University of Dublin

TRINITY COLLEGE

Integration of Blockchain and Named-Data Networking

Samuil Hristov

Final Year Project April 2019

Supervisor: Dr. Stefan Weber

School of Computer Science and Statistics

O'Reilly Institute, Trinity College, Dublin 2, Ireland

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Samuil Hristov

April 23, 2019

Permission to Lend and/or Copy

Acknowledgments

Firstly, I need to thank my supervisor, Dr. Stefan Weber for all of his hard work. For allocating time every week to help me with this project and for returning huge report drafts with ridiculously detailed feedback in no time.

Secondly, I'd like to thank my tutor, Dr. Séamus(Shay) Lawless. Without his help, I wouldn't have ever graduated from Trinity.

Thirdly, I'd like to thank my second reader Dr. Jonathan Dukes for not throwing me out of this University in first year, when I wrote a last minute abomination of an ARM Calculator for a project, that would only run if it did subtraction with a pair of three-digit,positive integers with non-zero digits where the difference was always positive.

Finally, I'd like to thank my friends and family for all their support in this arduous journey.

Samuil Hristov

University of Dublin, Trinity College

April 2019

Integration of Blockchain and Named-Data Networking

Samuil Hristov, B.A.(Mod.)
University of Dublin, Trinity College, 2019

Supervisor: Dr. Stefan Weber

Abstract: Information-Centric Networking (ICN) is a communication approach that makes content 'living' in a network the focus of communication, in contrast to the traditional communication between hosts based on IP addresses. In order to ensure the validity of content, it should be signed by its producer and in order to ensure that information is only accessible to a select number of consumers, content may need to be encrypted. Named-Data Networking (NDN) is an ICN implementation that provides a framework for the exchange of named content and a certificate-based security mechanism to sign and encrypt/decrypt content. The certificates in NDN are held at individual nodes and have to be requested by other nodes in a network in order to verify content, leading to additional latency once content has been retrieved.

This project has extended NDN's certificate management system by distributing certificates based on a distributed ledger i.e. a blockchain. Transactions such as the creation and removal are announced by nodes to miners which incorporate the transaction into new blocks and distribute these to nodes for inclusion into the Blockchain. Nodes have access to the current set of certificates through their Blockchain and can verify content through these certificates.

NDN currently implements a signature verification standard by using the standard X.509 Format for certificates. Certificates represent the validity of a public key that has been used to sign a piece of data. All Data packets in NDN are issued certificates. They are stored in the Network's public file system, known as the Public Information Base(PIB).

This paper recommends the additional hashing of these certificates in a Blockchain to improve security and reduce look-ups for each node in an NDN network when receiving Data packets.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Named Data Networking is an interesting new paradigm in the space of network architectures. Years on since Bell Labs' work on telephony, networking research had assumed that telephony is the right model for data networking[1]. This model implied that there had to be a determined route between two points for communication to happen - the setup for which was costly. This was proven not to be the case in Paul Baran's research paper titled "On Distributed Communication Networks" in 1964, which was widely disregarded until the practical application of his work in ARPAnet(Sept,'71).MIT Senior Researcher David Clark's paper on end-to-end principle confirmed the same and it became apparent that networking solved the telephony problem. However, we are still using this same architecture which was invented as a solution for a problem that is now five decades old, and the Internet of today is not facing the same challenges. CISCO predicted in 2013[2][3], that by the end of 2017, the annual traffic of the internet would exceed 1.4 zettabytes with almost 80% of that being video traffic.

This is why, in the age of content delivery, Named Data Networking and other Information Centric Networking architectures aim to move away from the source-destination pairwise method of IP communication which is inherently limited. Instead, NDN proposes a Name-based approach where each node in a network can request content based on the name of a piece of data it requires.

This project aims to improve on the security implemented in Named Data Networking.

The aim of this introduction is to give background for motivation as well as lay out the structure of this paper.

1.1 Motivation

Traditional NDN: The traditional NDN architecture presents a security architecture not dissimilar to the Central Authority architecture conceptualized by Loren Kohnfelder in 1978[4]. It provides a public file[5] system where all nodes can check the entries for other nodes issued by a central, trusted third party called the Certificate Authority(CA) which signs each entry or 'certificate'. This is all quite simple in a MiniNDN test environment, as described in the Certificates section in Chapter 2. In the case of a general or non-experimental NDN deployment[6] however, this is not the case.

The Network Manager would have to request a certificate verified by the CA. The manager, in turn, must also trust the CA and therefore the CA must be authentic and trusted by most networks. Trusted companies that provide this CA service include Verisign, GeoTrust,Symantec, etc...

The Network Manager would then provide a certificate for every node in the network which would in turn be signed by the root certificate provided by the authenticated CA. When a node wants to verify a certificate, like it would if it was receiving a Data packet, it would have to verify all of the certificates in the network hierarchy meaning it would also have to verify the root network certificate, for which it would have to contact the Certificate Authority. This means that verifying certificates can potentially involve a lot of communication.

This could be avoided by introducing a Blockchain which each node could keep a copy of. This would allow nodes to verify certificates without needing to contact the CA.

To visualize this, the following example is presented in Figure 1.1. An NDN Network with an external CA has a root namespace **/ndn/** and in it there is a node called A.

The root namespace is issued a certificate from the Certificate Authority which in this example is external e.g: Verisign. The root then issues a certificate for Node A. Node A in turn issues a certificate for the clock module in its namespace. These certificates allow all of the entities in the network to verify that their data is their own. If another node, like Node B for example, wants the time from Node A's clock, the node would simply express an Interest. When the clock replies with a Data packet, Node B would have to verify the data by verifying clock.cert, a.cert and root.cert. In order to verify root.cert,node B would need to contact the CA - which in the case of a real NDN environment might well be Verisign.

This project recognizes that this process isn't efficient use of bandwidth as nodes would constantly have to contact the CA, performing look-ups for the sake of root certificate verification and that this can be mitigated if each node had a Blockchain of verified certificates at its disposal.

1.2 Aims

Blockchain was an architecture proposed to store and verify transactions **reliably** in a decentralized currency system. Instead of transactions, this project aims to store certificates in a similar fashion. The goal is to do so efficiently, without increasing computational load on individual nodes in the system or increasing significantly the bandwidth use.

It is important to note that there isn't a monetary incentive for doing this "Proof-of-Work"[7] so each network should have a dedicated group of miners which verify blocks. The assumption here is that all(or most) of the nodes will not be malicious and will not be pooling their resources to attack the network. This means that in order for one to alter the list of certificates, they would have to have more computing power than the entire network of miners. This will allow for safer communication between nodes in a network.

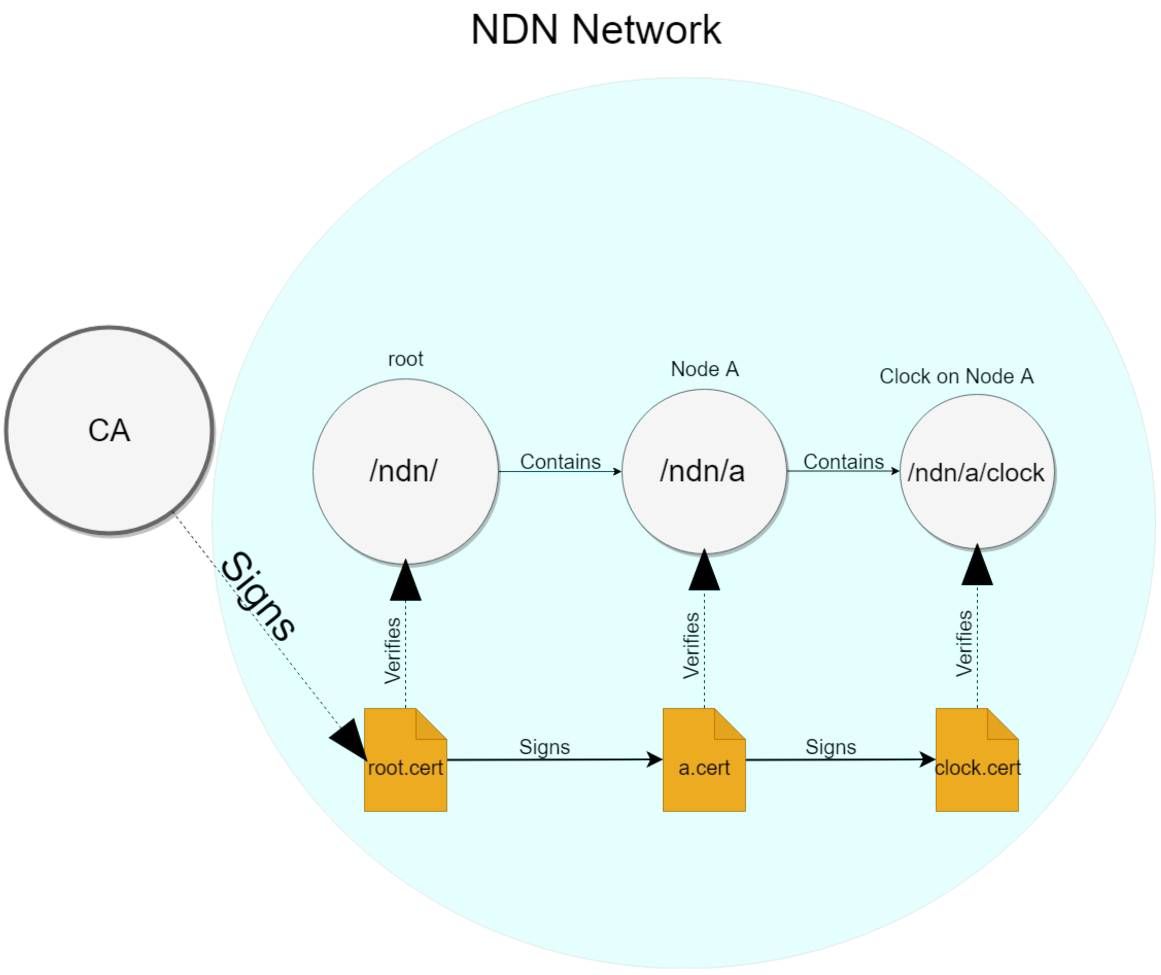


Figure 1.1: Certificate Hierarchy

1.3 Road-map

"Begin at the beginning," the King said gravely, "and go on till you come to the end: then stop."

- Lewis Carroll, *Alice in Wonderland*

This paper is structured as follows: State of the Art(Lit. Review), Design and Implementation, and Evaluation.

- Chapter 2 contains the background information required for the scope of this project.It also contains literary reviews of the papers which discuss the State of the Art. It looks at a ranking system for the papers reviewed and also provides critique for each one.

- Chapter 3 discusses in detail the design of the Blockchain solution in NDN. It outlines all aspects of the conception of the data types, including solutions, design challenges and alterations that were made along the way.
- Chapter 4 describes the implementation of this paper's solution
- Chapter 5 goes on to evaluate the working solution by discussing different experiments and topologies. It presents graphs which illustrate the performance differences in different topologies

The paper concludes with Chapter 6 which sums up the work that's been presented and outlines any future work that might be undertaken regarding the project.

Chapter 2

State of the Art

The State of the Art in NDN and Blockchain was thoroughly investigated when researching this project. This was done to establish what technologies are currently implemented in Named Data Networking and also to identify similar projects to our own solution.

It is important to note this was done to a lesser extent for Blockchain because it is a supplemental technology in the Cryptocurrency space and is relatively simplistic(i.e. a vector of blocks all hashed with the previous block's hash).The Blockchain "itself requires minimal structure"[8]. Therefore, the different nuances of the technology which deviate from the fundamentals weren't investigated as thoroughly as it was beyond the scope of the project specification which asked for a proof of concept Blockchain. A 'full fat' Blockchain solution involves miner nodes solving basic "Proof-of-Work" or cryptographic puzzles[9], based on a timestamped transaction, appending them to a chain, and advertising that chain to all nodes in the network.

This chapter is divided into Knowledge and Literary Review sections. The knowledge contextualizes this project by giving in depth background on all the technologies in use.

The Literary Review then describes each individual paper from which this information was obtained.

2.1 Background and Summary

In 2006, Van Jacobson likens the ICN solution for the IP problem to the Copernican solution for the Solar System problem.[10] What he suggests by saying this is that IP was a(good) solution but for an entirely different problem than what the Internet is today. "While it is entirely doable to predict the movement of planetary bodies by taking the Earth to be the center of the universe, it is incredibly complex. This is because the point-of-view is wrong." [11]

In a similar way, despite the plethora of problems that IP was able to address at the turn of the last century - in today's world it causes more problems than it solves. This is because the IP abstraction's benefits no longer outweigh the drawbacks. In fact, in order for one to even connect to the Internet, we need to break the IP abstraction and do a complicated Dynamic Host Configuration Protocol dance which dictates that in order for us to gain access to a website, we must first gain an IP, which we must ask for, of a DHCP, which will be given to us once we ask an Address Resolution Protocol server, to send our MAC to a DHCP server, which can then give us an IP, which we can then use to ask for the nearest DNS server to respond to our request for a website IP.

NDN does away with all of this by just having nodes broadcast the Names of the data they want in an Interest packet. It is a completely different way to think about Networking, because the network participants or nodes no longer care where the Data is going.

- Telephony concerns itself with having a direct end-to-end path in order to establish communication.
- IP concerns itself with having a connected graph and a destination address
- NDN doesn't concern itself with having a destination for a particular request but rather just propagating that request through the network.

Dissemination Networking is what Van Jacobson Proposes. Ask for Data - not an IP address. The background for this research splits off into two parts. The Blockchain technology is largely based on cryptography and hashing. NDN on the other hand has its roots in networking.

2.2 State of The Art - Knowledge

This section will give a brief overview of how these two technologies - Blockchain and NDN, and their components, work.

The paper splits the Knowledge section into a Blockchain description, followed by an overview of NDN and all of its main components.

2.2.1 Blockchain

A Blockchain is a distributed ledger of immutable digital records[12]. It was conceptualized in a paper called "Bitcoin - A Peer to Peer Electronic Cash System". The paper was written by Satoshi Nakamoto, an enigma of the cryptology world, who has even been speculated to be one of Trinity's own graduate crypto students - Michael Clear. The Blockchain is the main component of the Bitcoin cryptocurrency. It allows for decentralized transactions by solving the Byzantine Generals Problem[13]. This is also known as having Byzantine Fault Tolerance(BFT). The Byzantine Generals Problem was first theorized about in a paper by the same name, authored by Leslie Lamport, Robert Shostak, and Marshall Pease in 1982.

The Byzantine Generals Problem proposes a situation where a number of Generals from Byzantium have laid siege on an enemy city. The events of the problem develop somewhere in the Byzantine Empire, 1000 years ago.

Each General has an army. The problem they are faced with is taking an enemy city. Each of the Generals commands their own army. The Generals are aware that they can only take the city if they all order all of their respective armies to attack together in a **coordinated effort**. However, no General will commit to the attack without assurance that the others will also commit. In order to commit to an attack, the Generals must agree on a battle plan and a time of attack. There are a couple of problems with this:

- The messengers could be captured
- The Generals would need to reply with confirmations
- The Generals can change their minds, or be traitors and lie

The problem is that even if they agree on a time, no General can ever be sure that the others will attack with him because **confirmations are not instant.** This means that by the time a messenger arrives at General A's camp declaring General B's confirmation for an attack at a certain time, General B could've changed their mind and could've sent another messenger to relay the new message. This renders the original messenger's declaration invalid but General A doesn't know that. This cycle continues and the Generals can never be 100% certain when to attack because they can never know what the other Generals are thinking in that instant of time.

The Blockchain addresses the inability of traditional systems, without a central authority, to determine that a certain resource hasn't been spent more than once. The reason banks or "central authorities" can vouch for transactions, and guarantee against double spending, is because the transactions don't happen instantaneously. They are each, individually confirmed, and also, should the bank make a mistake, it covers it(insurance).

The reason this isn't the case in a Blockchain distributed ledger system like Bitcoin, is because miners do Proof-of-Work in a peer-to-peer network. The network timestamps each transaction, making them secure by hashing them into a hash-based proof-of-work. The resulting record cannot be altered without redoing this proof-of-work. This consensus mechanism[14] mirrors a voting system. There is a network, where nodes must know if an operation is valid. In order to do this, the nodes pool their CPU resources to do proof-of-work on transactions nodes accept to be valid, in order to verify them. If a transaction does not get their vote of confidence, they don't work on hashing that into the Blockchain. If the majority of nodes in the system are "good", then the largest Blockchain generated will be valid.

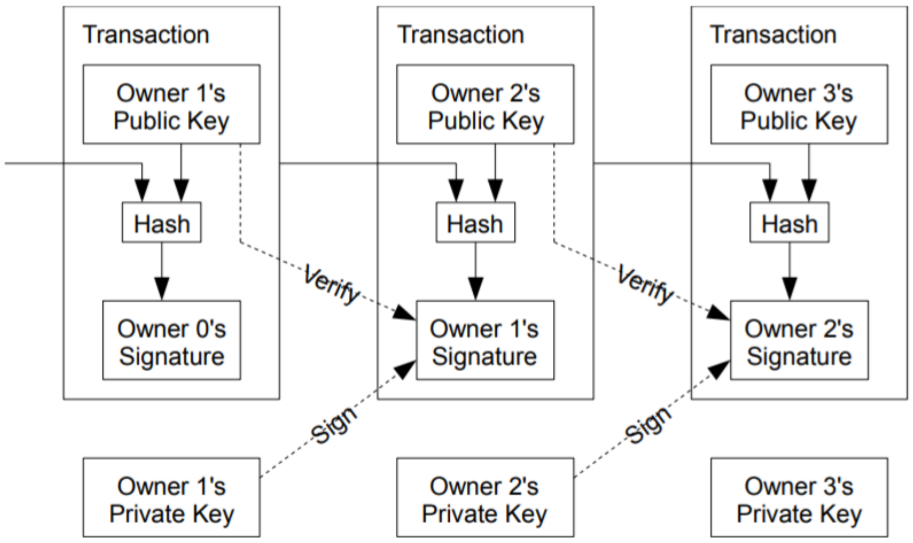


Figure 2.1: Blockchain structure[15]

2.2.2 NDN

Named Data Networking began life in 2010 under the NSF's Future Internet Architecture[16]. The leading effort in the NDN project has been UCLA Professor Lixia Zhang. Projects like NDN picked up traction after CCN began development at PARC, headed by Van Jacobson. NDN presents a paradigm shift for the Networking Stack. Instead of IP being the thin-waist of the stack, NDN suggests a data name abstraction, where the name of the content becomes this thin-waist.

NDN is composed of a number of modules described below. It is a name based architecture where nodes in a network can request data, not based on where it comes from but based on the name of the data. E.g.: a node requests **/ndn/a-site/important/information** instead of an IP address. This abstraction is hugely advantageous in today's Internet where content sharing is at the forefront of what the Internet delivers. NDN allows for nodes to respond to Interests, which are content requests, with Data packets which are the content. Nodes interact by sending Interests by Name. Data is retrieved in Data packets.

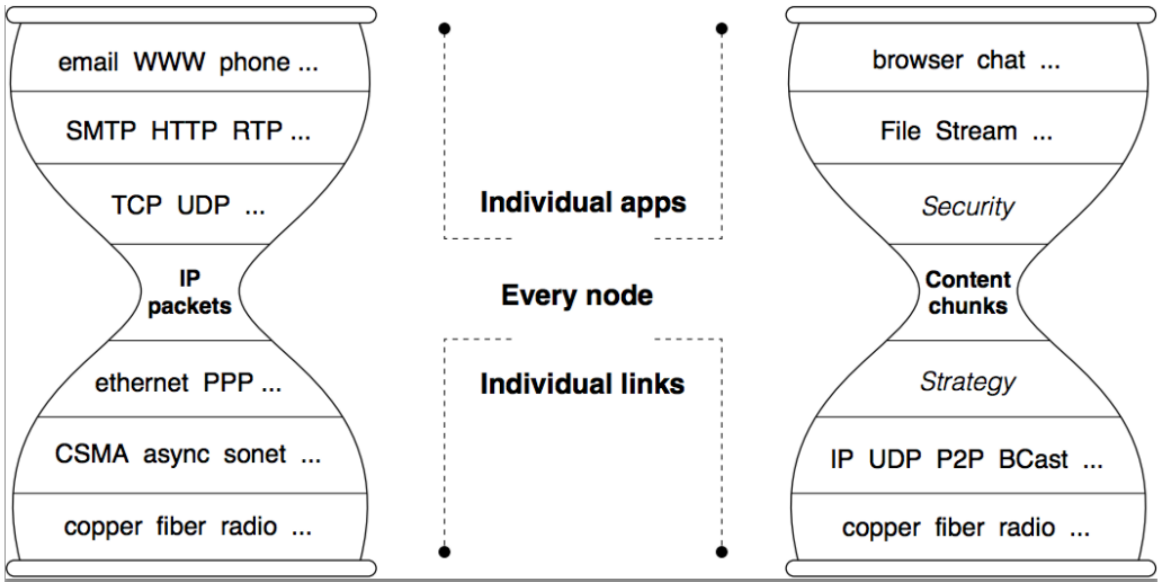


Figure 2.2: The Networking Stack[17]

2.2.3 Names

An NDN Name is a hierarchical name for NDN content, which contains a sequence of name components.[18] Each NDN name component consists of variable lengths and is separated by a '/' separator. This design is advantageous because it is human readable and allows for users to configure Interest and Data filters using regular expressions.
E.g: InterestFilter($^<ndn >< lectures >$) means listen for any interests beginning with /ndn/lectures. Named Data Networking is all about names. When presenting on NDN at MILCOM 2017 in Baltimore, Lixia Zhang said "the secret to NDN is in the names!" This is because this networking paradigm is entirely centered on the names of content. Each entity in a network has an identity, and each identity has a name. Each named entity in a network follows a hierarchical naming scheme.

Names are represented by the name class in NDN. They are made up by a URI string. In C++, they are defined using the C-style const char* array.

Packets are forwarded using Longest Name Prefix Matching(LNPM)[19].

It is similar to IP Longest Prefix Match where a routing algorithm will, upon needing to do a look-up, return the longest matching address. E.g: Consider the following IPv4

Forwarding Table(CIDR Notation):

192.168.20.16/28
192.168.0.0/16

Figure 2.3: Forwarding Table

If the router was to look-up 192.168.20.19, both entries will “match”, but the router will only return 192.168.20.16/28 since the submask /28 is longer than the other entry’s /16 submask[20].

A similar example could be given an Interest arriving at node B asking for **/ndn/a-site/lectures/telecomms/slides** and node B’s FIB could have entries for **/ndn/a-site** and **/ndn/a-site/lectures/telecomms/** in which case it will only return the second entry.

The problem with LNPM comes from the inherent differences between address lengths in IP and NDN. In most cases, NDN names will be longer than IP as the namespace is unbounded. The difference in size is so drastic, that TCAM or SRAM memory modules for the FIB would not be large enough if the number of rules is large. The NDN FIB has to use DRAM in order to accommodate for this size difference. With such a difference in size and complexity, look-ups can take O(k) string look-ups[21].

This is why NDN employs two techniques proposed in a Washington University paper by Haowei Yuan and Patrick Crowley. The first changes the LNPM design to a binary search of hash tables reducing lookups to O(log(k)) for prefixes with k components [22] and the second technique is level pulling to improve the average case. This makes the name prefix lookup solution scalable - a necessary feature for the NDN architecture to work.

2.2.4 Faces

Faces are abstractions of physical and virtual interfaces[23]. They provide the main methods for NDN communcation. Faces hold a connection to a forwarder and support Interest/Data exchanges[24].In order to communicate in an NDN network, each entity must have a Face. Because Faces abstract away physical and virtual interfaces, it allows NDN to be used as an overlay network over existing technologies like IP[25]

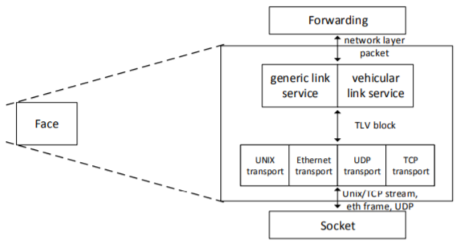


Figure 2.4: Face Architecture[26]

2.2.5 Interests

An Interest is a data packet. It is integral to the design of NDN. Interest packets are sent out when nodes require data. If a node wants a piece of information it will send out an Interest with the Name of the data it requires. That Interest is then propagated through the nodes in the network. When the Interest arrives at another node, the node checks its Content Store for a signed piece of Data with a Name matching that of the Interest's. If no Data is found in the Content Store, the node stores the Face on which the Interest arrived in its Pending Interest Table(PIT), and appends it to a list of other Faces that have requested the same Data. It then checks its Forwarding Information Base(FIB). If it has information on which node might have the Data the first node required, it will forward that Interest on to that node, if not it will discard the interest. If that node then ever comes across that data it will check its PIT and if the Interests are still "fresh", it will forward the Data packet to all of the Faces that have expressed an Interest for the Data. It will then store the Data in its content store and discard all entries in its PIT.

There are also Signed Interests, which used to be called Controller Interests. These Interests were designed because of the inherent inability of NDN nodes to send Data to each other without being prompted. This is why nodes can send Signed Interests which can request for another node to send and Interest for Data that the original node might have.

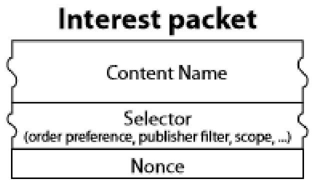


Figure 2.5: An Interest Packet[27]

2.2.6 Data

Data packets are the other integral packets to the NDN design. Data packets are sent out when there is an Interest for them. They contain a X.509 Certificate which signs each Data packet. They also have a freshness value and obviously also contain bits of data.

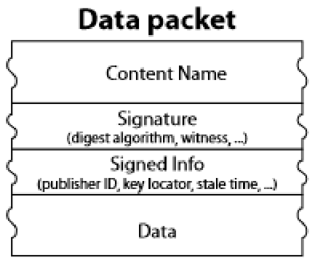
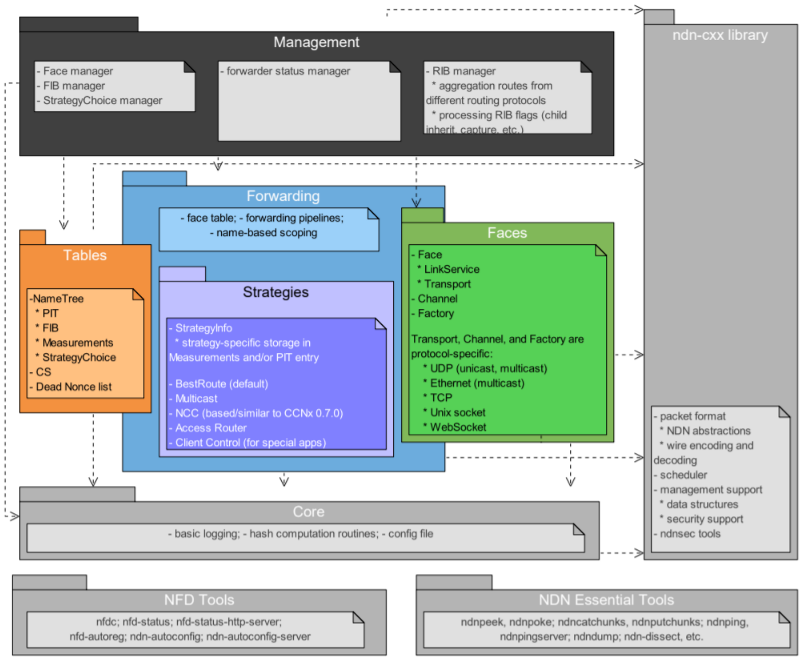

Data packet
Content Name
Signature
(digest algorithm, witness, ...)
Signed Info
(publisher ID, key locator, stale time, ...)
Data

Figure 2.6: A Data Packet[28]

2.2.7 Named-Data Forwarding Daemon

Named-Data Forwarding Daemon(NFD) is responsible for handling all packets in the network. An instance of NFD runs on every node in a network. When a packet arrives on a Face, the NFD checks if that packet is destined for the particular node and if not it discards it based on a policy to allow/disallow unsolicited Data packets. NFD is made up of the following modules: Core, Faces, Tables, Forwarding, Management, and Routing Information Base(RIB) Management.



Management
- Face manager
- FIB manager
- StrategyChoice manager
- forwarder status manager
- RIB manager
* aggregation routes from different routing protocols
* processing RIB flags (child inherit, capture, etc.)
ndn-cxx library
Forwarding
- face table; - forwarding pipelines;
- name-based scoping
Tables
-NameTree
* PIT
* FIB
* Measurements
* StrategyChoice
- CS
- Dead Nonce list
Strategies
- StrategyInfo
* strategy-specific storage in Measurements and/or PIT entry
- BestRoute (default)
- Multicast
- NCC (based/similar to CCNx 0.7.0)
- Access Router
- Client Control (for special apps)
Faces
- Face
* LinkService
* Transport
- Channel
- Factory
Transport, Channel, and Factory are protocol-specific:
* UDP (unicast, multicast)
* Ethernet (multicast)
* TCP
* Unix socket
* WebSocket
- packet format
* NDN abstractions
* wire encoding and decoding
- scheduler
- management support
* data structures
* security support
- ndnsec tools
Core
- basic logging; - hash computation routines; - config file
NFD Tools
nfdc; nfd-status; nfd-status-http-server; nfd-autoreg; ndn-autoconfig; ndn-autoconfig-server
NDN Essential Tools
ndnpeek, ndnpoke; ndncatchunks, ndnputchunks; ndnping, ndnpingserver; ndndump; ndn-dissect, etc.

Figure 2.7: NFD Structure and Components[29]

- Core - Provide common services shared between modules like hash computation routines, DNS resolver, config file, face monitoring, etc.. [29]
- Faces - the NFD implements the Face abstraction
- Tables - There are a number of tables that NFD is in charge of like the Content Store(CS), the Pending Interest Table(PIT) and the Forwarding Information Base(FIB).
- Management - Implements the NFD Protocol, allowing applications to configure NFD.
- RIB Management - The Routing Information Base is responsible for producing a consistent FIB, which is a non-trivial task because of the amount of ways it can be

updated such as through different routing protocols, command line, application's prefix registration, etc.. The RIB is a module on its own, however it is managed by NFD.

2.2.8 Named-Data Link State Routing

Named-Data Link State Routing(NLSR) is the NDN module which deals with routing when a topology is created. It reuses an already established routing algorithm - link state. Its basic functionality is to discover adjacencies and disseminate both connectivity and name prefix information[30]. NLSR works by sending out Link State Advertisements(LSA) when a network is created to collect reachability and connectivity[31] information. This means that each node has its Forwarding Interest Base populated with adjacent node names. NLSR takes time to converge proportional to CPU power in the Network.

There are two types of LSAs - *named* and *adjacency*. A name LSA will contain all the local prefixes registered locally with NLSR[32]. An adjacency LSA contains all active links of a router and is updated by an active ChronoSync module which updates active routes when NLSR sends 'hello' messages. If they time out three times, the connection has died and the adjacency LSA is updated. Whenever this LSA is updated for a router, the router advertises the new adjacency LSA to every node in the network. The latest versions of all adjacency LSAs are stored in the Link-State Database(LSDB). NDN can use the same routing algorithms that are used in IP such as Link-State and distance vectoring. There is one difference however, in the routing implementation for NDN. Whichever the implementation, routing must be able to offer multiple next hops(multipath) for nodes in the network, towards producers of Data. Fundamentally, the routing protocols are the same, with the exception that the Named-Data protocol must offer multipath.

NLSR offers a number of features, beneficial and crucial to NDN.

- Naming - NLSR uses hierarchically structured names to identify routers, routing

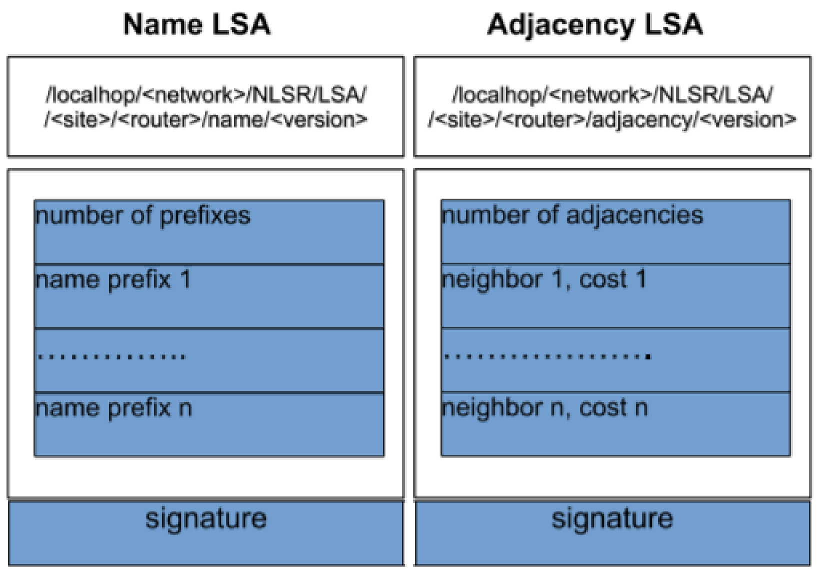


Figure 2.8: Link-State Advertisements[33]

processes, routing data and keys[34].

- Security - This feature comes as a by-product of the fact that each NLSR message is carried in an NDN Data packet which must be signed.

- Multi-Path - The main NLSR feature is multipath. While IP relies either on a single hop or limits its forwarding to multiple equal cost paths[35] in order to avoid loops - this is not an issue in NDN. While the architecture doesn't encourage forwarding loops, they aren't a big concern because the NDN architecture simply allows, when forwarding, for each node to make a decision on where to send a packet. This is what is implemented in NDN's loop detection.

2.2.9 Chronosync

Chronosync is a module used when there is a need for nodes to receive data or be updated at the same time. It is a vital module in networking configurations that run text messaging, group file sharing, screen-sharing, etc.. Chronosync runs on top of NFD and synchronizes Data and Interest packet sending and receiving. The Chronosync design splits into two parts: It maintains the state of a dataset, and it also has a logic module that responds to change[36]. Chronosync monitors datasets and discovers state changes. However, it doesn't alter the dataset - this is left to the application layer.

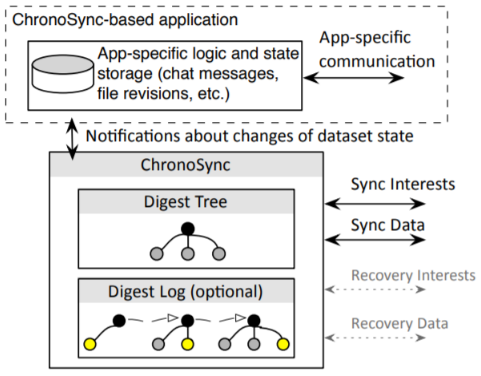

ChronoSync-based application
App-specific logic and state storage (chat messages, file revisions, etc.)
App-specific communication
Notifications about changes of dataset state
ChronoSync
Digest Tree
Digest Log (optional)
Sync Interests
Sync Data
Recovery Interests
Recovery Data

Figure 2.9: Chronosync Overview [37]

2.2.10 Content Store

The Content Store or cache is each node's local storage. It contains signed Data which the node can forward to any node expressing an interest for it. Caching is one of NDN's main features. There are two types - off path and on path caching. Off path caching, like the name suggests, disregards content path and just aims to replicate the content in a network[38]. On Path caching is limited to the content propagated along the delivery path[39] meaning the data cannot be cached outside the route from the node expressing the Interest for the Data and the producer of said Interest.

Policies

A number of policies exist for caching:FIX(0.9), DC and ProbeCache. A recent solution - ProbPD investigates content popularity as a heuristic for caching. All of the high-end solutions perform near identically.

2.2.11 Security

Ralph Merkle describes the problem with the classic[40] Authenticated Public Key Distribution protocol. He describes how each node in a network generates a public key and stores it in a file system. If two nodes wish to agree on a common key in order to interact, they look up the Public Key portion of the other node. Then each send a **session key** encrypted with the other node's public key. Once in agreement, this key is secret and authenticated and can be used by both nodes to communicate.

The problem with this approach is that a centralized file system is a single point of failure and is prone to attack. The attacks can be one of two - the public key elements can be altered in the file system(e.g. the attacking node could set another node's public key to be its own), and secondly, the private keys can be lost.

The alternative to this approach is implemented in NDN and it is to introduce Certificates and a Certificate Authority(CA). Certificates refer to the binding of a node's keys to its identity i.e. which key belongs to which node. It is very important that there is a robust method of determining this key-identity bond and perhaps even more importantly, to ensure that it is immutable. “In NDN, every entity that produces data needs to obtain an NDN certificate to prove the ownership of its namespace and cryptographic materials(public key)”[41]. The security process occurs as following: First, we start off with the Achilles Heel for any networking security protocol - **bootstrapping**. This is the process of obtaining all trust anchors and certificates.

Before that however, we need to just quickly define trust anchors(policies). They refer to the rules set by each entity to only accept packets of a desired format of names and name relationships[42]. It is also important to note that these rules are governed by each identity at the Application Layer.

Back to bootstrapping: In order to do this, nodes must obtain a namespace and then a certificate for that namespace from a CA that they trust[43]. The order for entities receiving certificates is hierarchical. This means that if a user(entity) has obtained a certificate, it can delegate certificates to other entities within its namespace. Because trust anchors are determined at the Application Layer - the only prerequisite[44] for security bootstrapping is allocating names. As long as an entity has a name, it can receive a certificate if allowed by the owner of the namespace. Each entity has its own trust anchors but should naturally trust the Certificate Authority. In the case of the root namespace - that is the recognized CA by the root user, as for the rest of the names in the namespace, that is the root namespace.

NDNCERT is a library found in NDN-CXX which provides the tools necessary for a name to obtain a certificate in an NDN network. It generates certificates for trust anchors automatically and manages them in a daemon[45]. It runs in an instance called an agent and maintains all certificates generated by NDNCERT.

As well as that NDNCERT can revoke certificates automatically if they are considered unfit. There is a check done on each certificate and if it is generated illegally, then in the case of the MiniNDN emulator, the code will throw an error and exit the session.

Data packets are signed at creation time[46]. This design choice is critical in the integrity of data packets in NDN because this means that a data packet physically cannot be sent off without being signed. The important bit here isn't so much that all data sent is signed as much as is the inverse - that all data received can be checked for a signature.

This makes Data verification twofold: Firstly, NDN uses Trust Anchors - so if a node is expecting data, it can define a trust anchor that states that the data can only arrive from one particular **name**. Secondly, and this is the traditional, cryptographic verification method, once the Data packet is received and passes the trust anchor check, the consumer of the Data packet retrieves the corresponding certificate for the producer which is identified by the key name section in the packet[47]. The Certificate will then recursively point to the root certificate and if all certificates along the way are valid that means that the Data packet itself is signed with a valid signature.

2.3 Related Work

There are a number of papers that I've come across that deal with security in NDN. As expected, there is plenty of work done in this field because it is a major concern for any network architecture.

The NDN guides were a big help when it came to researching security. Spyridon Mastorakis' paper on NDN Security Support outlines all major features found in the Security module of NDN. This paper cited the Merkle paper on Public Key Distrtibution with Tree Authentication which is what is currently used in NDN.

However, when it came to Blockchain integration in NDN, there wasn't much work done in the field. Hashing, on one hand, is a big part of NDN security because of the size of NDN Names as described above. To this degree, much of Ralph Merkle's work has been employed in NDN, but the Blockchain concept in general isn't seen much.

There is one particular paper titled BlockNDN by Kai Lei which describes a Blockchain abstraction for use in NDN in order to circumvent the IP architecture problem of lacking multicast support. This paper investigated whether Blockchain is a good fit for NDN. It outlined that the native support for multicast in NDN is a good starting point for the peer-to-peer Blockchain architecture and although it did not present a security solution involving Blockchain, this investigation was very useful when designing the solution for this paper. In BlockNDN, the concept of sync interests is introduced, where a node absent from the network would request a sync interest as opposed to request every single block in the chain individually. This paper goes into great detail of the entire process of creating blocks and publishing them. The BlockNDN project takes an interesting approach to broadcasting the information. Because of the nature of the Blockchain, Chronosync is not used to maintain a digest tree in order to maintain the system state. This is why, when a miner generates a node, it can simply send an interest with the node's hash value.

Another related solution which I used for some of its ideas was Alexander Afanasyev's paper titled NDNDelorean. This paper introduced a version concept for old data that needed to be authenticated. The version concept is quite useful when it comes to certificates. Because certificates can become invalid, it is important to be able to represent that in the Public Interest Base(PIB) and also the Blockchain. The problem with the Blockchain approach is that once appended to the Blockchain, a certificate can never be deleted without altering the whole Blockchain, which would render it invalid. This is why, instead we introduce "version control" which in this project's case involves a simple integer(which could also be a boolean) which keeps track of a block's certificate's version. The current version is 1 if valid and 0 if invalid.

2.4 Literary Review

This section has been divided in the different technical components that have been investigated as part of this Final Year Project. Apart from being split into **NDN** and **Blockchain**, I've also split NDN into: **Security, NFD, NLSR, Mini-NDN, Content Store**.

The following table aims to quantify the usefulness of each paper that has been looked at. This method has been directly inspired by Masters student Conor Mooney who did an excellent job at scoring and qualifying his papers while researching. The categories for each paper reviewed fall under one of three categories: analysis, implementation, review.

- Analysis - Refers to papers which analyse a technology

- Implementation - Refers to papers which discuss technical aspects of the technologies which were used. These papers were mostly the NDN Developer Guides.

- Review - Reviews classify papers which mainly contribute with an evaluation - useful when there are different technologies that one might use for a particular problem, allowing for the narrowing down of solutions.

All papers will be scored based on a retroactive relevancy heuristic i.e. how useful did these papers end up being to the problems presented in this project.

Paper	Type	Score
V. Jacobson - Networking Named Content	Analysis	5
D. Kim - Efficient and Secure NDN	Implementation	5
S. Weber - Caching	Implementation	3
K. Lei - Blockchain Based Key Management	Implementation	5
L. Zhang - Named Data Networking	Review	4
S.Nakamoto - Bitcoin	Implementation	4
L. Zhang - NDN	Implementation	4
K. Huang - Cyber Attack Business	Review	3
K. Lei - BlockNDN	Implementation	5
L. Wang - A Secure Link State Routing Protocol for NDN	Implementation	5
NFD Team - NFD Developer's Guide	Implementation	4
A. Afanasyev - NDN Technical Report 9	Implementation	4
Y. Yu - NDN Delorean	Implementation	4
S. Mastorakis - Security Support in NDN	Implementation	5
W. Diffie & M. Hellman - Privacy and Authentication	Implementation	3
R. Merkle - Protocols for PK Cryptosystems	Review	3
L. Kohnfelder - Central Authority	Implementation	3
B. Rainer & S. Petscharing - NDN In V2E	Review	4
A. Afanasyev & Z. Zhu - Let's Chronosync	Implementation	3

Table 2.1: Relevancy of Papers

2.4.1 NDN

Security

[Kim15]Efficient and Secure NDN by D. Kim - 2015 Seventh International Conference on Ubiquitous and Future Networks, pp. 118-120, Tokyo, Japan. 7-10 July 2015.

This paper is important to my State of the Art review because it clearly outlines the current security challenges in Named Data Networks. It suggests a new way of implementing security protocols which currently are only implemented at the application layer and aren't enforced. Because checking for which packets are signed at each packet transfer becomes recursive and very slow for any reasonable size transfer, this paper recommends only checking for signed data at critical points, incurring a smaller overhead on data transfer. This paper also presents an experiment on speeding up NDN by bundling Interest requests instead of burst firing interests for each packet. The paper concludes that this technique is upper-bounded by a $2^{\frac{1}{2}n}$ bundle size, yet delivers tremendous speed-ups in interests where the number of segments is larger than 4096.

[Mastorakis18] Security Support in Named Data Networking by Spyridon Mastorakis, Yanbiao Li, Lixia Zhang, Eric Newberry, Zhiyi Zhang, Haiteao Zhang, Alexander Afanasyev - NDN Technical Report, NDN-0057.

This paper was excellent for giving insight into NDN security. It described an implementation for NDNFit - an app which is designed to run on a user's phone or "data-collector". The user authenticates the collector to gather information, and then to encrypt and send that information to the user's laptop or "analyzer". The point of this implementation was the creation of a hierarchical trust model which employs not only cryptology but NDN's trust anchors to implement network security. The paper was very thorough in describing how a central authority(CA) authenticates certificates. It gives the NDNCERT library as an example of an authority which automatically verifies namespaces and trust anchors and signs certificates. Overall, this paper was extremely relevant to my work and very insightful into NDN security.

[Kohnfelder78] Towards a Practical Public-Key Cryptosystem by Loren Kohnfelder - MIT B.Sc. Dissertation

Despite the age of this paper, Kohnfelder gives a great overview of cryptographic techniques still used in cryptography today. Apart from his own contribution to the world of networking security, he also outlines mathematically, methods for encoding and decoding using cryptographic keys given in RSA, Diffie Hellman and Merkle. Most importantly though, in this paper, he introduces the concept of the certificate. It is this paper that recognizes the importance of a Certificate Authority, which is authorised to keep track and maintain a list of certificates(cryptographic entities) which guarantee that a a piece of content signed by a particular key is legitimately signed by that key. This sets the foundation for Merkle's paper on Certificate management.

[Merkle80] Protocols for Public-Key Cryptosystems by Ralph Merkle - 1980 IEEE Sysmposium on Security and Privacy

This paper is the natural evolution of the Kohnfelder paper. It also gives an overview of the broad scope of security solutions in place. Much like the Kohnfelder paper, despite its age, the techniques proposed are still in use today. Merkle describes different protocols and their benefits and drawbacks. He describes in depth: Simple Public Key Distribution, Authenticated Public Key Distribution, Public Key Distribution with Certificates, and Public Key Distribution with Tree Authentication. Merkle saw the potential of the certificates protocol and wanted to improve on it in this paper. He recognizes that despite the idea, the CA's decryption key is vulnerable to attack, which would result in system-wide loss of authentication. Merkle proposes a hashing function be used on the entire public file instead of the CA having to sign each entry in the Public File. This public file has a root which allows users in the network to use to derive the certificates. Once the nodes know the root, any attempt to alter the public file will now result in a different value for the root, which allows for easy detection. This paper directly builds on Kohnfelder's work and lays one of the foundations for network security implemented in NDN.

[DiffieHellman79] Privacy and Authentication: An Introduction to Cryptography by Whitfield Diffie & Martin. E. Hellman. Proceedings of the IEEE, Vol. 67, No. 3, Mar 1979

This paper goes into great detail on the mathematics of Public and Private Keys. It outlines different cryptographic methods and their cryptographic attack counterparts. Despite its age, it describes cryptographic methods, which are computationally secure, given that the cryptoanalyst trying to break the code has unlimited computational resources. This paper is the foundation for Public Key systems on the likes of which Merkle and Kohnfelder expand. This paper has been scored with a relevancy of 3, because despite the insight into the mathematics behind cryptographic keys, this isn't a technology that needed to be reinvented for this project. However, it was useful to know about the motivations behind these cryptological advancements, as well a refresher on the basics of Public Key systems. Overall, the paper is very detailed and offers great insight into the world of cryptology.

Overview

[Jacobson09] Networking Named Content by Van Jacobson, D.K. Smetters, James D. Thornton, Michael Plass, Nick Briggs, Rebecca L. Braynard - In CoNEXT '09: Proceedings of the 5th International Conference on Emerging Network Experiments and Technologies. Rome, Italy. 1-4 December, 2009.

The Van Jacobson paper on "Networking Named Content" is relevant to my State of the Art review, because it is the first paper to describe Content Centric Networking, on which Named Data Networking is based. This paper largely follows on from Dave Clarke's work in the field of the point to point communication problem. NDN is a direct evolution of both Clarke's work and Van Jacobson's work in CCN. It is implemented in much the same way, by fundamentally using very similar routing as IP, where nodes express Interests which are logged as faces in FIB tables for each NDN node, and are returned with a single Data packet over the shortest available path. "CCN is a networking architecture built on IP's engineering principles, but using named content rather than host identifiers as its central abstraction." NDN is also similar to CCN because it implements its 'soft state' model - meaning an expressed interest that isn't consumed by a Data packet is timed out, therefore the machine expressing an Interest must re-express that interest if it still requires the data. In conclusion, this paper is the foundation of Named Data Networking, which carries over many of the proposed features in CCN in its State of the Art form, including its Node Model, Transport, Sequencing, Routing and Security.

Content Store

[Weber14]A Survey of Caching Policies and Forwarding Mechanisms in Information-Centric Networks by S. Weber, A. Ioannou - 39th Annual IEEE Conference on Local Computer Networks. Edmonton, Canada, 8-11 September 2014.

The paper on Caching Policies and Forwarding Mechanisms was relevant to my work because it described in detail the current SOA of caching policies. As a survey, the paper outlines how currently the FIX(0.9), DC and ProbeCache are the best performers. However, none of these algorithms implement content popularity as a heuristic, the importance of which is proven and cited in the text. The results from the experiment that simulates different caching techniques show that Prob-PD shows very promising but very workload-dependant results, concluding that there's plenty of work to be done on the SOA of ICN caching. This paper was also useful as it gave suggestions for different topologies that might be used to test NDN functionality, for example having a 5 level binary tree with the root being the only initial content source with 1000 contents.

[Lei18] A Blockchain-based Key Management Scheme for Named Data Networking by K. Lei, J. Lou, Q. Zhang, Z. Qi. Proceedings of the 1st 2018 IEEE International Conference on Hot Information-Centric Networks(HotICN 2018). August 2018.

This paper was very relevant to my project as its research and work closely resembles my ideas of what my project should look like. It outlines a specific approach to the distributed ledger problem which isn't normally observed in PKI system. This paper suggests that instead of a root block(or genesis block), to instead have the incumbent nodes in the network come to a consensus on user validation. This is done through an authentication transaction where the user sends the network their public key time and version stamped. The network reaches a consensus and if the block with the user's public key is recorded, they are returned with a $<BlockHeight>$ and a $<TransactionHash>$

to signify that they've been accepted.

Chronosync

[Afanasyev13] "Let's ChronoSync: Decentralized Dataset State Synchronization in Named Data Networking" by Alexander Afanasyev & Zhenkai Zhu.Proceedings of 21st IEEE International Conference on Network Protocols(ICNP 2013).Göttingen, Germany. Oct 2013.

The ChronoSync paper was very insightful. It provided some perspective on how nodes maintain an updated system(similar to Blockchain in NDN) using a pending Interest packet, which when ChronoSync determines that a change of state has been made, fulfils that Interest with a Data packet with the new state. This could be implemented in the miner nodes for sending information in the network. Miner nodes could simply send off the hash value for the new block in another Interest packet containing the new state of the system or in this case the new hash value.

NLSR

[Wang18] "A Secure Link State Routing Protocol for NDN" by Lan Wang, Vince Lehman, A.K.M. Mahmudul Hoque, Beichuan Zhang, Yingdi Yu & Lixia Zhang. In IEEE Access, Vol. 6. Jan 2018.

This paper was excellent in describing how look-ups work. It went into detail about the challenges of named look-ups as the naming scheme implemented in NDN to keep track of entities is inherently larger than IP addresses, therefore Forwarding Tables become huge very quickly. This paper looks at hashing different names in the same tree hierarchy to preserve space. It compares IP routing algorithms to NDN routing algorithms, outlining their differences and similarities.

2.4.2 NFD

[NFDTEAM] "NFD Developer's Guide" by the NFD Team. NDN Technical Report, NDN-0021. Oct 2016.

The NFD Developer's Guide is very detailed. It describes thoroughly every component of the Forwarding Daemon. It starts of by describing Faces and the usefulness of that abstraction. It describes the FIB, the PIT, the CS. There is a full chapter on the Routing Information Base manager and its interaction with the the other tables in an NFD instance. The NFD also manages keys locally. The Develoepr's Guide truly is the compendium for NFD and provides really detailed information on all things NFD. Even though this project didn't involve changing the Daemon implementation at all, the NFD is still the module that forwards Data and so all security Data move through it so it was important to be familiar with its components.

2.4.3 Blockchain

Overview Paper

[Budish18] The Economic Limits of Bitcoin and Blockchain by E. Budish. The University of Chicago Booth School of Business. 5 June 2018.

This paper by itself offered very little in terms of insight for my project - i.e. the SOA of Blockchain or how to implement it in my project. However, this paper pointed me to some of the key and most important resources when researching blockchain such as Nakamoto's "Bitcoin: A Peer-to-Peer Electronic Cash System" paper.

[Nakamoto10] Bitcoin: A Peer-to-Peer Electronic Cash System, https://www.bitcoin.org

The Nakamoto paper is the paper which defined Bitcoin. It goes into great detail about the concept behind decentralized currencies. This paper proposes a paradigm shift from the current economic standard of having a middle man regulate transactions. The benefits of this include the ability of transactions to happen instantly, the reduction in administrator error, and also fees incurred by the bank. It is a welcome simplification of the economic architecture.

Value Chain

[K. Huang] Systematically Understanding the Cyber Attack Business: A Survey by Keman Huang, Michael Siege and Stuart Madnick, MIT

This paper describes in depth the current landscape of cyber attacks and their prevention as a service. It provided an overview of the cryptographic space. This was very informative as I had not dived into the world of cryptography outside of our Telecommunications modules. It was this paper that shed some light on different networking vulnerabilities and how they are tackled.

Chapter 3

Design

This chapter will discuss in detail, the high level approach of the design of the Blockchain in NDN. It will outline and justify different design choices that were made along the way and also the difficulty they presented or indeed alleviated.

The Design is sectioned into: The Problem, The Development Platform, The Data Structure and Content Delivery, which will be discussed in that particular order.

3.1 The Problem

The particular issue that this project concerns itself with is the security protocol. In particular, in the previous chapter, it was described that the current implementation relies on the CA method described by Kohenfelder in his B.Sc. dissertation and improved upon by Merkle's tree authentication[48].

Despite efforts made in the security of the CA and the public keys file, there hasn't been much regard for the efficiency of the network when it comes to security. There are two problems that this project aims to address:

- reduce lookups in the Public Information Base - alleviating the time constraint in having to request that information and go through it.
- to increase the Public Information Base's integrity - by hashing every certificate to

each other, and having those hashes be verified by miners authorized by the central authority. The result of this is that it is now exponentially harder for an attacker to compromise the PIB, by virtue of Blockchain **and also** because the PIB is now no longer the only point of failure.

3.2 The Development Platform

3.2.1 MiniNDN

There are a number of tools which can be used to experiment with NDN. These are the following: ndnSIM, Docker and MiniNDN. MiniNDN is an emulator. It is an extension of mini-net - a networking emulator. On top of mini-net, one can install MiniNDN and all of its modules:Chronosync, PSync, NDN-CXX, NDN-CPP, NFD and NLSR.

There are a couple of reasons why MiniNDN was chosen for this project. Firstly, it is very easy to set up. The knowledge required to get started with MiniNDN is minimal. It is largely based on MiniCCNx which is a fork of Mininet meaning there are plenty of resources available online in terms of reading material on getting started.

It goes without saying that MiniNDN was also chosen because it is open and free under the GNU General Public License. They also have a Redmine site which tracks and describes all bugs/features as well as a useful mailing list for any developers looking to experiment with the software.

There are also a number of MiniNDN specific tutorials that have been created by NDN's main coders - Alexander Afanasyev and Ashlesh Gawande. They go into quite a bit of depth regarding different utilities in MiniNDN. Ashlesh's tutorial mainly concerns experiments and topologies where Alex's tutorial goes a bit more in depth regarding node interactions.

There are drawbacks associated with using MiniNDN also. It is an emulator not a simulator, meaning all of the topologies tested are created in real time - with NLSR

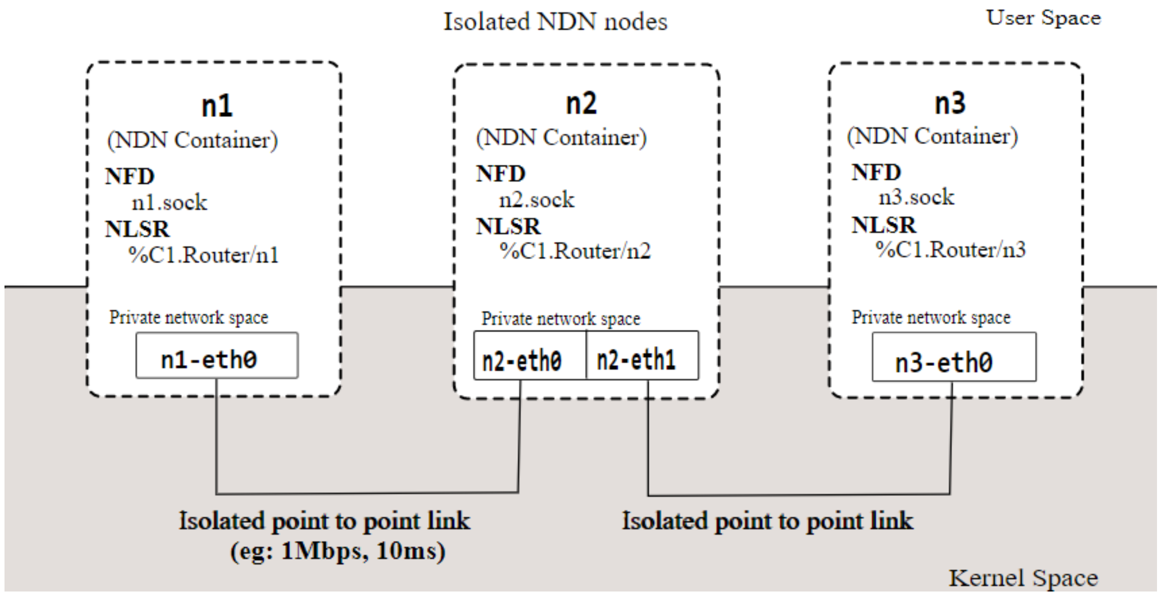


Figure 3.1: Basic MiniNDN architecture[49]

convergence happening in real time also. This means that for a portable machine, with an 8th generation hyper-threaded Intel i7 processor, it can take up to 70 seconds for NLSR to converge on a basic 4 node topology. Convergence time scales with CPU performance, so more CPU power should result in much quicker convergence.

As well as that, when defining topologies or writing experiments, the emulator must be reinstalled every time in order for these changes to be recognized.

MiniNDN works by creating an NDN container around nodes in a mininet simulator. Each node then runs an instance of NFD and NLSR. The user can then configure topologies - including amount of nodes, nodes' identities and adjacencies. As well as that the user can set parameters for hyperbolic routing which NLSR can run on if set in the configuration file.

MiniNDN is useful because we can run different programs on nodes without having to use the python experiments and reinstall MiniNDN every time we alter an experiment. Each node can run an xterm in the background by using the command "*<node> xterm &*" after which the user can export the home folder for each node and run any of the sample programs from the NDN libraries or indeed write their own.

3.3 The Data Structure

There are two data structures which were designed for the scope of this projects. These are the PibBlock and PibBlockchain. The PibBlockchain maintains a vector of hashed PibBlocks. It can return, at an index, a particular hashed block. It is maintained by the Public Information Base.

The PibBlock class deals with the Certificates. It stores all information about a certificate as well as its version and the hash of the previous block.

3.3.1 Public Information Base Block

This is the basic building block of the PibBlockchain. It contains the current block's hash, certificate, version and timestamp as well as the previous block's hash. Mining the block is also done in the block class. The difficulty of the block mining is determined by the proof of work algorithm which takes in a a difficulty argument which determines how many zeroes need to be mined by the algorithm until a block is valid. For the purposes and scope of this project however, this algorithm wasn't fully implemented.

The PibBlocks's constructor returns a pointer to a PibBlock's location in memory. For the initial Block, the PibBlock constructor takes no arguments and creates the genesis block which is based on a nonce cert. This is because we need to have an initial block with which we can hash the rest of the blocks. This bit of design wasn't particularly necessary as mentioned previously, the hashing algorithm wasn't implemented fully in the first place.

Because the PibBlock class doesn't use smart pointers(more on this later), the class must have an explicitly defined destructor. In this destructor, all of the PibBlock's components are deleted. It is however important to note that the PibBlock's destructor doesn't correlate to the PibBlock's invalidator. Once created, a PibBlock cannot be destroyed or it will invalidate the whole Blockchain. Instead, there is an invalidator function, which simply sets the version of a PibBlock to 0 to imply that it has been invalidated. If the same certificate needs to be validated again, it must go through the whole proof of work process and be hashed to a new block with a version number 1.

Perhaps the most important design element of the PibBlock was the displaying of information. Early iterations would have each PibBlock copy a certificate onto a new certificate instance before adding the Block to the Blockchain. This proved disastrous as NFD does not allow certificates that haven't been signed by the KeyChain to exist, and if any are found the NDN network is shut down immediately.

3.3.2 Public Information Base Blockchain

Public Information Base(PIB) - This is the public key infrastructure hierarchy where identities are stored. Each identity in a network contains within it a default key and a default certificate. The counterpart private key information is stored in the Trusted Platform Module(TPM). We do not concern ourselves with the TPM as we only need the public keys to verify a node's signature. This is standard security procedure in any network.

The PIB class in the NDN-CXX library is responsible for creating and publishing certificates. The design suggested by Dr. Weber was to create a wrapper so that any time a certificate is created, we could simply add it to the blockchain. Then the miner nodes could verify it and publish the given information. This however proved challenging in a number of ways.

The first issue I encountered had to do with instancing. Because we needn't necessarily have only one instance of a PIB, we have to make sure that each PIB's certificates go in that specific PIB's Blockchain. This is precisely why Alexander Afanasyev has designed the PIB class in a way that one cannot instantiate a Trusted Platform Module(TPM) outside the PIB class i.e. the constructor for the PIB is the only place where the TPM is also constructed. This means that we cannot have a PIB be matched with a TPM that isn't its counter part. I aimed to achieve the same goal. I did this by looking outside of

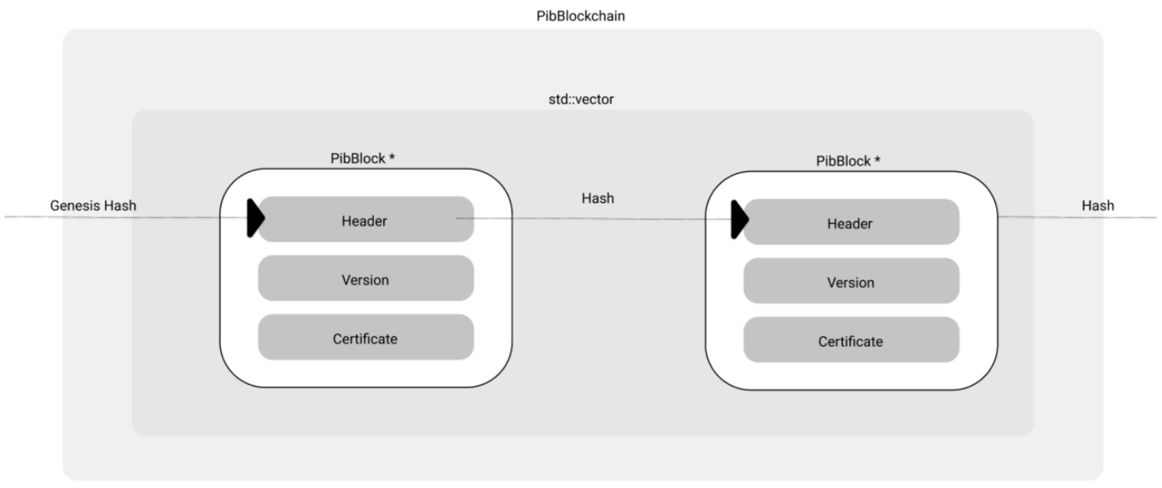


Figure 3.2: This figure shows how PibBlockchain interacts with NDN and the std library

the PIB class. The PIB class is instantiated and governed by the KeyChain class. This in turn means that the KeyChain class instantiates both the PIB and the TPM. This is why I tried to make the constructor for the PibBlockchain data type to use as an argument the KeyChain's address meaning it would be instantiated in the KeyChain with code that looks something like "PibBlockchain certChain = new PibBlockchain(this);"

However, the PIB Blockchain was comprised of blocks or PibBlocks, which were a separate data structure which made use of the Certificates class in order to store certificates or indeed to be able to parse them at all in the first place. Because PibBlock inherited from Certificates, and PibBlockchain inherited from PibBlock and KeyChain inherited from both PibBlockchain and Certificates, there was suddenly a circular dependency which could not be broken without completely scrapping the PibBlockchain constructor design which takes a pointer to the KeyChain as an argument.

This is where Dr. Weber's original "wrapper" idea came to mind and to good use. Instead of having to worry about the KeyChain pointing to the correct PibBlockchain for each PIB, we could just instead make a mutable PibBlockchain in the PIB class which would work on the exact same principle as Alex Afanasyev's idea to instantiate the TPM in the PIB. We simply do the same thing with the PibBlockchain and instantiate it in the

PIB. This way, we no longer have to worry having mismatched PIB and Blockchain.

Blocks on the other hand weren't at all a concern when it came to creating instances of the Blockchain. The purpose of the PibBlock class is twofold: Firstly, to encapsulate all of the data from each Certificate and secondly, to do all of the "heavy lifting". What this means is that the PibBlock class is responsible for the hashing of each block.

Of course this doesn't mean that there isn't a concern about which blocks go in which Blockchain. However, blocks are only created when Certificates are created. Certificates are created in the KeyChain.cpp. This means that each KeyChain has only one Pib-Blockchain to work with, because each KeyChain only instantiates one PIB. One PIB = One PibBlockchain. Therefore if we create PibBlocks in the KeyChain they will inherently be PibBlockchain specific and will be out of scope for any other KeyChains or PibBlockchains. This inherent property of C++ and indeed all object oriented programming made the challenge of making sure that each PibBlock is added to the correct Blockchain very simple.

"Within C++, there is a much smaller and clearer language struggling to get out" - Bjarne Soustroup[50]

Now that the allocation of PibBlocks to PibBlockchains was completed, and there were no longer any circular dependencies allowing for the code to be added to the security code hierarchy in NDN-CXX.

Shortly after adding the code to the project, it became evident that the solution wouldn't work in its current form. That is because the NDN Team have developed a pretty robust automatic Certificate Authority. I found this out when I realized that my PibBlock data structure was parsing certificates being created, allocating new memory for them and then copying the certificate. The problem with this approach is that the copied certificates have not been authorised by the Certificate Authority. This is why when running the simulation, it would abruptly exit.

So instead, the PibBlock class looks to the Certificate class for inspiration. The

Certificate class overloads the << operator for certs by extracting all data from them and printing it in chunks. PibBlock takes the functions used to extract data from the certificates and implements them in order to store the Certificate data in strings. Once stored, nodes can still access the signatures for each certificate in string form and verify them.

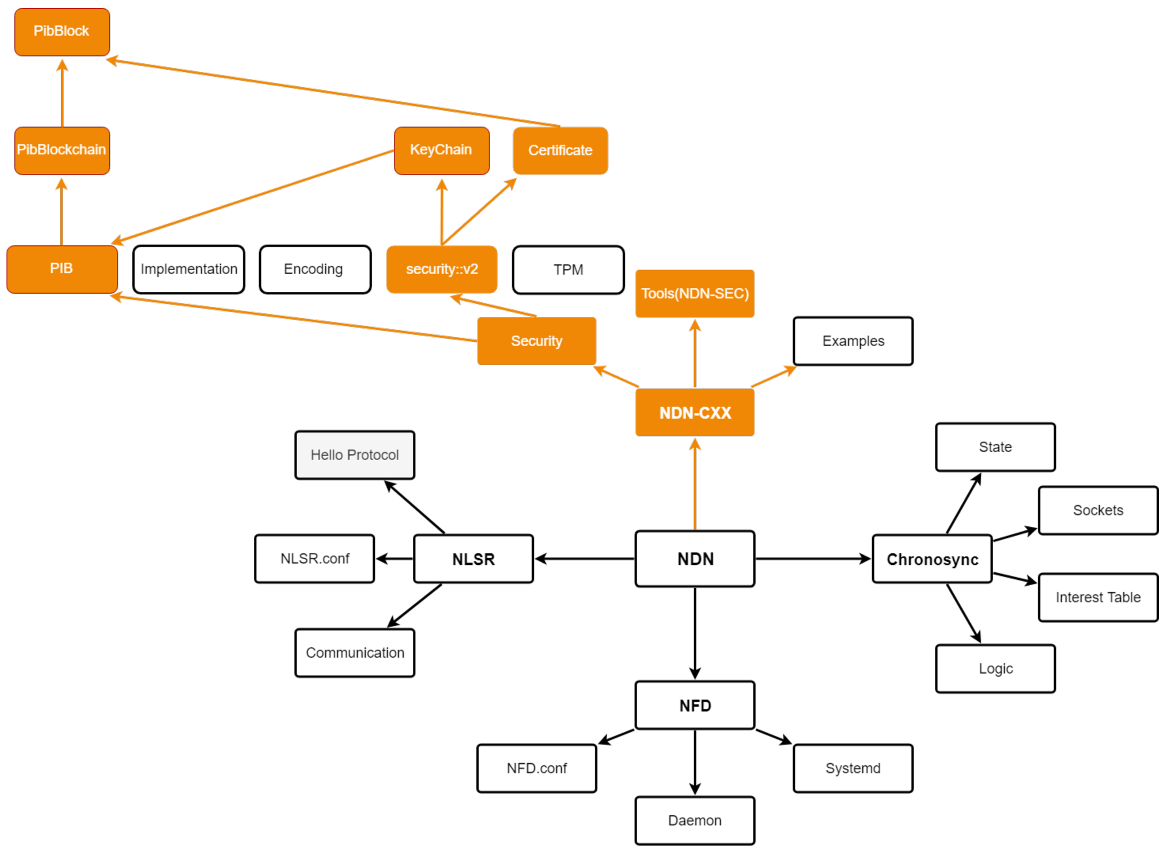


Figure 3.3: NDN Components: Orange Rectangles - Intimate knowledge of modules' code was required. Orange Rectangles w/ Red Frames - Modules where code was either altered or added for the scope of this project.

3.4 Broadcasting

Once the data is stored in the Blockchain, the next objective was to figure out a way to disseminate it across the network. This is the most important part of the project. The reason why the Blockchain is useful is so nodes can compare certificates against it. If they can't do that because they don't have that information, then the Blockchain is useless. The reason why the solution required was non-trivial was because of the communications design in NDN. In order for nodes to communicate, they must send out an Interest - which made the tasks more difficult than anticipated.

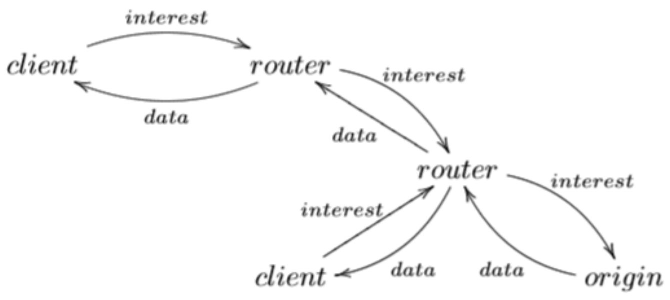


Figure 3.4: Node Communication[51]

3.4.1 Naive Approach

The proof of concept method of broadcasting the Blockchain using UDP, utilizes, in the case of MiniNDN, the file system. The Blockchain is broadcast to all nodes via the port and appears in the temp folder for all of the nodes. This is the naive approach and it is impractical because it just blasts data at the nodes and wouldn't work in a real environment.

3.4.2 Reconfigure NFD approach

The NFD is, as mentioned in the previous chapter, how Data and Interest packets get propagated in an NDN network. While investigating different options for broadcasting, I came across the configuration file for NFD which gives the user the option to redefine all nodes' caching policies. In it, there are a couple of listed policies. One only allows Data packets to be cached for which there are expressed Interests. The other allows Data to be cached regardless of whether there's been an Interest for it.

This project has explored one option for this approach. Because reconfiguring the NFD to make it so every node caches everything isn't feasible or wise, the option to create an altogether new policy was explored. However, there wasn't enough time to explore this option fully.

3.4.3 Signed Interest approach

This approach utilizes some IoT concepts in NDN that have been previously implemented and tested. Firstly, we must define a Signed Interest. This type of Interest is the second version of what was known as a "Control Interest". These specialized Interest packets were designed for Internet Of Things applications where nodes would have to communicate and sometimes control one another and the conventional Interest-Data packet communication didn't allow for this.

In a control system where a thermometer node has to tell a WiFi connected HVAC the temperature, so it can adjust accordingly, the conventional system leads to a paradox. This is because the thermometer node cannot simply send a Data packet to the HVAC, giving it the temperature, because this Data packet would be unsolicited. Also, it is not feasible to set up the HVAC to flood the network with temperature Interest packets at all times because this creates unnecessary strain on the network. It would be quite pointless to send out Interests(polling) at a random or set interval as well, as in most environments, temperature change doesn't happen at set intervals. This could make for an ineffective temperature control system if the temperature has changed from the desired temperature but the HVAC isn't due to poll the thermometer for a prolonged amount of time.

Instead, the thermometer node sends a "Control Interest"(now deprecated and known instead as a Signed Interest) where it tells the HVAC to ask for the temperature reading.

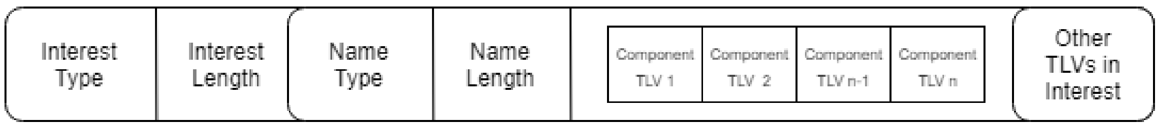


Figure 3.5: Signed Interest

We are met with a near identical situation. We have miners which need to broadcast a Blockchain to all nodes in a network. One approach could be to ensure that nodes request an updated Blockchain every time that they receive a Data packet. However, the delay from doing this would render the solution pointless and would provide little benefit in terms of a speed-up compared to the regular PIB look-up that nodes do(it would still improve on the single point of failure problem). So instead, when the miner verifies a new Certificate in the Blockchain, it sends out a Signed Interest to all nodes, to get them to request the information(Express an Interest) for the updated Blockchain.

The nodes in the network would have an InterestFilter which will verify that this Interest is indeed coming from an authenticated miner. Once the node receives the Signed Certificate, it will send an Interest packet to the miner to send out the updated Blockchain. This adds additional strain on the network, however, this is counterbalanced by the reduction in look-ups that nodes have to do in order to verify certificates.

3.4.4 ChronoChat

ChronoChat is perhaps the best way to approach this problem. The details of how ChronoChat works are described by A. Afanasyev and Z. Zhu in the 'Let's Chronosync' paper. They describe how ChronoChat creates a system, where every user or identity keeps an outstanding sync interest[52] with the current state. The benefits of this outstanding sync interest are immense. Firstly, this interest actually contains data with the system's current state, which means that as it propagates through the system with the help of NDN's inherent multicasting characteristics, it immediately updates every node with the system's most up-to-date state. Should any node introduce a change in the system, Chronosync will satisfy that node's outstanding interest with a Data packet containing the new system state. As this Data packet propagates through the system, so will all the other nodes be updated on the new system state.

In a similar fashion, miner nodes, using the Chronosync module, could maintain an outstanding Interest. When a node mines a block, it can satisfy that Interest with a Data packet which will then propagate through the system, updating all nodes with the new system state i.e. with the new, verified Blockchain.

3.4.5 Interest with Hash Value

In the BlockNDN project, it was made apparent that Chronosync isn't necessary for maintaining system state. However, in a system where it is imperative for all nodes to have the most up-to-date information at the same time, Chronosync is a better solution. The simpler option would be to just append the value of the Blockchain's newest hash to an Interest which will then be multicast through the system by the miner once they verify the newest block. This approach doesn't need Chronosync to maintain its state because Blockchain is naturally a digest chain reflecting the current owner's dataset [BlockNDN]

3.5 Challenges Overview

As described above, the main challenge for the NDN Blockchain project was broadcasting the Data. There were a number of approaches available with the ChronoChat approach being by far the most suitable for this project. Careful selection of a broadcasting method is crucial when extending a networking architecture. This is due to the overhead being introduced.

Deciding on where the Blockchain should live was also a big design challenge. There were many pitfalls when designing this, thoroughly described in Chapter 5.

Finally, and possibly just as important as the other two, a design element that was hardly looked at because of time constraints was deciding the difficulty of mining factor which will probably be left as future work.

Chapter 4

Implementation

You enter the first room of the mansion and it's completely dark. You stumble around bumping into the furniture, but gradually you learn where each piece of furniture is. Finally, after six months or so, you find the light switch, you turn it on, and suddenly it's all illuminated. You can see exactly where you were. Then you move into the next room and spend another six months in the dark.

- Andrew Wiles, *PBS Interview, After proving Fermat's Last Theorem*

This chapter leads on from the Design chapter and provides further in-depth explanation of the implementation for this project. It is structured as follows:

- Tech Used
- Blockchain Interaction with Data Structures
- Challenges
- Overview

4.1 Technologies Used

As stated in the Design chapter, MiniNDN was used in order to emulate a working NDN environment on which to prototype code. This chapter will provide more in-depth insight into the different technologies within MiniNDN and indeed all NDN which were used for this solution.

4.1.1 SHA256 Library

An external sha256 library was initially used in order to create hashed blocks in the Blockchain and in order to be able to do the proof-of-work. However, knowing that this wasn't the focus of the project, the idea was quickly scrapped.

4.1.2 ndnsec

Ndnsec is a command line tool for generating keys and certificates. There are 14 modules in ndnsec. They provide the necessary tools for maintaining NDN private and public keys and certificates for identities. When configuring a testbed to use NLSR-Security, what that actually does is utilize ndnsec in order to create and assign keys and certificates. This is done using predefined commands which generate keys and generate and sign certificates for each identity in a network and then dump the certificates in each identity's local files. The public keys are stored in the PIB and the private keys are stored in the TPM.

4.1.3 NLSR Security

As mentioned in the State of the Art Chapter, NDN - on an experimental level, relies on the Network Manager to choose a security standard. However, MiniNDN has some built in security features. These are found in *../mini-ndn/ndn-cxx/.* The C++ Library with eXperimental eXtensions provides some essential security tools. NLSR security is one way we can configure network security in NDN. NLSR security works in a trust hierarchy. This has many benefits, including being able to identify routers, keys and messages generated by a given routing process in the same network.[53]

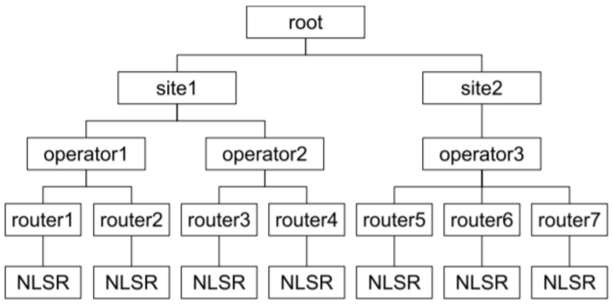
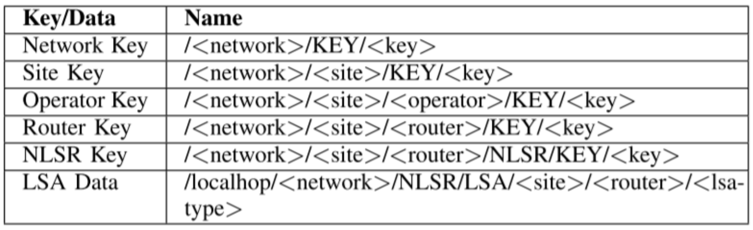


Figure 4.1: NLSR Trust Hierarchy[54]

The Trust Model is in place for intra-domain routing and relies on ndnsec to generate keys for NLSR. After some experimentation, it became apparent that NLSR security also imposes on the security protocol to assign a Certificate Authority in order to sign certificates. This is made apparent when printing out the certificates being generated in a long. The difference between having NLSR-Security configured and not shows in the certificates, which have a 'self' field which is generated when a certificate is self signed. When NLSR-Security is enabled, these certificates are also replicated with an 'NA' field which would normally be populated by the Certificate Authority name. In an experimental environment however, there isn't a specifically assigned CA and this is why this field is populated by 'NA'.

This security scheme is considered to be hierarchical because NLSR uses a "five-level hierarchical trust model reflecting the administrative structure"[55] of the network. At the top level, the root or trust anchor, issues certificates to sites. The sites are level two and they can have one or more operators(level three), like the clock example in Chapter 1. Each of these can then produce data, which can be routed by a router(level four), which can be forwarded in an NLSR routing process(level five) which produces LSAs containing the data which must be signed.



Key/Data	Name
Network Key	/<network>/KEY/<key>
Site Key	/<network>/<site>/KEY/<key>
Operator Key	/<network>/<site>/<operator>/KEY/<key>
Router Key	/<network>/<site>/<router>/KEY/<key>
NLSR Key	/<network>/<site>/<router>/NLSR/KEY/<key>
LSA Data	/localhop/<network>/NLSR/LSA/<site>/<router>/<lsa-type>

Figure 4.2: Keys generated with NLSR-Security[56]

4.2 Component Interaction with Data Structures

Incorporating the Blockchain Data Structure into the security module of ndn-cxx has been challenging. To a degree because ndn-cxx security is split between version 1 and 2 of the security module. The KeyChain is implemented in v2 but still has a header in v1 and uses the PIB and TPM from v1. There is also the problem of the security modules restarting themselves sporadically.

4.2.1 Public Information Base

The Public Information Base(PIB) is maintained in ndn-cxx. It is instantiated by the KeyChain. The PIB has two backend modules, pib-memory and pib-sqlite3. The PIB sends information to pib-memory which in turn sends information to pib-sqlite3 which in turn stores information in the pib.db. This is where the public keys are stored, i.e. the Public File. The PIB is managed by the KeyChain.

4.2.2 The KeyChain

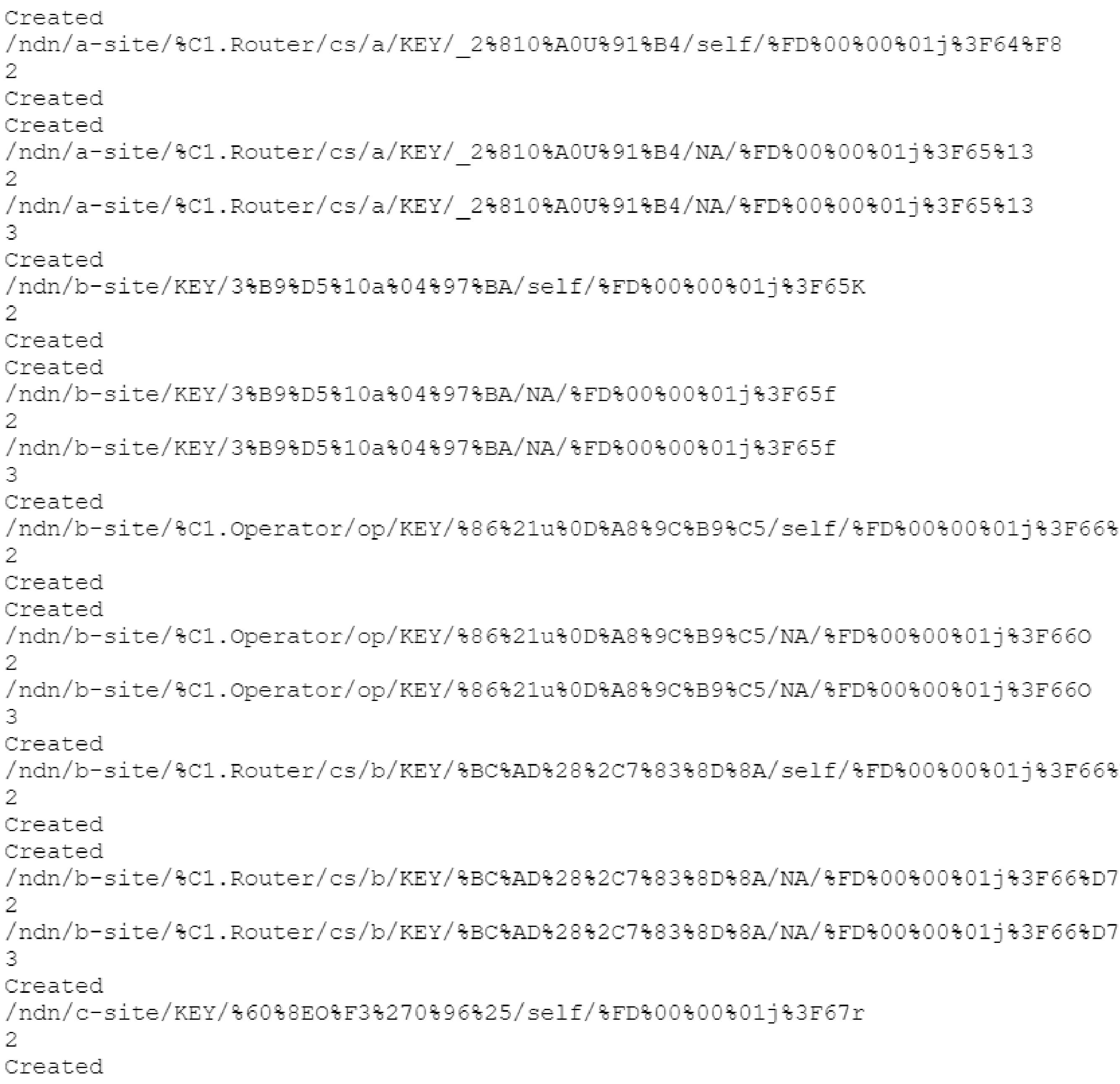
The KeyChain proved to be very problematic. This is the data structure in charge of instantiating the PIB and creating all certificates in a network, and making sure they are signed. If self-signing is allowed, this is done in the KeyChain class. The KeyChain also compares whether the Trusted Platform Module and the Public Information Base matched and if not, the KeyChain would restart and begin the security process all over again, by getting the PIB and TPM locations, their factories, creating them, then checking if they match again, until they do, and if so, the KeyChain creates the certificates for the network.

The problem with the KeyChain and implementing Blockchain begins with its design by the NDN team. The following is a log that was created using the std library as opposed to NDN's boost library approach to making logs. This was done in order to have a separate file to the already very busy NDN log. This log records every time the KeyChain constructor is used. The numbers on the screen represent the Blockchain size. The certificates in the log get printed any time a certificate gets added to the Blockchain. The problem here is that the KeyChain keeps getting instantiated. Because the KeyChain keeps getting reset, we see that the Blockchain is also being reset every time.

This is occurs because of the default security design in the NDN architecture. When MiniNDN is running NLSR-Security, it calls a new instance of the ndnsec tools for every entity. When the new instance is called, and the certgen function is used, that creates a new KeyChain. However, if the TPM and PIB don't match up then the KeyChain gets reset. This means that it is impossible to maintain a Blockchain in the KeyChain class, unless a specific security protocol is designed to run on this system.

The workaround for this has been to dump the PibBlocks into a completely separate data structure or even file in order for the nodes to be able to receive it. This is because the KeyChain class uses smart pointers to initialize PibBlockchains which in turn means that if a KeyChain goes out of scope as it would if it is running the reset() function when the TPM and PIB don't match up, the C++ Compiler simply deletes all references to the Blockchain.

A better solution would be to run the Blockchain outside of the security module, so it doesn't keep getting reset and wiped sporadically.



```
Created
/ndn/a-site/%C1.Router/cs/a/KEY/_2%810%A0U%91%B4/self/%FD%00%00%01j%3F64%F8
2
Created
Created
/ndn/a-site/%C1.Router/cs/a/KEY/_2%810%A0U%91%B4/NA/%FD%00%00%01j%3F65%13
2
/ndn/a-site/%C1.Router/cs/a/KEY/_2%810%A0U%91%B4/NA/%FD%00%00%01j%3F65%13
3
Created
/ndn/b-site/KEY/3%B9%D5%10a%04%97%BA/self/%FD%00%00%01j%3F65K
2
Created
Created
/ndn/b-site/KEY/3%B9%D5%10a%04%97%BA/NA/%FD%00%00%01j%3F65f
2
/ndn/b-site/KEY/3%B9%D5%10a%04%97%BA/NA/%FD%00%00%01j%3F65f
3
Created
/ndn/b-site/%C1.Operator/op/KEY/%86%21u%0D%A8%9C%B9%C5/self/%FD%00%00%01j%3F66%
2
Created
Created
/ndn/b-site/%C1.Operator/op/KEY/%86%21u%0D%A8%9C%B9%C5/NA/%FD%00%00%01j%3F66O
2
/ndn/b-site/%C1.Operator/op/KEY/%86%21u%0D%A8%9C%B9%C5/NA/%FD%00%00%01j%3F66O
3
Created
/ndn/b-site/%C1.Router/cs/b/KEY/%BC%AD%28%2C7%83%8D%8A/self/%FD%00%00%01j%3F66%
2
Created
Created
/ndn/b-site/%C1.Router/cs/b/KEY/%BC%AD%28%2C7%83%8D%8A/NA/%FD%00%00%01j%3F66%D7
2
/ndn/b-site/%C1.Router/cs/b/KEY/%BC%AD%28%2C7%83%8D%8A/NA/%FD%00%00%01j%3F66%D7
3
Created
/ndn/c-site/KEY/%60%8EO%F3%270%96%25/self/%FD%00%00%01j%3F67r
2
Created
```

Figure 4.3: KeyChain log

4.3 Challenges

Most of the design and implementation challenges for this project have been described above. To summarize, the two main challenges are communicating the Blockchain to all of the nodes in a network, and also appending the Blockchain class to the security library without recalling its constructor constantly, the way KeyChain does. These issues are solved by writing a wrapper for the PibBlockchain class, and also, upon limited trial, selecting the Chronosync approach in the dissemination of the Blockchain.

Chapter 5

Results

This chapter presents the result of the Blockchain in NDN experiment. The outcome of this experiment is interesting because it presents a different approach to verifying certificates. This project recognizes that, despite the apparent benefits of not having to contact the Certificate Authority for verification, there is also added complexity to the NDN security protocol. The results from this experiment investigate the trade off in this scenario, in different topologies using MiniNDN. Because of the serious time constraint imposed by our different modules, this project instead presents an ideal evaluation for the experiment.

5.1 Ideal Evaluation

Because of time constraints, only very limited testing was done with this project. The ideal evaluation for the Blockchain however, would involve testing a number of different realistic topologies. Ideally, one would request a certificate from the NDN team for the ability to test on the official testbed. This would allow for realistic results which are desirable. There are a number of different topologies that must be tested including the 'dumbbell' topology with two chokepoint routers. It is desirable to test different data sets, with different data sizes, and also different intervals of node communication. There must be a control experiment for each of the listed experiments. MiniNDN allows for huge amounts of configuration and customization for each node in a topology. A user can set different Hyperbolic Routing variables as well as delay and bandwidth between nodes and also links, allowing for extensive and robust network testing.

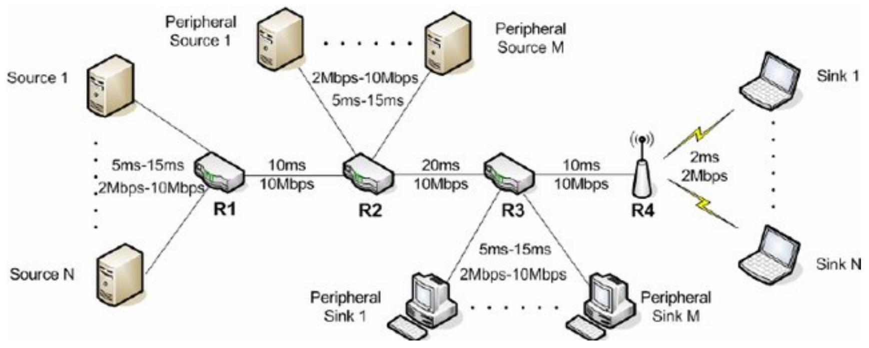


Figure 5.1: Sample Topology [57]

5.1.1 Overhead and Latency

By increasing the complexity of NDN, one inadvertently adds overhead. The trade-off of that is the amount of time saved by nodes that don't have to communicate with the CA in order to verify Data. The concept itself works and isn't new. Browsers pre-install certificates to reduce look-up times. The idea here is to do the same by adding as little complexity as possible. And so, this project aims to achieve a golden ratio of complexity to efficiency. Adding the Blockchain data type to the NDN architecture adds bloat. If

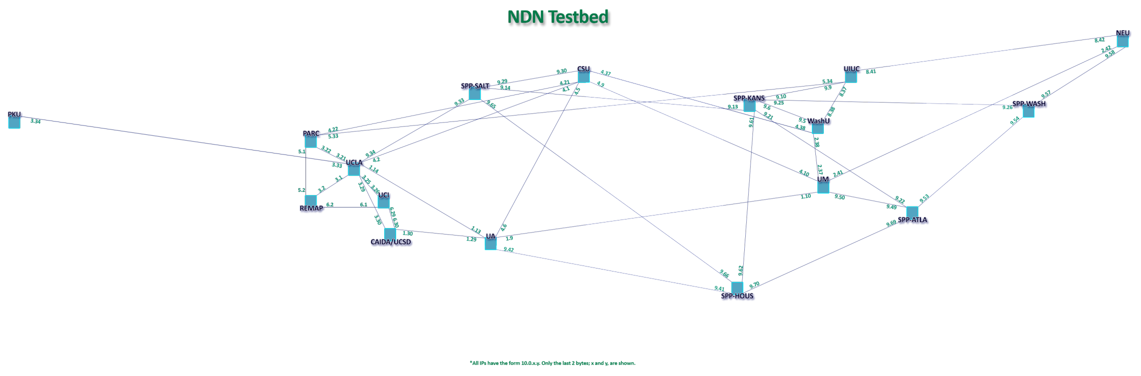


Figure 5.2: The Current NDN Testbed Topology [55]

every node has that bit of overhead of having the Blockchain stored in its content store, it begs the question whether there's enough of a speed up from look-up reduction time to justify it. Especially seen as we are broadcasting the Blockchain, and some nodes might not be or want to be communicating at all. Yet, they are to request the Blockchain when the miners publish a Block.

5.2 Discussion

The evaluation of the success of this project greatly depends on whether one values a bigger Content Store or quicker communication between nodes. This is because every node would have to maintain the blocks in their cache. Blockchain in NDN greatly reduces communication with external Certificate Authorities. However, it can greatly add to the complexity of a network. Also, if it's a case of needing all nodes to have the Blockchain information simultaneously, then Chronosync is required, adding more complexity to the system.

Chapter 6

Conclusions and Future Work

To conclude, a summary is presented which describes the work done in the context of NDN and Blockchain. Finally, a Future Work section presents future projects which could result from this project.

6.1 Conclusion

In this body of work, Named Data Networking and Blockchain have been presented as separate entities. The strengths and weaknesses of both have been discussed in depth as well as their different components.

This paper concerns itself with certificate management. A certificate is every entity that produces Data.

This project presents a solution to NDN's problem of tasking each Face to verify a hierarchy of certificates. Currently in NDN, each Face, in order to verify the certificate they are interested in, must verify every certificate before it in the tree hierarchy structure of certificates. When it gets to the root certificate, the Face must send an Interest packet to the CA in order to verify the root cert. The problem with this implementation is the look-up time introduced by having to verify the root certificate. This **look-up** introduces

considerable delays in a network.
This isn't always the case however, as NDN allows the network administrators to set up and configure the network's security in a number of different ways. They can be configured in a completely ad-hoc manner, where the Certificate Authority isn't a trusted authority, i.e. using an SDSI policy - Simple Distributed Security Infrastructure. This is mainly done in simulations in an experimental environment, and is neither secure or feasible to do in a real NDN deployment.

Alternatively, and more commonly in a real NDN deployment, networks use trusted Certificate Authorities such as Verisign, which grant certificates for local Certificate Authorities(root certs) which are then used to issue certificates to any entity that exists below the root in the namespace hierarchy of NDN. This is where the problem described above occurs, where a lot of needless communication between entities and the third-party Certificate Authority occurs

The solution presented in "Integration of Blockchain and Named-Data Networking" involves separate,dedicated nodes called miners, doing all of this verification on behalf of the network, appending the verified certificates to a Blockchain, and publishing and multicasting the verified chains to all of the nodes in a network which then no longer need to verify a hierarchy of certificates. This is because the Blockchain, by design, is tamper-proof, because all hash blocks in a Blockchain have been hashed with the previous block's hash and in order to tamper with or change a single block, a node would have to do all of the "Proof-of-Work" for all of the previously hashed blocks, which increases in difficulty exponentially. This is virtually impossible unless one has more computing power than an entire network of miners. This moves the single point of failure to all "miner" nodes. The network must guarantee that the miners' certificate verifications are correct(a trivial problem), which if done, would mean that no malicious node can tamper with or alter any certificate in the network.

6.2 Future Work

There are a number of directions this project could take. Firstly, it is important to note that the Blockchain doesn't hash blocks using a nonce to produce a set number of 0s. This means that the Blockchain algorithm doesn't do the "Proof-of-Work" described by Satoshi Nakamoto, required to create blocks in a regular Blockchain implementation. This is because this project has been a proof of concept solution. In future projects, this could be improved upon greatly, and the difficulty of the mining could be investigated to determine the trade-off of safety versus compute time. Since Blockchain isn't the main mechanism in making the network work, the way it is in a decentralized transaction system like Bitcoin, the hashing algorithm doesn't have to mimic the same difficulty. Additionally, a project could investigate the advantages and disadvantages of publishing the blockchain at a set time interval and which time intervals would best suit which networking topologies in ICN. In a Bitcoin environment, Blockchains are published once every 10 minutes. This particular time interval suits the dynamics of Bitcoin. However, how this affects an Information-Centric Network would be curious indeed.

An important investigation would be to determine the best way to disseminate Blocks across the network. There are a couple of methods investigated and outlined in this project which can be found in chapter 3.

Continuing on with the Blockchain data types - it's important to talk about C++ convention. The C++ language is constantly evolving and new revisions are released annually. Since C++11, smart pointers have become the de facto standard in memory allocation. It really is unacceptable to have a system where instances of a class or data type are created using the 'new' keyword and deleted using 'delete'. A future project would ameliorate the existing PibBlock data type and make it use either unique or shared pointers which rely on the C++ compiler to delete the allocated memory resources once the object is out of scope.

Additionally, it could be beneficial to investigate making this solution into a piggy-back module for any network.

Finally, the main reason for this project is to investigate whether it is worth implementing a Blockchain in the Public Interest Base. It would be advisable to utilize a huge computing resource like the MacNeill or Stoker in order to generate realistic topologies on which then to test the Blockchain integration and whether it is worth implementing as a viable option for speeding up communications and reducing the load on the network.

Appendix A

Abbreviations

Short Term	**Expanded Term**
CA	Central Authority
DNS	Domain Name System
ACK	Packet Acknowledgement
IoT	Internet of Things
URI	Universal Resource Indicator
ICN	Infomration Centric Networks
NDN	Named Data Networking
CCN	Content Centric Networking
NDN-CXX	C++ Library with eXperimental eXtensions
NFD	Named Data Forwarding Daemon
NLSR	Named Data Link State Routing
LSA	Link State Advertisement
HR	Hyperbolic Routing
Face	Interface(Physical/Logical)
FIB	Forward Interest Base
CS	Content Store(Cache)
PIT	Pending Interest Table

PIB	Public Information Base
RIB	Routing Information Base
PKI	Public Key Infrastructure
TPM	Trusted Platform Module
TLV	Type Length Value Encoding
TCP	Transmission Control Protocol
IP	Internet Protocol
LNPM	Longest Name Prefix Matching
DHCP	Dynamic Host Configuration Protocol
ARP	Address Resolution Protocol
DNS	Domain Name System
MAC	Media Access Control
RTT	Round Trip Time
RIB	Routing Information Base
SDSI	Simple Distributed Security Infrastructure

Bibliography

[1] [Zhang17a] "An Overview of Named Data Networking by Lixia Zhang." Proceedings of MILCOM 2017. Baltimore, MD, USA. Oct 2017.

[2] [Ioannou16] "A Survey of Caching Policies and Forwarding Mechanisms in Information-Centric Networking" by Andrianna Ioannou & Stefan Weber. IEEE Communications Surveys & Tutorials. Volume:18, Issue:4, Q4 2016.

[3] [C.Index13] "Cisco visual networking index: Forecast and methodology 2012-2017" by C. Index, White Paper. May 2013.

[4] [Kohnfelder78a] "Towards a Practical Public-Key Cryptosystem" by Loren M. Kohnfelder. B.Sc. Dissertation. MIT, Boston, MA, USA, May 1978.

[5] [Kohnfelder78b] "Towards a Practical Public-Key Cryptosystem" by Loren M. Kohnfelder. B.Sc. Dissertation. MIT, Boston, MA, USA, May 1978.

[6] [Weber19] "Thesis first draft corrections" by Stefan Weber. TCD, Dublin, Apr 2019.

[7] [Nakamoto09a] "Bitcoin: A Peer-to-Peer Electronic Cash System" by Satoshi Nakamoto, Oct 2008. `https://bitcoin.org/bitcoin.pdf`

[8] [Nakamoto09b] "Bitcoin: A Peer-to-Peer Electronic Cash System" by Satoshi Nakamoto, Oct 2008. `https://bitcoin.org/bitcoin.pdf`

[9] [Nakamoto09c] "Bitcoin: A Peer-to-Peer Electronic Cash System" by Satoshi Nakamoto, Oct 2008. `urlhttps://bitcoin.org/bitcoin.pdf`

[10] [Jacobson06a] "A New Way to Look at Networking" by Van Jacobson. Google Tech Talks Archive, Aug 2006. `https://youtu.be/oCZMoY3q2uM`

[11] [Jacobson06b] "A New Way to Look at Networking" by Van Jacobson. Google Tech Talks Archive, Aug 2006. `https://youtu.be/oCZMoY3q2uM`

[12] [Medium] "How The Byzantine General Sacked the Castle - A Look into Blockchain" by Debraj Ghosh. Medium.com, Apr 2016.

[13] [Shostack82] "The Byzantine Generals Problem" by L. Lamport, R.Shostack & M. Pease. ACM Transactions on Programming Languages and Systems, Vol. 4, No. 3. Jul 1982.

[14] [Nakamoto09d] "Bitcoin: A Peer-to-Peer Electronic Cash System" by Satoshi Nakamoto, Oct 2008. `https://bitcoin.org/bitcoin.pdf`

[15] [Nakamoto09e] "Bitcoin: A Peer-to-Peer Electronic Cash System" by Satoshi Nakamoto, Oct 2008. `https://bitcoin.org/bitcoin.pdf`

[16] [Zhang17b] "An Overview of Named Data Networking by Lixia Zhang." Proceedings of MILCOM 2017. Baltimore, MD, USA, Oct 2017.

[17] [Zhang14] "Named Data Networking" by Lixia Zhang, Van Jacobson & Alexander Afanasyev. Proceedings of ACM SIGCOMM 2014. Chicago, IL, USA, Aug 2014.

[18] [Named-Data.net] "Name Docs" by NDN Team. NDN Packet Format Specification. `https://named-data.net/doc/NDN-packet-spec/current/name.html`

[19] [Yuan15a] "Reliably Scalable Name Prefix Lookup" by Haowei Yuan & Patrick Crowley.Proceedings of ACM/IEEE Symposium on Architectures for Networking and Communication Systems(ANCS 2015). Oakland, CA, USA. May 2015.

[20] [Wikipedia] “Example for Longest Prefix Matching.” `https://en.wikipedia.org/wiki/Longest_prefix_match`

[21] [Yuan15b] “Reliably Scalable Name Prefix Lookup” by Haowei Yuan & Patrick Crowley.Proceedings of ACM/IEEE Symposium on Architectures for Networking and Communication Systems(ANCS 2015). Oakland, CA, USA. May 2015.

[22] [Yuan15c] “Reliably Scalable Name Prefix Lookup” by Haowei Yuan & Patrick Crowley.Proceedings of ACM/IEEE Symposium on Architectures for Networking and Communication Systems(ANCS 2015). Oakland, CA, USA. May 2015.

[23] [Rainer18a] “Challenges and Opportunities of Named Data Networking in Vehicle-To-Everything Communication: A Review” by Benjamin Rainer & Stefan Petscharing. Center for Safety and Communications Technologies, Austrian Institute of Technology. Vienna, Austria. October 2018.

[24] [ndn-docs] “Face Class” by NDN Team. NDN Common Client Libraries API 0.6.5. documentation. `https://named-data.net/doc/ndn-ccl-api/face.html`

[25] [V2E paperb] “Challenges and Opportunities of Named Data Networking in Vehicle-To-Everything Communication: A Review” by Benjamin Rainer & Stefan Petscharing. Center for Safety and Communications Technologies, Austrian Institute of Technology. Vienna, Austria. October 2018.

[26] [V2E paperc] “Challenges and Opportunities of Named Data Networking in Vehicle-To-Everything Communication: A Review” by Benjamin Rainer & Stefan Petscharing. Center for Safety and Communications Technologies, Austrian Institute of Technology. Vienna, Austria. October 2018.

[27] [Jacobson09a] “Networking Named Content” by Van Jacobson, Diana Smetters, James Thornton, Michael Plass, Nicholas Briggs & Rebecca Braynard. Proceed-

ings of the 5th International Conference on Emerging Networking Experiments and Technologies. Rome, Italy. December 2009.

[28] [Jacobson09b] “Networking Named Content” by Van Jacobson, Diana Smetters, James Thornton, Michael Plass, Nicholas Briggs & Rebecca Braynard. Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies. Rome, Italy. December 2009.

[29] [NFDTeam16] “NFD Developer’s Guide” by the NFD Team. NDN Technical Report, NDN-0021. Oct 2016.

[30] [Wang18a] “A Secure Link State Routing Protocol for NDN” by Lan Wang, Vince Lehman, A.K.M. Mahmudul Hoque, Beichuan Zhang, Yingdi Yu & Lixia Zhang. In IEEE Access, Vol. 6. Jan 2018.

[31] [Wang18b] “A Secure Link State Routing Protocol for NDN” by Lan Wang, Vince Lehman, A.K.M. Mahmudul Hoque, Beichuan Zhang, Yingdi Yu & Lixia Zhang. In IEEE Access, Vol. 6. Jan 2018.

[32] [Wang18c] “A Secure Link State Routing Protocol for NDN” by Lan Wang, Vince Lehman, A.K.M. Mahmudul Hoque, Beichuan Zhang, Yingdi Yu & Lixia Zhang. In IEEE Access, Vol. 6. Jan 2018.

[33] [Wang18d] “A Secure Link State Routing Protocol for NDN” by Lan Wang, Vince Lehman, A.K.M. Mahmudul Hoque, Beichuan Zhang, Yingdi Yu & Lixia Zhang. In IEEE Access, Vol. 6. Jan 2018.

[34] [Wang18e] “A Secure Link State Routing Protocol for NDN” by Lan Wang, Vince Lehman, A.K.M. Mahmudul Hoque, Beichuan Zhang, Yingdi Yu & Lixia Zhang. In IEEE Access, Vol. 6. Jan 2018.

[35] [Wang18f] “A Secure Link State Routing Protocol for NDN” by Lan Wang, Vince

Lehman, A.K.M. Mahmudul Hoque, Beichuan Zhang, Yingdi Yu & Lixia Zhang. In IEEE Access, Vol. 6. Jan 2018.

[36] [Afanasyev13a] "Let's ChronoSync: Decentralized Dataset State Synchronization in Named Data Networking" by Alexander Afanasyev & Zhenkai Zhu.Proceedings of 21st IEEE International Conference on Network Protocols(ICNP 2013).Göttingen, Germany. Oct 2013.

[37] [Afanasyev13b] "Let's ChronoSync: Decentralized Dataset State Synchronization in Named Data Networking" by Alexander Afanasyev & Zhenkai Zhu.Proceedings of 21st IEEE International Conference on Network Protocols(ICNP 2013).Göttingen, Germany. Oct 2013.

[38] [Ioannou14a] "Towards On-Path Caching Alternatives in Information-Centric Networks" by Andrianna Ioannou & Stefan Weber. Proceedings of 39th Annual IEEE Conference on Local Computer Networks. Edmonton, AB, Canada. Sept 2014.

[39] [Weber15] "Towards On-Path Caching Alternatives in Information-Centric Networks" by Andrianna Ioannou & Stefan Weber. Proceedings of 39th Annual IEEE Conference on Local Computer Networks. Edmonton, AB, Canada. Sept 2014.

[40] [Merkle79a] "Protocols for Public-Key Cryptosystems" by Ralph Merkle. In 1980 IEEE Symposium on Security and Privacy. Oakland, CA, USA. Apr 1980.

[41] [Spyridon18a] "An Overview of Security Support in Named Data Networking" by Spyridon Mastorakis, Zhiyi Zhang, Yingdi Yu, Haitao Zhang, Eric Newberry, Yanbiao Li, Alexander Afanasyev & Lixia Zhang. NDN Technical Report, NDN-0057, Rev. 2. Apr. 2018.

[42] [Spyridon18b] "An Overview of Security Support in Named Data Networking" by Spyridon Mastorakis, Zhiyi Zhang, Yingdi Yu, Haitao Zhang, Eric Newberry,

Yanbiao Li, Alexander Afanasyev & Lixia Zhang. NDN Technical Report, NDN-0057, Rev. 2. Apr. 2018.

[43] [Spyridon18c] “An Overview of Security Support in Named Data Networking” by Spyridon Mastorakis, Zhiyi Zhang, Yingdi Yu, Haitao Zhang, Eric Newberry, Yanbiao Li, Alexander Afanasyev & Lixia Zhang. NDN Technical Report, NDN-0057, Rev. 2. Apr. 2018.

[44] [Spyridon18d] “An Overview of Security Support in Named Data Networking” by Spyridon Mastorakis, Zhiyi Zhang, Yingdi Yu, Haitao Zhang, Eric Newberry, Yanbiao Li, Alexander Afanasyev & Lixia Zhang. NDN Technical Report, NDN-0057, Rev. 2. Apr. 2018.

[45] [Spyridon18e] “An Overview of Security Support in Named Data Networking” by Spyridon Mastorakis, Zhiyi Zhang, Yingdi Yu, Haitao Zhang, Eric Newberry, Yanbiao Li, Alexander Afanasyev & Lixia Zhang. NDN Technical Report, NDN-0057, Rev. 2. Apr. 2018.

[46] [Spyridon18f] “An Overview of Security Support in Named Data Networking” by Spyridon Mastorakis, Zhiyi Zhang, Yingdi Yu, Haitao Zhang, Eric Newberry, Yanbiao Li, Alexander Afanasyev & Lixia Zhang. NDN Technical Report, NDN-0057, Rev. 2. Apr. 2018.

[47] [Spyridon18g] “An Overview of Security Support in Named Data Networking” by Spyridon Mastorakis, Zhiyi Zhang, Yingdi Yu, Haitao Zhang, Eric Newberry, Yanbiao Li, Alexander Afanasyev & Lixia Zhang. NDN Technical Report, NDN-0057, Rev. 2. Apr. 2018.

[48] [Merkle79b] “Protocols for Public-Key Cryptosystems” by Ralph Merkle. In 1980 IEEE Symposium on Security and Privacy. Oakland, CA, USA. Apr 1980.

[49] [memphis.edu] “What is MiniNDN?” by University of Memphis. `http://minindn.memphis.edu/`

[50] [Soustroup] “The Design and Evolution of C++”. pp. 207.

[51] [Kutscher15] “A Survey of Information-Centric Networking” by Dirk Kutscher, Börje Ohlman, Claudio Imbrenda, Christian Dannewitz & Bengt Ahlgren. In IEEE Communications Magazine Vol. 50, No. 7. July 2012.

[52] [Afanasyev13c]“Let’s ChronoSync: Decentralized Dataset State Synchronization in Named Data Networking” by Alexander Afanasyev & Zhenkai Zhu.Proceedings of 21st IEEE International Conference on Network Protocols(ICNP 2013).Göttingen, Germany. Oct 2013.

[53] [Wang18g] “A Secure Link State Routing Protocol for NDN” by Lan Wang, Vince Lehman, A.K.M. Mahmudul Hoque, Beichuan Zhang, Yingdi Yu & Lixia Zhang. In IEEE Access, Vol. 6. Jan 2018.

[54] [Wang18h] “A Secure Link State Routing Protocol for NDN” by Lan Wang, Vince Lehman, A.K.M. Mahmudul Hoque, Beichuan Zhang, Yingdi Yu & Lixia Zhang. In IEEE Access, Vol. 6. Jan 2018.

[55] [Wang18i] “A Secure Link State Routing Protocol for NDN” by Lan Wang, Vince Lehman, A.K.M. Mahmudul Hoque, Beichuan Zhang, Yingdi Yu & Lixia Zhang. In IEEE Access, Vol. 6. Jan 2018.

[56] [Wang18j] “A Secure Link State Routing Protocol for NDN” by Lan Wang, Vince Lehman, A.K.M. Mahmudul Hoque, Beichuan Zhang, Yingdi Yu & Lixia Zhang. In IEEE Access, Vol. 6. Jan 2018.

[57] [Papadimitriou09] “Real Time Video Streaming over Heterogeneous Networks” by Panagiotis Papadimitriou & Vassilis Tsaussidis. Proceedings of 11th Inter-

national Conference On Advanced Communication Technology, 2009(ICACT 2009).Gangwon-Do, South Korea. Feb 2009.