Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Computer Science and Statistics

Implementing and Testing a Decentralized Web Protocol

Andrew Donegan
[redacted]
Supervisor: Professor Hitesh Tewari

April 25, 2020

A Dissertation submitted in partial fulfilment
of the requirements for the degree of
Master in Computer Science

Declaration

I hereby declare that this project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at `http://www.tcd.ie/calendar`.

I have also completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at `http://tcd-ie.libguides.com/plagiarism/ready-steady-write`.

Signed: 

Date: April 25, 2020

Abstract

Increasingly we are seeing governments around the world periodically shutting down Internet access to prevent the free flow of information. These shutdowns occur in the form of limiting access to particular websites such as social media or total blackouts of all network access in order to control citizens and the narrative around an event. This behaviour is enabled by the centralized nature of the current Internet architecture where a small number of gigantic technology firms control many of the popular applications in use today. Access to today's internet is traditionally provided by Internet Service Providers (ISPs). Governments can pressure these technology firms and ISPs to enforce restrictions on internet usage.

The Decentralized Web (DWeb) is a promising new approach to circumvent some of these issues, in which the Internet Protocol (IP) layer is replaced by a new network layer. The DWeb protocol design in this project makes use of mesh networks and Peer-to-Peer (P2P) links to connect community based routes in the network. Objects on the DWeb are indexed using blockchain technology in order to utilize a decentralized object storage method. The blockchain storage allows for easy searching and integrity checking of the data.

In this dissertation multiple network simulation tools are examined, the mesh network configuration is implemented, an initial prototype of the blockchain storage solution was created. However, combining the mesh network configuration with the blockchain solution was not successful. A network emulation configuration was successfully created using the Carrier Sense Multiple Access / Collision Detection (CSMA) protocol to connect the nodes in the network, instead of the mesh protocol, alongside blockchain storage and retrieval using Multichain streams.

The research that started with this project will certainly continue given the even stronger importance of physically secure and uncensored internet access in the period of the COVID-19 pandemic.

Acknowledgements

I would like to show my appreciation to my supervisor, Professor Hitesh Tewari, for his initial idea, the projects design and his guidance throughout this project.

I would like to thank Dr. Raman Singh for his help with this project.

I would also like to thank my parents for their support throughout my five years at TCD and my entire education. Thank you so much for the opportunities you have afforded me in my education and life.

Finally, college would not have been the gratifying experience it was without the support of my friends and girlfriend. Thank you for all the laughter and encouragement during these five years.

Contents

List of Figures

Acronyms

API Application Programming Interface.

CA Certificate Authority.

CCN Content-Centric Networking.

CLI Command-line interface.

CS Content Store.

CSMA Carrier Sense Multiple Access / Collision Detection.

DDoS Distributed Denial of Service.

DNS Domain Name System.

DoS Denial of Service.

DWeb Decentralized Web.

FIB Forwarding Information Base.

IoT Internet of Things.

IP Internet Protocol.

ISP Internet Service Provider.

LAN Local Area Network.

MAC Media Access Control.

NACK Negative Acknowledgement.

NDE ns-3 Docker Emulator.

NDN Named Data Networking.

NDNS Named Data Networking Domain Name System.

ndnSIM Named Data Networking Simulator.

ns-3 Network Simulator 3.

OID Object Identifier.

OS Operating System.

P2P Peer-to-Peer.

PIT Pending Interest Table.

PMK Pairwise Master Key.

QoS Quality of Service.

SAE Simulation Authentication of Equals.

UDP User Datagram Protocol.

VM Virtual Machine.

VPN Virtual Private Network.

WLAN Wireless Local Area Network.

WMNs Wireless Mesh Networks.

WWW World Wide Web.

1 Introduction

1.1 Background and Motivation

Internet access is essential for many people in today's world, however, the current centralized nature of the World Wide Web (WWW) can lead to impaired access. Governments or rogue players can censor the web, use the web as a surveillance tool or can control access through partial or complete shutdowns. Increasingly we are seeing governments around the world periodically shutting down access to the Internet to prevent the free flow of information. Limiting or completely blocking access to data networks is one tool governments can use to control both citizens and the narrative around an event. Between July 1, 2015 through June 30, 2016 there were 81 internet disruptions in 19 countries. "This includes 22 in India, 22 in Iraq, 8 in the non-ISIS-controlled parts of Syria, 6 in Pakistan, 3 in Turkey, and 2 each in Bangladesh, Brazil, North Korea, Republic of the Congo, Uganda, and Vietnam, among other places." "Between July 1, 2015 and June 30, 2016, [internet] shutdowns cost at least US$2.4 billion in GDP globally" [1]. Between the 15th and 23rd of November 2019, a total internet shutdown was introduced in Iran as an attempt to suppress fuel protests [2]. As seen in figure 1.1, Iranian connectivity went as low as 5% during this shutdown. Information could not be sent to the outside world from Iran and this restriction on internet access made it difficult to monitor human rights violations and coverage of the state of affairs as they happened. Having a centralized internet infrastructure that uses ISPs to provide internet access gives governments the power to restrict internet activity and this power should be curbed [2].

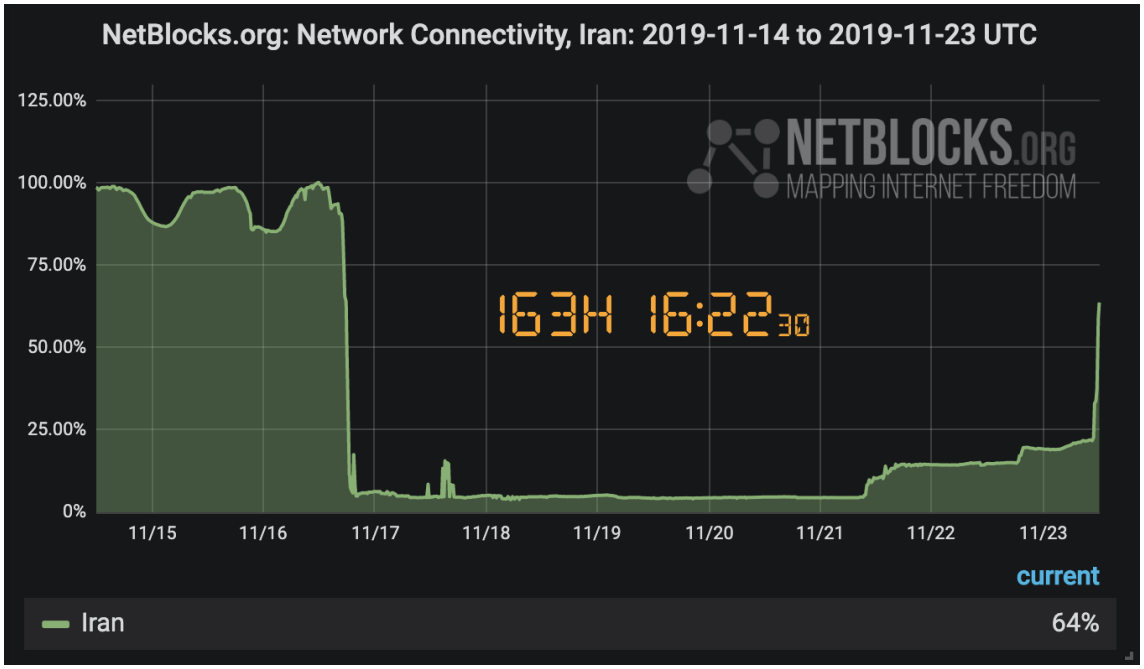

Figure 1.1: Internet connectivity in Iran between 15th - 23rd November

The Decentralized Web (DWeb) makes censorship more difficult as there are no single points of failure that can be targeted, such as ISPs. There is a large amount of research in the area of mesh networks combined with the blockchain, some examples are SmartMesh and RightMesh. These projects use mesh networks to provide connectivity in areas of low infrastructure, but the blockchain is used for buying or selling network access instead of using blockchain for decentralized object storage. These projects give users access without the need for external infrastructure. However, these still provide a single point of failure, the mobile applications providing access can be shut down by governments and therefore restrict user's access to these networks. Professor Tim Berners-Lee, inventor of the WWW, is leading a new web decentralization project called Solid[1]. Solid "realizes the web as originally envisioned and provides a platform for the next generation of truly empowering and innovative applications" [3]. The creation of this project by the inventor of the original WWW shows the need for a new, decentralized web.

The motivation behind this project is to create a protocol that prevents physical attacks and shutdowns of network infrastructure, providing increased physical security to networks.

[1] `https://solid.inrupt.com/`

1.2 Research Aim and Objectives

This project's aim is to implement and test a protocol for the DWeb that has no single entity in charge. It is designed to use mesh networks to provide Peer-to-Peer (P2P) connection infrastructure through DWeb routers and the blockchain for decentralized, immutable static object storage. To achieve this aim multiple objectives were set at the start of the project:

- Examine and understand newly proposed internet protocols to compare against, such as NDN.
- Examine and select a suitable network simulation tool to implement the DWeb protocol.
- Implement the existing IEEE 802.11s mesh network protocol [4] in the chosen network simulator.
- Use an existing blockchain framework to implement our desired object storage behaviour.
- Connect our mesh network simulation to our blockchain framework to create the desired DWeb architecture.
- Test the simulated protocol with different network behaviours such as different topologies, nodes joining and leaving the network and the number of nodes in the network.
- Compare this project's protocol against NDN, highlighting aspects such as packet loss and round trip times.

1.3 Dissertation Layout

Chapter 2 will discuss the core concepts and background information necessary to understand this project and look at state-of-the-art projects aimed at using mesh networks with the blockchain. Chapter 3 will examine the high-level design of this project. Chapter 4 will discuss the implementation of the project's design and will evaluate the outcomes of the implementations carried out. Chapter 5 concludes this dissertation and suggests future work that could be carried out following this research.

2 Background

2.1 Decentralized Web

The current Internet's infrastructure relies on the client-server model, where one server serves many clients. This centralized architecture leads to the existence of a single point of failure that is vulnerable to censorship or denial of service. If the server stops working, the system and clients can be affected. The Decentralized Web (DWeb) is a version of the web that uses Peer-to-Peer (P2P) communication to prevent the existence of a single point of failure. In P2P networks, the loss of any one node does not disrupt the network as a whole. Users can keep control of their data and communicate directly to avoid using a central server as a middleman for communication. The DWeb "is a vision of the next generation internet as a peer to peer network built around blockchain technology, where users own their own data, data is portable, computing and storage resources are provided by end-users within distributed networks, apps run locally on end-user devices and platforms are decentralized and autonomous" [5]. By removing the centralized servers, government censorship or shutdowns become much more difficult as data can be served from many places instead of one central source. "Today, the Web we use is not private, secure, reliable or free from censorship. It lacks a memory, a way to preserve our digital record through time. By distributing data, processing and hosting across millions of computers worldwide with no centralized control, a new Decentralized Web has the potential to be open, empowering users around the globe to control and protect their own personal data better than before" [6].

2.2 Wireless Mesh Networks

Wireless Mesh Networks (WMNs) are decentralized in design and operation, with no central point of management or failure and "are dynamically self-organized and self-configured, with the nodes in the network automatically establishing an ad hoc network and maintaining the mesh connectivity" [7]. Given WMNs decentralized design, they are

a good candidate for creating a decentralized web protocol that eliminates the need for an ISP and therefore hopefully eliminates the threat of a government shutdown of the internet. WMNs are made up of two types of nodes, routers and clients, mesh routers have additional routing functions. WMNs can achieve similar coverage to traditional network protocols while using lower transmission power by using multi-hop communications. Multi-hop routing is when a "network coverage area is often much larger then radio range of single node(s), so in order to reach some destination[s] [a] node can use other nodes as relays" [8]. Mesh routers are "usually equipped with multiple wireless interfaces built on either the same or different wireless access technologies" [7], this is to improve the flexibility of mesh networking.

2.2.1 Architecture

WMNs can be separated into three architecture types; Infrastructure WMNs, Client WMNs and Hybrid WMNs, as explained by Akyildiz and Xudong Wang [7].

Infrastructure WMNs, as shown in 2.1, can use several radio technologies to be built alongside the popular IEEE 802.11 frameworks. In figure 2.1, the dashed and solid lines indicate wireless and wired links, respectively [7]. A mesh network is used as a backbone for clients and other systems to connect to the internet. This approach allows for WMNs to be integrated with existing infrastructure through gateway functionality to allow for easier transitions from current internet infrastructure to a decentralized one.

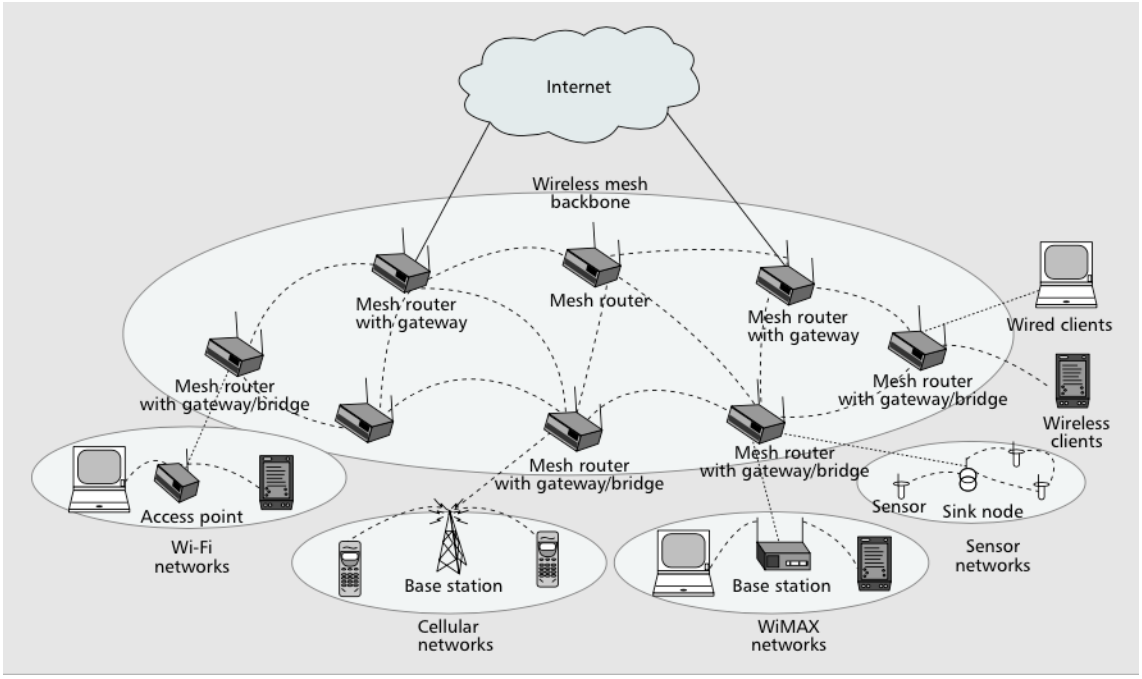

Figure 2.1: Infrastructure WMNs

Client WMNs, as seen in the bottom section of figure 2.2, allow for peer-to-peer networks between similar devices. Client nodes are the actual network and they "perform routing and configuration functionalities as well as providing end-user applications to customers" [7]. A mesh router is not needed for these networks.

Hybrid WMNs, as seen in figure 2.2, is a combination of the previous two meshing types. Mesh routers are not needed but can be used and current infrastructure connectivity is available as well as inside WMNs. "While the infra-structure provides connectivity to other networks such as the Internet, Wi-Fi, WiMAX, cellular, and sensor networks, the routing capabilities of clients provide improved connectivity and coverage inside WMNs" [7].

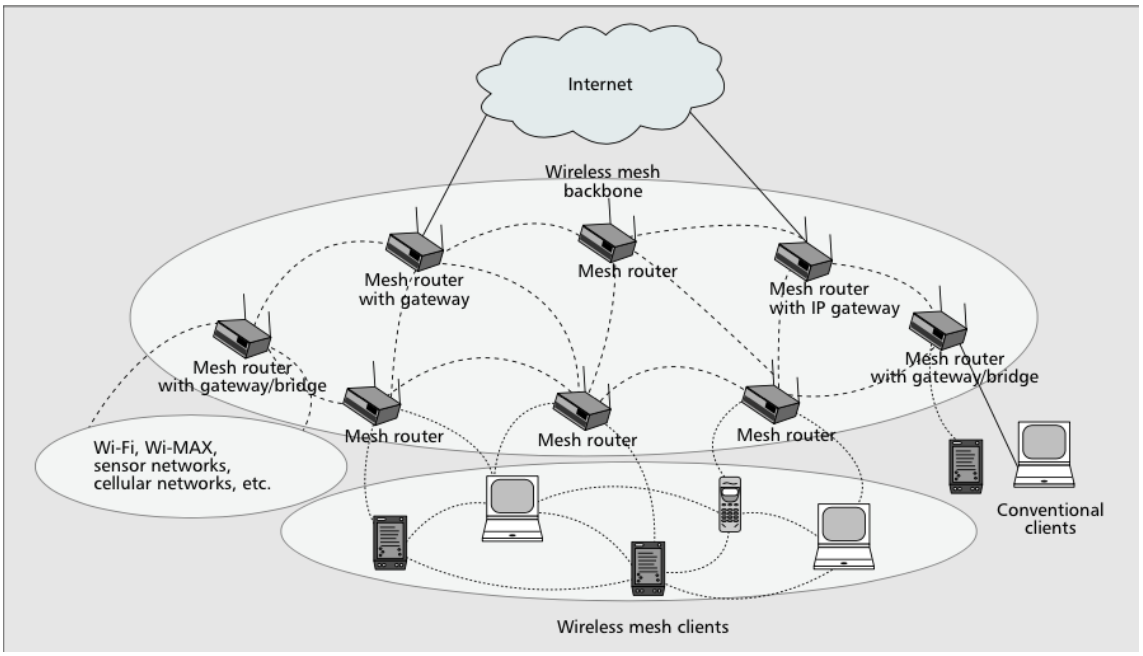

Figure 2.2: Hybrid WMNs

2.2.2 Ad hoc networks

Ad hoc networks do not rely on predefined infrastructure, for example wireless routers. "Individual nodes are responsible for dynamically discovering which other nodes they can directly communicate with" [9]. Nodes can send messages for other nodes that cannot communicate directly with each other. "Ad hoc networks are suited for use in situations where infrastructure is either not available, not trusted, or should not be relied on in times of emergency" [9].

2.3 Blockchain

In conjunction with WMNs, this project aims to use the blockchain as an immutable record of web objects where content can be uploaded to and retrieved from the record. The blockchain technology was first proposed by Nakamoto in 2008 when he published his bitcoin white paper [10]. Nakamoto described a solution to the double spending problem in the form of "an ongoing proof-of-work" [10]. The "blockchain serves as an immutable ledger which allows transactions take place in a decentralized manner" [11].

The blockchain, as seen in 2.3, is made up of a continuous sequence of blocks (records) that contain a header and body [11]. The block headers seen in 2.4, contains "block version: indicates which set of block validation rules to follow", "Merkle tree root hash: the hash value of all the transactions in the block", a "Timestamp", "nBits: target threshold of a valid block hash", "Nonce: a 4-byte field, which usually starts with 0 and increases for every hash calculation" and a "Parent block hash: a 256-bit hash value that points to the previous block" [11]. The block body contains a transaction counter and the transactions. In our use case these will store data uploaded by a content producer instead of transactions. Nodes can join and leave the blockchain at any time without disruption and accept an updated copy of the ledger when they rejoin [10].

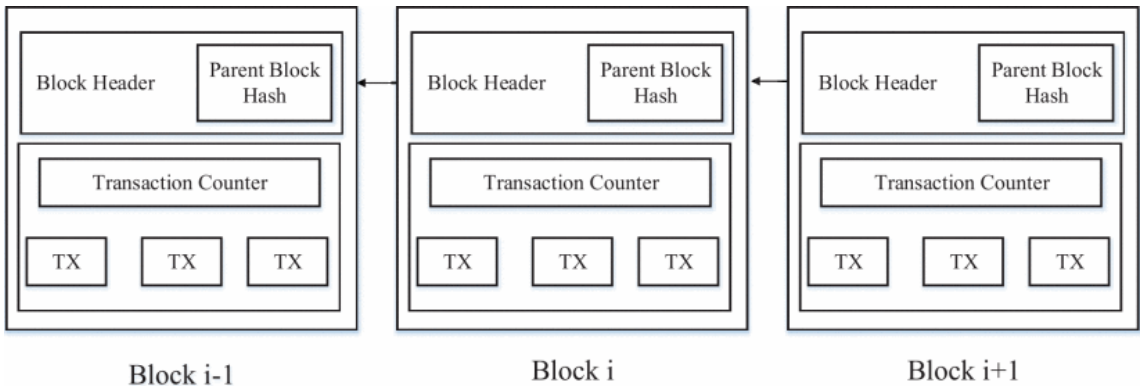

Figure 2.3: Blockchain Architecture

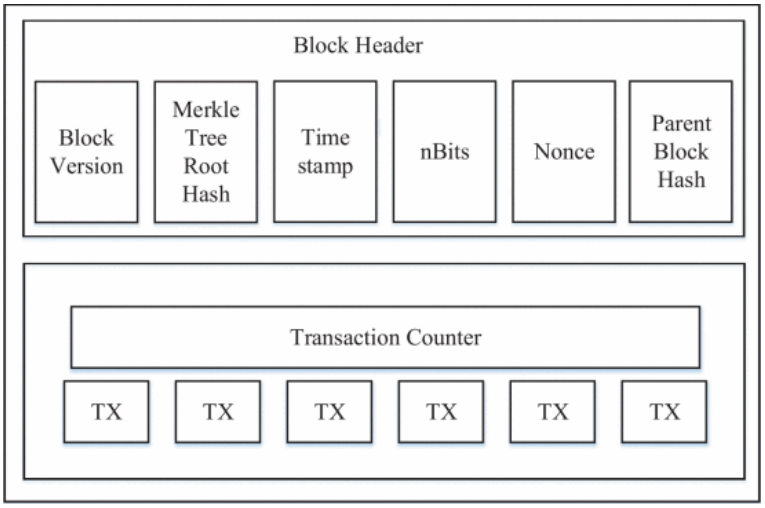

Figure 2.4: The Structure of a Block

One main advantage of using blockchain for content storage is its decentralized nature where no central trusted entity is needed. The blockchain is a trust-less system that avoids the need to have a central authority and therefore a central point of failure.

2.3.1 Transactions

Blockchain transactions are records of a user spending a coin and prevents double spending by being aware of all transactions in the chain, each transaction is publicly announced and this record is stored on the blockchain.

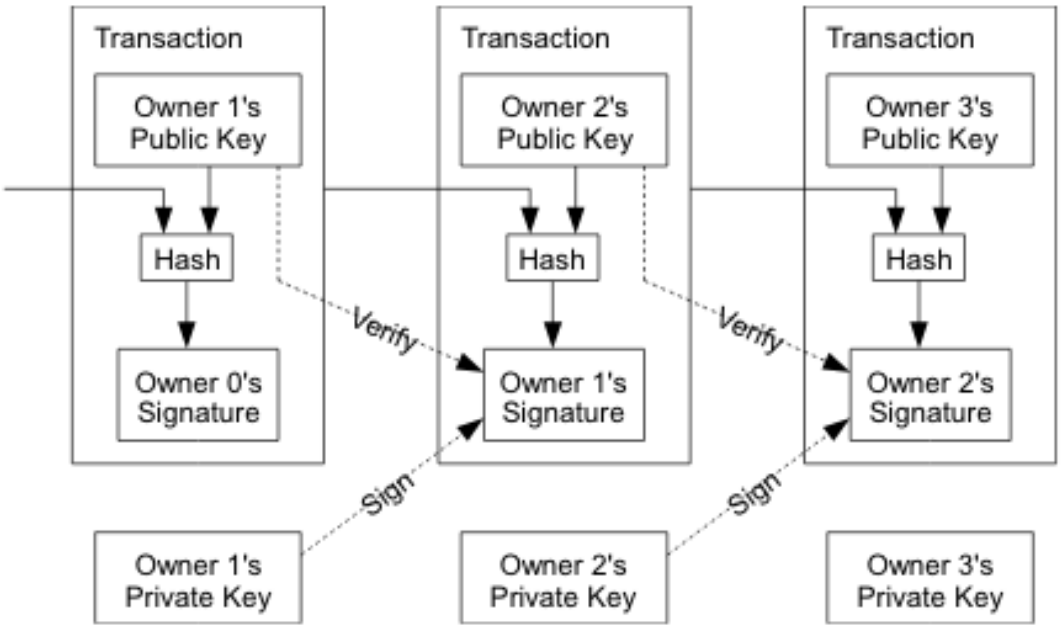

Figure 2.5: Blockchain Transactions

As seen in figure 2.5, each owner transfers the coin to the next by digitally signing a hash of the previous transaction and the public key of the next owner and adding these to the end of the coin [10]. Each transaction consists of a hash of the owners public key and the previous transaction, which is then signed by the previous owners private key. By including the previous transaction in the next hash, this makes the chain immutable. If someone attempts to alter a previous transaction in the chain it will have a knock on effect and cause the hash calculations of the next blocks to be incorrect.

2.4 Named Data Networking

Named Data Networking (NDN) is the framework against which we are comparing our decentralized web framework. "NDN is one of five projects funded by the U.S. National Science Foundation under its Future Internet Architecture Program. NDN has its roots in an earlier project, Content-Centric Networking (CCN), which Van Jacobson first publicly presented in 2006" [12]. NDN seeks to create an internet protocol to replace IP. "While IP has exceeded all expectations for facilitating ubiquitous inter connectivity, it was designed for conversations between communications endpoints but is overwhelmingly used for content distribution. Just as the telephone system would be a poor vehicle for the broadcast content distribution done by TV and radio, the Internet architecture is a poor match to its primary use today" [13]. NDN aims to remove IP datagrams containing address endpoints and replace them with a more general structure that can name chunks of data in a hierarchical manner [13]. "NDN is based on named content, which has no notion of host at its lowest level and no source and destination addresses. In NDN, because of this key feature, each content packet has a unique name, and it is forwarded by a lookup on its name" [14].

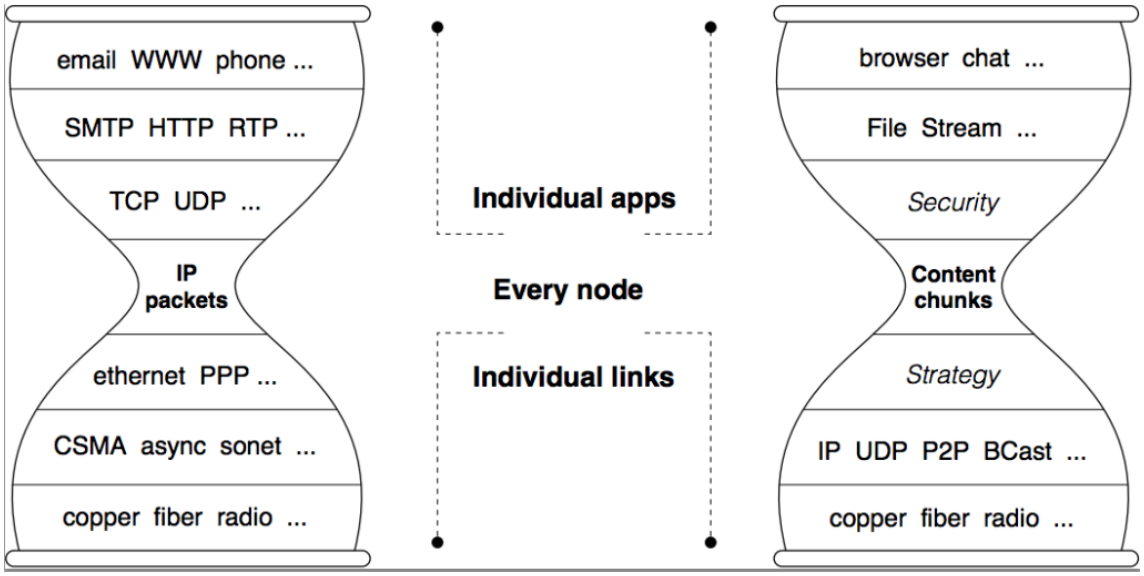

Figure 2.6: Internet and NDN Hourglass Architectures

The current Internet's thin waist has allowed for explosive growth in the layers surrounding the IP layer. As seen in figure 2.6, NDN retains the Internet's thin waist and replace IP packets with Content chunks [15]. NDN aims to keep this structure as it "evolves the thin waist to allow the creation of completely general distribution networks." "This conceptually simple change allows NDN networks to use almost all of the Internet's well understood and well tested engineering properties to efficiently solve not only communication problems but also digital distribution and control problems" [15]. Continuing to use the Internet's current structure that surrounds IP should make transitioning to NDN easier.

2.4.1 Architecture

Using NDN, a consumer can request content using interest packets containing the name of the desired data. The user then receives content through data packets containing the name of the content and the actual content itself that the user requested. You can see the full contents of the packets in figure 2.7. "A consumer puts the name of a desired piece of data into an Interest packet and sends it to the network. Routers use this name to forward the Interest toward the data producer(s). Once the Interest reaches a node that has the requested data, the node will return a Data packet that contains both the name and the content, together with a signature by the producer's key which binds the two" [12].

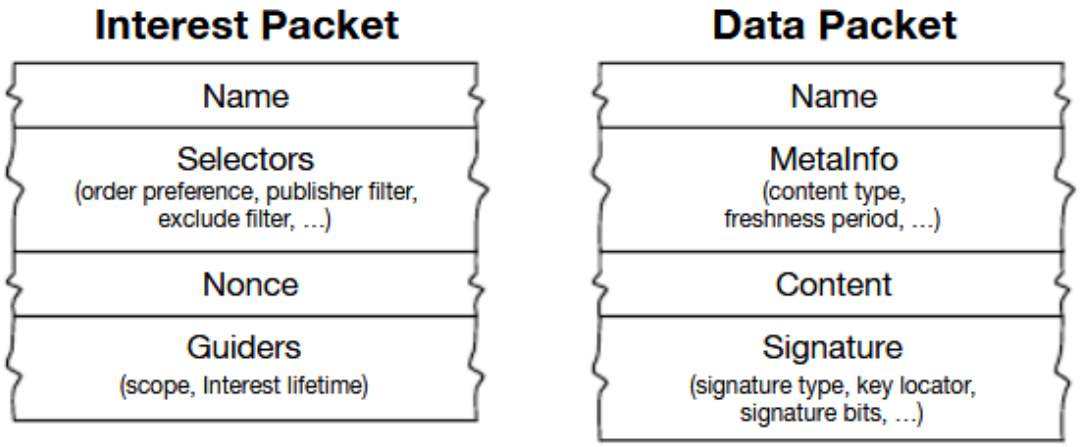

Figure 2.7: NDN Packet Architectures

NDN routers each have a Pending Interest Table (PIT), Forwarding Information Base (FIB) and a Content Store (CS). "The PIT stores all the Interests that a router has forwarded but not satisfied yet" [12]. "A router remembers the interface from which the request comes in, and then forwards the Interest packet by looking up the name in its FIB, which is populated by a name-based routing protocol" [13]. The router will cache the data in the CS once it has been sent to a consumer and can serve the data from the CS directly when another interest packet is received for the same data. It will also remove the corresponding PIT entry once the interest has been satisfied. An example of the forwarding process can be seen in figure 2.8.

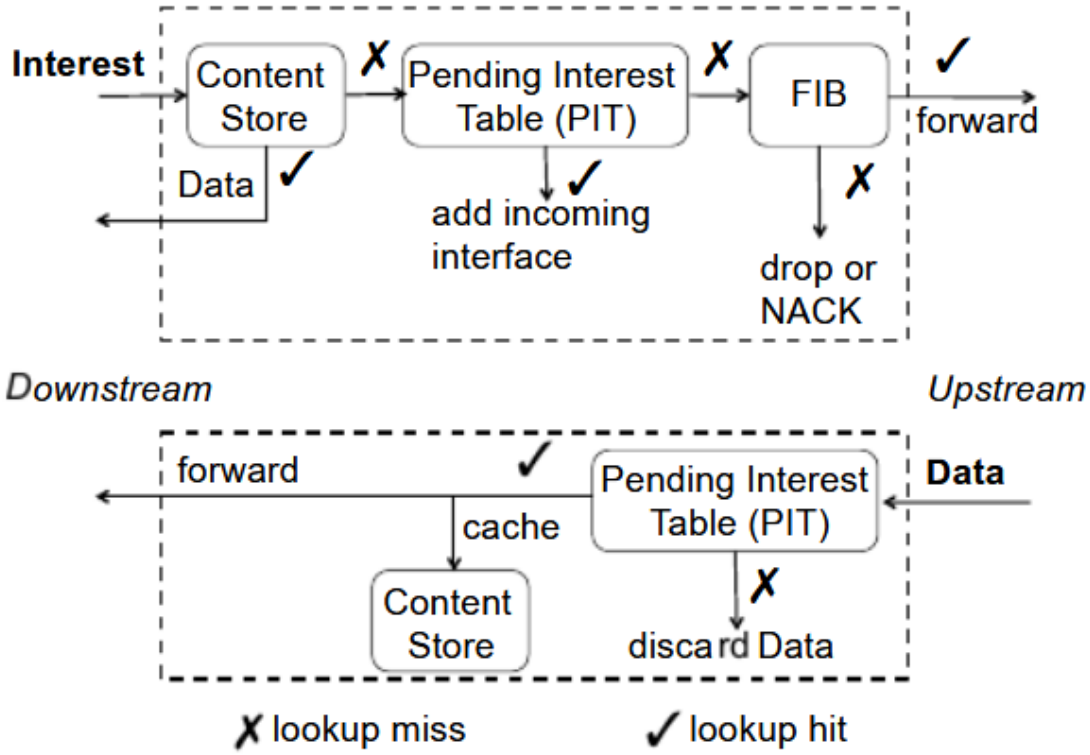

Figure 2.8: NDN Forwarding Process

As seen in figure 2.8, "when an Interest packet arrives, an NDN router first checks the Content Store for matching data; if it exists the router returns the Data packet on the interface from which the Interest came. Otherwise, the router looks up the name in its PIT, and if a matching entry exists, it simply records the incoming interface of this Interest in the PIT entry. In the absence of a matching PIT entry, the router will forward the Interest toward the data producer(s) based on information in the FIB as well as the router's adaptive Forwarding Strategy. When a router receives Interests for the same name from multiple downstream nodes, it forwards only the first one upstream toward the data producer(s). The FIB itself is populated by a name-prefix based routing protocol, and can have multiple output interfaces for each prefix" [12].

Names

Names are opaque to the network, routers do not associate any meaning with names. Applications can choose their own naming schemes. NDN does assume a hierarchical structure is used for naming, "e.g., a video produced by UCLA may have the name /ucla/videos/demo.mpg, where '/' delineates name components in text representations, similar to URLs" [12].

Security

Data in NDN cannot opt out of in-built security. This gives users a source of trust without needing to know the source of the data. Keys can be communicated as NDN data, making distribution simpler. This data-centric approach allows for applications to control data access using encryption and prevent tampering [13].

Routing

Routing is based on names, a NDN router announces name prefixes for the data it will serve. Propagating the announcements informs each router of its own FIB. "Based on information in the FIB and performance measurements, an adaptive forwarding strategy module in each router makes informed decisions about: which Interests to forward to which interfaces, how many unsatisfied Interests to allow in the PIT, the relative priority of different Interests, load-balancing Interest forwarding among multiple interfaces, and choosing alternative paths to avoid detected failures" [12]. If an interest cannot be satisfied, a NACK is sent back and the requesting router sends their interest elsewhere. "NDN inherently supports multipath routing. In NDN, Interests cannot loop persistently, since the name plus a random nonce can effectively identify duplicates to discard. Data does not loop since they take the reverse path of Interests" [13]. Interests only receive one data packet so traffic load can be controlled by setting the allowed number of pending interests [12]. The CS is used as a cache to store data packets to satisfy future requests for the same data in a more efficient manner than retrieving it from its producer.

2.4.2 NDNS: A DNS-Like Name Service for NDN

A Domain Name System (DNS) is a hierarchical and distributed look-up service to retrieve information given a name, such as IP addresses or cryptographic keys. The primary goal of standard DNS "is a consistent name space which will be used for referring to resources. In order to avoid the problems caused by ad hoc encodings, names should not be required to contain network identifiers, addresses, routes, or similar information as part of the name" [16]. DNS namespaces are split into zones, which are responsible for hosting resource records sets. Resource records are units of information in DNS zone files.

The NDN project had determined the need for an always-on lookup service "to support NDN routing scalability, to provide rendezvous for mobile data producers, and to provide persistent storage of critical data, such as public key certificates" [17]. NDN needs a lookup service because of routing scalability as the FIB size is limited and would not scale well. This is also needed for mobile publishing, as it is difficult to steer data towards

moving producers. Lastly, certificate provisioning as the data producer may not always be online when they need to be verified and the lookup service would provide public key certificate access for a consumer to verify data integrity.

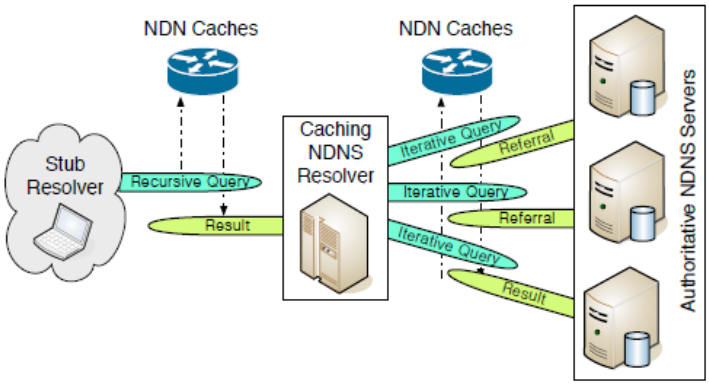

Figure 2.9: NDNS Overview

"Queries carry only the names to be queried, and 'NDN caches' represent caches at any network forwarders. Queries are forwarded based on their names, and can bring back matching answers from any intermediate caches." "By running on top of NDN, NDNS takes advantages of all built-in NDN features, including (1) efficient query Interest forwarding taking into account server and network availability information at network layer; (2) aggregation of same queries and multicasting query results; (3) in-network caching of query results; and (4) built-in authentication for query results carried in NDN data packets" [17].

2.4.3 Disadvantages of NDN

NDN has been in development since 2010 and is aiming to be the protocol used in the future, however, there are some disadvantages to the NDN protocol.

$$Overhead = \frac{\sum \text{bytes transmitted}}{\sum \text{bytes of content received}} - 1 \tag{1}$$

The NDN project aims to be compatible with today's infrastructure, "NDN can run over anything, including IP, and anything can run over NDN, including IP" [13]. However, this includes a large overhead, which is calculated as the ratio of total bytes transmitted to content bytes received, as shown in the equation above. "In CCNx, [CCNx is a software prototype which implements NDN], the default of a packet contains a maximum payload of 4096 bytes, a header of approximate 550 bytes and the interest packet segment with sizes from approximate 150 to 250 bytes. Thus, transmitting a NDN packet needs four

Ethernet frames, causing the total overhead to be about 23.6%. Besides, it's considered that the overhead is caused by retransmission of lost packets which is more expensive in NDN, due to the large 4096 bytes chunk payload" [14]. Another problem with NDN is that "HTTP is 10 times faster than NDN, because decoding and restructuring data and performing lookup of chunk names increase the computational overhead, as detailed in [18]" [14]. Also "NDN route tables are much larger than IPs', because names are longer and more numerous, which have multiple components similar with URLs resulting in inefficient lookup" [14]. The NDN protocol also uses the NDNS lookup service as the FIB has a limited size. Using a lookup service like this provides a single point of failure or attack, which is a solution this projects DWeb protocol aims to solve. "NDN, being an unstructured-flooding based routing protocol, neither guarantees content discovery nor ensures scalable routing table size and manageable update message overhead" [19].

2.5 State-of-the-Art

In this section, similar projects which have attempted to combine the Blockchain with Mesh networks but in different ways than this project's design will be examined.

2.5.1 RightMesh

RightMesh is "a Decentralized Mobile Mesh Networking Platform Powered by Blockchain Technology and Tokenization" [20]. RightMesh's goal, which is similar to other projects of this nature, is to use mesh networks to connect people that live in areas with poor infrastructure to their version of the internet. "With RightMesh, the devices people already carry around everyday form the infrastructure." Since RightMesh's network is a mesh network it is "self-forming, self-healing, and self-regulating, using whatever it has at its disposal" [20]. Users of RightMesh can utilize the RightMesh P2P network for connectivity between users without the need for the internet and can buy and sell data using transactions on the RightMesh blockchain.

RightMesh's mesh network is formed by connecting android phones and IoT devices together using Bluetooth and Wi-Fi. Data can be sent from one device to another with single-hop communications or by multi-hop transfers over many devices until the data reaches its destination [21].

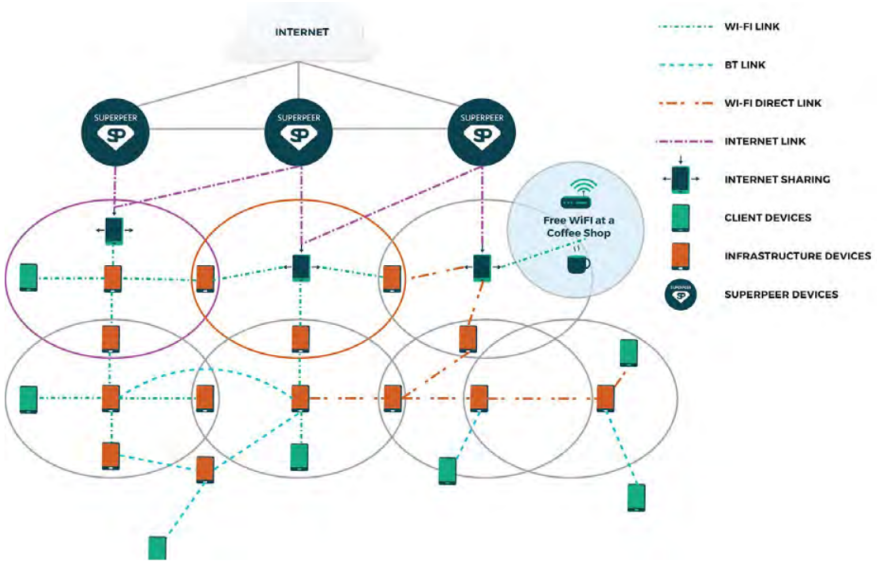

Figure 2.10: RightMesh Ecosystem

In figure 2.10, Superpeer nodes are run by RightMesh and others that have reliable connections. They act as proxies between internet traffic and RightMesh data requests. Internet sharing nodes provide access to the rest of the mesh for a price. Client are regular participants in the network that consume by purchasing data. Infrastructure devices do not share internet directly but forward packets on behalf of others [20].

RightMesh differs from this project as it plans to use the blockchain for object storage compared to a payment network to buy or sell data. RightMesh has a "seller node" network node that "sells internet data (or any other future mobile resource such as collated data, storage, processing, etc) at a rate determined by the seller." It also has an "Intermediary node" that "acts as a relay node in the network and contributes its device resources to the transmitting data across the mesh network" [20]. Having these types of nodes that provide network access in exchange for RMESH tokens, RightMesh's cryptocurrency token, can diminish the point of a decentralized mesh network that provides access to all nodes in the network. These nodes give a government or other organisation a target to shutdown network access. Another potential issue with this approach is a company or group could then become the main suppliers of RightMesh internet data and access and then a mesh network version of an ISP is created by supplying the majority of internet access to a certain region. If a monopoly of this kind appears then the group in control could begin to raise the price of RightMesh network access as the "rate [is] determined by the seller" [20].

2.5.2 SmartMesh

The SmartMesh project is similar to RightMesh, using a SmartMesh mesh network to connect users without infrastructure and a cryptocurrency token to incentivize the sharing of mesh network access. "SmartMesh is a blockchain-based IoT underlying protocol that enables smartphones, on board devices and other devices to connect to each other without the internet" [22]. The SmartMesh network is a "pay-per-forward" network, with no required infrastructure where users connect through phones. A user pays their neighbouring node to send out packets, carrier nodes are mobiles that forward data packets for other nodes and internet nodes have internet access and can offer this access to others [22].

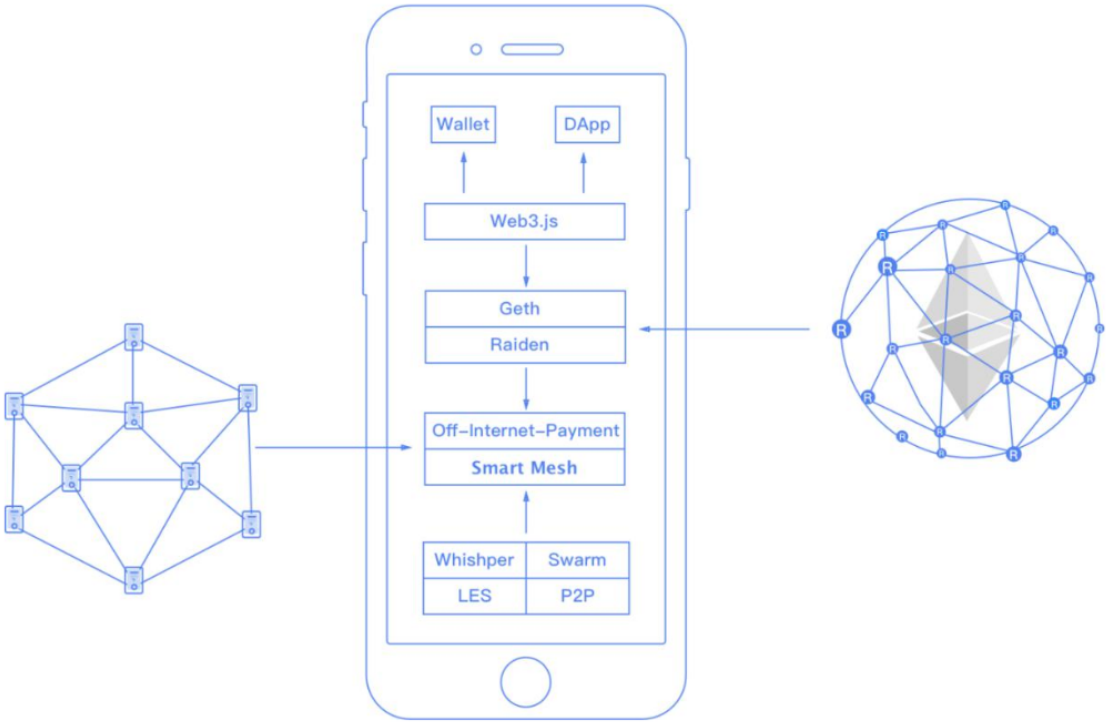

Figure 2.11: SmartMesh Technology Stack

In figure 2.11, Off-internet payments occur in the SmartMesh network. The SmartMesh architecture is made up of Whisper, Swarm, Light Ethereum Sub-protocol (LES) and P2P connections [22].

SmartMesh embeds the Ethereum light node, which is based on the LES, which is a protocol designed for a light client and only downloads the block header instead of the entire block when the blockchain is synchronized. The decentralized application interacts with the contract layer through Web3.js [23]. The wallet is where the user stores their private key, makes transfers, queries and other operations. Swarm hosts personal files in

the share storage space and Whisper is used to build a peer-to-peer messaging network. The SmartMesh token is based on Ethereum to incentivize users to purchase products and services in the SmartMesh ecosystem [22].

Similar to RightMesh, SmartMesh differs from this DWeb project as it uses the blockchain as a transaction system for users buying and selling network access instead of using it for decentralized object storage.

2.5.3 Skycoin

Skycoin is a blockchain ecosystem that provides several cryptocurrency based services. Skywire is one of these services and "aims to create an incentivized mesh network that is faster, more affordable, more accessible, and offers higher QoS than the current Internet" [24]. At time of writing this dissertation, the whitepaper specifically for Skywire is not available so more detailed analysis cannot be provided. Figure 2.12 shows the mesh configuration used by Skywire [24]. The Skywire network consists of a communication protocol "that avoids the limitations of Transmission Control Protocol/ Internet Protocol (TCP/IP) communication protocols." It also consists of a payment system to compensate users providing resources to the network, a hardware platform to provide networking, storage and computing power and an application ecosystem to help adoption. The Skywire communication protocol uses Multi-protocol Label Switching where routes through the network are determined prior to sending traffic [24].

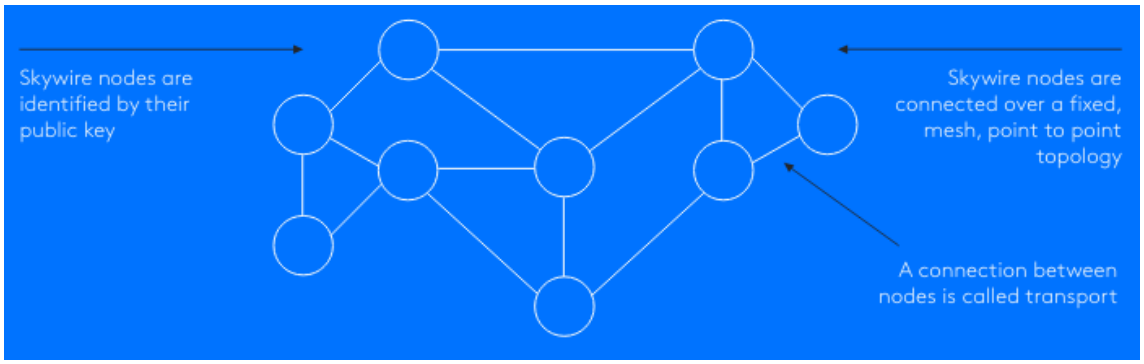

Figure 2.12: Skywire Mesh Configuration

Similar to RightMesh and SmartMesh, Skywire uses an incentive scheme for users to provide resources to the network where "every node effectively acts as a micro-ISP capable of automatic metering, billing, and settlement" [24].

2.5.4 Althea

Althea's motivation is also to stop the use of traditional ISPs by creating "decentralized ISPs" in communities by using Althea routers that pay each other for bandwidth. The Althea system firmware runs a routing protocol that automates network configuration and handles router payments for bandwidth. "Every Althea network connects to the Internet through a backhaul - a commercial-grade subscription to the Internet backbone" "Users are also incentivized to grow the network by connecting their neighbors to the Althea network" [25].

Althea also follows model similar to RightMesh, SmartMesh and Skycoin of using cryptocurrency as payment for bandwidth, where "nodes only pay neighbors for forwarding packets" [26]. User nodes are installed by people that want to purchase internet access on Althea. Relay nodes are installed by people that want to earn money by forwarding internet traffic. Gateway nodes provide a backbone connection from Althea's physical layer to the outside internet and exit nodes are connected to gateway nodes over VPN tunnels and perform network address translation [26]. An overview of this configuration can be seen in figure 2.13.

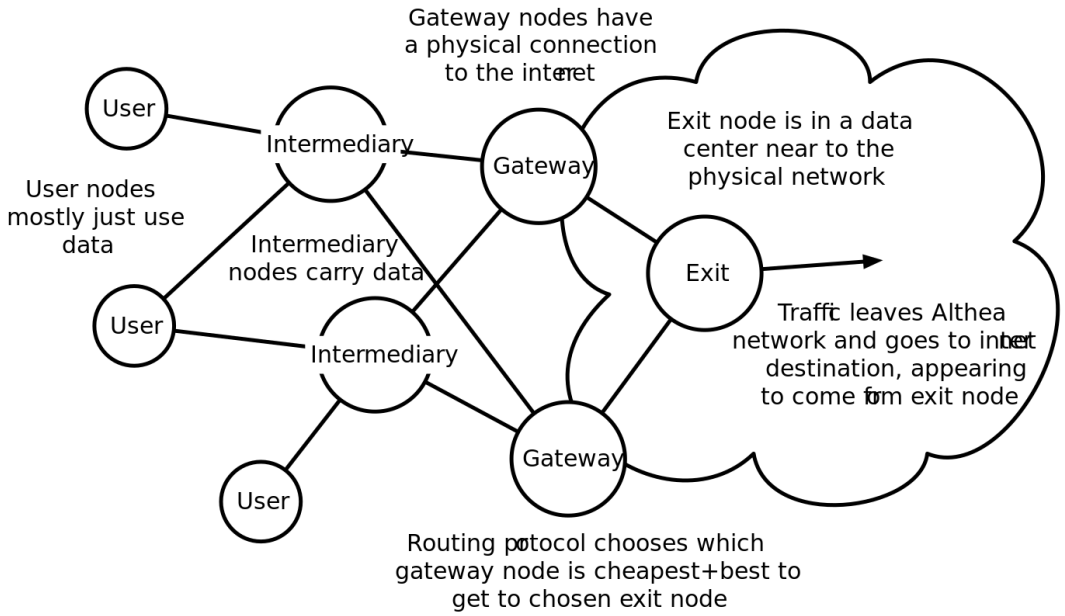

Figure 2.13: Althea Network Topology

Althea uses a proof of stake blockchain built on the Cosmos platform and is maintained by validators who validate the blockchain. This power is given by holders of the Althea governance tokens, who delegate tokens to validator(s) of their choice. Validators and those delegating Althea governance tokens to them earn transaction fees from each payment [26].

In [26] there is no mention of a decentralized storage solution and the Althea network appears to rely on connecting to a traditional ISP through gateway nodes. This design for a decentralized network still allows for the existence of single points of failure such as a centralized server for storage or ISPs that governments can use to censor or block internet access.

2.5.5 BlockMesh

Another project combining mesh networks and the blockchain is BlockMesh, which aims to "enable cost-free communication by creating a global mesh network which allows our users to skip typical cellular and ISP 'toll roads'." BlockMesh uses the same blockchain incentive model where "users will get Mesh Tokens (MESH) for every MB of data that passes through their mobile device" [27].

The BlockMesh project does not want to remove the traditional internet infrastructure that uses ISPs, but build on top of it and provide mesh networks in areas with poor infrastructure. "Users and IoT devices can earn 'Mesh Token' for supporting the network and pay in 'Mesh Token' for access to conventional internet at a fraction of the cost. Users with unlimited Wi-Fi can offer their community or clients internet access with no passwords needed. With 'mesh' the internet connection can then be passed beyond the hardware range" [28]. A custom BlockMesh router called MeshEX, provides access to the mesh network for surrounding users and the users "utilising the routers in homes or in their small business will be rewarded for each MB support on the DATA free mesh network via the hotspot created." "Users wanting conventional internet with in MeshEX range will be charged in Mesh Token per megabyte" [28].

This architecture does not remove the single point of failure that ISPs introduce and could still allow for government censorship or shutdowns. There is no mention of a decentralized storage system in the BlockMesh whitepaper and relying on centralized servers for object retrieval keeps another method of censorship within the network that governments can use.

2.5.6 Ammbr

The Ammbr project seeks to solve the same problem as the other state-of-the-art projects, providing internet access to areas of poor infrastructure. They seek to achieve this in a similar fashion by using mesh networks and the blockchain. Ammbr pays particular attention to ensuring node operators complying with local regulations through their DAO (Decentralized Autonomous Organisation) feature. "The DAO will enforce granting and

revocation of access for non-compliance with local laws or network rules, as defined by Ammbr's rulings" [29]. This access management could be abused and used as a form of censorship if a node operator's access is revoked due to a government ordering Ammbr to revoke a specific operator.

Ammbr also uses the blockchain to buy and sell network access using their proprietary cryptocurrency, AMR. "Billing terms are determined based on various criteria to include bytes transferred, guaranteed bandwidth, and time of day rating." "Node operators in all positions on the Ammbr Network, whether client access nodes, transit (or "intermediate") nodes, or gateways, will have the ability to independently set their own rates in accordance with their operating expenses and desired rate of return" [29]. One issue with operators setting their prices is the potential for unequal treatment of traffic, removing net neutrality from the network.

The Ammbr project also has partnerships with Huawei, a company that wants to introduce a new internet architecture called "New IP". "The proposal has caused concerns among western countries including the UK, Sweden and the US, who believe the system would splinter the global internet and give state-run internet service providers granular control over citizens' internet use" [30]. If the introduction of this new internet architecture is successful, this partnership could lead to the Ammbr network being centrally controlled.

2.5.7 Summary

The state-of-the-art projects that were discussed and this project have similar goals, to create a decentralized network using mesh networks and the blockchain. However, these projects use the blockchain as an incentive scheme for users to provide resources to the network as opposed to using it for decentralized object storage. These projects do not seem to acknowledge the benefits of using the blockchain's decentralized nature and how it can be used for object storage in order to create a fully decentralized web.

These projects also seek to provide internet in areas of poor infrastructure however, governments could censor internet in these areas if they still rely on the current internet architecture that uses centralized storage and DNS lookups.

3 Design

In this chapter, a high-level overview of the DWeb protocol design will be presented. This project aims to realise the DWeb by using the blockchain in combination with mesh networks. This design for this project was created by the project's supervisor, Professor Hitesh Tewari.

3.1 How Can We Realise the DWeb?

Blockchain technology has shown how we can move from a centralized decision making network to a distributed, transparent and more inclusive system. In this project, we want to apply the same philosophy to the Internet and create a decentralized web infrastructure for static objects. In this project, we will make use of mesh networks and point-to-point links that will prevent physical attacks to the network and create highly replicated and immutable blockchain backed data stores. Highly replicated and immutable blockchain backed data stores will prevent logical attacks to access information. The high-level view of the system architecture can be seen in figure 3.1.

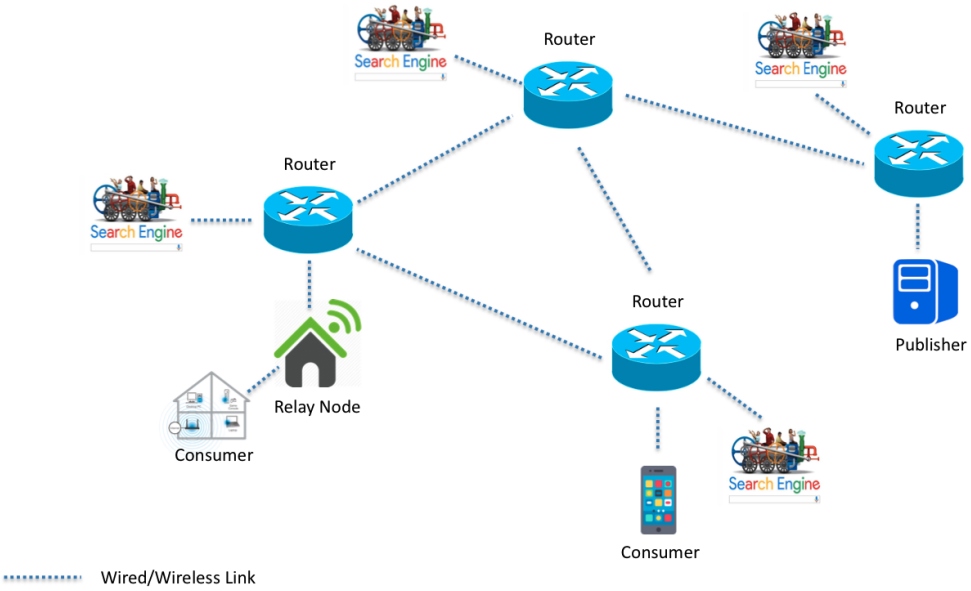

Figure 3.1: High-level overview of the DWeb system architecture

A typical DWeb network topology could consist of users in a housing estate each have their own Wi-Fi router that wirelessly connects to a neighbouring Wi-Fi router, which in turn connects to another router, and so on. One or more routers in the estate will act as the gateway router(s) and will run the DWeb stack. They will have a directed wireless link to one or more other DWeb routers in the adjoining neighbourhoods. This way we will be able to form a mesh network of DWeb routers, behind each of which there will be a number of users.

Each consumer can request and search for objects using a search engine. Each DWeb router is connected to at least one other router to create a mesh network. The publisher advertises its produced objects to its nearest router which can then satisfy requests from consumers. Search Engines used in the design of this project are publicly searchable entity attached to a DWeb router to allow efficient access to objects.

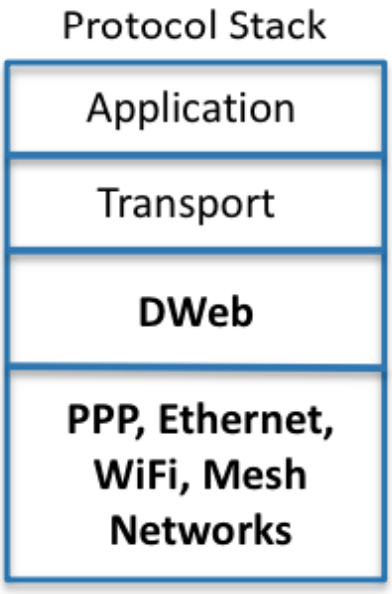

Figure 3.2: DWeb protocol stack

DWeb routers will have a protocol stack, as seen in figure 3.2, consisting of a DWeb layer which replaces IP. The upper layers (Application and Transport) remain as before. This protocol makes use of MAC addresses to use as a flat naming space. Communication links operate on a P2P basis using available wired and wireless technologies.

3.2 DWeb Router

Each DWeb router is connected to at least one other router, when each router follows this rule, a mesh network is created. This DWeb protocol uses MAC addresses to replace IP to route packets via P2P links. Mesh networks interconnect over IEEE 802.11s. This creates a Wireless Local Area Network (WLAN) network [4] and 802.11s is based on the 802.11 MAC protocol. The 802.11 MAC protocol uses MAC addresses as source and destination addresses. A mesh frame source addresses field "contains an IEEE MAC individual address that identifies the MAC entity from which the transfer of the [MAC Service Data Unit] MSDU". The destination address "field contains an IEEE MAC individual or group address that identifies the MAC entity or entities intended as the final recipient(s) of the MSDU" [31].

Creating a local network using mesh networks and routing via MAC addresses removes the need for IP addresses and therefore DNS lookups that remove Internet Service Providers (ISPs). With mesh networks implementing a P2P protocol and sending data directly from one router to another, there is no middleman like ISPs needed to route the traffic. By removing ISPs this could eliminate the ability of Governments' to perform Internet

shutdowns by telling an ISP to stop providing their service.

DWeb routers also maintain paths to other popular routers in the DWeb by making use of a time and space limited cache.

Routers maintain a synchronized copy of the blockchain ledger that contains a fingerprint of each object that has been uploaded to the DWeb.

Finally, routers have a publicly accessible search engine that consists of transactions extracted from the synchronised ledger for quicker access.

3.3 Edge Nodes

There are two types of edge node in this protocol, a publisher and a consumer. A publisher is a type of edge node that publishes Web objects. Publishers send a copy of any new object to its default router to be included in the blockchain. The default router is a gateway that links its local mesh network to another network. Publishers may have a direct or indirect connection to a DWeb router that connects to another mesh network.

A consumer has knowledge of routers on the DWeb and can query search engines to locate objects and requests its default router to retrieve objects.

3.4 Blockchain Storage Design

For this project, the blockchains use case is for immutable web object and data storage. A user can request an object using a search engine attached to a DWeb router to allow for efficient access to objects.

The blockchain in this project is a permissioned ledger to which blocks can only be added by authorized DWeb routers and the opportunity to add a block is controlled in a round robin fashion, meaning no proof-of-work is required. Each block has a number of transactions, where a transaction is a set of parameters to identify a web object, as see in figure 3.3. When a new object is created by a publisher, it sends a copy to its default router. The router then creates a new transaction and broadcasts it to the P2P network.

Figure 3.3: Contents of a Blockchain Transaction

A transaction contains the Object, associated metadata and a 256-bit cryptographic Hash of the object and metadata, to create a unique identifier. The cryptographic hash acts as integrity check, as well as unique Object Identifier (OID), the OID represents a flat naming space. Each new transaction has to be digitally signed with the secret key of the router that it originates from. The signature can be verified by all other DWeb routers by accessing the public-key certificate of the router. Any unsigned transaction is automatically rejected by the network and this prevents content pollution by unauthorized publishers.

3.4.1 Adding a New Blockchain Transaction

A router broadcasts a new transaction to the P2P network for it to be included in the next block, as seen in figure 3.4.

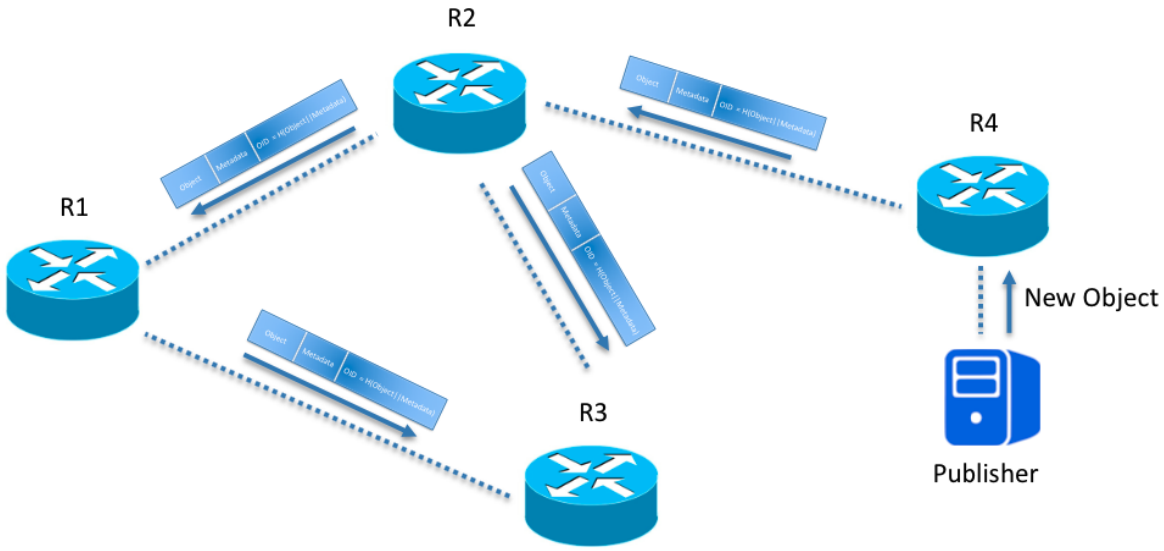

Figure 3.4: Example of adding a Blockchain Transaction

Once a transaction has been locked into a blockchain, other routers can note the address of the upstream router from where the transaction originated. Using this address, routers can add an entry for the object to their search engine and store a local copy.

3.5 Routing

If an object is cached in the default router's search engine, the object can be served immediately, if not a request will be broadcast on the P2P network with the object's OID and router's address. Each router along the path adds its address prior to forwarding the interest request. A router that has the specified object can satisfy the request by simply reversing the path to get the object to the requesting consumer, as seen in 3.5. This is known as dynamic source routing [32], however, the addresses used in this projects design are MAC addresses not IP addresses.

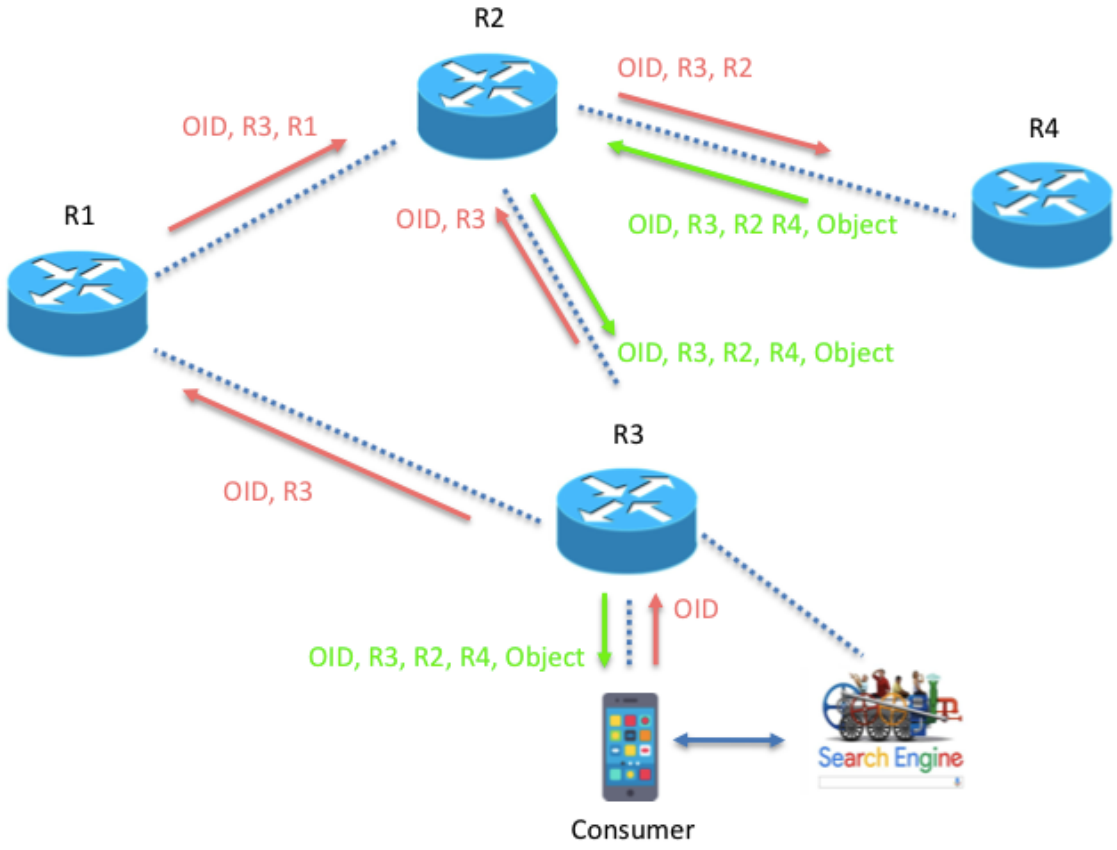

Figure 3.5: Fetching a Non-cached Object

A consumer searches for an object and the request is broadcast to the P2P network, seen in red. The request is satisfied by Router 4 and the object is sent back by reversing the request path, as seen in green.

3.6 New Router Joining the DWeb

A router joining the DWeb must obtain a public-key certificate from a trusted certification authority (CA). A cert binds the MAC address of a router to its public key. The certificate is broadcast to the P2P network and locked into the blockchain.

A new router must also download a full copy of the blockchain. It should index transactions from the blockchain into its publicly accessible search engine according to a known algorithm to mirror other DWeb search engines.

3.7 Finding an Object on the DWeb

The easiest way to find an object is for a consumer to query a DWeb search engine. The engine returns a set of links based on the search query. Each link consists of an object identifier and associated metadata. `Link = (OID, Metadata)`. On clicking a link a request will be sent out to the default router or another directly connected router that will retrieve the object.

3.8 Network Partitioning or Joining

It is possible that parts of a DWeb network may lose connectivity to the DWeb from time to time. A node may rejoin the network at a later stage or never reconnect with the network.

During a network partition, all updates to the ledger will be "local" to the routers in the sub-network. When connectivity is re-established with the DWeb the sub-network nodes must use the longest blockchain being advertised. The longest blockchain is determined by the number of object storage transactions that have taken place, not the number of nodes in the network. The reconnected nodes broadcast any "local transactions" to the P2P network for them to be included in the longest chain.

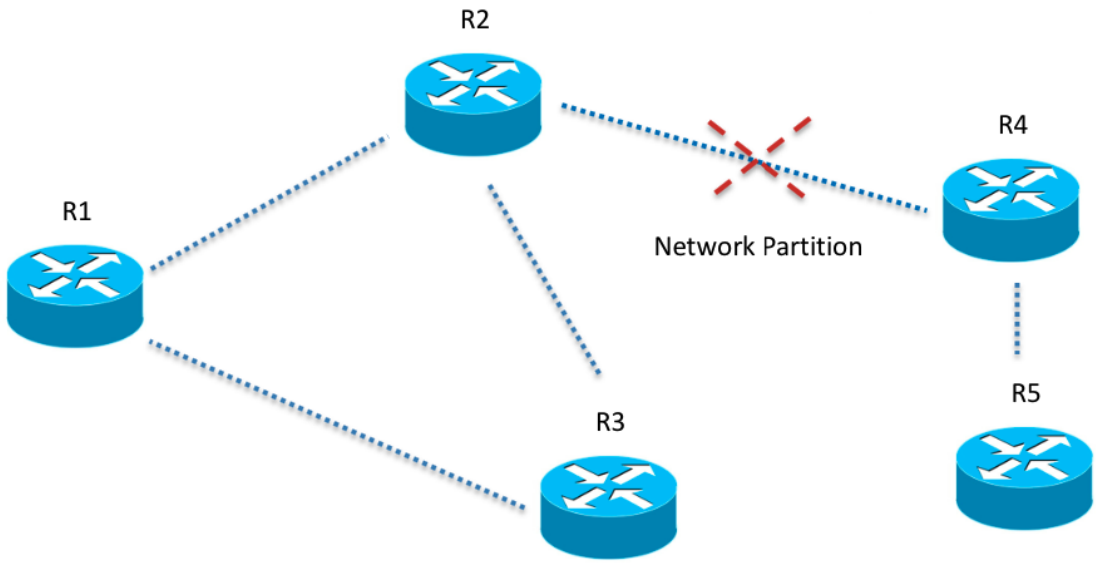

Figure 3.6: Example of a network partition

In figure 3.6, router R4 and R5 become disconnected from the main network. All updates to the ledger in routers R4 and R5 will be local to their sub-network. When connectivity is re-established, the longer blockchain is used, in this example this will be R4 and R5's chain. Therefore, the "local" blockchain from R1, R2 and R3 will be advertised to be included in the longer chain. Each router in the network will then have a synchronised copy of the blockchain with all transactions that happened during the partition.

3.9 Trust

DWeb gateway routers are trusted entities in the network. They create blockchain transactions on behalf of other routers and are relied upon to act in a fair manner. This privilege is granted when a DWeb router joins the network and obtains a public-key certificate from a trusted CA.

These privileges can be revoked by the DWeb community by removing the public-key certificate of a "blacklisted" gateway, if for example the trust score of the router falls below a threshold value. Certificate revocation can be made possible by adding a "dummy" certificate into the blockchain for the MAC address associated with the blacklisted router. The nodes on the network will use the new public-key certificate and see that the router is blacklisted and will not include its transactions in the blockchain.

All routers in the DWeb have the ability to become a gateway router by obtaining a public-key certificate of their own. A publisher can run their own gateway router and send it any object references that it wishes to be added to the blockchain. Similarly, a consumer can run a full gateway node and keep a synchronized copy of the blockchain to

search for objects on the DWeb. In this manner a consumer or publisher no longer need to trust other entities in while retrieving or adding objects to the DWeb.

3.10 Security and Privacy Considerations of the Design

3.10.1 Inbuilt Security Features

Once a requesting router receives their requested object, they create a hash of the metadata and object and compare the hash with the OID of the object. The OID is a hash of the original object and its metadata so if the hashed values match, the router knows it has been served the original object and the object has not been modified, therefore providing a data integrity check. In addition, limiting who can add objects to the DWeb reduces the chance of something malicious being added to the network. Having an authorised router sign each transaction means other routers can verify the objects uploaded by any router to ensure authenticity.

IEEE 802.11s uses the Simulation Authentication of Equals (SAE) key establishment algorithm between peers. "Besides mutual authentication, SAE provides two mesh stations with a Pairwise Master Key (PMK) that they use to encrypt their frame. As its name indicates, SAE does not rely on a keying hierarchy like traditional 802.11 encryption. Instead, it implements a distributed approach that both mesh stations may initiate simultaneously. Because of the pairwise encryption, each link is independently secured. As a consequence, 802.11s does not provide end-to-end encryption" [4].

3.10.2 Mesh Network Security Considerations

MAC Address Spoofing

A potential security issue with this project is MAC address spoofing in order to attempt to receive packets or data that was intended for a different router. DWeb routers in a mesh network will maintain paths to other popular routers in their network and also the path to the requesting router, as seen in figure 3.5. Each DWeb router has its own, unique MAC address and another router cannot join the network with the same MAC address. In order for an attacker to attempt to receive a packet that was meant to someone else, they would first have to somehow remove the consumers' router from the network then join the network with their own router and fake MAC address. However, when a new

router joins the DWeb network, it must obtain a public-key certificate from a trusted CA. The certificate binds the router's MAC address to its public key. The certificate is broadcast to the P2P network and is locked into the blockchain. Due to this process, an attacker trying to pose as another router will have a lot of difficulty, as they would need to somehow link themselves to the certificate that has already been locked into the blockchain for the real user's router. MAC address spoofing attacks become difficult when each router has their own certificate as attacker's have to attempt to get their fake router linked to a real router's certificate when trying to join a network maliciously.

However, there is privacy concerns with this way of using MAC addresses. This does not allow for random MAC address usage in order to make tracking and fingerprinting of users more difficult. Using the DWeb in this design leads to a trade-off of greater physical security and stability of a network for reduced potential privacy of individual users.

Denial of Service

A Denial of Service DoS attack is another potential security issue with mesh networks. "A common technique is to flood the target system with requests. The target system becomes so overwhelmed by the request that it could not process normal traffic" [33]. Routers MAC addresses could be blocked if they are sending too many requests, however, this isn't a good solution. Also, this would not work with any form of Distributed Denial of Service (DDoS) attack, there would be too many systems to block and most of them would probably be innocent users whose systems have been compromised without their knowledge. DDoS attacks "typically originate from large clusters of computers that have been commandeered by the attackers. These zombie systems are then controlled from a centralized point allowing a huge force of bandwidth to be concentrated in a single place. These zombie armies, or botnets, can utilize thousands of machines" [34]. So instead of the flood of requests coming from one system, it comes from many. Using a larger number of computers makes it harder to block the attacker's address and can cause more disruption.

3.10.3 Blockchain Security

Majority Attack

"With Proof of Work, the probability of mining a block depends on the work done by the miner" [35] and therefore people create mining pools to increase computation power. However, if a mining pool has more than 51% of the computing power, it can control the blockchain and can then decide if the block is permissible or not. With a majority, an

attacker can modify transaction data, possibly to cause a double spending attack, stop the block verifying a transaction or stop a miner mining [35]. This issue is mitigated in this project's design as it uses a round robin consensus for the opportunity to add a block to the blockchain. According to Yaga et al. [36], the "round robin does not work well in the permission-less blockchain networks used by most cryptocurrencies. This is because malicious nodes could continuously add additional nodes to increase their odds of publishing new blocks."

Attack Surface

According to Dai et al. [37], a distributed storage mechanism, similar to this projects use for the blockchain, creates a broader attack surface and gives a potential attack more alternatives to access data or decipher network structure from transactions in the blockchain.

3.11 Summary

Nodes in this design are connected by a mesh network protocol and use the blockchain for immutable object storage. Routers replace the IP layer with the DWeb layer and route using MAC addresses. Publishers produce web objects for the DWeb and consumers request the objects using a search engine. Only authorized routers can add to the blockchain and this prevents malicious actors from uploading dangerous objects to the blockchain. Each router wishing to join the DWeb must obtain a public-key certificate from trusted certificate authority. Routing in the DWeb uses dynamic source routing but replaces IP addresses for MAC addresses. During network partitions, updates to the ledger will be local to the sub-network. Once a reconnection occurs, the longest blockchain is used and all local transactions are synced. Comparing hashes of objects and their metadata against the object's OID gives consumers piece of mind that they are receiving the original object.

In the next chapter, the successes and difficulties during the implementation of this design will be discussed.

4 Implementation and Evaluation

The design chapter gave a high-level overview of the DWeb protocol and justification for the design choices taken. In this chapter, the implementation of these designs and the evaluation of each implementation will be discussed.

4.1 Mesh Network

In order to test the design of this DWeb protocol, a network simulation tool is needed to create a virtual mesh network, with DWeb router behaviour installed on the nodes. In order to create these simulations, ns-3 was used as it provided mesh network behaviour without the user needing to create any custom configurations.

4.1.1 Network Simulator 3

Network Simulator 3 (ns-3) is an open source, "discrete-event network simulator, targeted primarily for research and educational use" [38]. ns-3 is the simulation tool used to create this projects DWeb mesh network. ns-3 is used to design and test network simulations. These simulations can be created by making C++ or python programs where the creator can fully customize the behaviour, topology and network stack used in their simulations.

Some features of ns-3 include attention to realism, where "nodes are designed to be a more faithful representation of real computers, including the support for key interfaces such as sockets and network devices, multiple interfaces per nodes, use of IP addresses, and other similarities." Another feature is software integration, which supports "the incorporation of more open-source networking software such as kernel protocol stacks, routing daemons, and packet trace analyzers, reducing the need to port or rewrite models and tools for simulation" [39].

Core Concepts

The key abstractions in ns-3 are Nodes, Applications, Channels, Net Device and Topology Helpers. A node in ns-3 is the basic computing device abstraction to which a user can add functionality such as protocol stacks or applications. “In ns-3 the basic abstraction for a user program that generates some activity to be simulated is the application” [40]. Applications run on nodes to simulate certain predefined behaviour. In ns-3, a node connects to an object for communication. This object is a Channel, which manages sub-network object communication and connecting nodes to them. The net device abstraction is for the software driver and simulated hardware and is installed in a node to allow the node to communicate with others via Channels. Topology helpers make setup of simulations easier that provide methods to assign IP addresses, connect Net Devices to Channels, etc. [40]. As seen in figure 4.1, a node has an application, protocol stack and net device installed. Nodes communicate with each other over Channels [39].

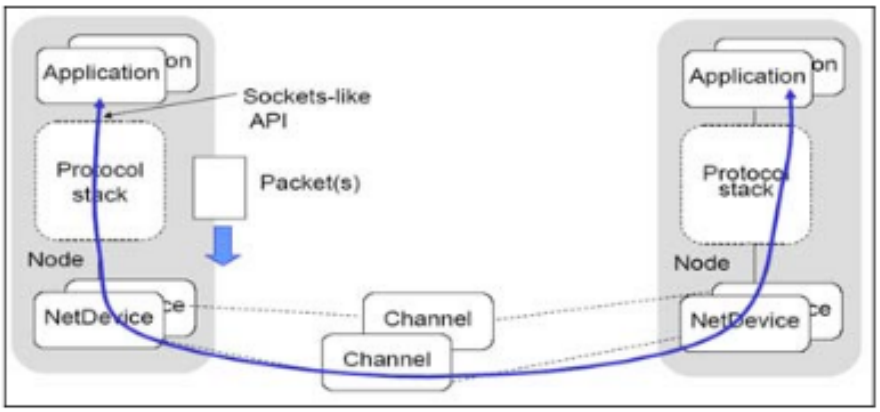

Figure 4.1: ns-3 Basic Architecture

ns-3 provides different modules that can be used in different simulations, such as the Mesh Device module that provides MAC-layer routing functionality. This is the main reason ns-3 was chosen as the simulation tool to attempt to create the DWeb described in the design section. Using the examples provided by ns-3, creation of a mesh network was not difficult. The next task was to link the ns-3 simulation with Multichain, our blockchain framework of choice for this project. This was when the guide on how to "make ns-3 interact with the real world" was discovered which introduced the concept of emulation and using the ns-3 TAP mechanism [41].

Emulation Support

ns-3 can be integrated into test bed and virtual machine environments by providing two net devices, Emu Net Device, which allows ns-3 to send data on a ‘real’ network and Tap

Net Device, which allows a 'real' "host to participate in an ns-3 simulation as if it were one of the simulated nodes" [39].

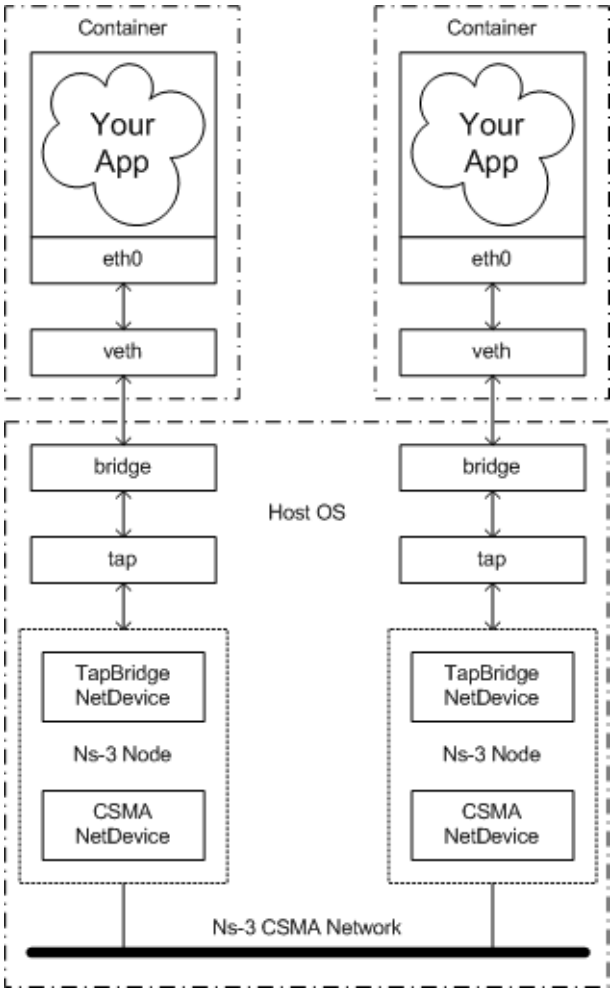

Figure 4.2: ns-3 Connection to External Containers

Figure 4.2 [42] demonstrates how ns-3 nodes can connect to an external application that is located in a container. This process is how ns-3 simulations can connect to external applications, by placing the applications inside containers. This was originally designed to work with Linux containers, however, this project is using Docker containers, which is further discussed in the ns-3 Docker Emulator section. "This uses a special net device called a TapBridge; Tap coming from the tun/tap device which is the Linux device driver used to make the connection from ns-3 to the Linux guest operating system, and Bridge since it conceptually extends a Linux (brctl) bridge into ns-3" [41]. This TapBridge net device appeared to be the best method to connect our external application running Multichain with our ns-3 mesh simulation.

4.2 Blockchain Storage

4.2.1 Multichain

Multichain is an open source blockchain framework to create private blockchains [43]. Multichain is the blockchain application chosen to implement this DWeb protocol's static object storage. A user can create blockchains, connect to them and create streams for general data storage and retrieval, which is ideal for our use case. Multichain streams will be how this protocol will simulate blockchain object storage.

Multichain uses a "round-robin consensus scheme, rather than proof-of-work as in bitcoin" [44]. The round-robin schedule, which enforces miners to create blocks in rotation, ensures one participant in the blockchain does not monopolize the mining process by having a "constraint on the number of blocks which may be created by the same miner within a given window" [45]. A diversity threshold value, between 0 and 1, is used to determine the strictness of the schedule. A value of 1 ensures every permitted miner is included in the rotation and a value of 0 represents no restriction, a miner could create a monopoly. However, a value of 1 is not advised as the blockchain could freeze if a miner is inactive. Multichain suggests a value of 0.75 as a reasonable compromise. This diversity threshold also "helps in a case where the network splits temporarily into disconnected islands, perhaps due to a communications failure. This mishap will lead to a fork in the chain, as each island is unable to see the other's transactions and blocks. Once the network is reunited, the fork with the longer chain will be adopted as the global consensus. The diversity threshold ensures that the longer blockchain will belong to the island containing the majority of permitted miners, since the other island's chain will quickly freeze up" [45]. This behaviour is ideal for this project as nodes can leave and join WMNs at any time.

Streams

Multichain streams "provide a natural abstraction for blockchain use cases which focus on general data retrieval, time-stamping and archiving, rather than the transfer of assets between participants" [46]. In this project's implementation, streams are used to store object data that a user can request from a stream in their network. Multichain streams solve the problem of network splits and using the largest blockchain as described in Chapter 3. Any number of streams can be made in a blockchain and they can be used for three different types of databases on a chain:

- A key-value database, in the style of NoSQL.

- A time series database that focuses on the ordering of entries.
- An identity-driven database where entries are classified according to their author.

For this project, the key-value pair or the identity-driven database method could be used.

Each item in a stream has the following characteristics: [46]

- One or more publishers who have digitally signed that item.
- An optional key for convenient later retrieval.
- Some data, which can range from a small piece of text to many megabytes of raw binary.
- A timestamp, which is taken from the header of the block in which the item is confirmed.

Each node in a blockchain can choose what streams to subscribe to and can retrieve items from a stream in multiple ways: [46]

- Retrieving items from the stream in order.
- Retrieving items with a particular key.
- Retrieving items signed by a particular publisher.
- Listing the keys used in a stream, with item counts for each key.
- Listing the publishers in a stream, with item counts.

Similar to when a fork occurs in a network and the larger blockchain of the fork is used as the consensus, streams will be updated when the fork reconnects. If an isolated node publishes an item to a stream, when it reconnects to the network, every other node subscribed to the same stream will show this new published item. This behaviour can be seen when some data is published to a stream when the ns-3 emulation is not running, and is therefore not providing network links between the DWeb nodes.

Multichain Commands

The following is a list of commands used to create a blockchain, connect to the blockchain from another node, create a stream and publish to a stream. A more detailed list of commands can be found on the multichain website.[1]

- Create a new blockchain: `multichain-util create <chain-name>`

[1] `https://www.multichain.com/getting-started/`

- Initialize the blockchain: `multichaind <chain-name> -daemon`
- Connect to a blockchain: `multichaind <chain-name>@[ip-address]:[port]`
- Enter the CLI for a blockchain: `multichain-cli <chain-name>`
- Create a new stream: `create stream <stream-name> '{"restrict":"write"}'`
- Publish some JSON to a stream: `publish <stream-name> <key-name> '{"json":{"name":"John Doe","city":"London"}}'`

4.3 Connecting the Mesh Network to the Blockchain

The main roadblock in this project was figuring out how to link the mesh network simulation in ns-3 with the multichain blockchain. Multiple tools were discovered, ns-3 Docker Emulator being the one selected for this project.

4.3.1 ns-3 Docker Emulator

ns-3 Docker Emulator (NDE) is the main tool used to implement this projects design. NDE is a network emulator that uses ns-3 and Docker that is developed by Jose Alfredo Alvarez Aldana [47][2]. NDE works by connecting an application that is on a Docker container to each node in the ns-3 simulation using a Linux Bridge, as seen in figure 4.3. A python file runs the required commands and uses an ns-3 simulation file, along with command line arguments to create the desired emulation. A user can monitor and interact with the emulation to obtain the desired data.

[2] `https://github.com/chepeftw`

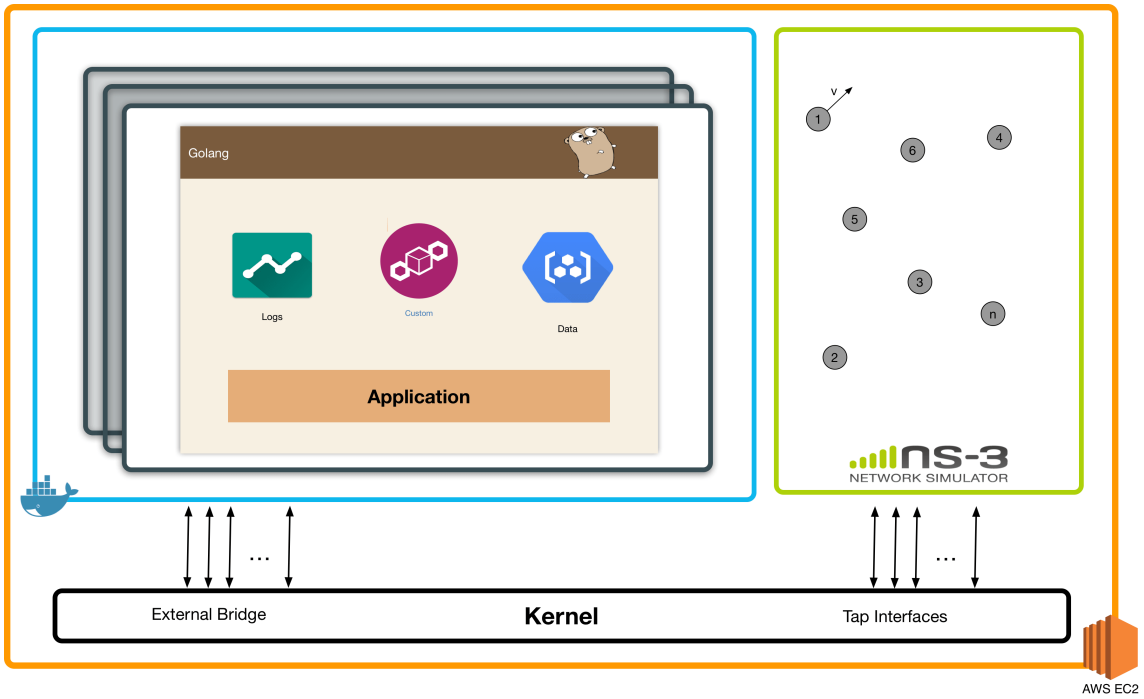

Figure 4.3: ns-3 Docker Emulator Design

In figure 4.3, the ns-3 simulation is connected to the application on a Docker container through a Linux Network Bridge [48]. The application in the Docker container for this project has a multichain blockchain running on it with a stream that each node in the ns-3 network is subscribed to and can get objects from.

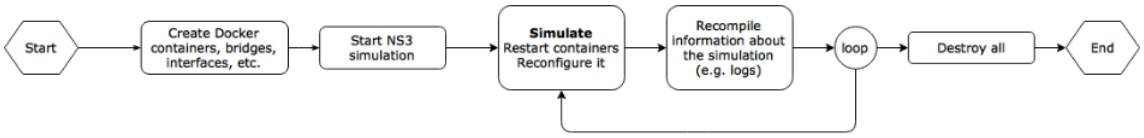

Figure 4.4: ns-3 Docker Emulator Program Pipeline

Figure 4.4 show the stages of a emulation using NDE [48]. The first step is creation of a Docker container, with the desired application installed, and a bridge and tap interface for each node in the network. The next step is to start the ns-3 simulation to initialise the nodes, install the selected protocol on the nodes and install the ns-3 `TapBridgeHelper` onto the nodes. The next step is the emulation. NDE is "highly scalable" as it "restarts the containers and makes sure everything is in place. By doing this, it restarts the app you are running inside the containers, allowing you to run a test. Then without destroying everything you just re-invoke the same command to start again" [48]. The emulation runs for the specified amount of time, during which a user can interact with the nodes, by pinging between nodes or monitoring a networks behaviour. Finally, once an emulation

is finished, the Docker containers, bridges and taps are destroyed.

The user can specify command line several arguments for the program, which are as follows:

- "operationStr", The name of the operation to perform, options: create, ns3, emulation and destroy.
- "-n", The number of nodes to simulate.
- "-t", The time in seconds of NS3 simulation.
- "-to", The timeout in seconds of NS3 simulation.
- "-s", The size in meters of NS3 network simulation.
- "-ns", The speed of the nodes expressed in m/s.
- "-np", The pause of the nodes expressed in s.
- "-c", The count of simulations.
- "-j", The number of parallel jobs.

For this projects implementation, the Docker container is an Alpine Linux image with multichain installed with one blockchain and stream that each node in the network is subscribed to. The nodes in the projects design use mesh networking for communication, however, an NDE emulation was not successfully completed using mesh routing between nodes.

The emulation configuration that was completed within the project's time-frame consists of an ns-3 file that uses the CSMA protocol that creates the network that the Docker container nodes communicate over. The default ns-3 CSMA example[3] was used to create a network between nodes where they could interact with the multichain blockchain. The file has been modified to allow for a variable number of nodes, simulation time and name for the `TapBridge` interfaces. A `TapBridge` is created and installed on each node in the simulation and then the simulation is run for the specified amount of time. Figure 4.5 is a high-level diagram of this project's implementation.

[3] `https://www.nsnam.org/doxygen/tap-csma-virtual-machine_8cc_source.html`

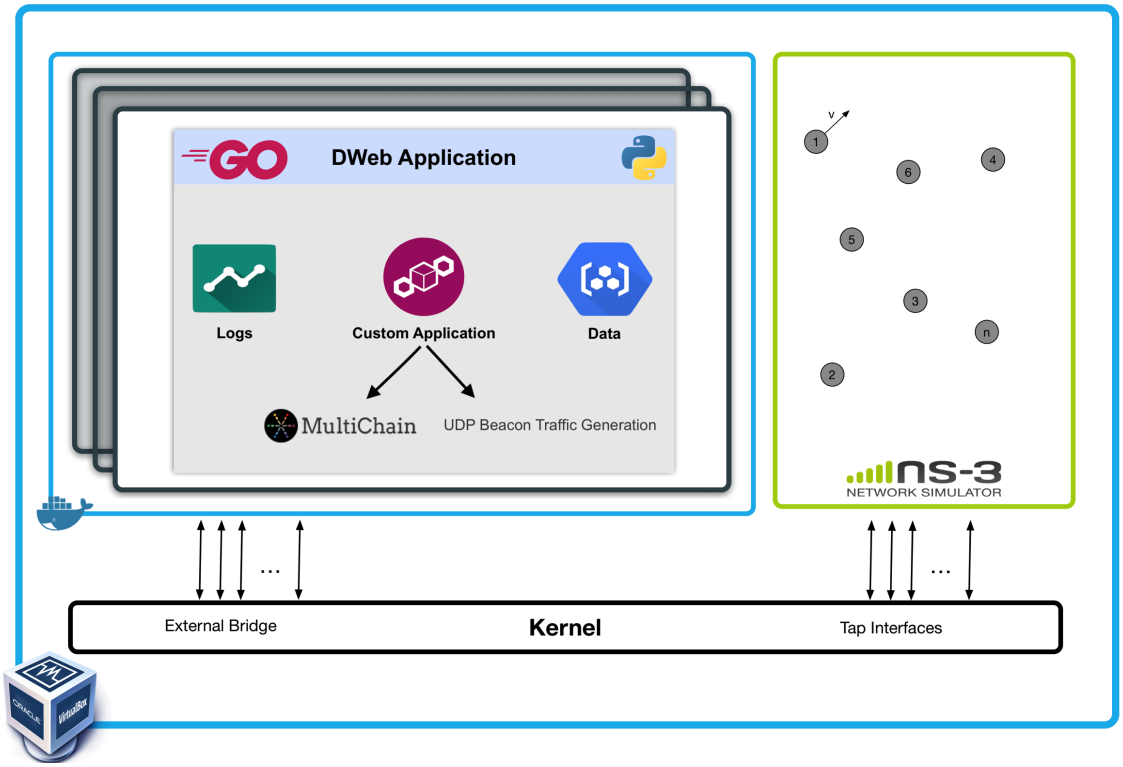

Figure 4.5: Component Diagram of Implementation

In order to create a form of traffic generation, a UDP Beacon program[4], also made by the creator of ns-3 Docker Emulator, Jose Alfredo Alvarez Aldana, was used. This program broadcasts a UDP packet periodically and listens to any incoming UDP packets. The original program has been modified to publish a message to a multichain stream when each packet is sent, using the origin address as the key and the message as the value. When a UDP packet is received by a node it performs a lookup on the stream using the origin address as the key in order to retrieve the data. This behaviour has been created as a starting point for traffic generation and blockchain object storage.

Docker

Docker is a service that provides operating system virtualisation that packages software into containers. It is a tool for running applications in an isolated environment with advantages similar to virtual machines (VMs). "A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings" [49]. Containers are used to package software with all the parts it needs, such as libraries, programs and

[4] `https://github.com/Donegaan/Beacon`

other dependencies. Providing a packaged standardised application allows the developer to be confident that their application will run uniformly on any other machine, regardless of custom settings.

Docker uses a client-server architecture, as seen in figure 4.6. "The Docker client talks to the Docker daemon, which does the heavy lifting of building, running, and distributing your Docker containers. The Docker client and daemon can run on the same system, or you can connect a Docker client to a remote Docker daemon. The Docker client and daemon communicate using a REST API, over UNIX sockets or a network interface" [50].

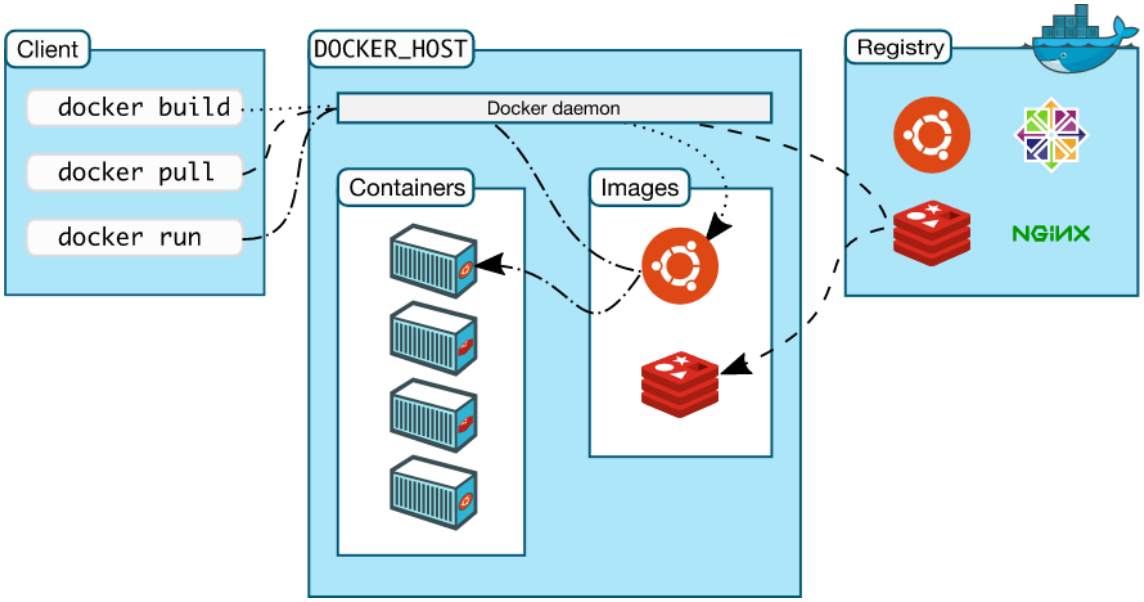

Figure 4.6: Docker Architecture

The Docker daemon listens for Docker API requests and manages Docker images, containers, networks and volumes. Most users interact with Docker through the client by using commands that are sent to the daemon, which then carries out these tasks. A Docker registry stores Docker images, the website Docker Hub[5] being the most popular [50].

Docker Images "are read-only template[s] with instructions for creating a Docker container" [50]. Users can create their own images by creating a `Dockerfile` that defines the steps to create and run an image.

Docker Containers are instances of an image that can be run using the Docker API or CLI. "A container is defined by its image as well as any configuration options you provide to it when you create or start it. When a container is removed, any changes to its state that are not stored in persistent storage disappear" [50].

[5] `https://hub.docker.com/`

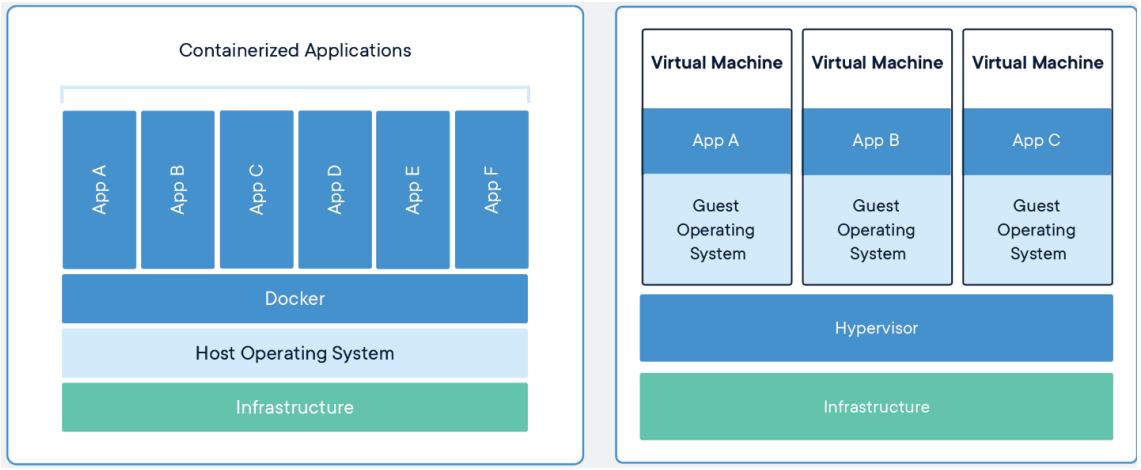

Figure 4.7: Docker Container vs Virtual Machines

In figure 4.7, Containers are an abstraction at the app layer that packages code and dependencies together. Virtual machines (VMs) are an abstraction of physical hardware turning one server into many servers [49].

Docker containers do not have a full copy of an OS, and are therefore much smaller than a VM, which is typically tens of gigabytes. VMs also tend to have slow startup times. "Containers and virtual machines have similar resource isolation and allocation benefits, but function differently because containers virtualize the operating system instead of hardware. Containers are more portable and efficient" [49]. Docker is available on macOS and Windows through the Docker Desktop application and available natively on Linux.

Linux Bridges

Linux can be used to create a virtual network switch, called a bridge. Bridges connect network links together which creates a local area network (LAN). "Conceptually, the major components of a network switch are a set of network ports, a control plane, a forwarding plane, and a MAC learning database" [51]. Traffic is forwarded using the ports, the control panel normally runs the Spanning Tree Protocol, which calculates the minimum spanning tree for a LAN, in order to prevent physical loops crashing a network. Input frames are processed by the forwarding plane and decides which port the input frame is forwarded to. The MAC database tracks the host locations in the LAN, for "each unicast destination MAC address, the switch looks up the output port in the MAC database. If an entry is found, the frame is forwarded through the port further into the network. If an entry is not found, the frame is instead flooded from all other network ports in the switch, except the port where the frame was received" [51].

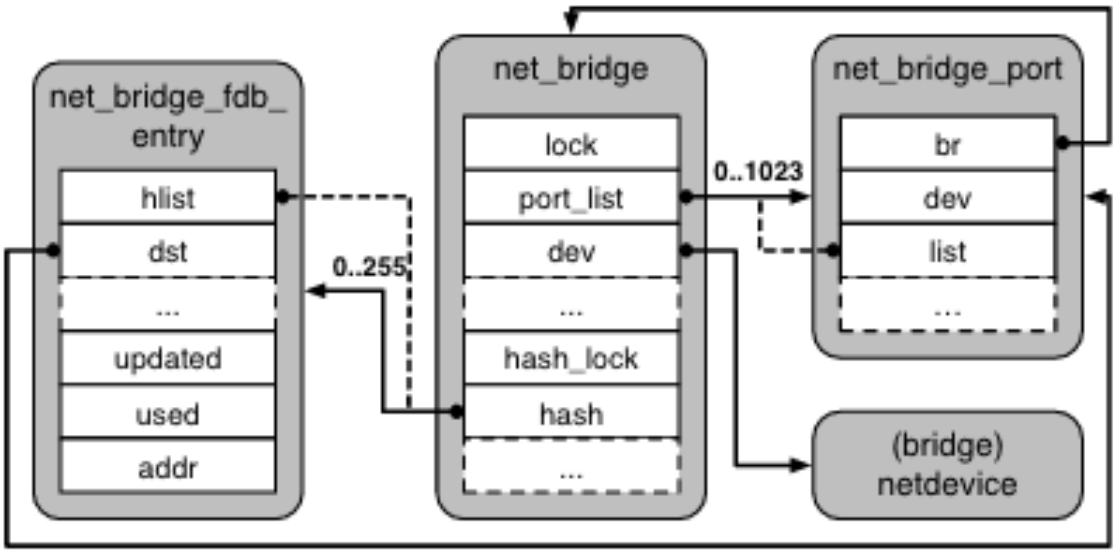

Figure 4.8: Linux Bridge Data Structures

As seen in figure 4.8, the primary data structure in each bridge is the `net_bridge`, which stores bridge configuration information, a doubly linked list of bridge ports (`net_bridge_port` objects), a pointer in `dev` to the `netdevice` and forwarding database in the `hash` field. The lock is used to synchronise configuration changes, such as changing bridge parameters [51].

The `net_bridge_port` has a back reference to the bridge it belongs to, the network interface the port uses to communicate frames and the `port_list` from the `net_bridge`.

The `net_bridge_fdb_entry` represents a single forwarding table entry, which consists of a MAC address of the host, a `hlist` field that points to the position of the object in a hash table. The `updated` and `used` fields are for timekeeping, the `updated` field is used to delete if they are inactive for a certain amount of time.

Input/output control (`ioctl`) is the configuration interface that creates and destroys bridges, and adds or removes network ports to or from a bridge. The `sysfs` interface is for the management of bridge and port parameters. An overview of this can be seen in figure 4.9 [51].

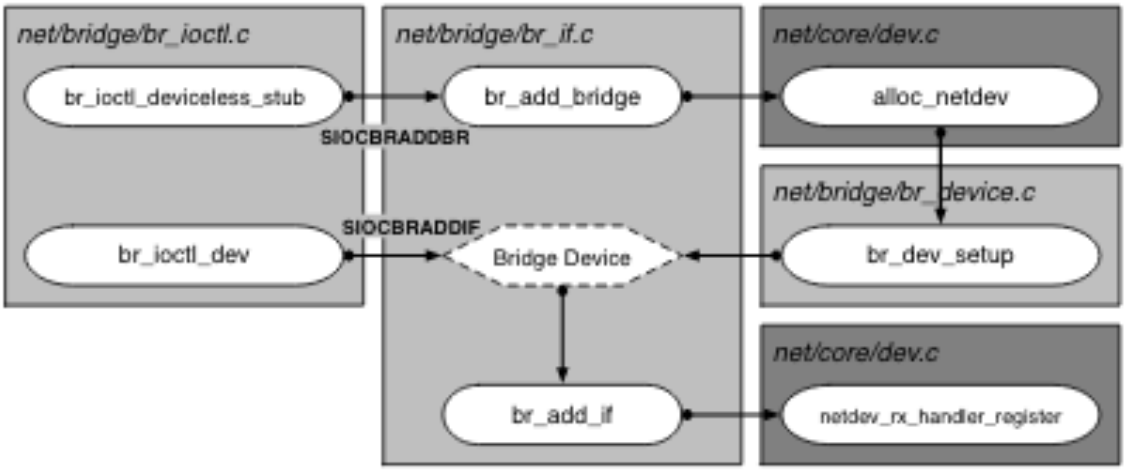

Figure 4.9: Linux Bridge Configuration; adding a bridge and bridge port.

A Linux bridge can be created using the `brctl addbr <bridge-name>` command.

Linux TAPs

A Linux TAP is a virtual network interface and carries Ethernet frames. The TAP delivers packets to the OS's network stack to emulate them coming from an external source [52]. In this projects case, it is used in the ns-3 `TapBridge` class to "make it appear that a real host process is connected to an ns-3 net device. [A] Linux process writes a packet to a tap device and this packet is redirected to an ns-3 process where it is received by the `TapBridge` as a result of a read operation there. The `TapBridge` then sends the packet to the ns-3 net device to which it is bridged. In the other direction, a packet received by an ns-3 net device is bridged to the `TapBridge`. The `TapBridge` then takes that packet and writes it back to the host using the Linux TAP mechanism. This write to the device will then appear to the Linux host as if a packet has arrived on its device" [53].

A TAP is created by using the `tunctl -t <tap-name>` command.
A TAP interface can be added to a bridge using the `brctl addif <bridge-name> <tap-name>`.

4.4 Tools Investigated but Not Used

4.4.1 Named Data Networking Simulator

Named Data Networking Simulator (ndnSIM) was the tool that was first looked at to simulate our DWeb protocol, as we wanted to compare our protocol's performance against

NDN. ndnSIM is built on top of ns-3 to specifically test NDN based simulations. ndnSIM changes the ns-3 packet format to the NDN packet format specified by the NDN project [54]. "ndnSIM is implemented as a new network-layer protocol model and can run on top of any available link-layer protocol model (point-to-point, CSMA, wireless, etc.). In addition, the simulator provides an extensive collection of interfaces and helpers to perform detailed tracing behavior of every component, as well as NDN traffic flow" [55].

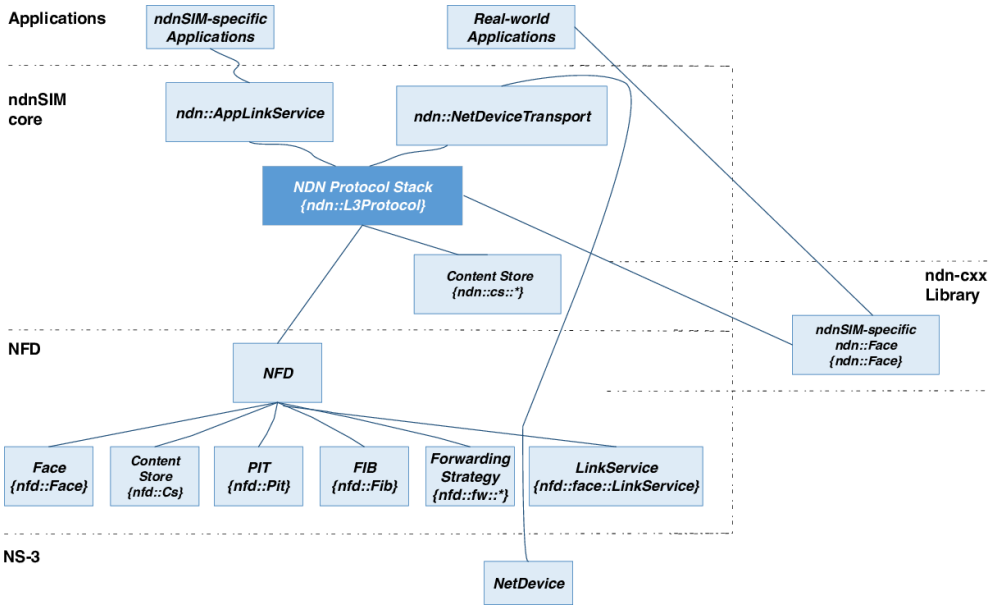

Figure 4.10: ndnSIM Architecture Components

As seen in figure 4.10, ndnSIM has its foundations based on the ns-3 Net Device abstraction. From there it is built up to create NDN specific configurations to make NDN simulations easier to create.

ndnSIM was originally used to understand the NDN protocol and to try to replicate existing NDN research papers experiments. This was done to find out what aspects NDN research papers measured when presenting their results. Once some familiarity with ndnSIM was gained, some time was spent creating mesh network simulations. When mesh network simulations were working using the NDN protocol, the next step was to attempt to use IEEE 802.11s MAC routing instead of the NDN protocol. However, once the code for ndnSIM was examined, it was determined that modifying the code to use the 802.11s protocol would be too difficult. This is when ns-3 was looked at and the mesh device module was discovered which better suited this project's needs.

4.4.2 Dockemu

Dockemu was another emulation tool that was examined when searching for a suitable tool to implement and test this DWeb protocol design. It was discovered at the same time as NDE as it is a main motivator in the creation of the NDE tool [38]. "Dockemu utilizes virtualization with Linux Containers through Docker and Linux Bridging along with ns-3 for the emulation of layers 1 and 2 of the OSI model" [56]. Dockemu is similar to NDE's design that combines Docker to virtualize Linux containers and Linux network bridges with ns-3 to create emulation. During the initial setup of NDE, there were some configuration problems with creating new Linux Taps and Bridges for the Docker containers. Due to these configuration difficulties, Dockemu was considered as an alternative tool to implement the design. A top-level view of the Dockemu framework can be seen in figure 4.11 [56].

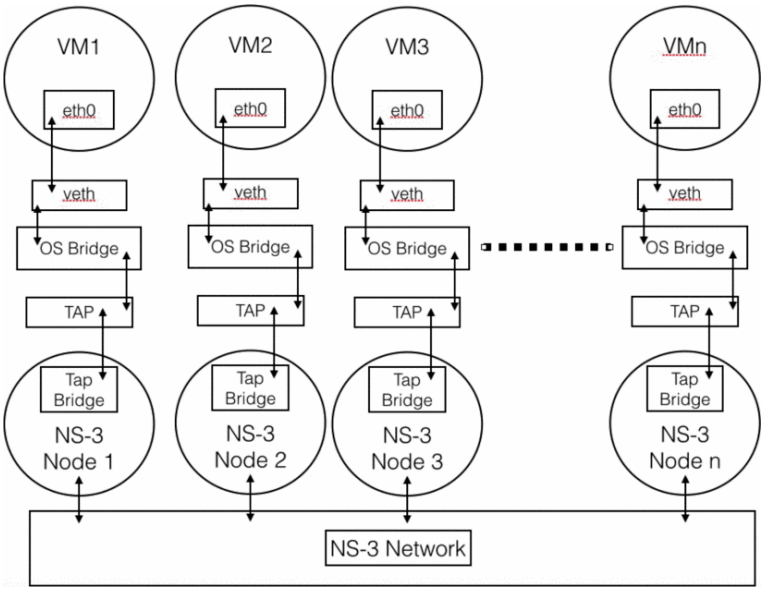

Figure 4.11: Top Level View of the Dockemu Framework

Initial configuration of Dockemu required some changes[6], however, once they were implemented it provided enough of an understanding to overcome the setup difficulties with the NDE tool. The main learning outcomes from configuring Dockemu was the need for administrator access to create bridges and taps to be used with the external nodes and using the ns-3 CSMA example configuration to allow nodes to communicate. Once this was applied to NDE, its initial configuration was complete and could be used to create custom emulations.

NDE was ultimately chosen instead, as Dockemu lacked any documentation for the code aside from its research paper [56], which provides some high level explanation of how to

[6] `https://github.com/Donegaan/dockemu`

use the tool but lacks technical details about creating custom scenarios. In the paper, it states a "tutorial will be available at the URL of the Research Lab[7]" [56], however, at the time of writing this dissertation, no such tutorial is available.

4.5 Evaluation

The implementation of this projects design that was achieved during this project uses Docker containers with Multichain installed and nodes can communicate over the network that is configured by the ns-3 simulation file. The containers act as nodes in the network and communicate using the CSMA LAN protocol, as the mesh protocol does not create the necessary connection between nodes. There is some traffic generation in the form of a UDP beacon program where a packet is sent and its data, alongside its origin address is published to the multichain stream each node in the network is subscribed to. When a packet it received, the data is searched for and retrieved from the stream. Logs from this traffic generation can be seen in figure 4.12.

```
INFO** 0415 11:19:16.604744 48 Beacon.go:184 Starting UPD Beacon
INFO** 0415 11:20:16.610425 48 Beacon.go:126 Our random message is 25167
INFO** 0415 11:20:16.612751 48 Beacon.go:87 10.12.0.1 -> Message: Hello network! 25167 from 10.12.0.1
INFO** 0415 11:20:17.136305 48 Beacon.go:139 Stream created sucessfully
INFO** 0415 11:20:17.437683 48 Beacon.go:148 Object published to stream sucessfully
INFO** 0415 11:20:17.565902 48 Beacon.go:87 10.12.0.1 -> Message: Hello network! 877 from 10.12.0.2
INFO** 0415 11:20:18.326425 48 Beacon.go:93 Object requested from stream sucessfully
INFO** 0415 11:20:18.412031 48 Beacon.go:87 10.12.0.1 -> Message: Hello network! 31648 from 10.12.0.3
INFO** 0415 11:20:19.242232 48 Beacon.go:93 Object requested from stream sucessfully
INFO** 0415 11:20:19.383518 48 Beacon.go:87 10.12.0.1 -> Message: Hello network! 26720 from 10.12.0.4
INFO** 0415 11:20:23.549579 48 Beacon.go:93 Object requested from stream sucessfully
INFO** 0415 11:20:23.549705 48 Beacon.go:87 10.12.0.1 -> Message: Hello network! 74461 from 10.12.0.5
INFO** 0415 11:20:25.770831 48 Beacon.go:93 Object requested from stream sucessfully
INFO** 0415 11:20:27.354567 48 Beacon.go:87 10.12.0.1 -> Message: Hello network! 26720 from 10.12.0.4
INFO** 0415 11:20:28.714083 48 Beacon.go:93 Object requested from stream sucessfully
INFO** 0415 11:20:30.446316 48 Beacon.go:87 10.12.0.1 -> Message: Hello network! 25167 from 10.12.0.1
INFO** 0415 11:20:33.903717 48 Beacon.go:93 Object requested from stream sucessfully
INFO** 0415 11:20:33.903791 48 Beacon.go:87 10.12.0.1 -> Message: Hello network! 31648 from 10.12.0.3
INFO** 0415 11:20:35.362267 48 Beacon.go:148 Object published to stream sucessfully
INFO** 0415 11:20:35.373352 48 Beacon.go:93 Object requested from stream sucessfully
INFO** 0415 11:20:35.538254 48 Beacon.go:87 10.12.0.1 -> Message: Hello network! 74461 from 10.12.0.5
INFO** 0415 11:20:40.626506 48 Beacon.go:93 Object requested from stream sucessfully
```

Figure 4.12: Publisher Logs from Traffic Generation

The logs show the UDP beacon program being started, the sending of messages, the creation of the Multichain stream and the request and publishing of objects from/to the stream. The logs for a consumer node are the same as the publisher logs except for the object publishing logs as only publishers can publish objects to the stream.

There were some implementation issues that meant the original design was not implemented in this project's time-frame. The main factor that prevented further implementation was that combining parts of the project was not achieved. A mesh network simulation was created but could not be used for communication with the NDE emulations.

[7] `http://rlict.galileo.edu/`

ndnSIM was determined to be unsuitable to implement this project's design as it had been created for a more specific purpose. Trying to change the behaviour of this tool would have been difficult and time consuming so an alternative was required and this is when ns-3 was chosen.

Another issue during implementation was the misinterpretation of the behaviour of the mesh network configuration when being used with NDE. When NDE was initialised, the next step was using a mesh configuration for the Docker nodes to communicate over. A mesh configuration was created and due to a misinterpretation of the original UDP beacon program, it was believed to be working. Once the beacon program was more carefully examined, it was determined that the mesh configuration was not working correctly, as nodes could not communicate with each other, only with themselves.

Due to time constraints and configuration issues with the simulations no tests were carried out on the desired framework and therefore no data was collected to compare this DWeb framework against NDN.

The process of implementing this framework could have been done better. Instead of fully committing to one simulation tool at the very beginning (ndnSIM), a time of initial research and comparison should have taken place which would have reduced the time spent configuring and testing other tools to ultimately find out they were not suitable for the required task. Another part of the research process that could be improved is better time management between researching existing projects, such as NDN, and implementing the desired DWeb framework.

During this project's implementation, several existing tools were used and this introduces a variability in the time needed to understand, configure and use software tools developed by others. This process took too much of the available time for this project and therefore left insufficient time for development of the DWeb framework.

4.6 Summary

This chapter presented the achieved implementation in this project's period and an evaluation of the initial emulation system created and the process in creating the initial system. The implementation was not compared against NDN due to time constraints.

The simulation system implemented provides a starting point for future work to continue development to create a full implementation of this project's design and compare it against new existing frameworks such as NDN.

Some investigation was carried out into determining the cause of these implementation

issues and the main factors were determined to be time management, the process of choosing tools to use for the implementation and the overhead of configuring the chosen tools.

5 Conclusion

5.1 Overview

This project's aim was to implement a version of the DWeb framework's design in order to be tested and compared against existing internet frameworks, such as NDN. This project's motivation includes increasing a network's physical security, preventing government censorship or complete shutdowns of internet and promoting net neutrality. An initial version of the simulation system was created utilising the CSMA protocol for node communication and multichain for blockchain storage. A simple form of network traffic simulation is created by using a UDP beacon program where objects are published and retrieved from a multichain stream when a message is sent or received.

Most of this project's research objectives were achieved during this project's period. NDN was examined, as it is a new potential framework for the internet. The DWeb design presented at the beginning of this project was examined and understood. Several network simulators were investigated to find a suitable one to implement the DWeb protocol, such as ndnSIM and ns-3, with ns-3 being determined as the satisfactory tool. Using ns-3, the IEEE mesh configuration was achieved. Multichain was selected as the blockchain framework to use and had existing behaviour for object storage that suited this project's needs. Linking the blockchain solution to the mesh network simulation is the next step needed to progress this implementation further towards its goal.

The implemented system was not tested as it did not include the necessary parts of the design to warrant data collection in order to be compared against NDN.

Future work is needed to create the desired framework for testing and to meet the research objectives of this project, however, this dissertation can be used as a starting point for further implementation.

5.2 Future Work

The current implementation of this novel DWeb protocol does not contain all the design aspects and therefore should be improved upon. There are several main aspects that can be examined when adding to the current simulation configuration that are detailed below in order to create a system that fully satisfies the design requirements of this DWeb protocol.

5.2.1 ns-3 Mesh Functionality in Combination with Docker Containers

One main piece of architecture that makes this DWeb protocol is the use of mesh networks. Using the mesh protocol to allow nodes in the network to communicate would be a major step in creating a full implementation of the design.

The first step would be to investigate why the CSMA protocol works in allowing the Docker nodes to send messages between each other. Then once that is established, examine the potential areas that could be altered to allow mesh communication. This could be modifications in the ns-3 mesh configuration or changes to the behaviour of Linux Bridges that link the ns-3 simulation to the external Docker nodes.

The implementation of this behaviour should be the first priority of any future work as it applies a key part of the DWeb protocol behaviour to the simulation test bed.

5.2.2 Implement Blockchain Behaviour within ns-3 Itself

A potentially different future work path that could be taken is attempting to implement the desired blockchain storage behaviour in the ns-3 program itself. By creating the blockchain behaviour within the ns-3 program, this avoids the need for a tool such as NDE that links an external application to an ns-3 simulation.

This could potentially remove the current problem faced where external nodes that are Docker containers cannot communicate over the mesh protocol. The simulation file could contain all the necessary behaviour to implement this project's design and avoid the need for configuration of multiple external tools.

Some examples exist[1], these could be used as a starting point and changes in the programs actions could be introduced as needed to create the desired blockchain storage behaviour.

[1] `https://github.com/sapgan/NS3-IoT-Simulator`

5.2.3 Test and Compare Against NDN or Similar Protocols

Once a fully feature implementation is created then data collection can begin in order to analyze the protocols behaviour. A certain set of performance metrics should be decided on and then used when comparing against the NDN protocol. This comparison should provide an analysis of the performance and viability of the DWeb protocol and whether it is an improvement over NDN.

Bibliography

[1] Darrell M West. Internet shutdowns cost countries $2.4 billion last year. *Center for Technological Innovation at Brookings, Washington, DC*, 2016.

[2] NetBlocks. Internet being restored in iran after week-long shutdown - netblocks, 2019. URL `https://netblocks.org/reports/internet-restored-in-iran-after-protest-shutdown-dAmqddA9`. Accessed: 2020-03-23.

[3] Solid. About | solid, 2020. URL `https://solid.inrupt.com/about`. Accessed: 2020-04-19.

[4] G. R. Hiertz, D. Denteneer, S. Max, R. Taori, J. Cardona, L. Berlemann, and B. Walke. Ieee 802.11s: The wlan mesh standard. *IEEE Wireless Communications*, 17(1):104–111, February 2010. doi: 10.1109/MWC.2010.5416357.

[5] Systems Innovation. Decentralized web, 2020. URL `https://systemsinnovation.io/decentralized-web-articles/`. Accessed: 2020-03-21.

[6] Decentralized Web Summit. Decentralized web summit 2018: Global visions/working code, 2018. URL `https://www.decentralizedweb.net/about/`. Accessed: 2020-03-22.

[7] I. F. Akyildiz and Xudong Wang. A survey on wireless mesh networks. *IEEE Communications Magazine*, 43(9):S23–S30, Sep. 2005. doi: 10.1109/MCOM.2005.1509968.

[8] Uroš M Pešović, Jože J Mohorko, Karl Benkič, and Žarko F Čučej. Single-hop vs. multi-hop–energy efficiency analysis in wireless sensor networks.

[9] R. Ramanathan and J. Redi. A brief overview of ad hoc networks: challenges and directions. *IEEE Communications Magazine*, 40(5):20–22, May 2002. ISSN 1558-1896. doi: 10.1109/MCOM.2002.1006968.

[10] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. URL https://bitcoin.org/bitcoin.pdf. Accessed: 2020-02-23.

[11] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang. An overview of blockchain technology: Architecture, consensus, and future trends. In *2017 IEEE International Congress on Big Data (BigData Congress)*, pages 557–564, June 2017. doi: 10.1109/BigDataCongress.2017.85.

[12] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, kc claffy, Patrick Crowley, Christos Papadopoulos, Lan Wang, and Beichuan Zhang. Named data networking. *SIGCOMM Comput. Commun. Rev.*, 44(3):66–73, July 2014. ISSN 0146-4833. doi: 10.1145/2656877.2656887. URL https://doi.org/10.1145/2656877.2656887.

[13] Lixia Zhang, Deborah Estrin, Jeffrey Burke, Van Jacobson, James D Thornton, Diana K Smetters, Beichuan Zhang, Gene Tsudik, Dan Massey, Christos Papadopoulos, et al. Named data networking (ndn) project. *Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC*, 157:158, 2010.

[14] Ge Ma, Zhen Chen, Junwei Cao, Zhenhua Guo, Yixin Jiang, and Xiaobin Guo. A tentative comparison on cdn and ndn. In *2014 IEEE international conference on systems, man, and cybernetics (SMC)*, pages 2893–2898. IEEE, 2014.

[15] NDN Project. Named data networking: Execute summary - named data networking (ndn). URL https://named-data.net/project/execsummary/. Accessed: 2020-02-25.

[16] Paul Mockapetris et al. Domain names-concepts and facilities. 1987. doi: 10.17487/RFC1034.

[17] A. Afanasyev, X. Jiang, Y. Yu, J. Tan, Y. Xia, A. Mankin, and L. Zhang. Ndns: A dns-like name service for ndn. In *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–9, July 2017. doi: 10.1109/ICCCN.2017.8038461.

[18] H. Yuan, T. Song, and P. Crowley. Scalable ndn forwarding: Concepts, issues and principles. In *2012 21st International Conference on Computer Communications and Networks (ICCCN)*, pages 1–9, July 2012. doi: 10.1109/ICCCN.2012.6289305.

[19] M. F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and B. Mathieu. A survey of naming and routing in information-centric networks. *IEEE Communications Magazine*, 50(12):44–53, December 2012. ISSN 1558-1896. doi: 10.1109/MCOM.2012.6384450.

[20] Dr. Jason Ernst, Dr. Zehua (David) Wang, Saju Abraham, John Lyotier, Chris Jensen, Melissa Quinn, and Dana Harvey. Rightmesh whitepaper. URL https://www.rightmesh.io/docs/RightMesh_WP6.pdf. Accessed: 2020-03-03.

[21] RightMesh. How it works | rightmesh. URL https://www.rightmesh.io/technology/how-it-works. Accessed: 2020-03-04.

[22] SmartMesh Foundation PTE. LTD. The smartmesh project, 2017. URL https://smartmesh.io/SmartMeshWhitePaperEN.pdf. Accessed: 2020-03-09.

[23] Ethereum. web3.js - ethereum javascript api, 2020. URL https://web3js.readthedocs.io/en/v1.2.6/. Accessed: 2020-04-25.

[24] Skycoin Project. Skycoin business whitepaper edition 1.2, 2020. URL https://downloads.skycoin.com/whitepapers/Skycoin-Whitepaper-v1.2.pdf. Accessed: 2020-04-11.

[25] Althea. Althea - how it works, 2020. URL https://althea.net/how-it-works. Accessed: 2020-04-11.

[26] Althea. Althea whitepaper v1.5, 2019. URL https://althea.net/whitepaper. Accessed: 2020-04-11.

[27] BlockMesh. Welcome to blockmesh, 2020. URL https://www.blockmesh.io/How-it-works.php. Accessed: 2020-04-11.

[28] BlockMesh. Blockmesh - whitepaper, 2017. URL https://www.blockmesh.io/pdf/BlockMesh-White_Paper-1.pdf. Accessed: 2020-04-12.

[29] Ammbr. Ammbr - whitepaper, 2018. URL https://ammbr.com/docs/2018/11/Ammbr_Whitepaper.pdf. Accessed: 2020-04-12.

[30] Anna Gross and Murgia Madhumtia. China and huawei propose reinvention of the internet | financial times. *URl: https://www.ft.com/content/c78be2cf-a1a1-40b1-8ab7-904d7095e0f2*, 2020.

[31] IEEE standard for information technology—telecommunications and information exchange between systems local and metropolitan area networks—specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, pages 1–3534, Dec 2016. ISSN null. doi: 10.1109/IEEESTD.2016.7786995.

[32] David Johnson, Yin-chun Hu, David Maltz, et al. The dynamic source routing protocol (dsr) for mobile ad hoc networks for ipv4. Technical report, RFC 4728, 2007.

[33] M. S. Siddiqui and C. S. v. Security issues in wireless mesh networks. In *2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07)*, pages 717–722, April 2007. doi: 10.1109/MUE.2007.187.

[34] Richard A Paulson and James E Weber. Cyberextortion: an overview of distributed denial of service attacks against online gaming companies. *Issues in Information Systems*, 7(2):52–56, 2006.

[35] Iuon-Chang Lin and Tzu-Chun Liao. A survey of blockchain security issues and challenges. *IJ Network Security*, 19(5):653–659, 2017.

[36] Dylan Yaga, Peter Mell, Nik Roby, and Karen Scarfone. Blockchain technology overview. *arXiv preprint arXiv:1906.11078*, 2019.

[37] F. Dai, Y. Shi, N. Meng, L. Wei, and Z. Ye. From bitcoin to cybersecurity: A comparative study of blockchain application and security issues. In *2017 4th International Conference on Systems and Informatics (ICSAI)*, pages 975–979, Nov 2017. doi: 10.1109/ICSAI.2017.8248427.

[38] nsnam. About | ns3, 2020. URL `https://www.nsnam.org/about/`. Accessed: 2020-02-23.

[39] Rachna Chaudhary, Shweta Sethi, Rita Keshari, and Sakshi Goel. A study of comparison of network simulator-3 and network simulator-2. *International Journal of Computer Science and Information Technologies*, 3(1):3085–3092, 2012.

[40] nsnam. ns3 tutorial, 2019. URL `https://www.nsnam.org/docs/release/3.30/tutorial/ns-3-tutorial.pdf`. Accessed: 2020-03-11.

[41] nsnam. Howto make ns-3 interact with the real world, 2017. URL `https://www.nsnam.org/wiki/HOWTO_make_ns-3_interact_with_the_real_world`. Accessed: 2020-04-07.

[42] nsnam. Howto use linux containers to set up virtual networks - nsnam, 2017. URL `https://www.nsnam.org/wiki/HOWTO_Use_Linux_Containers_to_set_up_virtual_networks`. Accessed: 2020-03-27.

[43] Coin Sciences. Multichain, 2020. URL `https://www.multichain.com/`. Accessed: 2020-02-23.

[44] Coin Sciences. Getting started with multichain | multichain, 2020. URL `https://www.multichain.com/getting-started/`. Accessed: 2020-02-23.

[45] Gideon Greenspan. Multichain private blockchain-white paper. *URl: http://www.multichain.com/download/MultiChain-White-Paper.pdf*, 2015.

[46] Gideon Greenspan. Introducing multichain streams | multichain, 2016. URL https://www.multichain.com/blog/2016/09/introducing-multichain-streams/. Accessed: 2020-03-16.

[47] Jose Alfredo Alvarez Aldana. Ns3 docker emulator | github repository, 2019. URL https://github.com/chepeftw/NS3DockerEmulator. Accessed: 2020-03-11.

[48] Jose Alfredo Alvarez Aldana. Ns3 docker emulator, 2019. URL https://chepeftw.github.io/NS3DockerEmulator/. Accessed: 2020-03-11.

[49] Docker Inc. What is a container | app containerization | docker, 2020. URL https://www.docker.com/resources/what-container. Accessed: 2020-03-11.

[50] Docker Inc. Docker overview | docker documentation, 2020. URL https://docs.docker.com/engine/docker-overview/. Accessed: 2020-03-14.

[51] Nuutti Varis. Anatomy of a linux bridge. In *Proceedings of Seminar on Network Protocols in Operating Systems*, page 58, 2012.

[52] Maxim Krasnyansky and Maksim Yevmenkin. Universal tun/tap device driver. *URL: http://www.kernel.org/pub/linux/kernel/, FILE: people/marcelo/linux-2.4/Documentation/networking/tuntap.txt*, 2007.

[53] nsnam. ns-3 ns3::tapbridge class reference, 2019. URL https://www.nsnam.org/docs/release/3.30/doxygen/classns3_1_1_tap_bridge.html#details. Accessed: 2020-03-15.

[54] NDN. Ndn packet format specification, 2020. URL https://named-data.net/doc/NDN-packet-spec/current/. Accessed: 2020-04-25.

[55] Spyridon Mastorakis, Alexander Afanasyev, Ilya Moiseenko, and Lixia Zhang. ndnsim 2.0: A new version of the ndn simulator for ns-3. 01 2015.

[56] M. A. To, M. Cano, and P. Biba. Dockemu – a network emulation tool. In *2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops*, pages 593–598, 2015.

A1 Appendix

A1.1 Simulation Code Repository

The following is the URL for the code repository of this project: `https://github.com/Donegaan/NS3DockerEmulator`

Most of the work was done on the 'bchain-get-stuff-working' branch.

A1.2 UDP Beacon Repository

The following is the URL for the code repository of the UDP beacon program used for traffic generation: `https://github.com/Donegaan/Beacon`