Title
Named Data Networking on AIR

Author
Vakayil Palakandy, Priyesh

Publication Date
2017

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
SANTA CRUZ

NAMED DATA NETWORKING ON AIR

A thesis submitted in partial satisfaction of the requirements for the degree of

Masters of Science

in

COMPUTER ENGINEERING

by

Priyesh Vakayil Palakandy

March 2017

The Thesis of Priyesh Vakayil Palakandy is approved:

Professor J.J. Garcia-Luna-Aceves, Chair

Professor Chen Qian

Professor Brad Smith

Tyrus Miller
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

Abstract

Named Data Networking on AIR

by

Priyesh Vakayil Palakandy

This thesis examines Automatic Incremental Routing (AIR) [1] algorithm that run on a Named Data Networking (NDN) [2] architecture. To study the performance of the AIR algorithm we simulated multiple nodes in the network simulator NS3 [6]. ndnSIM [5] was used to simulate NDN routing between the nodes. We compared the number of hop counts required from an arbitrary source node to an arbitrary destination node using AIR routing with respect to Dijkstra's [3] shortest path first algorithm. We found that AIR routing rarely takes over one hop count more than the hop counts obtained from Dijkstra's algorithm. From this empirical result, we conclude that AIR routing provides us with an alternative approach to name-based link-state routing, and gives us the advantages of automatic routing while not severely impacting performance.

Chapter 1

Introduction

The limitations of the current IP Internet architecture with respect to contemporary data traffic, calls for re-architecting the Internet with a novel design that caters to the new content-dissemination demands placed on the network. The data networking (NDN) architecture [2] has been proposed as a possible approach to meet such demands and its basic design has assumed the modification of traditional link-state routing to populate the forwarding information bases (FIB) used by routers to determine the next hops that requests for content (Interests) should traverse towards the producers of content.

On the other end of the spectrum, wireless communication and mobile applications are becoming prominent owing to the increasing popularity of mobile phones, "Internet of Things" and wireless sensors among others. The automatic incremental routing (AIR) protocol [1] has been proposed to provide more efficient routing in MANETs by labeling nodes with prefix labels that make the routes from sources to destinations automatic through simple comparisons of the destinations labels with the labels of relaying nodes. For the automatic routes to work, the source must first find out the prefix label of the intended destination.

This thesis evaluates the viability of the AIR protocol for the NDN architecture by simulating it using a network simulator running on the NDN architecture. A real world topology was used to compare the routing performance of AIR over Dijktra's shortest path algorithm. Chapter 2 summarizes important limitations of the Internet architecture. Chapter 3 describes the NDN architecture. Chapter 4 summarizes prior work related to AIR and describes the operation of AIR. Chapter 5 discusses our implementation of AIR and describes the results of simulations used to compare AIR with the results obtained using Dijkstra's shortest-path algorithm on the network topology. The results of our simulations show that AIR can be used instead of a link-state routing protocol in NDN, resulting in signaling savings without a large penalty incurred in the quality of paths traversed by Interests and data packets.

Chapter 2

Limitations of the Internet

Modern Internet applications employ the TCP/IP suite of communication protocols, of which, much was initially designed in the late 1960s. Though a lot of improvements have been made over the years to the original model, the fundamental principles on which TCP/IP was designed has persevered throughout these revisions. One of the defining principles in the original design that has persisted throughout the years, is the notion of an infrastructure oriented computation and/or networking model[13]. During the early days of the Internet links and storage were expensive and content often resided on single hosts. These constraints greatly influenced the original design of the Internet[13]. Thus, networking protocols were designed to efficiently use hardware resources, enhance network sharing and accommodate host to host communication. One outcome of such a design is employing host based addressing for IP packets.

Since then, hardware costs have plummeted and connectivity has increased significantly. Computing and networking models have been shifting from the age of mainframes and time-shares to the age of cloud based computing and personal computing with smartphones. Data is no longer restricted to text. Voice, images and video traffic are being extensively distributed on the Internet, in fact

80%[12] of the current Internet traffic is video content. Advanced digital coding mechanisms and the ease of access to create content on the web is improving day by day[9]. These advances have expedited the proliferation of volume and distribution of content through Internet networks. Recent studies forecast Internet traffic reaching the order of hundreds of exabytes by the year 2020[11]. The Internet is thus no longer a means of point to point communication between two entities. These advances have caused a paradigm shift from infrastructure centric networking to information centric networking[13]. Thus future research should focus on innovative ways to process information efficiently, reduce transfer of duplicate data wherever possible, and improve accessibility and searchability of content.

The physical medium of network access for end users is also shifting from wired to wireless. There has been an exponential growth in wirelessly connected devices such as mobile phones, wireless sensors and IOT devices. Cisco predicts smartphone traffic will exceed PC traffic by 2020[12]. These trends indicate a growing need for mobile services and applications that are location independent and within social contexts. But, current wireless networking technologies are yet to be fully autonomic and scalable. Virtualization technologies have also surged in the past decade and IPV4 addresses of migrating virtual hosts are no longer confined to single physical hosts, again supporting the need for location independent addressing.

Our reliance on wireless technologies and Internet applications has considerably increased over the years. Some applications include online banking services and social media services where privacy and security demand utmost consideration. Content security is thus a growing concern. Current technologies employ indirect mechanisms such as securing hosts or the connection to gain trust in the content being transferred [9]. There is thus a research opportunity to redesign underlying Internet technologies wherein content is inherently made secure enabling location independent security [13].

The hourglass structure of TCP/IP has been instrumental in the success of the Internet. Such a structure imposes a simplistic network layer (layer 3) enabling an all inclusive and compatible layer amongst different protocol suites and dissimilar underlying physical technologies [13]. Since, networking in the Internet is moving towards an information centric model there is incentive to move to an hourglass structure that focuses on supporting heterogeneous end user application and content rather than connectivity among hosts [13].

The unprecedented number of connected devices on the Internet has brought about imminent challenges for future growth. Notable concerns include: ever increasing routing table sizes and poorly scaling services such as the DNS service [13]. The current design does not integrate routing name resolution with content distribution well and this has caused unwieldy workarounds and application specific solutions such as Content Delivery Networks(CDNs) to be deployed to overcome some of these challenges placed on TCP/IP networks.

Chapter 3

Named Data Networking

NDN is an Internet architecture that aims to overcome the shortcomings of the current TCP/IP model. NDN is based on Content Centric Networking (CCN) proposed by Van Jacobson[7]. Zhang et al. [9] paper titled "Named Data Networking Project" discusses NDN project's vision, architecture and research agenda of which the architecture is summarized in Section 4.1.

IP realizes communication endpoint names as addresses in IP datagrams. Names in NDN datagrams are hierarchical, yet arbitrary identifiers used to name data instead of endpoints. With such a naming scheme, data is promoted as the fundamental entity of the Internet architecture, in contrast to IP, where the host of the data is featured. This shift lends itself elegantly to the way applications are written. For example, most applications are designed to primarily process content and offload the responsibility of operating on the content's location to lower layers.

Security is incorporated within the NDN architecture and not designed as an afterthought. Traditionally, the approach to security is to employ an all-encompassing model where the transmission link is secured rather than the content. NDN names impart necessary security context empowering NDN consumers to verify the content's provenance and isolate it to a single producer.

Exclusive naming of data chunks prevents looping and allows routers to request for content and distribute them through any of its interfaces, unlike IP where restrictive single-path forwarding is enforced.

3.1 NDN Architecture

NDN maintains the hourglass structure of IP and the separation of forwarding and routing planes. NDN is a receiver driven communication architecture.

To solicit for content a consumer names the content it is seeking and sends out an Interest packet to NDN routers. NDN routers forward these Interests based on a Forward Information Base(FIB) which is populated employing a name-based routing protocol. Each router records the Interest and the interface on which it arrived. Once a content provider or producer receives an Interest[4], it sends back the corresponding content and the producer's signature along the forward path of the Interest, finding its way back to the consumer. The content and the Interest are cached along the NDN routers for a set period of time. Interests are recorded in a Pending Interest Table(PIT) in each of these routers. Requests for the same content from other consumers can now be retrieved from these caches, thus economically using the network infrastructure and reducing the total round trip delay. This approach aims to enhance multicast communication, content distribution, mobility in addition to making the Internet fault tolerant and resilient.

3.2 NDN Names

NDN names are abstracted from the underlying network allowing applications to independently design names. NDN uses hierarchical names to facilitate compartmentalization of data chunks or to identify different aspects of the content, for example, versioning or sequencing of the data. Names need to be unique only if the content is global.

3.3 Pending Interest Table

A PIT contains entries of arriving Interests and the interface on which it was received. If there is no valid PIT entry for a received data, it will be dropped. Hence PIT entries need to persist till the arrival of corresponding content from the producer. But, it should be evicted out to allow for newer ones. The size of the PIT table thus limits the load a NDN router can accommodate.

3.4 Content Store

Upon receiving an Interest the router initially inspects its Content Store for matching content. If the requested data is available, it forms a response and sends it back to the router or consumer which requested for it. Contemporary routers have similar buffer memory that caches data, but they cannot be capitalized for additional requests for the same data.

Chapter 4

Prior Work

There has been considerable work on routing using virtual and geographical coordinates. S. Das [17] et al. compares the performance of various routing approaches based on coordinate information. Some of the noteworthy ones are listed below.

Tribe [18] uses a virtual network representation to populate the relative location of nodes in a network infrastructure. Since nodes are at addresses in continuous intervals, routing tables are of smaller sizes. This approach does not scale well with topology changes because need to be recalculated for addition or deletion of nodes.

BVR [19] uses virtual coordinates to overcome the limitations (such as line of sight requirements) of geographical coordinates informed routing such as XYLS [21] and GLS [20], but these methods have collisions in coordinates either causing suboptimal routing or adding extra signaling to resolve collisions.

Kademlia [14], Tapestry [15] use a DHT over a virtual topology on top of a physical network. But signaling overhead occurs when links have to communicate multi-hop paths.

AIR [1] was first proposed for mobile ad hoc networks (MANET). However, it is equally applicable to wired networks and is arguably one of the best-performing protocols based on virtual coordinates. Accordingly, we chose it as a replacement of traditional link-state routing in the context of NDN.

We provide a summary description of AIR based on the original paper that introduced it [1]. In the AIR protocol, first, a root node is elected among a group of nodes. All nodes are labeled based on its location from the root node. Since the applications are unaware of these labels AIR maintains a Distributed Hash Table (DHT) across some nodes known as anchor nodes. Anchor nodes store names and the corresponding labels. When a node needs to make its presence known to the network it hashes its identifier to get a label. This label points to the address of an anchor node. It then passes its name and label information to the anchor node. When a source node wants to connect to a destination it hashes the destination identifier to get the anchor node's prefix label. It then subscribes to the anchor node to get the prefix label of the destination. Once it has a destination label, it can use AIR's incremental routing algorithm to route to the destination.

4.1 Assignment of Prefix Labels

First among the nodes connected, a root node is elected. This idea can be extended to multiple root nodes, as per Multi-root Automatic Incremental Routing (MAIR)[8]. The root node election is similar to STP in that neighbor to neighbor hello messages are used to create and maintain the LDAG. A node looks at its neighbor table and checks if any of its neighbors are labeled. If they are not labeled, it waits for a local time to expire before it elects itself as the root node. The node with the lowest node identifier becomes the parent node for its neighbors. Once a root node is elected its neighbors are marked as its children and corresponding labels are used to name the root's children. For example, as

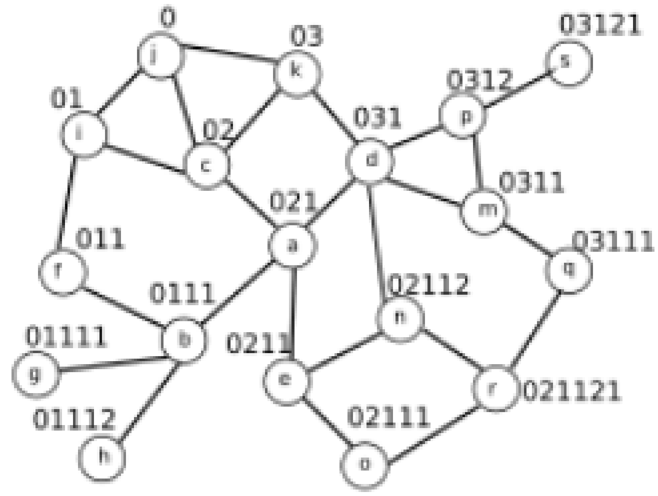


Figure 4.1: Assigning Prefix Labels [1]

shown in Figure 4.1 , the root is named with a prefix /0, and its children are named 0/1, 0/2 and 0/3. Once a node is named, its neighbors are marked as its children and named with respect to itself as shown in Figure 1a.

4.2 Building Distributed Hash Tables

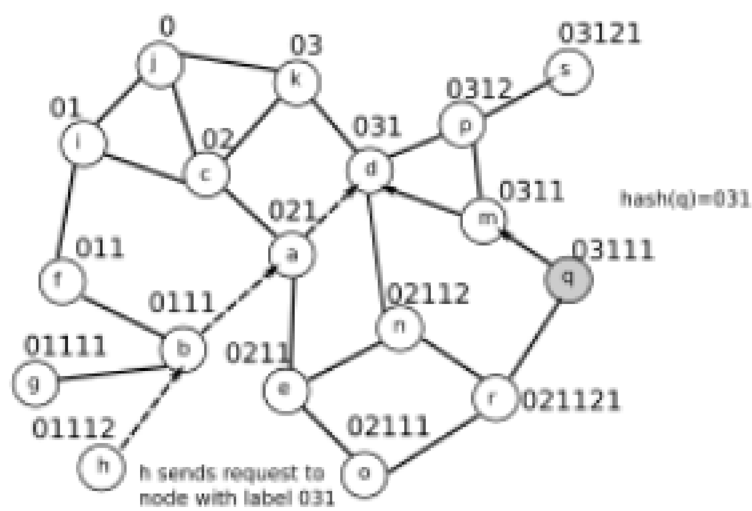


Figure 4.2: Using Hash Function to Route to Anchor [1]

Every node publishes its prefix label to an anchor node. The node obtains the prefix label of an anchor node by hashing its identifier. In Figure 4.2 Node 'q' hashes its identifier 'q' using a hash function to obtain 0/3/1 which is the prefix label of node 'd'. Thus node 'd' becomes the anchor node for node 'q'. If its prefix label changes due to some reason it updates the anchor node. Multiple stores that contain mappings of node identifiers to prefix labels need to be maintained to avoid bottlenecks and single points of failure.

Source nodes obtain the prefix labels of destinations by subscribing to anchor nodes. It uses the same hashing function to obtain the anchor node from the destination node identifier. In 4.2, node 'h' needs to identify node 'q'. It uses the hash function to identify node 'd' as the anchor node. It then subscribes to the anchor node to obtain the prefix label of the destination.

4.3 Routing in AIR

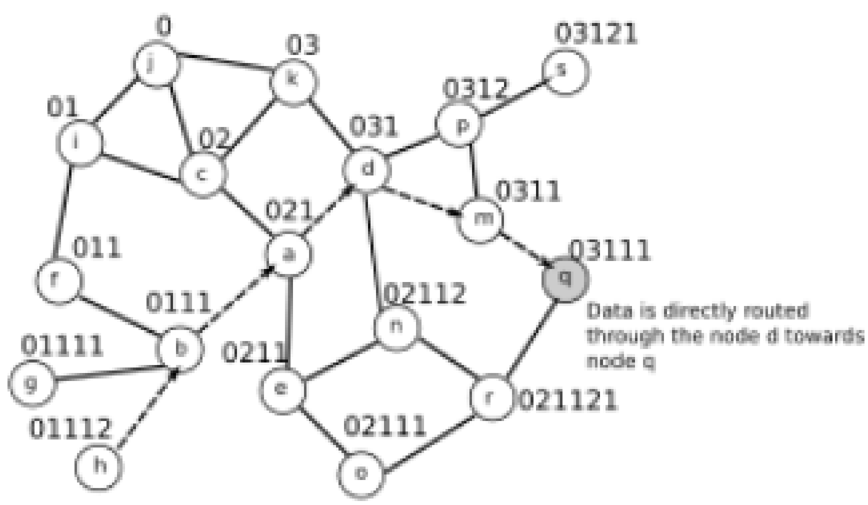


Figure 4.3: Routing Unicast Traffic [1]

Every node keeps a table of its direct neighbors or one hop neighbor table and its two hop neighbors. When a node wants to route a packet to a destination 'q', it finds the maximum matching prefix of its two hop neighbors with respect to the destination and selects the next hop as the neighbor that routes to the two hop neighbor as shown in Figure 4.3. If two hop neighbor nodes offer the same matching prefix then the next hop is randomly selected. To obtain a richer path diversity, longest prefix matching is performed on two-hop neighbors rather than directly routing through parents.

Chapter 5

Implementation and Results

ndnSIM is an open source simulator over the NS-3 framework. It attempts to simulate basic NDN protocol operations. This thesis implements the AIR algorithm on ndnSIM. The project source code is hosted at `https://github.com/priyesh16/thesis` .

The first step in the implementation process was to run AIR on ns3 and verify the feasibility of the AIR algorithm. To do this, we implemented the AIR algorithm on an 18 node topology similar to the one in [1]. Figure 5.1 shows the topology we used.

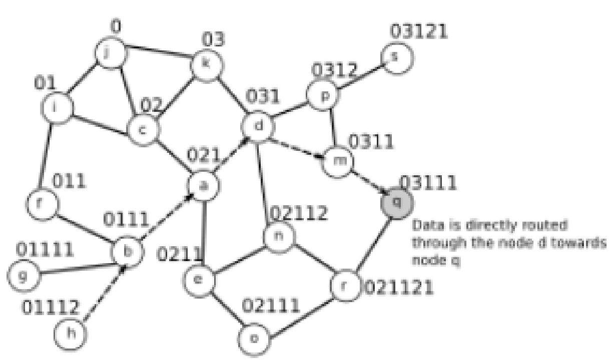
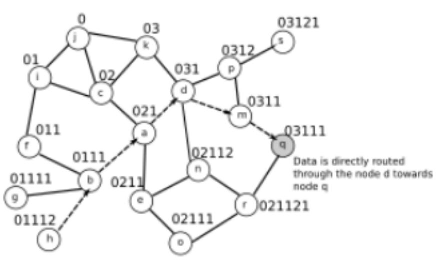


Figure 5.1: Topology used to test AIR routing

Hard-coded prefix names were initially used to prototype the working of the AIR algorithm. All names are stored in a trie data structure to compute longest prefix matches. NS3's topologyReader class was used to parse a text file containing the network topology. The topology reader provides a list of nodes and the links it is connected to. Using the APIs provided by these links, each node maintains a list of its direct neighbours and two-hop neighbours. The node with the lowest node identifier was elected as the root node. Then, the list of child nodes for each node are computed from the prefix names and direct neighbours. Given a producer nexthops are computed for every node.

Figure 5.2 shows the computation of nexthop for node 'a' given producer 'q'. From the list of two-hop neighbours of 'a', the node with the longest prefix match for destination 'q' (0/3/1/1/1) is node 'm'(/0/3/1/1). The direct neighbour for node 'a' reaching node 'm' is node 'd'. Hence node 'd' was computed as the next hop for node 'a'.




```
Destination Prefix is/0/3/1/1/1
```

```
Sourcename a
    1HopNbr b
        2HopNbr f : /0/1/1
        2HopNbr g : /0/1/1/1/1
        2HopNbr h : /0/1/1/1/2
    1HopNbr c
        2HopNbr i : /0/1
        2HopNbr j : /0
        2HopNbr k : /0/3
    1HopNbr d
        2HopNbr k : /0/3
        2HopNbr p : /0/3/1/2
        2HopNbr m : /0/3/1/1
        2HopNbr n : /0/2/1/1/2
    1HopNbr e
        2HopNbr n : /0/2/1/1/2
        2HopNbr o : /0/2/1/1/1
Longest Prefix found is/0/3/1/1
Next hop is d
```

Figure 5.2: Computing Nexthops Based on Two Hop Neighbour Table

Every node and its corresponding nexthop is added as an entry to the FIB table. Finally, using ns3's Simulator class an Interest packet is sent from a consumer to a producer. We tested the arrival of this Interest packet at the producer's end. Figure 5.3 shows the output of NS3's PyViz simulator, showing AIR routing from consumer 'j' to producer 'r'

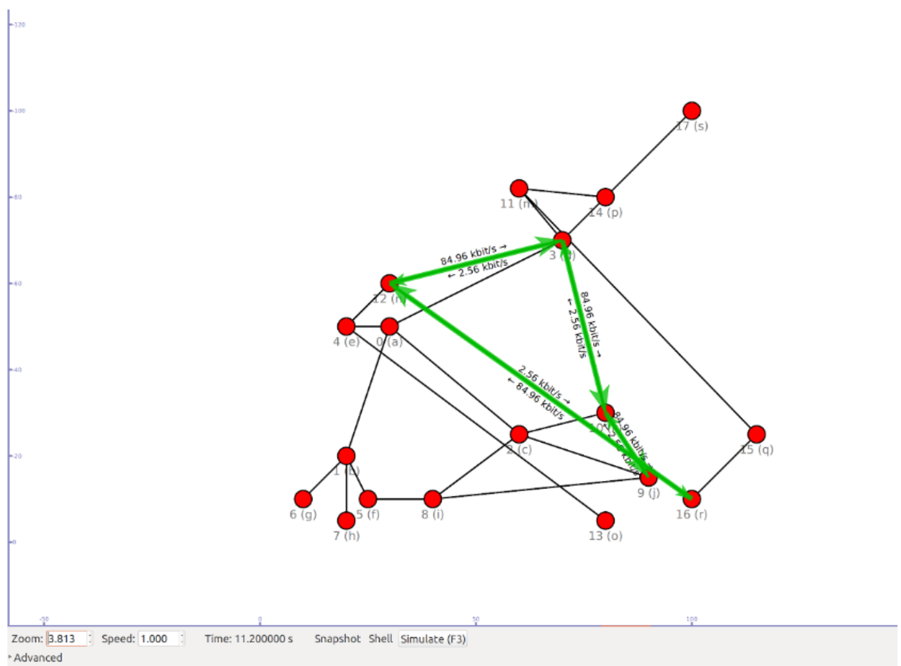


Figure 5.3: Simulation of AIR Routing

Since, AIR routing need not take the fastest route to a destination, we also looked at how it fared when compared to a shortest routing algorithm such as Dijkstra's. We used the built in GlobalRoutingHelper class provided by ndnSIM to implement Dijkstra's SPF algorithm and compared the forwarding path against AIR. Figure 5.4 shows the path taken by using Dijkstra's algorithm and Figure 5.5 shows the path taken by AIR. We see that AIR requires to go through an extra hop to reach the destination.

```
PriDebug: ForwardingStrategy::OnInterest() : forwarded to : 9
PriDebug: ForwardingStrategy::OnInterest() : forwarded to : 10
PriDebug: ForwardingStrategy::OnInterest() : forwarded to : 3
PriDebug: ForwardingStrategy::OnInterest() : forwarded to : 12
PriDebug: ForwardingStrategy::OnInterest() : forwarded to : 16
```

Figure 5.4: Path Taken by Dijkstra's algorithm

```
PriDebug: ForwardingStrategy::OnInterest() : forwarded to : 9
PriDebug: ForwardingStrategy::OnInterest() : forwarded to : 2
PriDebug: ForwardingStrategy::OnInterest() : forwarded to : 0
PriDebug: ForwardingStrategy::OnInterest() : forwarded to : 3
PriDebug: ForwardingStrategy::OnInterest() : forwarded to : 12
PriDebug: ForwardingStrategy::OnInterest() : forwarded to : 16
```

Figure 5.5: Path taken by AIR algorithm

```
Legend :
0 a    1 b    2 c    3 d    4 e    5 f
6 g    7h     8 i    9 j    10k    12m
13n    14o    15p    16q    17r
```

Once, we confirmed the working of AIR routing over NDN, we implemented automatic naming and name resolution by consumer subscription to anchor nodes. We used a very simplistic hash function over node identifiers to obtain anchor nodes.

Then we compared AIR performance over Dijkstra's using a real world network topology consisting of 153 nodes. Four nodes were made anchor nodes. We made every node a consumer and computed hop counts to every other node. The bar graph of hop counts obtained by routing from all nodes to a given producer is shown in Figure 5.7.

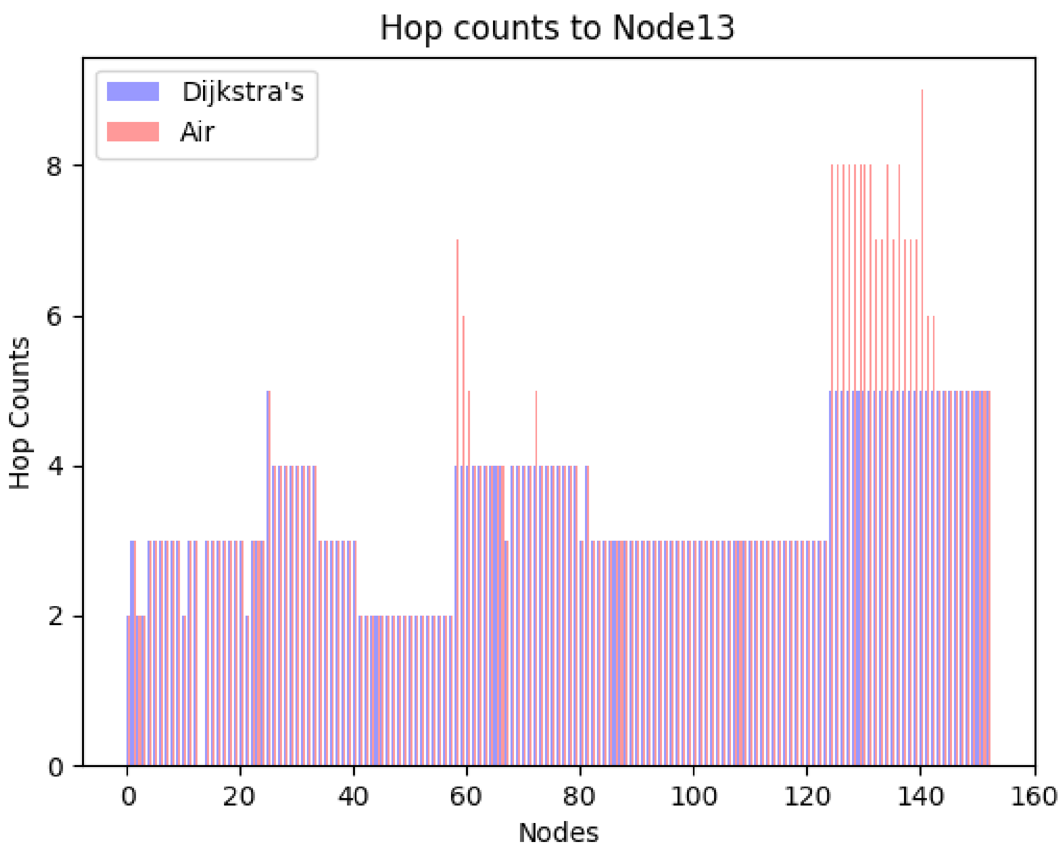


Figure 5.7: Bar Graph showing Hop Counts to Node 13 from Every Node

We can glean more information about the nature of AIR and Dijkstra's routing by sorting the above graph based on hop count. Figure 5.8 shows the sorted graph. From the sorted graph, we see that when the nodes are neighbouring each other (less than three hops apart in Figure 5.8) both Dijkstra's and AIR route indistinguishably. As the distance between nodes increase, AIR takes more hop counts than Dijkstra's. Also, the difference in hop counts increase as the distance between the nodes increase.

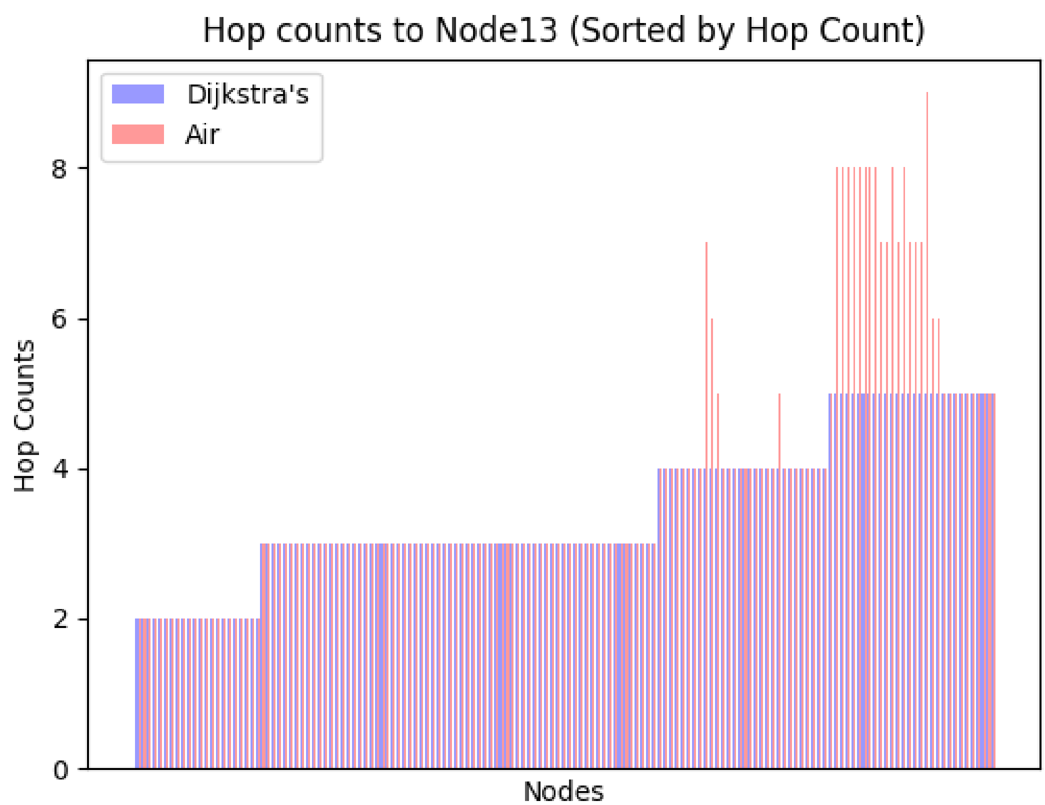


Figure 5.8: Bar Graph showing Hop Counts Sorted

Figure 5.9 shows the mean and standard deviation of hop counts of AIR and Dijkstra routing from every node to every other node as destination. We see that the mean hop count of AIR for every destination takes at most one hop count more than Dijkstra's. When we look at the maximum and minimum standard deviation of both algorithms the difference is almost always less than one hop count. This shows that AIR does not perform significantly poorly when compared with Dijkstra's

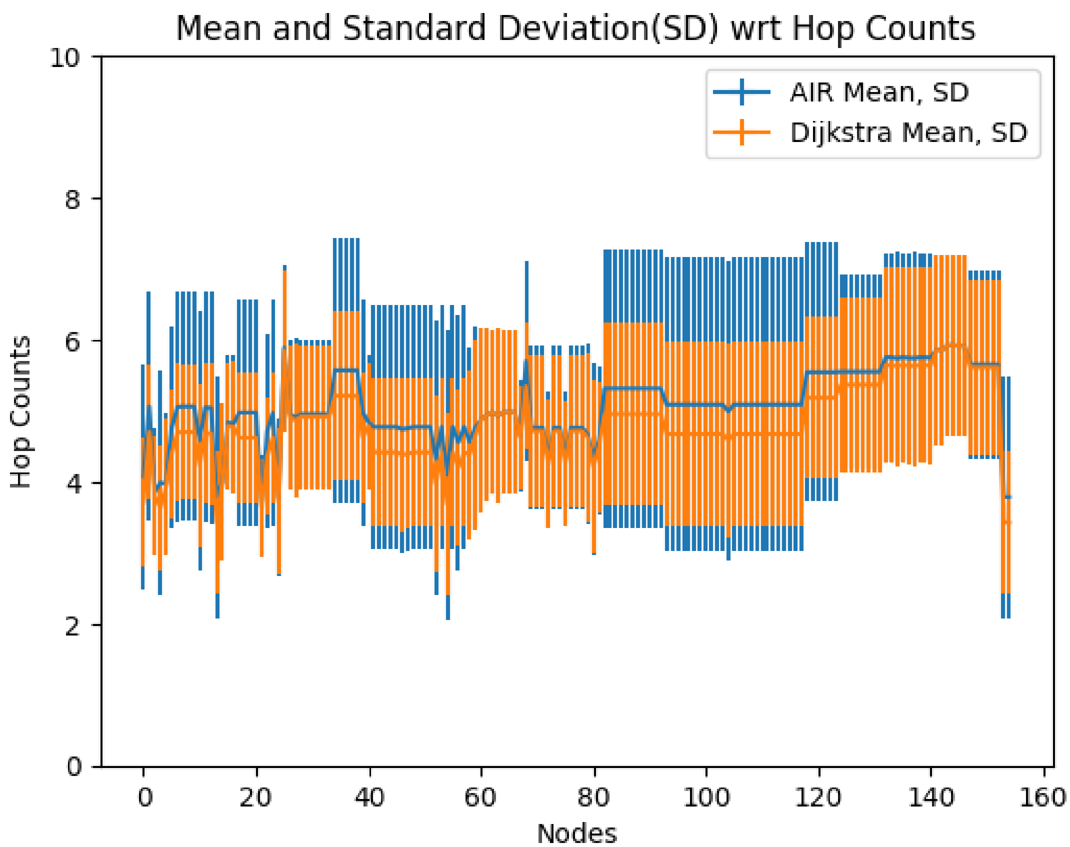


Figure 5.9: Mean and Standard Deviation of Hop Counts

Chapter 6

Conclusion and Future Work

The main objective of this thesis was to implement AIR as an alternative to name-based link-state routing for NDN. We theorized that AIR does not ensure shortest paths and thus we needed to compare its performance with traditional IP networks to see if the path stretch introduced by AIR was small enough to ensure that AIR could be a feasible solution for routing in NDN.

We showed that a consumer can always find a route to a producer using AIR routing on an NDN architecture. We also showed that the path obtained by AIR may not be the optimum path. However, the empirical evidence from simulation results indicates that the path stretch incurred by AIR is small even when the shortest paths from consumers to producers (sources and destinations) are short. Accordingly, AIR constitutes a viable replacement for name-based link-state routing in NDN, because it can render similar paths with far less signaling overhead to update routing and forwarding state, and less storage and processing overhead to forward Interests.

We used a single root as the origin of prefix label, as proposed in [1]. However, a multi-root approach could help reduce the path stretch incurred in AIR and reduce the likelihood that excessive signaling must be incurred after the failure of the root or links connecting neighbors to the root. A multi-root approach for AIR is discussed in [8], and is an important topic for further study in the context of NDN. The hash function we used to obtain anchor nodes was a very simplistic one. If the hash function is improved, DHTs in the topologies becomes more optimally distributed improving routing and reduces signaling. This is also an important topic for further research.

Bibliography

[1] J.J. Garcia-Luna-Aceves and D. Sampath, "Scalable Integrated Routing Using Prefix Labels and Distributed Hash Tables for MANETs", Proc. IEEE MASS 2009 , Oct. 2009.

[2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, Networking Named Content, in Proc. ACM CoNEXT, 2009.

[3] Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs", Numerische Mathematik. 1: 269-271. doi:10.1007/BF01386390.

[4] G. Carofiglio, M. Gallo, and L. Muscariello. ICP: Design and Evaluation of an Interest Control Protocol for Content-Centric Networking. In IEEE NOMEN 12, March 2012.

[5] A. Afanasyev, I. Moiseenko, and L. Zhang. ndnSIM: NDN simulator for NS-3. Technical Report NDN-0005. (NDNsim)

[6] F. Urbani, W. Dabbous, and A. Legout. (2011, November) NS3 DCE CCNx quick start. INRIA.

[7] Jacobson, Van. "A New Way to look at Networking". You Tube. Google Talk.

[8] R. Ghosh and J. J. Garcia-Luna-Aceves, "Automatic Routing Using Multiple Prefix Labels," Proc. IEEE Globecom 2012 , Dec. 3-7, 2012.

[9] Lixia Zhang et. al. Named Data Networking (NDN) Project, PARC Tech Report, October 30, 2010

[10] Alex Hinds, Michael Ngulube, Shaoying Zhu, and Hussain Al-Aqrabi, "A Review of Routing Protocols for Mobile Ad-Hoc NETworks (MANET)", International Journal of Information and Education Technology, Vol. 3, No. 1, February 2013.

[11] Cisco. "The Zettabyte Era - Trends and Analysis". http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html

[12] Cisco. "Cisco VNI Forecast and Methodology". http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html

[13] J.J. Garcia-Luna-Aceves. CMPE252B: Principles of Computer Communication. 252B-01-class.pdf, Spring 2016

[14] P. Maymounkov and D. Mazi‘eres. Kademlia: A peer-to-peer information system based on the xor metric. In 1st International Peer-to-Peer Symposium (IPTPS 2002), pages 5365, 2002.

[15] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and. Technical report, University of California at Berkeley, Berkeley, CA, USA, 2001.

[16] A.C. Viana, M.D. de Amorim, S. Fdida. Indirect routing using distributed location information. Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003. (PerCom 2003). 26-26 March 2003

[17] S. Das, H. Pucha, and Y. Hu. Performance comparison of scalable location services for geographic ad hoc routing. INFOCOM 2005, 2:12281239 vol. 2, March 2005.

[18] S. M. Das, H. Pucha, and Y. C. Hu. Distributed hashing for scalable multicast in wireless ad hoc networks. IEEE Transactions on Parallel and Distributed Systems, 19(3):347362, 2008.

[19] Rodrigo Fonseca , Sylvia Ratnasamy , Jerry Zhao , Cheng Tien Ee , David Culler , Scott Shenker , Ion Stoica, Beacon vector routing: scalable point-to-point routing in wireless sensornets, Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation, p.329-342, May 02-04, 2005

[20] J. Li, J. Jannotti, D. S. J. D. Couto, D. R. Karger, and R. Morris, "A scalable location service for geographic ad hoc routing," in Proc. of ACM MobiCom, August 2000.

[21] I. Stojmenovic, "A routing strategy and quorum based location update scheme for ad hoc wireless networks," SITE, University of Ottawa, Tech. Rep. TR-99-09, September 1999.