hasn't been notified of. P2 will try to route the message back to P1 since it knows the route P2-P1-P3-P4, and P1 will send it back to P2

again. Thus, we have a loop.

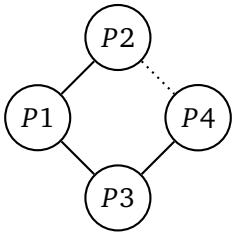


Figure 3.1: Topology with the dotted dead route

Path length and the path cost are the distance a message must travel to reach its destination. To provide acceptable performance, the network path length and cost must be kept at a minimum, especially in networks with high latency and multiple hops. The ability of the network to structure itself in a way that minimizes the distance between peers is a metric that will be measured in the experiments.

Routing and control data traffic is considered overhead and should be reduced to a minimum. The amount of control traffic generated by the protocol is of interest as it takes up bandwidth that otherwise could be used for data. The experiments will measure the amount and distribution of routing and control traffic generated.

3.5 Result analysis

Data created during the tests will be collected from the prototype application. The results from the experiments will be analyzed and discussed to establish how well the selected routing and connection protocol answers the research questions.

Chapter 4

Experiments and results

The experiments described in this chapter will attempt to measure the choices of metrics and criteria described in the methodology chapter. The tests are grouped into two main categories. First, the basic functionality and performance of the network is measured. The second category of tests measures the resilience of the network. The experiments will start with simple setups to establish the functionality and validity of the metrics that will be used. Here a baseline is established that can be compared with a network containing more peers and connections.

4.1 Implementation

The basis of the implementation of a routing protocol will be the code that was developed by Fallang [2] for a peer-to-peer overlay network using Tor. A routing mechanism is added to select circuits for transmitting and receiving data. This will extend the application with a dynamic routing option instead of the static one it has now. The goal is also to make routing decisions in the overlay layer of the network, instead of the network (IP) layer. For these experiments, a version of Link State routing called Shortest Path First (SPF) [36] is implemented. It uses Dijsktra's Shortest Path First algorithm to calculate the shortest path between two nodes in the network. Each node keeps a database of the paths to its neighbors, along with information such as the cost of the path to these peers. Changes in a peer's environment are distributed through Link State Advertisements (LSA) which are flooded throughout the network until all peers have received them. The receiving peers then calculate the shortest paths to other peers using Dijkstra's algorithm based on their view of the network. SPF creates a weighted graph of the network. The weights of each of the edges represent the cost that the route has between the connecting vertices. There are several options for metrics that can be used to calculate the cost of a route. This implementation will use link latency as a cost metric. This is the round-trip time between two neighbors as measured in milliseconds by sending a request-response message between them. This method of routing is commonly used in networking and is implemented in the Open Shortest Path First (OSPF) protocol [37] in many routers. OSPF is the most widely used

interior gateway protocol on the Internet [23]. The routing protocol will take care of communicating application-level messages with their payload to other peers in the network.

The code developed for this thesis is available as open source at `https://github.com/andrgaar/imt4905`. The modified TorPy client library from the original application by Fallang is used.

The application was modified to use Python threads within the main process, thereby having asynchronous running of different threads of execution. This allows the client process to maintain several independent connections to other rendezvous points, and simultaneously have routing logic running alongside the connection threads. The application is divided into two parts where one part takes care of routing and connection control, and the other takes care of setting up the connection and circuits communicating with the rendezvous points. The routing and connection protocol components run as four separate threads within the main application:

- *Receiver*
- *Sender*
- *Heartbeat*
- *Connection*

Receiver thread listens for incoming messages on the routing stream and distributes them to various handlers depending on the type of message received. The *Sender* thread periodically sends out LSAs to neighboring nodes within a specified update interval. The *Heartbeat* thread will periodically send out *heartbeats*, or keep-alive, messages to neighbors to indicate that the path is alive. Neighbors that do not receive a heartbeat within a certain time period will assume the link is dead. The neighbor will be removed from the internal topology graph and then announced to the rest of the network in the next LSA sent out by the peer. The *Connection* thread keeps track of neighbor connections and selects new peers to join or switches rendezvous points and circuits when necessary.

In addition to the routing and connection threads, there are two threads for communicating with the rendezvous points. The *RendezvousEstablish* thread sets up a new rendezvous point, and the *RendezvousConnect* thread connects to an established rendezvous point.

4.1.1 Protocol messages

The application defines a set of application-level messages that are used by the protocol to communicate state changes, peer connection status, and other connection control messages. The messages are Link State Announcements (LSA), Heartbeat/keep-alive (HB), connection control (JOIN), and handshake (HELLO).

LSA messages

Link State Announcements (LSA) are messages that contain information about a peer and its neighboring connections. LSA messages update the network about which routes are available and the cost associated with those routes. When an LSA is received it is added to the receiving peer´s LSA database and forwarded to other neighbors of that peer. The receiving peer calculates its view of the network by adding, updating, or removing routes in its internal graph representation of the network for the peer that is the source of the LSA.

HB messages

Heartbeat (HB) messages are sent out periodically to indicate that a route is up and the peer on the other end is alive. If a peer doesn´t receive a heartbeat from a neighboring peer within a specified period of time, that neighbor is considered dead. An LSA is then flooded to the network containing the dead route(s). All the other peers receiving this LSA can then update their topological map of the network. Heartbeats are also used to measure link latency. This is equivalent to an ICMP ping where the heartbeat is sent with a *request* field indicating that the receiving neighbor should send back a response. The requesting peer calculates the round-trip time (RTT) from the response it receives from the neighbor. The path cost to that neighbor is then updated when enough samples of heartbeat messages are received.

JOIN messages

The JOIN message is a connection control message that tells the receiving peer to open a connection and join a rendezvous point. The message contains the Tor relay identity and cookie value identifying the rendezvous point relay that the receiving peer can use to connect to the peer that sent the JOIN message.

HELLO messages

After two circuits are joined through a rendezvous point the connecting peers need to establish each other's identities. This is done with HELLO messages the peers send each other. The HELLO message contains the Peer ID of the peer joining. This simple handshake establishes identities and does not exchange cryptographic key material.

4.1.2 Routing of messages

Each peer will be able to route a message based either on the shortest path, or flooding. In case of flooding, a TTL value is used to avoid flooded messages traveling the network indefinitely.

4.1.3 Connection setup

Figure 4.1 describes a normal setup between two peers. The establishing peer will connect to an RP through a guard node (GN). The connecting peer can use a guard node to increase anonymity or connect directly to the RP.

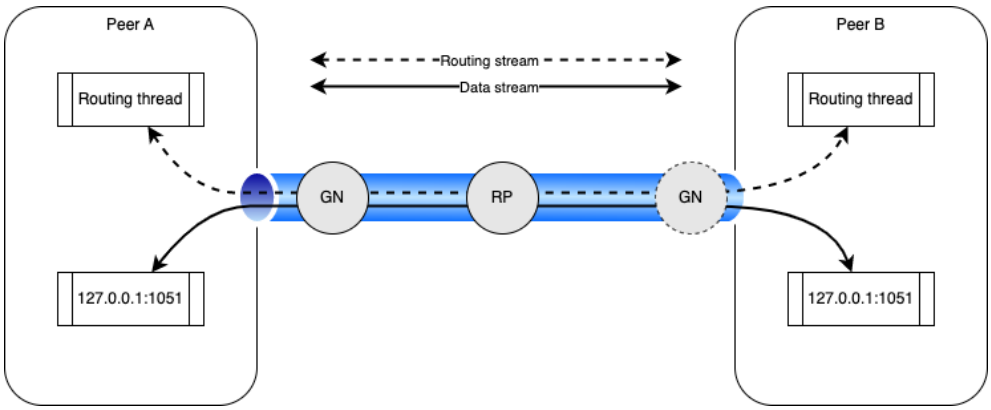


Figure 4.1: Communication between peers

The connection setup is achieved by first creating the routing and connection threads, and then establishing the connections to peers with the rendezvous connect/establish threads. Each main process has one set of routing threads, i.e. Receiver, Sender, Heartbeat, and Connection. The rendezvous threads contact a Tor relay to either establish a rendezvous point (RP) or connect to an already established RP. The peer establishing the RP will create the circuit to the RP and wait for another peer to join. The connecting peer connects to the RP and waits for the establishing peer to open a stream within the circuit. When the connecting peer joins the establishing peer will receive a RENDEZVOUS2 cell as a signal that someone has joined. The process of establishing and joining an RP is described in Fallang´s paper [2].

Figure 4.2 describes the peer setup flow. After receiving the RENDEZVOUS2 cell the establishing peer creates a stream inside the circuit for communicating with the joining peer´s router. The establishing peer sends the connecting peer a RELAY_BEGIN cell with an address and port. In our case, this is a dummy address and port since the application doesn´t need an actual TCP socket endpoint. The connecting cell replies with a RELAY_CONNECTED cell containing the dummy IP address and a Time to Live (TTL) value. Upon receiving the RELAY_CONNECTED cell the establishing peer sends a HELLO message handshake over the created stream identifying itself. The connecting peer receives this and sends a HELLO message back with its own identity. The identities of each party have now been established. Routing and connection information between the peers will be exchanged over this stream during the time they are connected.

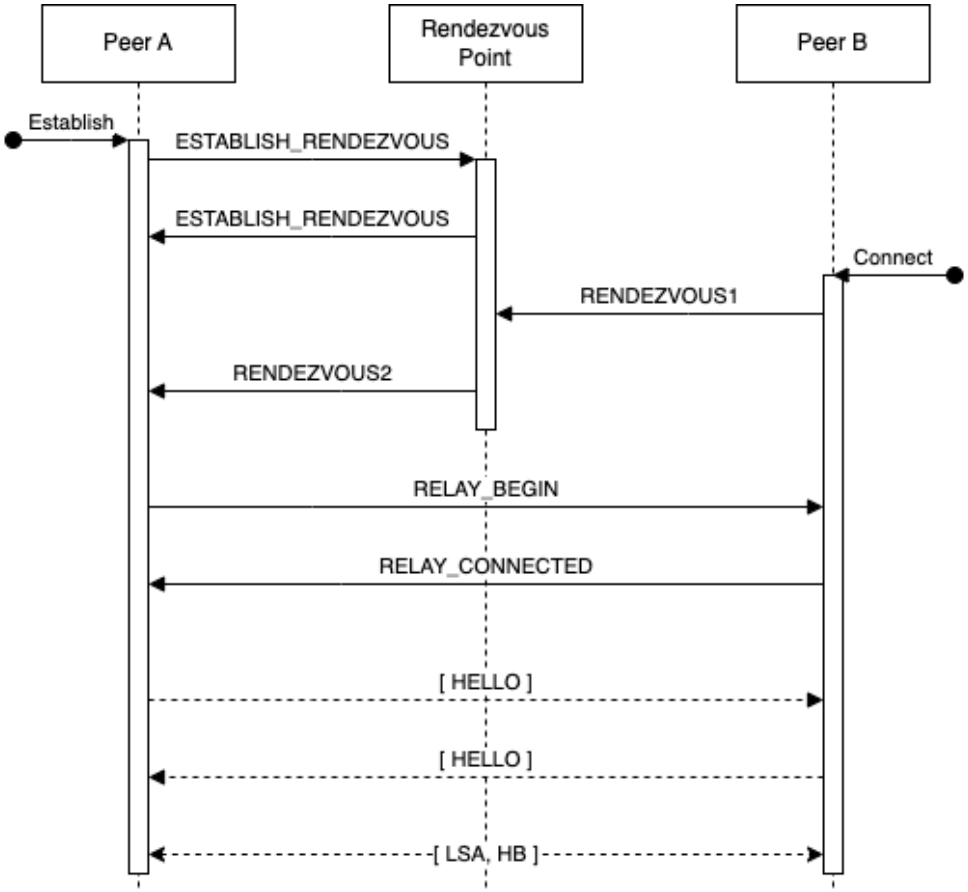


Figure 4.2: Peer connection setup

4.1.4 Rendezvous point and circuit switching

The peer that initiates the switch will always be the establishing peer of the current circuit. When a circuit switch is due it will select a new rendezvous point from available relays in the Tor consensus document. These relays are selected from those that have Stable, Valid, and Running flags. It will then send a JOIN message to the opposite peer telling it which relay to join. Then the other peer connects to the rendezvous point and exchanges HELLO messages. When HELLO messages have been exchanged the new circuit becomes the active path between the two peers. The old circuit and rendezvous point will die when it times out and the circuit is torn down. The connection threads exit and releases resources held.

4.1.5 Selecting peers and rendezvous points to join

This algorithm runs periodically to check if each peer has enough neighboring paths. If a peer needs more neighbor connections to fulfill the requirement it will choose a rendezvous relay, select a non-neighboring peer and send a JOIN message instructing it to connect to this rendezvous point. It works as shown in algorithm 1.

Algorithm 1 Redundant path algorithm

```
connections ← numberOfNeighborConnections()
while connections < MIN_NEIGHBOURS do
    relay, cookie ← chooseRelay(flags)
    establishRendezvousPoint(relay, cookie)
    max_latency ← 0
    for all peer ∈ non_neighbors do
        l ← getLatency(peer)
        if l > max_latency then
            join_peer ← peer
            max_latency ← l
        end if
    end for
    sendJOIN(join_peer)                ▷ Send a JOIN message to the peer
end while
```

4.2 Test environment

For the tests, we run the application on a virtualized environment. The guest virtual machine (VM) runs Debian 11 64-bit on Azure. The VM runs on 4 vCPUs with 16 GiB of RAM. Each peer runs as a separate process inside the VM. The VM has connectivity to the internet in order to contact the Tor relays. All connections for setting up the P2P network are initiated from inside the VM network. Therefore, it is not necessary for the VM to be exposed to incoming connections from

the internet. Python v3 is used for the development and runtime platform for the application.

4.3 Connection setup between peers

The setup we use between peers for all the experiments is the *two-hop* connection between peers and Tor relays explained earlier. The peer establishing the rendezvous point connects through a Tor guard relay (GN) to the rendezvous point relay (RP). The connecting peer joins the rendezvous point directly without intermediate relays as described in figure 4.3 where P1, as the establishing peer, connects to P2. In all the following experiments this is inferred when displaying links between peers.

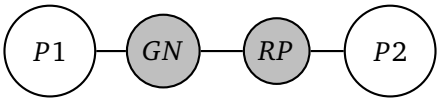


Figure 4.3: Inter-peer connection topology

4.4 Functionality and path metrics

In the functionality and path metrics experiments, the baseline functionality and validity of selected path cost metric is tested.

4.4.1 Link cost metrics

The first experiment is to validate the chosen link cost metric. In this experiment, we use the link latency as measured by the round-trip time to the neighbor as the link cost metric. The heartbeats sent between peers are used to calculate a Round-Trip Time (RTT) for the connection. This is similar to an ICMP echo request-reply. The heartbeat is sent out with a millisecond time reference in the transmission from the sender. When a peer receives the heartbeat it immediately returns a heartbeat with the response. Once the response is received by the sender the RTT is calculated and added to a running total. When a set number of heartbeats and RTTs has been gathered the link cost of the connection is updated with the RTT. The updated link cost needs to be announced to the rest of the network. An LSA is sent out notifying others of the updated cost of the connection. Each peer can then update the network graph with the new cost.

Latency vs geographic distance

The topology will be configured with rendezvous points in different geographic distances from the peers as in figure 4.4. The relays are selected from the highest announced bandwidth relays in their country. Since all traffic will originate from

the same location the distances will be calculated from Oslo, Norway to the country where the rendezvous point relay is located. P1 is the only peer using a Tor guard node. The others connect directly to the rendezvous point. The guard node is selected from the same location as P1. The test will sample the RTT data produced by the application and correlate them to the geographical distance between the origin and the rendezvous point.

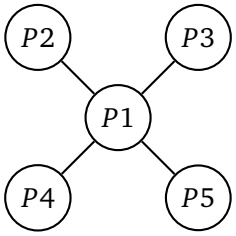


Figure 4.4: Peer topology for the latency experiment

Figure 4.5 shows the distribution of round-trip times compared to the geographical distance to a rendezvous point relay. In the figure, the measured RTT distribution tends to increase along with increasing geographical distance. Calculating the Pearson r - value for this dataset we get $r = 0.22$ which indicates that there is a weak positive correlation between the RTT measured and distance. The p-value for this dataset indicates that the correlation is statistically significant.

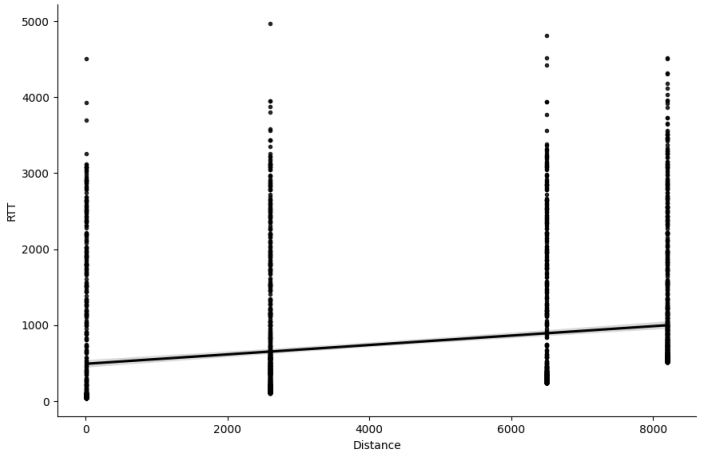


Figure 4.5: Distribution of measured latencies and geographical distance

In table 4.1 we see the measured mean RTT for each path. As expected the mean RTT corresponds roughly to the distance traveled. We would expect the round-trip time measured in this test to differ from ICMP ping. ICMP ping shows the network layer latency. Since our round-trip time is done in the application layer, it probably shows a more realistic metric taking into account the overhead of encrypting and decrypting data cells, congestion, and general performance of

the peer.

Table 4.1: Average Round-Trip Time (RTT) between peers in different geographical locations

Path	Country	Distance (km)	Average latency
P1-P2	Norway	20	482 ms
P1-P3	Japan	8200	1110 ms
P1-P4	USA	6500	732 ms
P1-P5	Spain	2600	717 ms

Route selection

The path to any peer is calculated to be the shortest path based on the cost metric. When using a dynamic cost metric like latency, the path calculation can change with the variability of the latency along the path. For this test, a 5-peer network, described in figure 4.6, is used to see which path the network uses between peers P1 and P5. From the test, we can see in figure 4.7 that the shortest path selected alternates between P2, P3 and P4. The share of time spent as the next-hop peer for P1 is fairly equal to one another as seen in table 4.2.

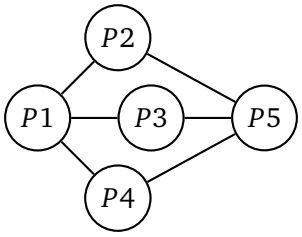


Figure 4.6: Route stability test

Table 4.2: Share of time as route next-hop for P1

Peer	Time as next-hop (s)
P2	438 (42%)
P3	272 (26%)
P4	326 (31%)

Shortest path measurement vs actual latency

In this experiment, the selected path based on latency sampling is compared with other alternative paths to check whether the routing protocol uses the most optimal route to the destination peer. The optimal route will have the overall lowest latency compared to every other possible path. This will indicate whether the

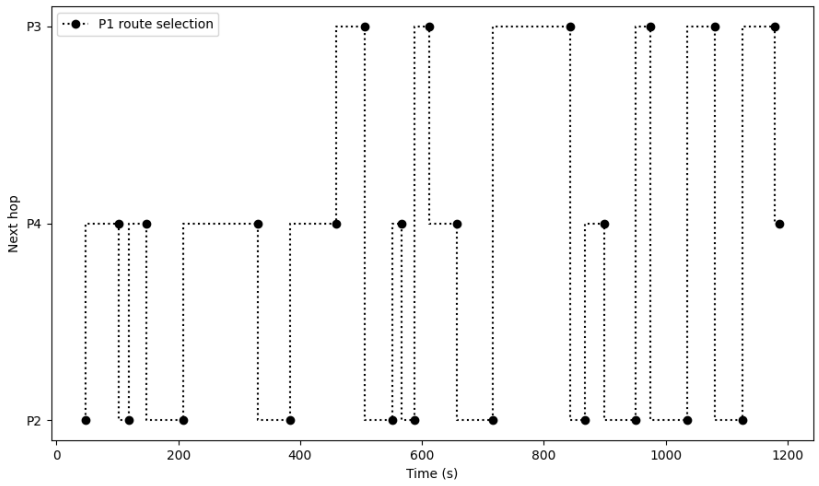


Figure 4.7: Route stability of P1-P5

shortest path cost calculation actually produces the lowest latency path through the network. The network is a 5-peer setup as shown in 4.8.

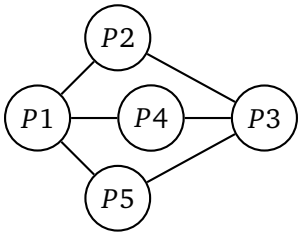
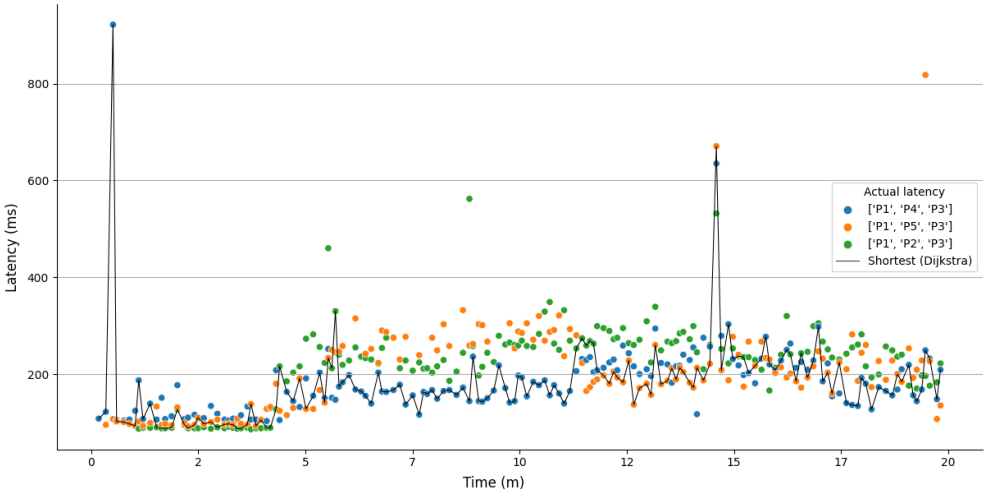


Figure 4.8: Peer setup for route latency test

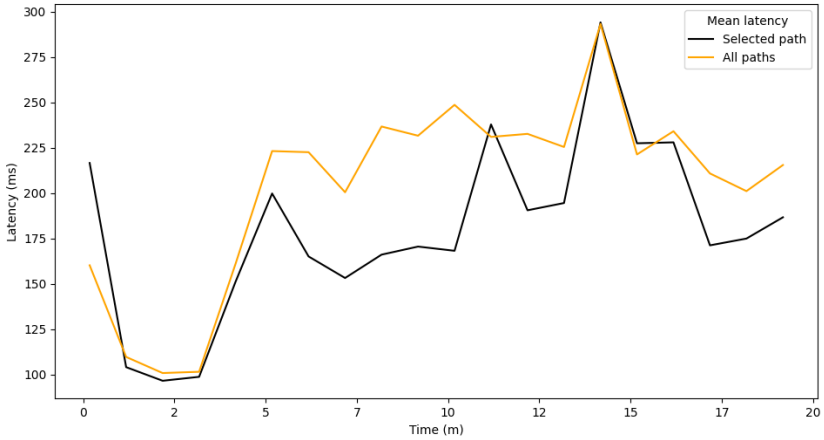
Peer P1 sends LOOKUP messages to P3 via P2, P4 and P5. Hence, there are three paths that can be selected. A message is sent through each path simultaneously and the time traveled from source to destination is registered. The rendezvous points switch every 5 minutes to vary latency.

The results can be seen in figure 4.9 showing the latencies of different paths. Figure 4.9a shows the actual latencies for all messages sent between P1 and P3. The path that is currently used as the shortest route is marked by the black dotted line. In figure 4.9b the mean average latency per minute of all the messages routed is compared to the messages traveling along the shortest path calculated by the routing algorithm.

The test shows that the route selected as the shortest is the one with the lowest latency most of the time. The conclusion is that the route selection algorithm is adequate for selecting the lowest latency route a majority of the time.



(a) Actual latency of all paths



(b) Mean latency of shortest path compared to others

Figure 4.9: Comparisons of latencies of different routes

4.4.2 Routing and control traffic

Routing and control traffic is data that is used for routing and circuit management. To get an indication of how much traffic this constitutes, the inbound and outbound traffic for a single peer is measured under different network sizes.

The graph in figures 4.10a shows traffic generated, without any query lookup messages, between 2 peers. Only route updates, heartbeats, and Tor circuit management traffic is recorded. From this, we can see there are regular peaks every 10 seconds, when the heartbeat is transmitted, and an increase around the 2-minute interval when circuits and rendezvous points are switched. When increasing the peers in the network to eight the routing and control traffic shape remains the same, but volume increases as seen in 4.10b. Table 4.3 shows that there is quite a significant increase in the routing and control traffic being exchanged between the peers.

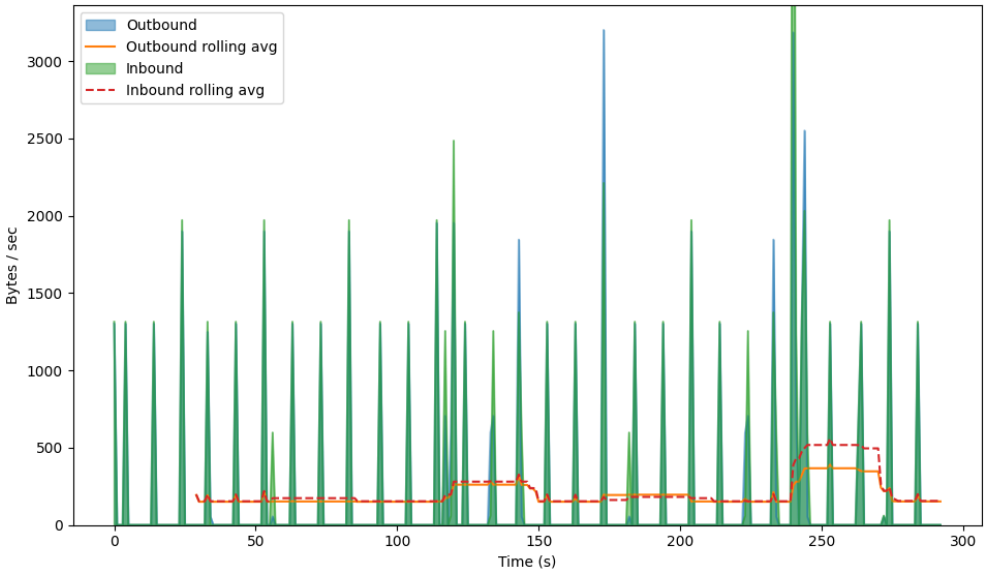
Table 4.3: Average throughput of routing and control traffic per peer

	Outbound (Bytes/s)	Inbound (Bytes/s)
2 peers	193	213
4 peers	947	1386
8 peers (min 2)	2068	2708
8 peers (min 4)	4457	4267

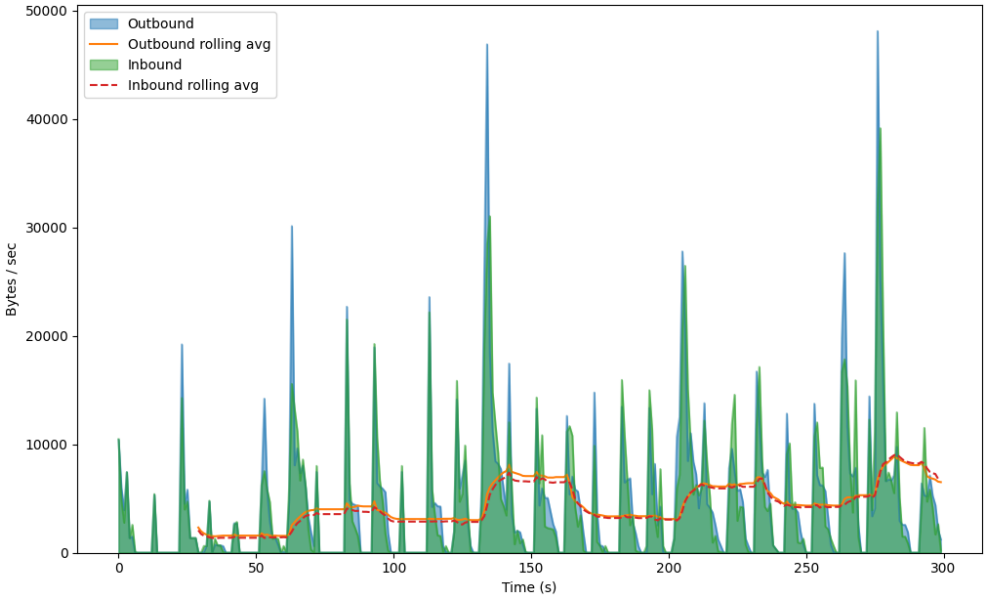
Figure 4.11 shows the distribution of messages received by a single peer. In the 2-peer setup with one connection between them, the majority of messages are heartbeats that are sent by adjacent neighbors. Heartbeats are not flooded therefore they only vary with the number of neighbor connections a peer has. Link State Announcements (LSA) are flooded through the network from both neighbors and non-neighbors and vary by the number of peers in the routing area for which route announcements are received. This can be seen in the distribution of messages in the 8-peer setup with four connections for each peer. Here the LSA messages dominate along with JOIN messages.

4.4.3 Rendezvous point and circuit switching

Rendezvous point and circuit switching test whether the peers can negotiate a new rendezvous point to meet at and switch the traffic through this rendezvous point. The test uses a simple 2-peer network where P1 and P2 share a path P1-P2. Every two minutes the peer that has established the rendezvous point between them, in this case P1, selects a new rendezvous point to use. The rendezvous point selection does not use any other criteria than relays with flags Stable, Valid, and Running. The peer creates a new connection thread that establishes the rendezvous point and waits for a peer to join the rendezvous point. A JOIN message is sent to P2 telling it to connect to the new rendezvous point relay and the cookie



(a) Routing and control traffic for 2 peer topology



(b) Routing and control traffic for 8 peer topology

Figure 4.10: Routing and control traffic for 2 and 8 peer topologies

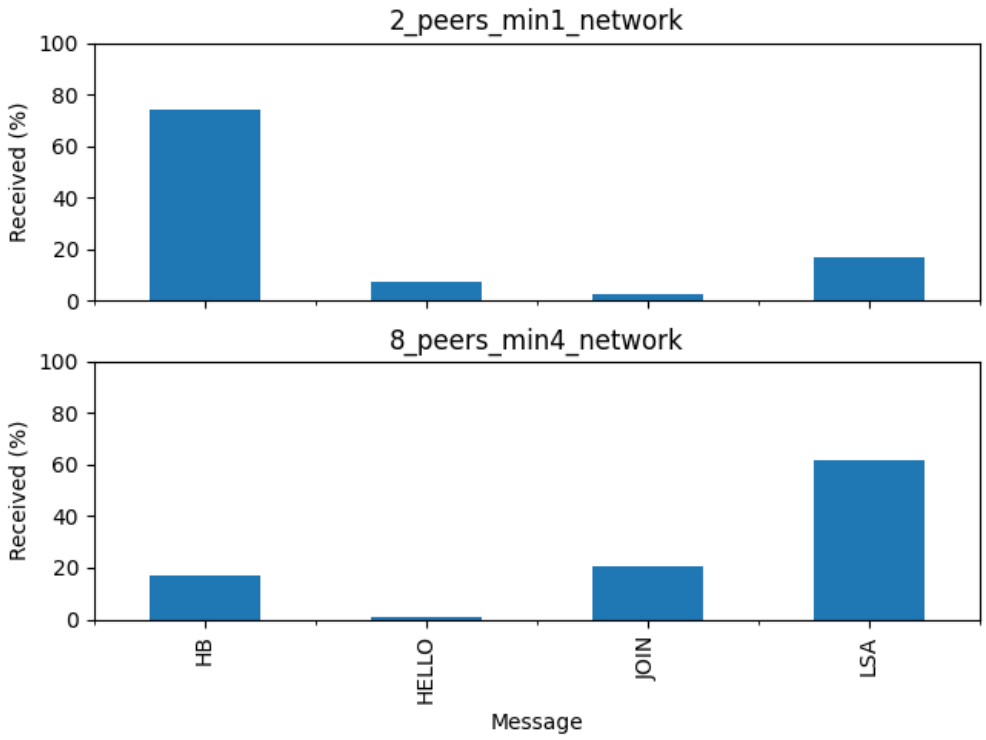


Figure 4.11: Distribution of messages received by peer

value identifying it. The effect on the path cost estimation can be seen in figure 4.12.

The default path cost for a new connection is 1000 before the peer has been able to sample the latencies. Every time the path cost jumps to 1000 it means that a new route and rendezvous point is being used. During this test, the rendezvous point is rotated four times.

The rendezvous points selected are listed in table 4.4. The path cost for the US relay is higher than the others, as would be expected due to geographical distance.

Table 4.4: Rendezvous point selection for P1-P2

Time (s)	Rendezvous Point	Country	Advertised b/w
0	despacitor	France	49.5 MiB/s
120	JimmyBulanik	Latvia	29 MiB/s
243	11Square	US	10 MiB/s
369	Quetzalcoatl	Netherlands	26.1 MiB/s
495	Karlstad1	Sweden	6.76 MiB/s

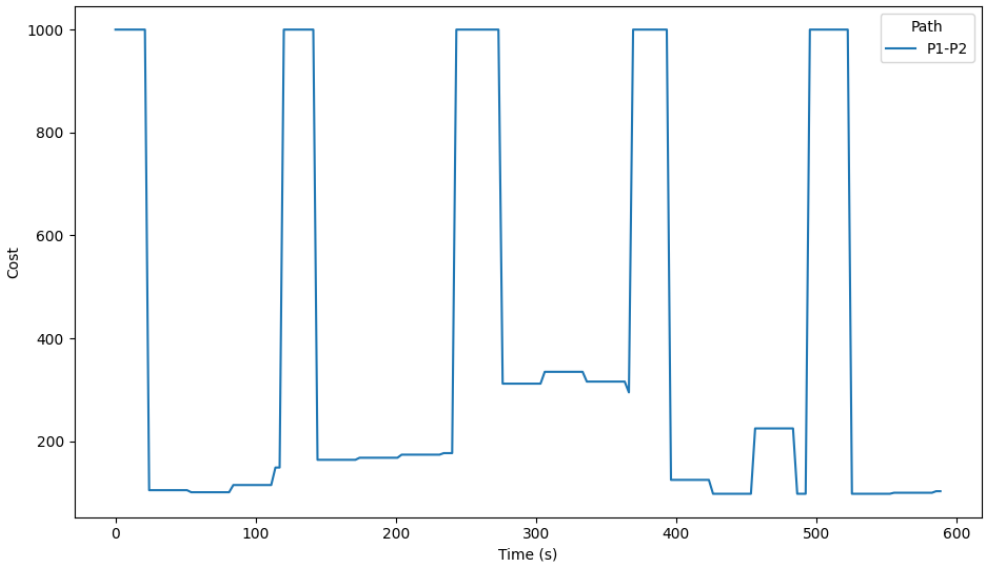
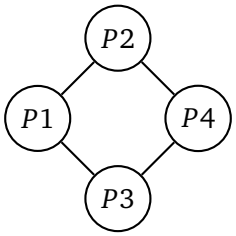


Figure 4.12: Path cost for circuit-switched routes

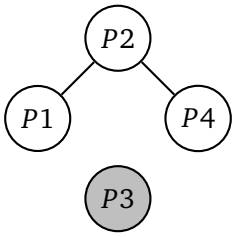
4.4.4 Routing with failed paths

In this test, we introduce a path failure through the shortest path. P1 is sending messages to P4. Figure 4.13 shows the topology where all four peers are active and the lowest cost path is P1-P3-P4. After the failure of P3 the shortest path changes to P1-P2-P4 as shown in 4.13b.

As we can see from figure 4.14 the path taken by the routing algorithm uses the lowest cost path over P1-P2-P4. After the failure occurs there is some time before P1 detects the dead route. It took 50 seconds between the first lost message and P4 began receiving messages again. This is where the timeout value is set to 15 seconds after the last heartbeat is received from the peer. P1 also needs to calculate a new shortest path which adds to the delay. This can probably be reduced by reducing the time delay between dead route detection and the calculation of paths.



(a) Peer topology before failure



(b) Peer topology after failure

Figure 4.13: Peer topology before and after the failure of path

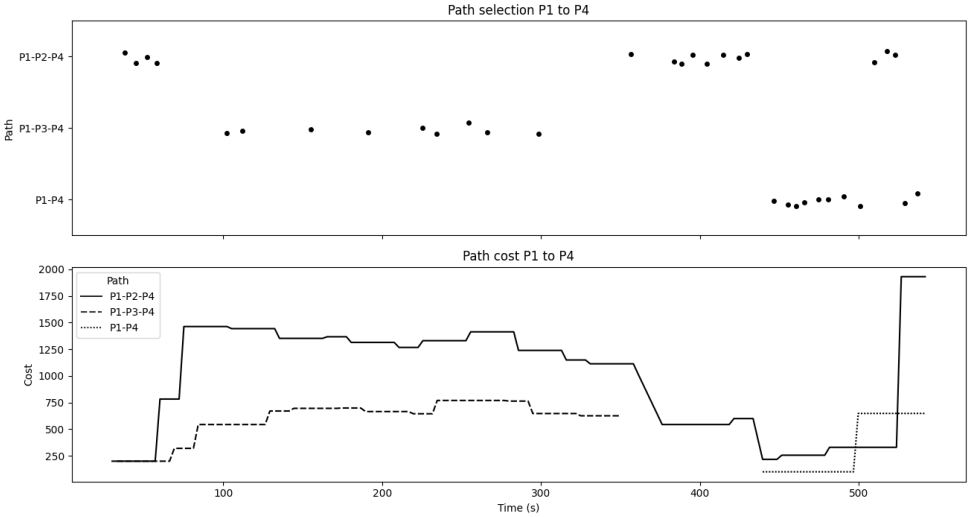


Figure 4.14: Path failure between P1 and P4

4.5 Resilience and reliability

The other category of experiments is to test whether the network can reliably handle link and node failures and recover from them. For these experiments, an 8-peer topology is used as in figure 4.15. In order to simulate link failures and node churn the connecting peer links are set to fail after a random period of time. The Mean Time to Failure (MTF) is set to 300 seconds for each peer in these tests. This is to test the network´s ability to re-establish failed links and keep a connected network. The tests are run for 30 minutes each time.

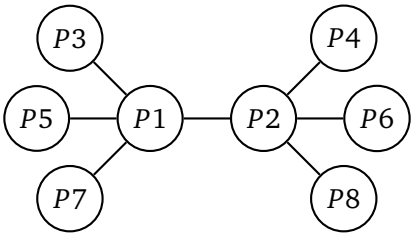


Figure 4.15: 8 - peer topology

The variable that is adjusted in these tests is the node degree, i.e. the number of connections each peer has to other peers. Each set of tests is run with a different minimum connection parameter for the peers. Based on this we can see if there are significant changes in other metrics concerning performance and resilience.

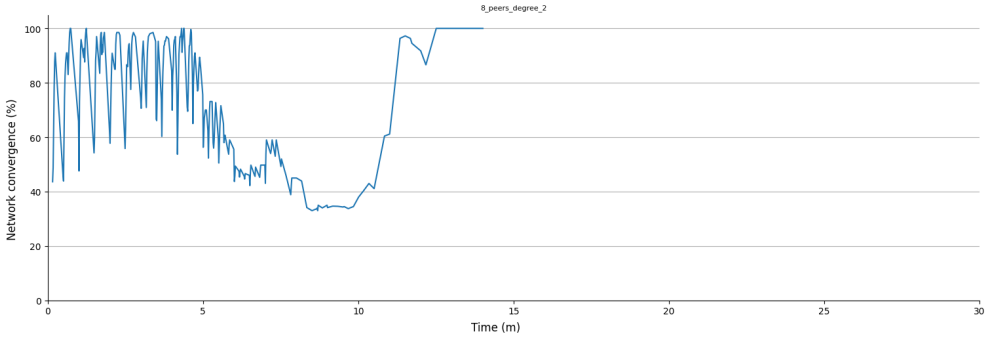
4.5.1 Network convergence

Network convergence is measured by the percentage of peers that have the current SN number for each peer. SN is the serial number that identifies the latest route topology update sent by a peer. Whenever a peer updates its topology it increments the SN of its graph and sends a route update (LSA) with this number. The receiving peer stores the SN to its database of LSAs so that it knows if it is a new update, or already seen update. For the convergence of networks with different node degrees, we can see in figure 4.16 rises and falls with the recalculation of path cost and node churn. The convergence percentage stays on average above 75% for all node degrees.

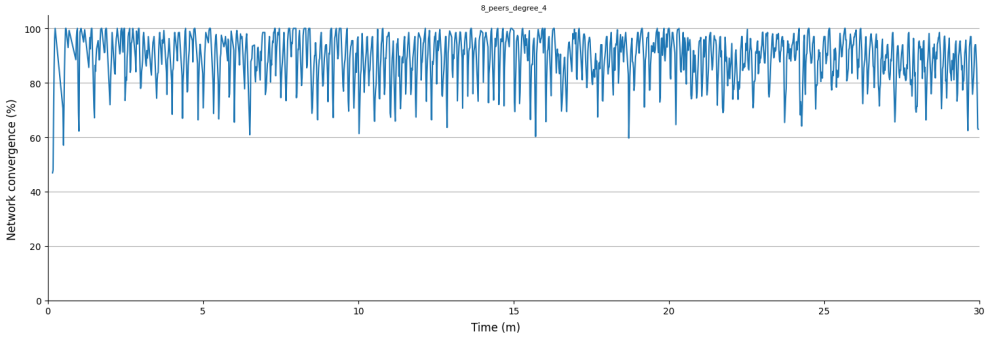
In figure 4.16a with a node degree of 2, the convergence falls after about 5 minutes and then climbs to 100%. This is because the network cannot maintain itself and the peers become disconnected from the rest of the network, as shown in section 4.5.2.

4.5.2 Resilience to failures

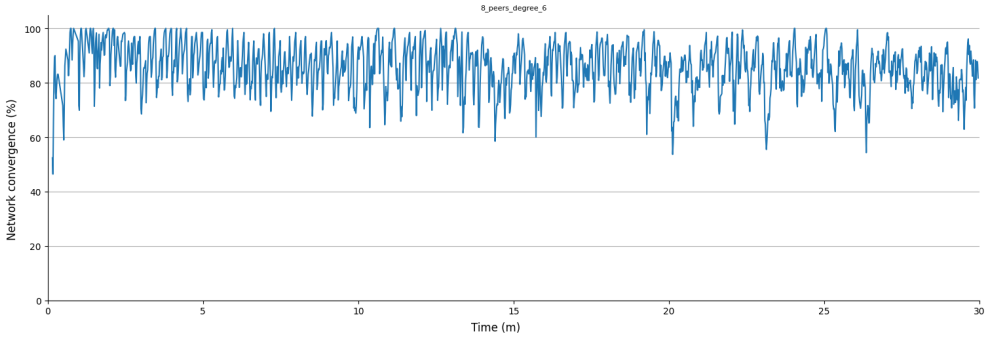
The node degree of the peers in the network is measured to test the network´s resilience to the failure of links between peers. For each of the tests, the peers are set with a specific minimum connection requirement to maintain during the



(a) Minimum 2 connections



(b) Minimum 4 connections



(c) Minimum 6 connections

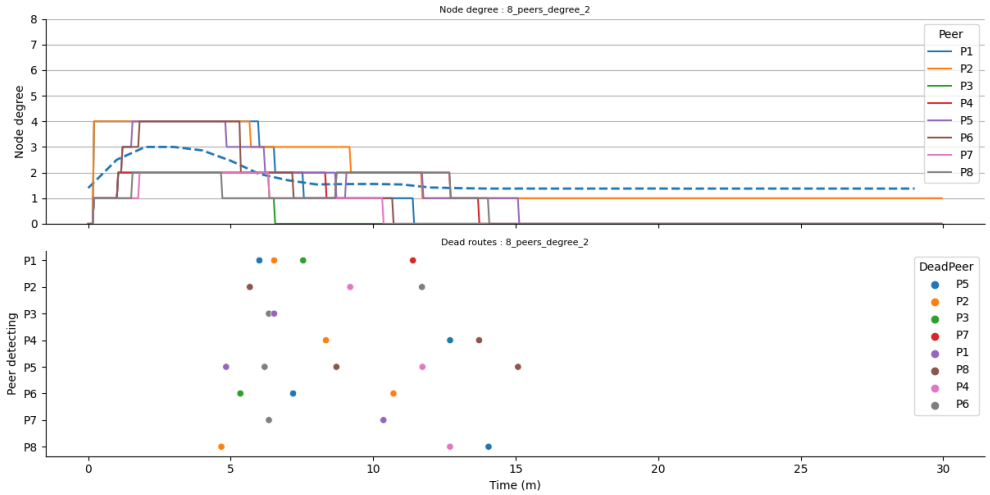
Figure 4.16: Convergence of network at different node degrees

lifetime of the peer. If the number of connections falls below this requirement, the peer will try to establish new connections to non-neighboring peers in the network. The number of actual connections it maintains is called the *node degree* of the peer. When it drops to zero the peer is essentially disconnected from the network and will have to rejoin it by other means. Whenever a peer no longer receives Heartbeat messages from another peer it marks the route to that peer as a *dead route*. The number of connections the peers can maintain to others, along with the dead route detections can be seen in figure 4.17.

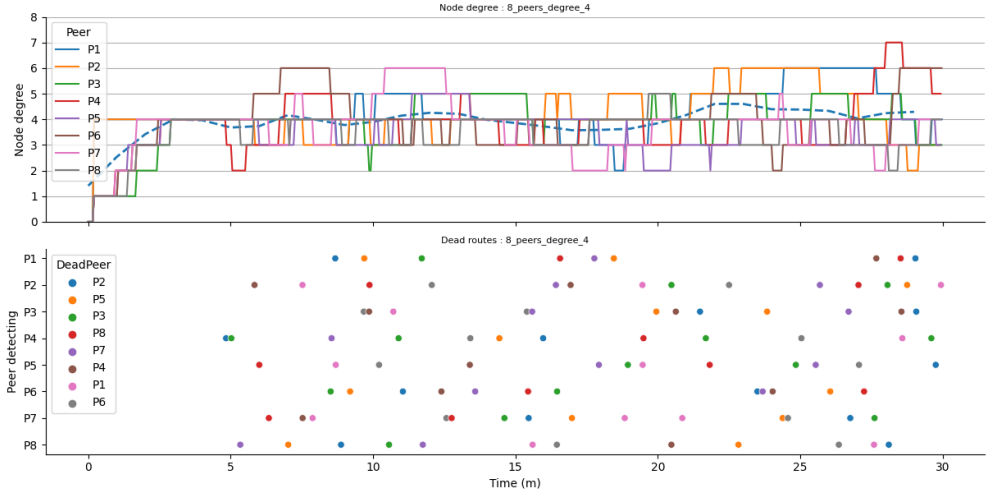
For the network with only 2 minimum connections, the node degree drops to zero for most of the peers after about 15 minutes. This means the network cannot recover from the failed links and dies out when most of the peers cannot make new connections. However, with 4 and 6 minimum connections the peers are able to recover from link failures and continue working.

4.5.3 Clustering and network structure

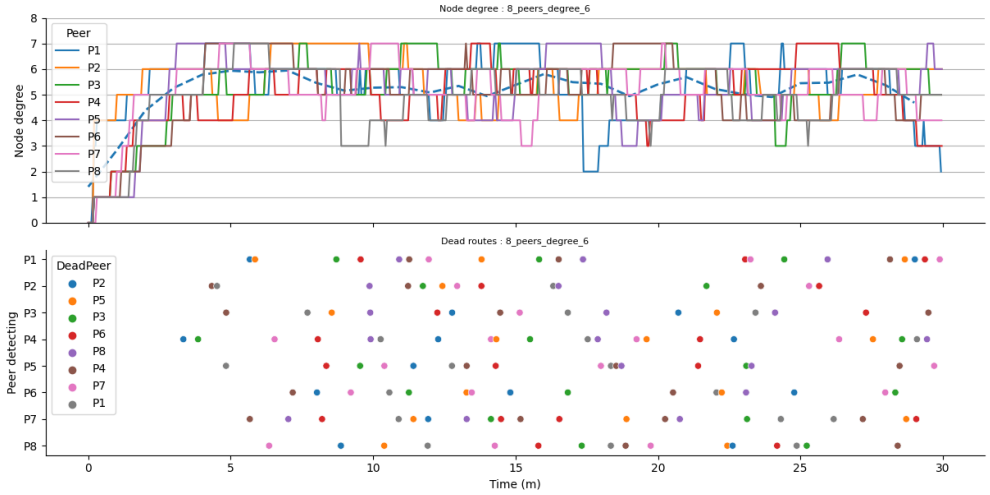
The clustering coefficient tells how connected a node or graph is. A clustering coefficient of 1 is where all neighbors of the node are connected in every possible way. Figure 4.19 shows the clustering coefficient of the network and each node. The clustering coefficient in 4.19a is an anomaly and cannot create any clustering around the peers. As expected, the network with node degree $d = 6$ in figure 4.19c is the most connected, and almost all peers reach a coefficient of 0.8, which means the network is densely connected. High clustering is essential for network cohesion and tolerance for link outages to disconnect parts of the network. In figure 4.18, the network in 4.19c is shown with both low and high clustering coefficients. In figure 4.18a, only one link drop is enough to disconnect either a single node or half of the network from each other. In figure 4.18b, all the peers have multiple connections to others in the network resulting in a more robust topology.



(a) Minimum 2 connections



(b) Minimum 4 connections

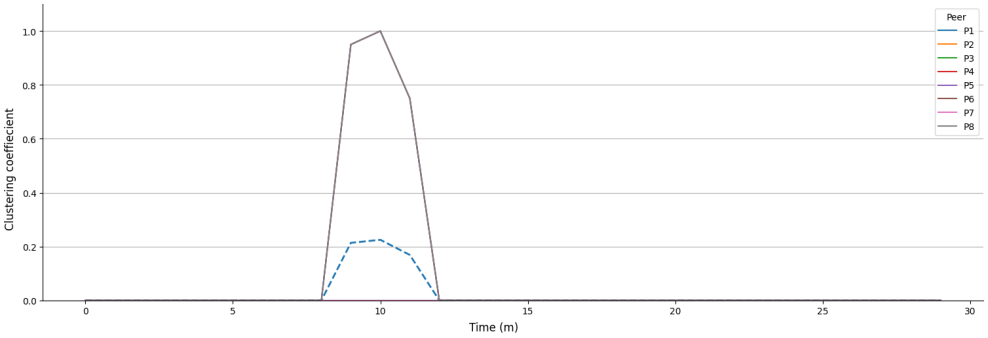


(c) Minimum 6 connections

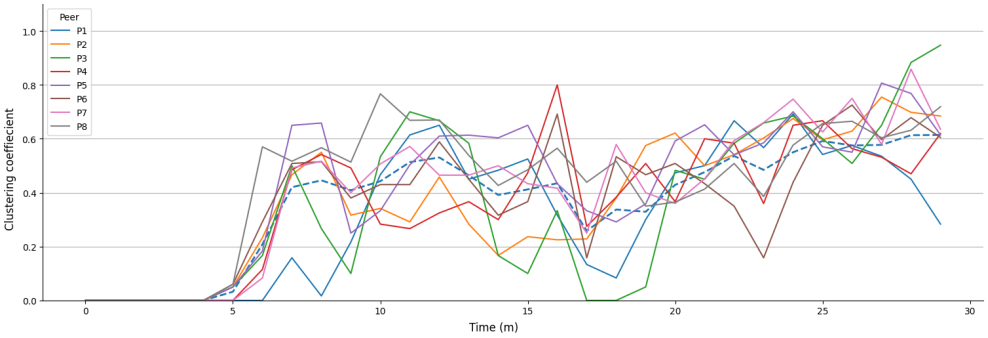
Figure 4.17: Node degree and average node degree (dashed line) with failures

(a) Low clustering **(b)** High clustering

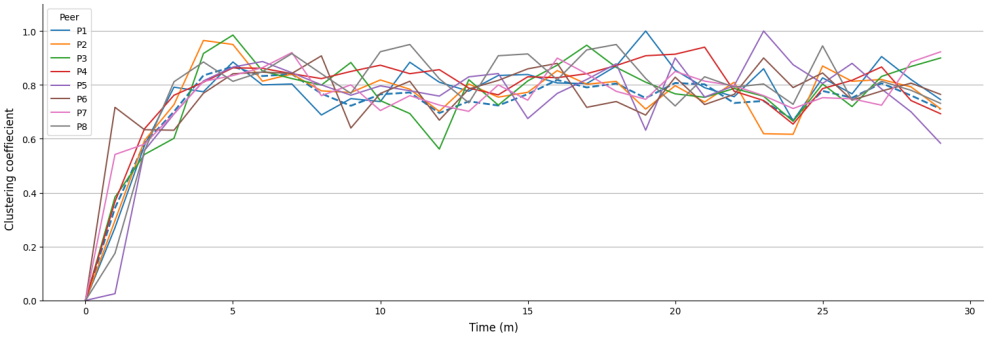
Figure 4.18: Density of connections in networks with low and high clustering coeff.



(a) Minimum 2 connections



(b) Minimum 4 connections



(c) Minimum 6 connections

Figure 4.19: Clustering of network and peers with different node degrees

4.6 Performance and reliability

Performance and reliability of message delivery are measured in these experiments. Latency of message delivery is used as a metric for performance, and reliability is measured by the number of sent LOOKUP messages that reach their destination.

4.6.1 Latency of messages delivered

Table 4.5 shows the measured latency averages for peers with different node degrees. Networks with lower node degrees have a lower latency on average, but this must be compared with the distribution along the lifetime of the peers as shown in figure 4.20. The network with a node degree of d = 2 has a high variability in latencies and does not survive long before all nodes become disconnected.

Table 4.5: Average latency (ms) / minute for message delivery between peers with different node degrees (d)

	d = 2	d = 4	d = 6
Mean	156.8	220.3	412.8
Stddev	209.1	80.9	264.3
Min	44.8	98.0	87.4
Percentile 25%	80.3	162.3	167.4
Percentile 50%	104.7	233.8	356.5
Percentile 75%	125.0	276.4	560.8
Max	873.5	392.6	1050.0

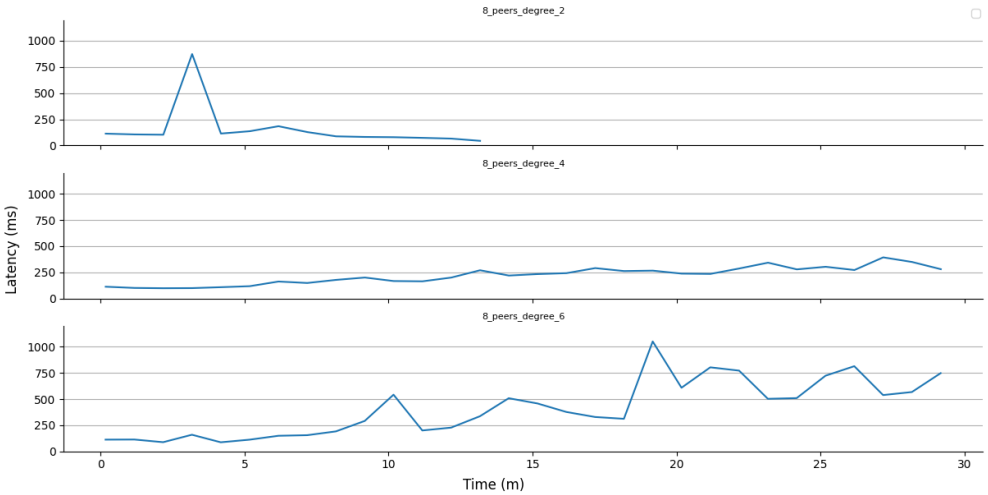


Figure 4.20: Average latency (ms) / minute of message delivery in networks with random failures

The network node degree d = 4 has the lowest latency and variability on average for the two others. One would expect more connections to translate into shorter latencies because of more options, but this could result from different relays selected in the Tor network.

4.6.2 Reliability of message delivery

Reliability is measured in the number of LOOKUP messages sent to another peer that reaches its destination. Table 4.6 shows the network's reliability. The reliability is low for a network with node degree d = 2, and the other networks with node degrees 4 and 6 have few success rate differences.

Table 4.6: Average reliability (% success rate) / minute for message delivery between peers with different node degrees (d)

	d = 2	d = 4	d = 6
Mean	28.6	71.5	73.9
Stddev	36.6	14.6	14.4
Min	0	49.2	43.1
Percentile 25%	0	62.3	65.8
Percentile 50%	0	66.6	72.9
Percentile 75%	61.8	80.2	84.2
Max	100	100	100

The reason for message delivery failure appeared to be when the route to the next hop or destination peer had died but before the peer had discovered and recalculated the topology.

4.6.3 Path length vs. path cost

A benefit of denser networks is lower path length between peers. Fewer hops should yield lower latency, all else being equal. The average path length is compared to the average path cost to test this. The testing shows that the network will seek to join distant peers, based on latency, and lower the average path length along with the average path cost. An example of this can be seen in figure 4.21 where there´s a drop in average path length at around 10 minutes when the network re-configures itself. Path cost differences can be seen before and after, where the cost is lower after the average path length has fallen.

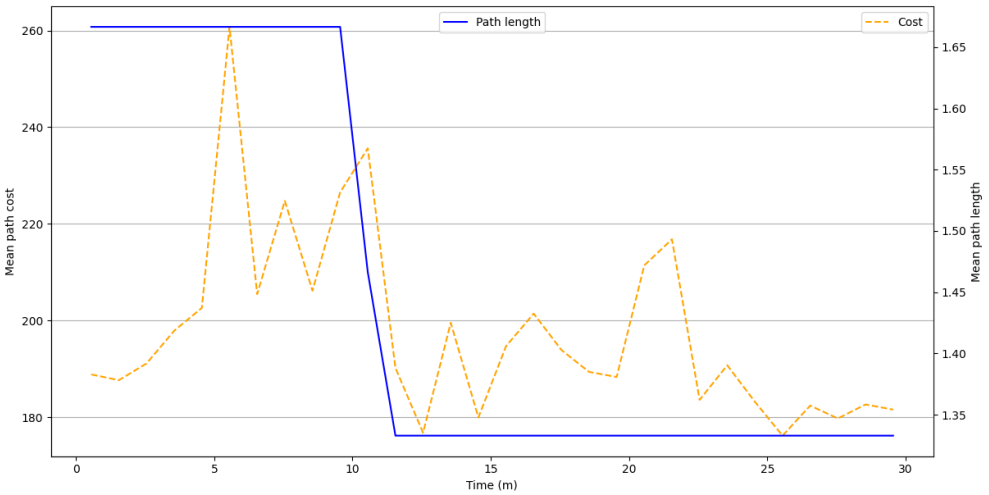


Figure 4.21: Average path length and cost

Chapter 5

Discussion

The experiments wanted to test how data can be routed efficiently in a peer-to-peer overlay network using Tor relays, and which metric could be used for this. Secondly, how stability and resilience could be ensured in the network. A prototype application implementing a routing protocol based on link-state updates and latency measurements between peers was used to answer the research questions.

There are many possible methods of routing and control to ensure both efficiency and stability. The selection of which method to experiment with was based on the challenges with these particular kinds of networks compared with other, non-anonymous networks. The need to minimize the number of hops between the source and destination is crucial as each intermediate Tor relay or peer incurs a relatively large price in latency. Therefore, a network of topology-aware peers was considered the most suitable to test. Link latency was used, but other cost metrics are discussed. The data from the experiments showed that there was some validity in using latency as a cost metric for selecting the most efficient path in the network.

Secondly, the experiments wanted to test whether the connection protocol could ensure a stable network that is resilient against link and node failures. A connection control protocol was tested that could negotiate a rendezvous point with other peers in the network and maintain a desired degree of connectivity for each peer. Connecting peers in a Tor network, or similar, poses an extra challenge compared to non-anonymous P2P networks. Direct connections cannot be made as they would reveal the IP addresses of peers. Instead, a rendezvous point must be negotiated and a connection established through it. Experiments showed that the connection control protocol could establish new connections between peers and recover from path failures. Partial or full network failures were avoided when the desired node degree was set sufficiently high.

The protocol has implicit load balancing and switching of Tor circuits that can give some security against traffic correlation attacks by passive attackers controlling a subset of Tor relays or peers.

The testing method used was to experiment with a prototype on a live Tor network instead of a simulated environment. This gives more realistic data in terms

of latency and reliability of Tor relay nodes than a simulation. Although several iterations of the tests were run there is always the question of data sample size. The test network used was fairly small compared to an actual network. Also, the usage pattern of a peer-to-peer network can differ greatly depending on many factors. Some of the benefits and challenges of using this type of routing are discussed in this chapter.

5.1 Performance

The prototype used was not optimized for performance. The choice of software and testing environment made it difficult to scale up the network without hurting the measurements. Several improvements can be made programmatically to increase performance. However, it is believed that the experiments gave a sufficient indication of the validity of the choices made.

5.1.1 Convergence

Having a high convergence ratio is important for calculating correct routes. The network´s ability to stay converged was relatively good. It took only a few seconds to converge the network after a route update. This was only with a small network. With larger networks, one would expect longer convergence times and less overall convergence.

5.1.2 Path cost metric

A good link cost metric is important to get the best shortest path estimate for the route. Link latency was used in these experiments, but there are different metrics one can use to calculate this, both static and dynamic.

A coarse and easy static metric to use is the number of hops between peers. This does not take into account the latency or bandwidth of the relays between peers. To get a link cost metric for the circuit one would have the combine the views on both sides of the circuit for peer A and peer B. Peer A knows how many hops to the rendezvous point (RP) on its side of the circuit, but only peer B knows how many hops there are to the RP on its side. This would have to be combined in the link-state update sent to the network.

Bandwidth is a static metric that can be used. For Tor relays the bandwidth is announced in the Tor network consensus document. Tor relays advertise their bandwidth for which they are able to forward traffic. The *observed* bandwidth is calculated from a measurement of the highest throughput achieved over a ten-second period in the last five days. Average bandwidth and burst bandwidth can be configured by the relay operator. The advertised bandwidth is the minimum of either the observed or average bandwidth of the relay. TorFlow is used as a remote measuring tool to calculate relay performance by creating a circuit through

the relay and transferring a file of known size. The relay weight is then calculated by combining the measurement speed with the average of other relays and multiplying this ratio with the advertised bandwidth of the relay. The observed bandwidth and weights are published in the consensus document. A paper by Jansen and Johnson [38] measured the accuracy of Tor bandwidth estimates. It found significant variation in the relay´s advertised bandwidth, which indicated inaccurate estimation. Higher variation was significantly associated with relays of low capacity or who were online less frequently. If the advertised bandwidth is reported correctly it could be a good indicator of link cost and used in calculating the shortest path. This metric is easy to retrieve by querying the consensus document. However, it is not updated frequently and does not necessarily reflect the current throughput of the relay. In addition to the Tor relays, the peers in the network would have to announce their bandwidth as well to get a more precise path cost.

The geographical distribution of peers and RPs is a possible metric for link cost. A hypothesis is that geographical closeness equals lower latency. A peer would choose the route that traverses the shortest geographical distance between two peers. A host can be geo-located by its IP address. Using this has implications for an anonymous P2P network. To remain anonymous the peers would normally not disclose their geographical location, much less their IP address. The only one that is not anonymous in the circuit is the RP relay. A geographical distance can be calculated to this by a node, but not from the RP to its peer. As such, it could not choose an RP, knowing it is close to its peer or not. An exact location may not be desirable for a node to disclose, but a wider geographical location, like a region, could be an option. A node could, for instance, tell its neighbor that it resides in Europe. A peer would not choose a path going through the US or Asia to communicate with this peer.

Link latency is a dynamic metric that is calculated based on the observed latency of the circuit between two peers. It could be measured between the peers during the lifetime of the circuit. The measured latency could be sent out with the link-state update as the link cost in milliseconds, for instance. The round-trip time (RTT) should also correspond to the geographical distance as longer RTTs are expected with peers further apart geographically. The RTT of a circuit can vary significantly at different times depending on how loaded the circuit or peer is. Frequent updates of the link latency are beneficial in making routing decisions but can be detrimental to performance because of signaling noise in the network due to frequent route updates. A balance between signaling noise and freshness of link latency information is needed. A benefit of using link latency is that traffic can be distributed more dynamically through less congested routes at the time.

Sending out updates to link costs can be resource intensive both in transmission and re-calculation of graphs. A sensible middle ground in how frequent updates occur must be set. Too often and the network is swamped with updates, too seldom and link costs become outdated and less relevant.

5.1.3 Peer selection

Peer selection in the experiments is based on using the highest-cost non-neighboring peer in the network as a candidate for joining. The hypothesis is that this will, over time, reduce the average latency in the network. And also, connect distant peers closer together. The experiments showed that average path cost and length were considerably reduced by joining peers with long paths. Any further improvement in latency was not noticeable after the network had settled.

5.1.4 Number of concurrent connections

In non-anonymous peer-to-peer networks, there is little overhead to connect to other peers. In networks where relays are used to hide the identities of users, it is more costly to establish new connections. Therefore, there needs to be a balance between how many connections a peer holds with other peers and what is established dynamically per demand. This depends on the need for redundancy of links and the nature of communication between peers. Short-lived exchanges of data between two peers may accept a higher latency through multiple hops compared with longer-lived connections that exchange data frequently.

Multiple connections did have an impact on the performance in testing. Encryption and decryption of relay cells can be resource intensive for lower-spec hardware. Therefore, it is not always feasible to have a high node degree for all peers in the network. The node degree can easily be adjusted based on the client´s desire and resources.

5.1.5 Network churn

Network churn can negatively affect the routing protocol performance by frequently updating the routes, resulting in network instability and re-calculation of routes. This can be a problem in peer-to-peer networks because of the transient nature of nodes. In our peer-to-peer network, using Tor as a substrate, the reliability of the relays along the circuit can also be a factor. During testing the Tor relays generally behaved well, but there were some relays that were problematic.

The experiments used a rather crude model of churn that doesn´t necessarily reflect the dynamics of a real P2P network. Simulating a network depends on size, structure, usage, and many more factors.

5.1.6 Problems with cell digest

For each RELAY cell in Tor, there is a digest field that contains the first four bytes of the running SHA-1 digest for all data destined for that hop in the circuit. The digest is seeded with the forward digest (Df) of the sending relay which is created during the handshake when setting up the circuit. Digests are used to check that the payload data originates from that relay and has not been altered.

During the execution of the tests, there were often problems with the digest mismatch of the payload received from the relays. When there is a digest mismatch the cell is discarded and the message therefore does not get through to its destination. The cause of this was not found, but it is likely due to an implementation error and not a man-in-the-middle attack. However, this can cause serious problems with routing performance depending on the type of message the relay cell was carrying. A choice was made to disable this check in order to have the experiments go through. However, this should be remedied since it´s important for the security of the Tor relay communication and defense against man-in-the-middle attacks.

5.1.7 Relay cell size

If the cell payload size is too large, we get this error, for instance:

```
relay payload length cannot be more than 498 (502 got)
```

Therefore, the message payload must be kept below the maximum cell size. With large topologies, the application-level messaging should be able to span multiple cells or reduce cell size to within maximum.

5.1.8 Reliability

The failure rate of message delivery was largely caused by failed routes where the dead route had not yet been detected. Since there is no retry mechanism the message simply did not get forwarded. Using sending failure as a detection mechanism for dead routes would be an improvement, and lead to faster re-calculation of the topology.

5.2 Network stability and resilience

Resilience in a network is the ability of the network to provide and maintain an acceptable level of service in the face of various faults and challenges to regular operation [35]. An overlay network can either have proactive or reactive maintenance of failed nodes and links [5]. The type of maintenance tested in this thesis is reactive maintenance, where failed connections are replaced after detecting a peer failure. Detection of failures can either be probe-based or usage-based. Probe-based sends a ping, or keep-alive, to the peer. Suppose a probe times out; then the peer link is considered dead. Usage-based detects a dead connection if the acknowledgment of a sent message times out. In these experiments, probe-based failure detection was used. Probe-based will see a dead link faster, but it comes with the overhead of sending keep-alive messages. The keep-alive was also used as a latency probe to measure the delay between peers in our experiments. Without this latency probe, the detection method could also be to check for incoming traffic from the node. The problem is that all links could become dead

without the peer discovering it before it´s disconnected from the network and unable to establish new connections. Therefore, the probe-based detection model was considered most appropriate for our case.

5.2.1 Degrees of connectivity

In the experiments, a node degree of 4 and above prevented the network from collapsing due to random link failures. A higher degree of connectivity would ensure a more reliable network that is more resilient against failures. However, the overhead of maintaining many connections can become burdensome for some low-resource peers, as mentioned in section 5.1.4.

5.2.2 Failure detection and failover

An efficient and resilient network implies that failures are quickly detected and alternate routes are found. The experiments showed that the application could detect and find alternate paths within a minute of a failed route. Detection time could be reduced by decreasing the timeout value before the route is marked as dead, or reacting to failed delivery of data. However, missing keep-alive messages can be temporary and this can cause route-flapping if the value is set too low. Unstable routes also increase route updates and network re-calibration.

5.2.3 Disjoint networks

Disjoint networks have subsets of the network which have no connectivity. This can happen if the peers connecting the subsets drop out. This risk increases with increasing network diameter and lesser node degree. In our implementation, whenever a peer needs to create a new connection, the protocol selects a non-neighboring peer with the highest latency as a candidate for joining. The reasoning is that this peer also has the highest distance in the network from the connecting peer. Thereby increasing the connectivity between distant parts of the network so that the risk of disjointed networks is reduced.

5.2.4 Recovery of disconnected peers

When a peer's connections to the network drop to zero, it is essentially disconnected from the network with no way to connect. It would have to connect again using the method of bootstrapping it used initially. A solution to this could be to have *hanging rendezvous points*. When a peer bootstraps itself into the network, it connects to an established rendezvous point by whichever means the network uses to onboard new peers. When the peer leaves this RP, the peer that established this RP can re-establish it and keep it open indefinitely. If a peer gets disconnected from the network, it would know of the original rendezvous point it used and re-connect to this to get back into the network.

5.3 Scalability

The scalability of the routing protocol is essential. As seen in the experiments, the number of route updates increased the most with the increasing number of nodes. Although the experiments used only a small network of nodes, we can make some calculations and assumptions when scaling to more extensive networks. If the number of peers is n_t and the node degree of peer i is d_i, each peer sends a route update every 30 seconds to all neighbors. Then peer i will receive at least $n_t - 1$ number of route updates per minute, but also multiplied by the number of simple routes to other peers, as the route updates will be received via different routes, but a maximum of d_i. The number of route updates sent by i will equal the node degree of the peer. If one assumes a highly connected network, then the total amount of route updates received and sent by i per minute U_i equals:

$$U_i = 2((n_t - 1) * d_i) + 2d_i$$

The projected number of route update messages sent per minute by a peer is shown in figure 5.1. In addition, there are heartbeats, handshakes, and join messages. Scaling to more extensive networks may require segmentation into smaller routing areas or zones, as described in section 2.8 of Chapter 2.

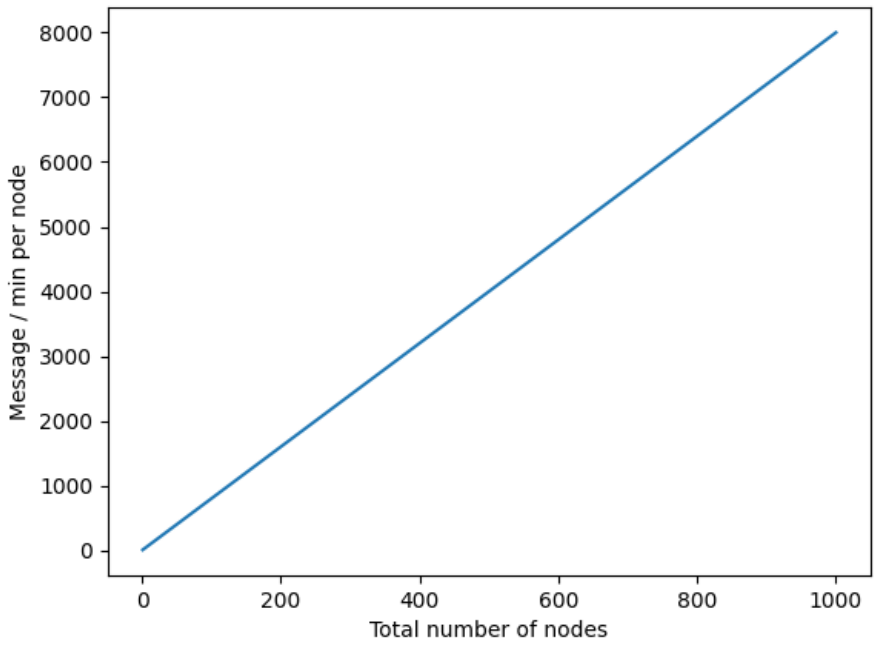


Figure 5.1: Estimation of route update messages sent and received per node depending on network size

The number of route updates can also be reduced by increasing the number

of latency measurements sampled before a new route update is sent. In the experiments, the update of latencies was rather "aggressive" to compress the run time of the tests.

5.3.1 Enhancements to routing information exchange

The way routing information is exchanged between peers can be modified to increase performance. The application exchanges routing information by sending the complete neighboring link database to the network. It could only send out an incremental update instead of a complete LSA each time there is a topology change in the network. The size of the LSA message would decrease, and the receiving peers would only need to consider the single link state data exchanges could use compression to reduce the size of the LSA sent. Especially for peers with many neighboring routes, there is a substantial potential for reducing the size of the route updates sent out.

5.4 Routing security

The routing protocol in this prototype has no specific security measures to handle the challenge of route authenticity or leakage of network topology, among others. These are some of the challenges and solutions.

5.4.1 Route security

The OSPF protocol that this protocol builds on has, by design, some security strengths [23]. Flooding the network with LSAs will ensure that malicious actors cannot prevent routing updates from reaching others as long as paths from the originator don't go through the attacker. OSPF has a *fight-back* mechanism that makes a router immediately send out a new LSA if it receives an instance of its own LSA that is newer than was last sent out. Thereby canceling out the false LSA sent out by the attacker. Also, an LSA only holds the immediate neighboring routes of a router. Therefore, an attacker must publish the LSA of many routers to affect the network topology's view significantly.

5.4.2 Leaking of network topoloy

Leaking of network topology can reveal the network identities and connections of the users to an adversary. The problem is whether this can relate to disclosing the users' real identities, e.g., IP addresses. There are anonymous routing protocols that have been suggested, but most of them are for MANET networks. Ideally, the routing updates should not reveal any other information than the next hop neighbor to use to reach a destination. Although our implementation does give away the overall topology of the network, there is no identifying information other than peer IDs. If these IDs were persistent, they could theoretically be used to

identify a user, but the assumption is that they are temporary or not constant over a long time. Using the Tor network as a substrate already gives good anonymization of peers. Also, segmenting peers into areas or zones can reduce the leaking of topology information.

5.4.3 Establishing identities

The protocol establishes identities by exchanging a handshake using HELLO messages. These messages do not check for the authenticity of the identifying party since there hasn't been enabled authentication using private keys, for instance. A future implementation should allow peers to identify themselves by some authentication mechanism. An authentication could have been implied during bootstrapping if the connecting peers used an out-of-band agreement on where to meet. The rendezvous point and cookie value would be a kind of ticket the two peers used to identify themselves.

An improvement to the security of this protocol is to have end-to-end encryption of messages that traverse multi-hop routes. For instance, when a JOIN message is sent to a peer to agree on a rendezvous point, an intermediate peer could intercept and read it. This peer could then connect to that rendezvous point and impersonate the peer for which the JOIN message was intended, given the network lacks any trusted identities.

5.4.4 Rendezvous point selection

In this implementation, the requestor of the join operation also chooses the rendezvous point (RP) to use. This is a potential security risk as the requestor could direct the other peer to use a relay it controls. With the two-hop implementation used here, the other peer connects directly to the RP, revealing its IP address. For instance, another approach where the connecting peer chooses the RP relay could be used.

5.4.5 Directional tunnels

This application uses bi-directional tunnels where request and response travel to and back via the same circuit. To prevent a malicious actor from intercepting and analyzing traffic from both sender and receiver, it could route the traffic along circuits that do not share the same relay in any direction. In terms of the application, this can be achieved using directed graphs where each edge has one direction between any two vertices, as described in section 2.6.2 of Chapter 2. The drawback is that the round-trip time for any exchange between two peers increases because the route will not be the shortest path, which would slightly modify the protocol.

5.4.6 Path switching

Path switching means periodically changing the route between two peers in the overlay or the substrate network. This implementation dynamically switches routes in the overlay based on the shortest path's latency. The paths were changed regularly in the experiments, but this depends on the network topology. More stable latencies result in more stable paths.

For the Tor substrate network, the path switching is done by changing rendezvous point relays between peers. The Tor network uses a 10-minute window for circuits [34], which is also adopted by this implementation. Switching circuits would decrease the window of opportunity for an adversary to make correlation attacks, for instance. But, it would also increase the likelihood of selecting a circuit the adversary controls.

Chapter 6

Conclusion

This thesis aimed to answer the research questions of how data could be routed efficiently in a peer-to-peer overlay network using Tor, which metric to use for this, and how stability and resilience could be ensured. These questions were answered by developing a prototype application that implemented a probe-based, link-state routing protocol adapted for the Tor network. The routing protocol used latency measurements between peers over Tor rendezvous points as a metric for calculating shortest path routes. This routing method showed a high degree of success in selecting the most efficient path in the network.

The routing protocol was combined with a connection control protocol for fast detection of dead connections and dynamically connecting peers over Tor rendezvous points. The connection control protocol would ensure that peers remained connected to the network despite link failures and churn. Latency was used as a metric for joining peers, thereby reducing the overall path length and cost in the network. The connection control protocol managed to keep the network stable when desired node degree was set sufficiently high.

The implemented routing and connection control protocol showed that data could be routed efficiently in an anonymous overlay network over Tor using a latency metric. Stability and resilience can be ensured by maintaining enough connections between peers and actively recovering dead routes when detected. However, scaling the network up can be challenging due to the number of route updates generated. A segmentation of the network into areas, or zones, could be a solution for this.

Chapter 7

Future work

Establishing trust between peers was not in the scope of this thesis. For instance, trusted peers and signed route updates can improve the protocol's security against routing attacks. A method of securing route updates and ensuring peer identities could be a topic for future research, along with testing the network against different routing attacks.

The efficiency of this protocol relies on the peers knowing the network topology. Anonymity can be preserved better by not leaking the network topology to the whole network [39]. Other routing methods only reveal the next-hop relay in the path to a peer. How this can be reconciled with shortest-path routing can be investigated.

Getting a good metric for the path cost associated with a route is essential in routing traffic efficiently. This thesis used a sampling of latency over a short period. Although latency can indicate a delay between peers, it does not necessarily indicate the throughput capacity of the link. Research can be done into making the measurements better or combined with other metrics to estimate route cost better.

The performance can likely be improved by a more intelligent selection of rendezvous points when peering. The prototype used randomly selected relays from the Tor consensus. Peers could negotiate, or select on some other criteria, a relay that lowers the latency or throughput between them. Dynamically grouping peers based on communication patterns can also increase performance.

The protocol was tested on a small network. As noted in the discussion, this method does not necessarily scale well due to signaling traffic and frequent route calculations and updates. Future research could examine how it can be scaled to more extensive networks, for example, with incremental route updates, segmentation, etc.

Work can be done to develop the application-level protocol further and combine the routing part with the data channel communication.

Bibliography

[1] Torproject, *Torspec/tor-spec.txt at main · torproject/torspec*, Mar. 2023. [Online]. Available: https://github.com/torproject/torspec/blob/main/tor-spec.txt.

[2] F. Fallang, 'Security of dark net overlay networks,' M.S. thesis, Norwegian University of Science and Technology, 2022.

[3] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek and H. Balakrishnan, 'Chord: A scalable peer-to-peer lookup service for internet applications,' *ACM SIGCOMM computer communication review*, vol. 31, no. 4, pp. 149–160, 2001.

[4] J. Galán-Jiménez and A. Gazo-Cervero, 'Overview and challenges of overlay networks: A survey,' *Int J Comput Sci Eng Surv (IJCSES)*, vol. 2, pp. 19–37, 2011.

[5] W. Galuba and S. Girdzijauskas, 'Peer to peer overlay networks: Structure, routing and maintenance,' in *Encyclopedia of Database Systems*, L. LIU and M. T. ÖZSU, Eds. Boston, MA: Springer US, 2009, pp. 2056–2061, ISBN: 978-0-387-39940-9. DOI: 10.1007/978-0-387-39940-9_1215. [Online]. Available: https://doi.org/10.1007/978-0-387-39940-9_1215.

[6] *Peer*. [Online]. Available: https://www.merriam-webster.com/dictionary/peer.

[7] C. Wang and B. Li, 'Peer-to-peer overlay networks: A survey,' *Department of Computer Science, The Hong Kong University of Science and Technology, Hong Kong*, vol. 9, 2003.

[8] R. Schollmeier, 'A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications,' in *Proceedings First International Conference on Peer-to-Peer Computing*, 2001, pp. 101–102. DOI: 10.1109/P2P.2001.990434.

[9] 'Napster: Music from every angle.' (Jul. 2022), [Online]. Available: http://www.napster.com/.

[10] M. K. Reiter and A. D. Rubin, 'Anonymous web transactions with crowds,' *Commun. ACM*, vol. 42, no. 2, pp. 32–48, Feb. 1999, ISSN: 0001-0782. DOI: 10.1145/293411.293778. [Online]. Available: https://doi.org/10.1145/293411.293778.

[11] *Udp over tor*, May 2020. [Online]. Available: https://gitlab.torproject.org/tpo/core/torspec/-/blob/main/proposals/339-udp-over-tor.md.

[12] N. Mathewson and M. Perry, 'Towards side channel analysis of datagram tor vs current tor,' torproject.org, Tech. Rep., 2018.

[13] *Graph theory*. [Online]. Available: https://www.cs.yale.edu/homes/aspnes/pinewiki/GraphTheory.html.

[14] A. S. Tanenbaum and D. Wetherall, 'Routing algorithms,' in *Computer Networks*. Pearson India Education Services Pvt, Limited, 2014.

[15] D. Leonard, Z. Yao, V. Rai and D. Loguinov, 'On lifetime-based node failure and stochastic resilience of decentralized peer-to-peer networks,' *IEEE/ACM Transactions on Networking*, vol. 15, no. 3, pp. 644–656, 2007. DOI: 10.1109/TNET.2007.893199.

[16] R. Schollmeier, I. Gruber and M. Finkenzeller, 'Routing in mobile ad-hoc and peer-to-peer networks a comparison,' in *Web Engineering and Peer-to-Peer Computing*, E. Gregori, L. Cherkasova, G. Cugola, F. Panzieri and G. P. Picco, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 172–187, ISBN: 978-3-540-45745-9.

[17] M. Al Mojamed and M. Kolberg, 'Structured peer-to-peer overlay deployment on manet: A survey,' eng, *Computer networks (Amsterdam, Netherlands : 1999)*, vol. 96, pp. 29–47, 2016, ISSN: 1389-1286.

[18] J. Xu, 'On the fundamental tradeoffs between routing table size and network diameter in peer-to-peer networks,' in *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, vol. 3, 2003, 2177–2187 vol.3. DOI: 10.1109/INFCOM.2003.1209238.

[19] S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Shenker, 'A scalable content-addressable network,' in *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, 2001, pp. 161–172.

[20] A. Rowstron and P. Druschel, 'Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems,' in *IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing*, Springer, 2001, pp. 329–350.

[21] N. Dang, S. Vu and N. Hoaison, 'Building a low-latency, proximity-aware dht-based p2p network,' vol. 0, Oct. 2009, pp. 195–200, ISBN: 978-0-7695-3846-4. DOI: 10.1109/KSE.2009.49.

[22] D. Xuan, M. Krishnamoorthy and S. Wang, 'Analyzing and enhancing the resilience of peer-to-peer systems to routing attack,' Technical Report, Tech. Rep., 2002.

[23] G. Nakibly, A. Kirshon, D. Gonikman and D. Boneh, 'Persistent ospf attacks.,' in *NDSS*, 2012.

[24] M. Herrmann and C. Grothoff, 'Privacy-implications of performance-based peer selection by onion-routers: A real-world case study using i2p,' in *Privacy Enhancing Technologies*, S. Fischer-Hübner and N. Hopper, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 155–174, ISBN: 978-3-642-22263-4.

[25] H. Beitollahi and G. Deconinck, 'Ferris wheel: A ring based onion circuit for hidden services,' *Computer Communications*, vol. 35, no. 7, pp. 829–841, 2012, ISSN: 0140-3664. DOI: https://doi.org/10.1016/j.comcom.2012.01.008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366412000096.

[26] *Protocols/brp.md · master · briar / briar-spec · gitlab*. [Online]. Available: https://code.briarproject.org/briar/briar-spec/blob/master/protocols/BRP.md.

[27] I. Clarke, O. Sandberg, B. Wiley and T. W. Hong, 'Freenet: A distributed anonymous information storage and retrieval system,' in *Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability Berkeley, CA, USA, July 25–26, 2000 Proceedings*, H. Federrath, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 46–66, ISBN: 978-3-540-44702-3. DOI: 10.1007/3-540-44702-4_4. [Online]. Available: https://doi.org/10.1007/3-540-44702-4_4.

[28] *I2p*. [Online]. Available: https://geti2p.net/en/.

[29] T. De Boer and V. Breider, 'Invisible internet project (i2p),' *System and Network Engineering*, pp. 1–16, 2019.

[30] A. Montresor, 'A robust protocol for building superpeer overlay topologies,' in *Proceedings. Fourth International Conference on Peer-to-Peer Computing, 2004. Proceedings.*, 2004, pp. 202–209. DOI: 10.1109/PTP.2004.1334948.

[31] *Nist retires sha-1 cryptographic algorithm*, Dec. 2022. [Online]. Available: https://www.nist.gov/news-events/news/2022/12/nist-retires-sha-1-cryptographic-algorithm.

[32] J.-J. Lee, H.-H. Chiou, Y.-S. Yang, Y.-C. Su and C.-L. Lei, 'A sector-based routing model over structured peer-to-peer networks,' in *Proceedings of the International Conference on Mobile Technology, Applications, and Systems*, ser. Mobility '08, Yilan, Taiwan: Association for Computing Machinery, 2008, ISBN: 9781605580890. DOI: 10.1145/1506270.1506396. [Online]. Available: https://doi.org/10.1145/1506270.1506396.

[33] W. Holmes, *Tryquiet*, 2022. [Online]. Available: https://github.com/TryQuiet/quiet.

[34] May 2023. [Online]. Available: https://support.torproject.org/about/change-paths/.

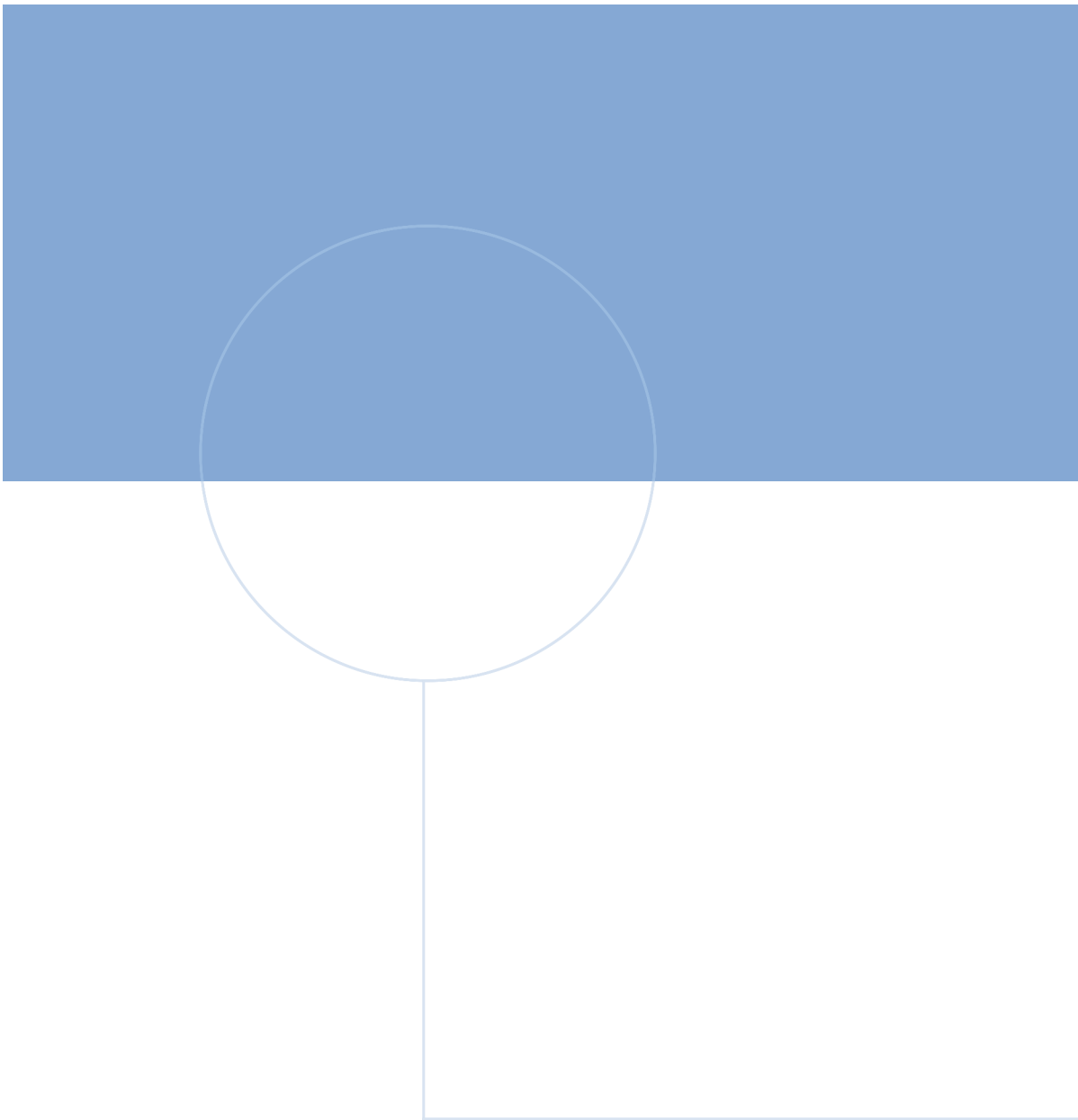
[35] D. Lummen, 'An analysis of link and node level resilience on network resilience,' B.S. thesis, University of Twente, 2020.

[36] J. T. Moy, *OSPF: anatomy of an Internet routing protocol*. Addison-Wesley Professional, 1998.

[37] J. Moy, 'Ospf version 2,' RFC Editor, STD 54, Apr. 1998, http://www.rfc-editor.org/rfc/rfc2328.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2328.txt.

[38] R. Jansen and A. Johnson, 'On the accuracy of tor bandwidth estimation,' in *International Conference on Passive and Active Network Measurement*, Springer, 2021, pp. 481–498.

[39] R. Schlegel and D. S. Wong, 'Anonymous overlay network supporting authenticated routing,' *Information Sciences*, vol. 210, pp. 99–117, 2012, ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2012.04.042. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025512003374.



NTNU

Norwegian University of
Science and Technology