A Grouped Hamming Network

by

Bryan Logan

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2010

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

A distributed hash table (DHT) is a type of peer-to-peer (P2P) network that, like traditional hash tables, maps *keys* to *values*. Unlike traditional hash tables, however, the data is distributed across a network with each node being responsible for a particular range of keys. Numerous other DHTs have been presented and have become the cornerstone of wildly popular P2P file-sharing applications, such as BitTorrent. Each of these DHTs trades-off the number of pointers maintained per node with the overhead and lookup time; storing more pointers decreases the lookup time at the expense of increased overhead.

A Grouped Hamming Network (GHN), the overlay network presented in this thesis, allows for the number of pointers per node to be any increasing function of n, $\mathcal{P}(n) = \Omega(\log n)$. The system presented assumes that nodes fail independently and uniformly at random with some probability $q = 1 - p$. Three different schemes for routing in a GHN are presented. For each routing scheme a theoretical estimate on the probability of failure is given and optimal configurations in terms of n and $\mathcal{P}(n)$ are given. Simulations of GHNs with various configurations indicate that the given estimates are indeed accurate for reasonable values of q and that the optimal configurations are accurate.

Acknowledgements

I would like to thank my supervisor, Ian Munro, for his guidance and patience in helping me complete my thesis. I would also like to thank my fellow student Lucasz Cwik for the many afternoons spent discussing DHTs and their related problems; without those meetings I likely would not have ended up with this topic. Finally, I would like to thank my family – speficially Mom, Dad, Grandma and Grandpa – for their unwavering support in helping me further my education.

Contents

Contents **v**

List of Figures **vii**

List of Algorithms **xi**

1 Introduction **1**

2 Background **4**

3 System Overview **7**

3.1 Notation and Conventions 9

3.2 Network Layout 12

3.3 Constraints 14

4 Routing **25**

4.1 The Probability of a Path Existing 26

4.2 The Naïve Algorithm 28
4.2.1 Calculating the Probability of Failure 29
4.2.2 Minimizing the Probability of Failure 36
4.3 The Always-Forwards Algorithm 40
4.3.1 Calculating the Probability of Success 41
4.3.2 Maximizing the Probability of Success 43
4.4 The Never-Backwards Algorithm 47
4.4.1 Calculating the Probability of Failure 49
4.4.2 Maximizing the Probability of Success 53

5 Simulations **55**

5.1 The Probability of Any Path Existing 57
5.2 Simulating the Naïve Algorithm 58
5.3 Simulating the Always-Forwards Algorithm 60
5.4 Simulating the Never-Backwards Algorithm 61

6 Conclusion **70**

6.1 Future Work 70
6.2 Summary 71

References **73**

List of Figures

3.1 The Lambert W function. The solid blue line is the positive branch, $\mathcal{W}(x)$, and the dashed red line is the negative branch, $\mathcal{W}_{-1}(x)$. 9

3.2 Two different ways of routing from N_i to N_k where N_i and N_k are in neighbouring cliques. The solid blue lines represent the path taken and the light dotted lines represent the other connections that are not used. 11

3.3 A row taken from a GHN with width 3. The filled red circles represent nodes in the network. Nodes in the same clique are surrounded by a black circle and intra-clique connections are shown as black dashed lines, while connections between adjacent cliques are shown in green and longer connections are shown in blue. 12

3.4 A full GHN in two dimensions with width 2 and clique size of 3. The filled red circles represent nodes in the network. Nodes in the same clique are surrounded by a black circle and intra-clique connections are shown as black dashed lines, while connections between nodes in adjacent cliques are shown in green. 13

4.1 A sample path through a GHN with $k = 2$ and $d = 2$ where the message originates at the dashed circle in C_1 and is destined for C_3. 34

4.2 A summary of all configurations of node failures in a path of length three with $k = 2$, where A means the node is alive and D means the node is dead. The *Route?* column indicates whether it is possible to route a message through this configuration; Y indicates that it is possible to route a message, while N indicates it is not. 35

5.1 Simulation results for finding the probability of a path existing in a GHN. Each chart shows the probability of routing successfully as a function of q and k with the line representing the predicted lower bound on the probability of success and the dots representing the mean of 500 trials. The columns of dots occur when numerous trials were performed with the same q and k, but different d; trials with a larger d value are found at the bottom of the column, while trials with low d values occur at the top. 62

5.2 Simulation results for finding the probability of a path existing in a GHN. Each chart shows the probability of routing as a function of k for $d = 3$ and varying values of q. Each marker represents the success rate observed over 500 trials. Multiple markers occur when their values of d, k and q are the same, but w is different. 63

5.3 Simulation results for finding the probability of a path existing in a GHN. The solid line shows the predicted success rate as a function of the width, clique size, and node failure rate, while the markers represent the success rate over 1000 trials. Note that multiple markers in a column occur when when w, k, and q are the same, but d is different. Higher d values occur towards the bottom of the column. 64

5.4 Simulation results for the naïve algorithm. Markers represent the success rate over 1000 trials and the dotted line is the expected success rate from equation (4.2.1). Multiple dots occur when a trial was conducted with d, k, and q were the same, but w was different. Since the naïve algorithm is not impacted by w the observed variation is purely by chance. 65

5.5 Simulation results for the always-forwards algorithm. Each chart shows the probability of routing successfully as a function of q and k with the line representing the predicted lower bound on the probability of success and the dots representing the mean of 1000 trials. The columns of dots occur when numerous trials were performed with the same q and k, but different d; trials with a larger d value are found at the bottom of the column, while trials with low d values occur at the top. 66

5.6 Simulation results for the always-forwards algorithm. Each chart shows the probability of routing as a function of k for $d = 3$ and varying values of q. Each marker represents the success rate observed over 1000 trials. Multiple markers occur when their values of d, k and q are the same, but w is different. 67

5.7 Simulation results for the never-backwards algorithm. The solid line shows the predicted success rate as a function of the width, clique size, and node failure rate, while the markers represent the success rate over 1000 trials. Note that multiple markers in a column occur when when w, k, and q are the same, but d is different. Higher d values occur towards the bottom of the column. 68

5.8 Simulation results for the never-backwards algorithm as a function of k. Each chart shows the predicted success plotted as a function of k for a particular value of q. The markers represent the success rate over 1000 trials. Note that multiple markers in a column occur when k, and q are the same, but d is different. Higher d and w values occur towards the bottom of the column. The solid line represents the predicted value from equation (4.4.1) with $w = 2$ while the dashed line represents the predicted value for large w from equation (4.4.2). 69

List of Algorithms

1 Naïve Routing 29
2 Always-Forwards Routing 41
3 Never-Backwards Routing 48

Chapter 1

Introduction

A distributed hash table (DHT) is a type of peer-to-peer (P2P) network that, like traditional hash tables, maps *keys* to *values*. Unlike traditional hash tables, however, the data is distributed across a network with each node being responsible for a particular range of keys. Thus retrieving the value associated with a specific key involves determining which node is responsible for the key and requesting the data from that node.

For example, Chord, one of the first DHTs, assigns a random unique identifier (ID) to each node and arranges the nodes in a ring by increasing ID [18]. In its simplest form each node maintains a single pointer to the node with the next largest ID, called its *successor*. Each node is responsible for the values associated with the range of keys between its ID and its successor's ID. Retrieving the value associated with a particular key is then a matter of walking around the ring until the node responsible for the key is located.

Numerous other DHTs have been presented (e.g. [12, 13, 15, 5, 16, 11, 4]) and have become the cornerstone of wildly popular P2P file-sharing applications, such as BitTorrent. Each of these DHTs trades-off the number of pointers maintained per node with the

overhead and lookup time; storing more pointers decreases the lookup time at the expense of increased overhead. Many DHTs require nodes to maintain a logarithmic number of pointers (e.g. [18, 12, 13, 15]) and therefore also require a logarithmic number of messages to be sent in order to retrieve a file. Other DHTs (e.g. [5, 4]) have opted to decrease the lookup time to at the expense of increased overhead by increasing the number of pointers per node to $\mathcal{O}(\sqrt{n})$.

The key observation in this tradeoff is that increasing the number pointers per node increases the overhead at each node in terms of both memory consumption and bandwidth. The former occurs as a result of keeping the node's contact information (e.g. IP address and port), while the latter occurs as a result of ensuring the integrity of the pointer. Each node must periodically check that it is able to reach the nodes it points to and perform some sort of clean-up action if it is unable to do so. Thus as the number of pointers increases the overhead also increases in terms of bandwidth used.

The authors of Kelips argue that increasing the number of pointers to $\mathcal{O}(\sqrt{n})$ per node does not require an excessive amount of memory [5]. In fact, for a system with 10^5 nodes and 10^7 files only 1.93 MB of memory is used to store the routing information at each node [5]. Although this is small – especially given that current desktop computers have on the order of gigabytes of memory – they do not consider the increased overhead in terms of the bandwidth required to maintain the increased number of pointers.

Accordion, on the other hand, is a Chord-like DHT that adjusts the number of pointers each node stores based on its maximum available bandwidth – its *bandwidth budget* – while maintaining an expected $\mathcal{O}(\log n)$ lookup time [8]. Since it uses a ring structure like Chord, nodes do not require pointers to particular parts of the network because, in the worst case, each node passes a message to its successor. As a result of the relaxed

constraints on the routing table nodes in an Accordion network choose to store pointers probabilistically; a node N_i that learns of another node N_j at distance x is stored with probability proportional to $\frac{1}{x}$.

This thesis proposes a new DHT called a Grouped Hamming Network (GHN) where each node stores $\mathcal{P}(n)$ pointers, and $\mathcal{P}(n)$ is allowed to vary as a function of the number of nodes in the network, n. Storing the same number of pointers at each node, rather than varying it node by node, is beneficial in two different ways. First, it forces each node to be responsible for the same amount of work. In Accordion, it is possible for a node to request the lowest allowed bandwidth budget in order to minimize its own bandwidth, while simultaneously reducing lookup times by abusing its high-bandwidth neighbours. In a GHN every node is responsible for the same number of nodes and thus there aren't any nodes that can be exploited in this manner. Secondly, an equal number of pointers per node creates a regularly structured network that naturally lends itself to numerous possible routing schemes which, because of the straight-forward structure, yield strong theoretic results.

The remainder of this thesis is broken in to five chapters. Chapter 2 gives some background on the use of hypercubes in networks. Chapter 3 presents the GHN at a more detailed level and introduces the constraints on the problem. Chapter 4 introduces three different routing schemes and gives a theoretical analysis of their probability of routing success. Chapter 5 gives the results of simulations to support the claims made in chapter 4 and, finally, chapter 6 concludes the thesis.

Chapter 2

Background

A generalized w, d-hypercube is a hypercube of width w in d dimensions[1]. Although originally presented as a topology for VLSI communication networks in multi-processor environments [3], the same topology can easily be used to construct an overlay network where each node is a physical machine rather than a processor. In this situation, however, the biggest concerns are the bandwidth overhead associated with maintaining pointers to other nodes, and the non-trivial probability of nodes failing. This is radically different from the original use where the primary concern is wiring density as a high wiring density results in a more expensive product [3]. Furthermore, the probability of a processor failing is extremely small and, except in enterprise and safety-critical applications, results in nothing more than an inconvenience to the user.

Models of packet routing in hardware are generally quite different than those over wide area networks. A common model, for example, is one where neither edges nor nodes fail but only one message can be sent along a connection in any given time step [6]. The

[1]For the sake of clarity, this has been renamed from a k-ary n-cube as presented in [3]

problem, then, is to find the minimal number of time steps required to route n messages to n different nodes. In routing models over wide-area networks, however, more than one message can usually be sent along a connection, but edges, nodes, or both edges *and* nodes may fail with some non-trivial probability. Questions associated with this model can involve, for example, finding the probability of routing success for a particular routing algorithm, designing a routing algorithm with a minimal error rate, or determining the number of messages that need to be sent in order to guarantee routing success.

The network presented in this thesis is intended as a *service-based* network where there are multiple nodes providing identical services (e.g. hosting a file, providing CPU cycles) and clients do not care which node provides the service, as long as the service is received. This is similar to a DHT where clients are interested in receiving data, but generally not interested in who is providing it. Thus a message is said to be routed *successfully* through the network if it reaches any node that provides the desired service; otherwise the message is *unsuccessfully* routed.

Numerous DHTs have been presented (e.g. [12, 18, 13, 15, 5, 16, 11]) and have become the cornerstone of wildly popular P2P file-sharing applications, such as BitTorrent. Although these DHTs have been extended to improve throughput while reducing latency [2], improve security [20], improve performance under various levels of churn [22, 14] etc., and analyzed both theoretically [9] and empirically [17, 7], none have allowed for the number of pointers per node to vary as a system-wide function, nor have theoretical estimates on the probability of routing under random node failures been given. In general, different DHT implementations trade off between the number of pointers each node maintains and the reliability of the network.

The overlay network presented in this thesis allows for the number of pointers per node

to be any increasing function of n, $\mathcal{P}(n) \geq \lceil \lg n \rceil$. This flexibility is important as it easily allows for the network administrators to fine-tune the trade off between reliability and overhead. Consider, for example, a large private network constructed on a slow connection; naturally, low-overhead would be important. Now consider what would happen if the network were upgraded and the bandwidth was no longer an issue. If the overhead were not adjustable then the choices would be to either rewrite all of the software to use a different DHT, or deal with unnecessarily low reliability. Using a network that can fine tune this trade off, however, it would only be a matter of restarting the software with a modified $\mathcal{P}(n)$.

Chapter 3

System Overview

As previously described, a Grouped Hamming Network (GHN) is similar to a w, d-hypercube however each address in the cube maps to a *clique* of nodes, rather than a single node. A GHN is completely described by three parameters: k, the number of nodes in each clique, w, the width of the cube, and d, the number of dimensions; hence the size of the network is $n = kw^d$, and letting $k = 1$ and $w = 2$ gives a traditional binary hypercube. In practice n, the number of nodes, is given and the remaining parameters are positive integer values derived from n and $\mathcal{P}(n)$. Using a network of cliques significantly improves the reliability of the network as the probability of routing to a neighbouring clique increases with k. Additionally, a modest increase in k results in a large improvement in the probability of success of even the most simple routing schemes. The latter property allows for strong theoretical results because simpler algorithms still result in high probabilities of success.

A GHN is a *service-based* network which differs from a general network in that nodes are looking to receive a service and do not care who provides it. Specifically, each clique provides a service (e.g. host a file) and nodes within the same clique are indistinguishable

in terms of the quality of service provided. In a distributed file system, for example, a file could be mapped to a clique by hashing the file name to a clique ID. Another use – and, in fact, the original purpose of this network – is to select nodes uniformly and independently at random from the network in order to perform fair statistical sampling. Assuming that each clique contains an equal number of nodes, this is easily accomplished by selecting a clique at random and selecting a node from the clique at random. This thesis does not address the problem of assigning services to cliques and assumes that nodes are able to independently determine which clique is responsible for the desired service.

The model of the system presented assumes that nodes fail independently and uniformly at random with some probability $q = 1 - p$ at the time step before the message is sent and that the state of the network does not change while a message is being routed. Additionally, it is assumed that failed nodes have failed completely and that a node sending a message to a failed node is aware that its message was not delivered. This eliminates the possibility of faulty or adversarial nodes that are able to receive messages, but may route them incorrectly. Finally it is assumed that links do not fail; that is, if nodes N_i and N_j are alive and connected they are able to communicate across it.

The remainder of this thesis addresses three different routing strategies and the probability of a path existing in a GHN. For each routing scheme a theoretical estimate on the probability of failure is given and optimal configurations for d, w and k in terms of n and $\mathcal{P}(n)$ are given. Simulations of GHNs with various configurations indicate that the given estimates are indeed accurate for reasonable values of q and that the optimal configurations are accurate.

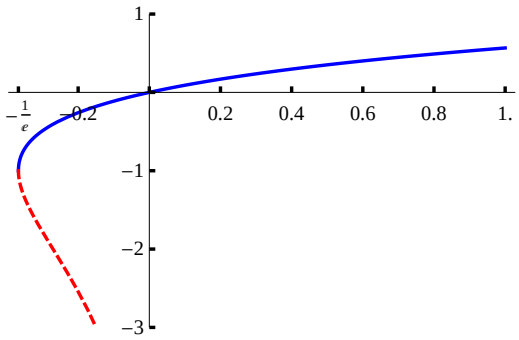


Figure 3.1: The Lambert W function. The solid blue line is the positive branch, $\mathcal{W}(x)$, and the dashed red line is the negative branch, $\mathcal{W}_{-1}(x)$.

3.1 Notation and Conventions

In order to prevent any misunderstandings this section will briefly note all conventions used in this thesis. The variables d, w, k, and n represent the number of dimensions, the width, the size of a clique, and the number of nodes in the network, respectively. The function $\mathcal{P}(n)$ represents the number of pointers stored per node. It assumed that n and $\mathcal{P}(n)$ are given. If $w = 2$ and $k = 1$ then the GHN is a traditional binary hypercube in d dimensions and is deonted Q_d.

The logarithm of x to the base b is denoted $\log_b x$ and the special cases of $b = e$ and $b = 2$ are denoted $\ln x$ and $\lg x$, respectively. The exponential function $y = e^x$ will, when more convenient, be represented by $y = \exp(x)$.

The Lambert W function of x is denoted by $\mathcal{W}(x)$ [1]. This function is used to solve

transcedental equations of the form

$$ye^y = x$$

$$y = \begin{cases} \mathcal{W}(x) & \text{if } x \geq 0, \\ \mathcal{W}(x) \text{ or } \mathcal{W}_{-1}(x) & \text{if } -\frac{1}{e} \leq x < 0. \end{cases}$$

The two branches $\mathcal{W}(x)$ and $\mathcal{W}_{-1}(x)$ on the interval $-\frac{1}{e} \leq x < 0$ denote the positive and negative branches, respectively (see figure 3.1). All applications of the Lambert W function will be explicit regarding which branches are valid. Note that if $-\frac{1}{e} \leq x < 0$ then $\mathcal{W}_{-1}(x) < y$ holds when $y > -1$. The derivative of the Lambert W function can be found by implicit differentiation and is given by the expression

$$\frac{\partial}{\partial x}\mathcal{W}(x) = \frac{\mathcal{W}(x)}{x(\mathcal{W}(x)+1)}.$$

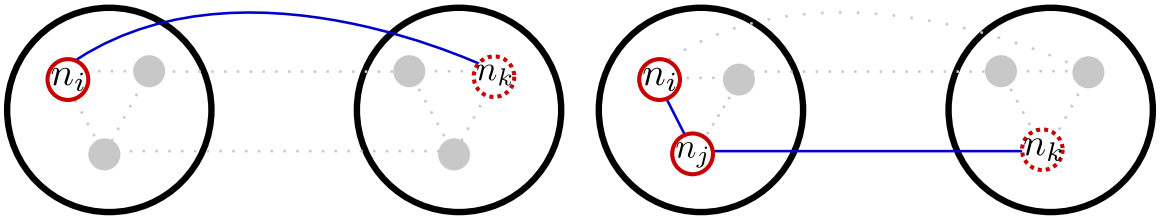
The probability of an event E occurring is given by $P(E)$ while the probability of event E *not* occurring is $Q(E)$. Naturally these probabilities are related by the expression $P(E) = 1 - Q(E)$.

The binomial coefficient is denoted

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

and represents the number of ways k elements can be selected from a set of n elements.

A node N_i that is a member of clique C_x is denoted using set notation as $N_i \in C_x$. If multiple nodes and cliques are introduced then, unless otherwise stated, it as assumed that they are distinct. For example if $N_i \in C_x$ and $N_j \in C_y$ then it is assumed that $i \neq j$



(a) Direct routing. (b) Indirect routing.

Figure 3.2: Two different ways of routing from N_i to N_k where N_i and N_k are in neighbouring cliques. The solid blue lines represent the path taken and the light dotted lines represent the other connections that are not used.

and $x \neq y$. The *cliquemates* of $N_i \in C_x$ refers to all nodes $\{N_j \mid N_j \in C_x, N_j \neq N_i\}$.

If a clique's ID must be given then it will be introduced as $C_{\langle x_1,x_2,\ldots,x_d\rangle}$, where $\langle x_1, x_2, \ldots, x_d\rangle$ is the d-component ID of the clique. The distance between two cliques $C_{\langle x_1,x_2,\ldots,x_d\rangle}$ and $C_{\langle y_1,y_2,\ldots,y_d\rangle}$ is calculated as the Hamming distance between their clique IDs and is denoted as $\mathcal{H}\left(C_{\langle x_1,x_2,\ldots,x_d\rangle}, C_{\langle y_1,y_2,\ldots,y_d\rangle}\right)$. The Hamming distance between two vectors is the number of components in which the vectors differ. For example $\mathcal{H}\left(C_{\langle 1,2,2,1\rangle}, C_{\langle 1,5,2,3\rangle}\right) = 2$. The *rowmates* of a clique C_x refers to all of the cliques $\{C_y \mid \mathcal{H}(C_x, C_y) = 1\}$. Geometrically, this is the set of cliques that are in the same row as C_x in some dimension.

A message is said to be routed *directly* from $N_i \in C_x$ to $N_k \in C_y$ if the message is sent across the connection from N_i to N_k. Figure 3.2(a) illustrates direct routing. Alternatively a message is said to be routed *indirectly* from $N_i \in C_x$ to $N_k \in C_y$ if it goes via some intermediary node $N_j \in C_x$. Figure 3.2(b) illustrates indirect routing.

A path through a GHN is identified by the cliques traversed, rather than the nodes the message passed through. For example if a message m_1 travels through nodes $N_i \in C_x$ and $N_j \in C_y$, and message m_2 travels through nodes $N_p \in C_x$ and $N_q \in C_y$ then messages m_1 and m_2 are said to have taken the same path, even though they did not pass through any

of the same nodes.

3.2 Network Layout

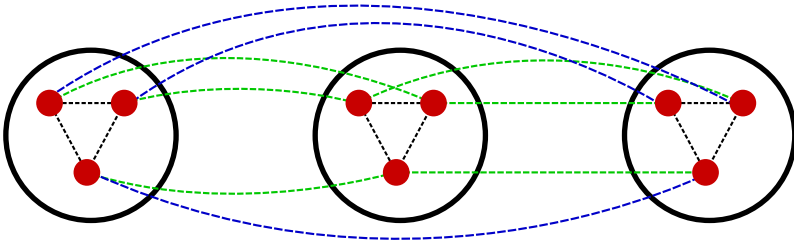


Figure 3.3: A row taken from a GHN with width 3. The filled red circles represent nodes in the network. Nodes in the same clique are surrounded by a black circle and intra-clique connections are shown as black dashed lines, while connections between adjacent cliques are shown in green and longer connections are shown in blue.

A GHN is a highly-connected network. Every node maintains a connection to each of its cliquemates as well as to one node in each clique in each of its rows. Figure 3.3 illustrates a row taken from a network of width 3. If this were taken from a GHN with d dimensions then each clique would be a member of d similar rows.

In a full GHN where there are exactly kw^d nodes, each cliquemate maintains a connection to a unique node in neighbouring cliques. For example if $N_i, N_j \in C_x$, $N_k \in C_y$, and C_x and C_y are in the same row, then only one of N_i or N_j will have a connection to N_k. Figure 3.4 shows a full GHN in two dimensions with width two and a clique size of three.

Handling Partially-Filled GHNs

A partially-filled GHN is when there are not exactly kw^d nodes and will occur often in practice. This results in fractional values for k, w, and d, which must be handled carefully

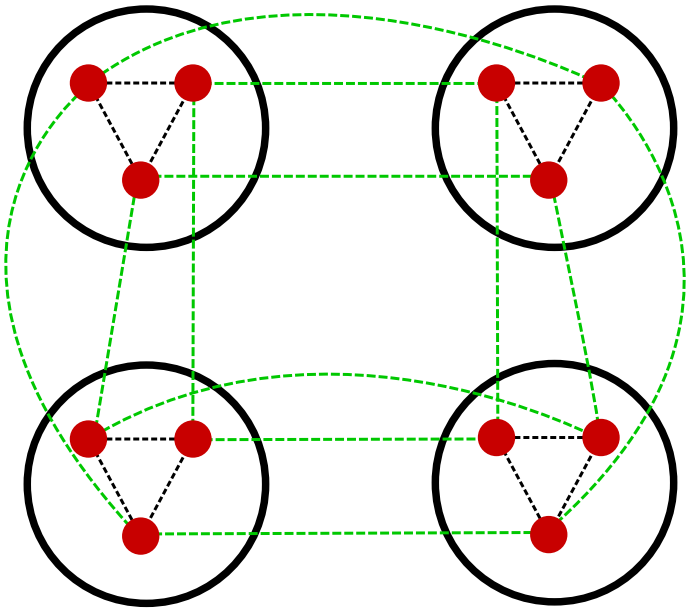


Figure 3.4: A full GHN in two dimensions with width 2 and clique size of 3. The filled red circles represent nodes in the network. Nodes in the same clique are surrounded by a black circle and intra-clique connections are shown as black dashed lines, while connections between nodes in adjacent cliques are shown in green.

in order to ensure a correct network structure. If w or d must be rounded then they will be rounded *down* as rounding up may result in cliques of size $k < 1$. If k is fractional then not all clique sizes will be the same; some are]emphsmall and contain $\lfloor k \rfloor$ nodes while others are *large* and contain $\lfloor k \rfloor + 1$ nodes.

If a large clique C_L is adjacent to a small clique C_S then there will be two nodes $N_i, N_j \in C_L$ that point to $N_k \in C_S$. This is required to ensure that each node is able to route to its neighbouring cliques both directly and indirectly.

3.3 Constraints

Since there are numerous parameters used to describe the GHN, this section is intended to collect all of the constraints on these variables in one place. The parameters are

k the number of nodes in each clique,

w the number of cliques in a row,

d the number of dimensions,

n the total number of nodes in the network,

$q = 1 - p$ the probability of a node failing, and

$\mathcal{P}(n)$ the number of pointers maintained per node.

These variables are related by the equations

$$n = kw^d, \tag{3.3.1}$$

and

$$\mathcal{P}(n) = (k-1) + (w-1)\,d \approx k + wd. \tag{3.3.2}$$

The approximate value of $\mathcal{P}(n)$ is used in much of this thesis as it significantly reduces the complexity of the solutions.

Constraints on these parameters are listed here for reference and proved afterwards.

$$1 \leq n \tag{3.3.3}$$

$$1 \leq \mathcal{P}(n) < 2\sqrt{n} \tag{3.3.4}$$

$$2 \leq w \leq n \tag{3.3.5}$$

$$1 \leq k \leq n \tag{3.3.6}$$

$$2 \leq d \leq \lg n. \tag{3.3.7}$$

Constraint 1. *The width of a grouped Hamming network is at least 2.*

Proof. Rearranging equation (3.3.1) gives

$$d = \frac{\ln n - \ln k}{\ln w}.$$

Since setting $w = 1$ would cause d to be undefined it must be the next largest integer; therefore w is at least 2. □

Constraint 2. *If $\mathcal{P}(n) \geq 2\sqrt{n}$ then the solution is trivial.*

Proof. If $\mathcal{P}(n) = 2\sqrt{n}$ then the network can be arranged such that $k = \sqrt{n}$ and $w = \sqrt{n}$. This solution is trivial and by forcing $\mathcal{P}(n)$ to be less than $2\sqrt{n}$ the solution becomes invalid. □

Since any solutions with $\mathcal{P}(n) \geq 2\sqrt{n}$ are trivial, only solutions with $\mathcal{P}(n) < 2\sqrt{n}$ are considered. It should be noted, however, that when $\mathcal{P}(n) = 2\sqrt{n}$ the network takes the same form as the distributed hash table Kelips [5].

Since equations (3.3.1) and (3.3.2) form two constraints on three variables, it is possible to express all of the parameters of the network in terms of the givens, n and $\mathcal{P}(n)$, as well as only *one* of the other parameters.

Lemma 3.1. *The parameters of a GHN can be expressed in terms of k, n, and $\mathcal{P}(n)$ as*

$$w = \frac{(1 + \mathcal{P}(n) - k)\,\mathcal{W}_{-1}\left(\frac{\left(\frac{n}{k}\right)^{\frac{1}{k-\mathcal{P}(n)-1}}}{k-\mathcal{P}(n)-1}\right)}{\ln k - \ln n}$$
$$d = \frac{\ln n - \ln k}{\ln w}.$$

Proof. Solving equation (3.3.1) for d gives

$$\begin{aligned} n &= kw^d \\ \frac{n}{k} &= w^d \\ \ln n - \ln k &= d \ln w \\ d &= \frac{\ln n - \ln k}{\ln w} \end{aligned}$$

and (3.3.2) for d yields

$$\begin{aligned} \mathcal{P}(n) &= (k-1) + (w-1)\,d \\ \mathcal{P}(n) - k + 1 &= (w-1)\,d \\ d &= \frac{\mathcal{P}(n) - k + 1}{w-1}. \end{aligned}$$

Equating these gives

$$\frac{\ln n - \ln k}{\ln w} = \frac{\mathcal{P}(n) - k + 1}{w - 1}$$
$$\frac{w-1}{\ln w} = \frac{\mathcal{P}(n) - k + 1}{\ln n - \ln k}.$$

Let

$$x = \frac{\mathcal{P}(n) - k + 1}{\ln n - \ln k}$$

then

$$\begin{aligned}
\frac{w-1}{\ln w} &= x \\
w - 1 &= x \ln w \\
e^w e^{-1} &= \left(e^{\ln w}\right)^x \\
e^w e^{-1} &= w^x \\
e^w &= e w^x \\
e^w w^{-x} &= e \\
e^{-\frac{w}{x}} w &= e^{-\frac{1}{x}} \\
-\frac{w}{x} e^{-\frac{w}{x}} &= -\frac{1}{x} e^{-\frac{1}{x}} \\
-\frac{w}{x} &= \mathcal{W}_{-1}\left(-\frac{1}{x} e^{-\frac{1}{x}}\right) \\
w &= -x\mathcal{W}_{-1}\left(-\frac{1}{x} e^{-\frac{1}{x}}\right).
\end{aligned}$$

Substituting in x and simplifying gives the closed form

$$w = \frac{(1 + \mathcal{P}(n) - k)\,\mathcal{W}_{-1}\left(\frac{\left(\frac{n}{k}\right)^{\frac{1}{k-\mathcal{P}(n)-1}}}{k-\mathcal{P}(n)-1}\right)}{\ln k - \ln n}.$$

□

Theorem 1. *Each node in a GHN stores $\mathcal{P}(n) \geq \lg n$ pointers.*

Proof. This problem amounts to minimizing $\mathcal{P}(n)$. Solving this directly through traditional calculs techniques proves rather difficult as the derivatives are complex; thus the argument is done logically.

Increasing k by one has the effect of increasing $\mathcal{P}(n)$ by 1 while increasing n by a factor of

$$\frac{(k+1)\,w^d}{kw^d} = \frac{k+1}{k}.$$

Since $\lim_{k\to\infty} \frac{k+1}{k} = 1$, the effect on n as k grows is negligible, while $\mathcal{P}(n)$ still increases by one.

Increasing w by one has the effect of increasing n by a factor of

$$\frac{k\,(w+1)^d}{kw^d} = \frac{(w+1)^d}{w^d}.$$

With d constant, $\lim_{w\to\infty} \frac{(w+1)^d}{w^d} = 1$ and increasing w by one also results in a negligible increase to n, while $\mathcal{P}(n)$ increases by one.

Finally, increasing d by one has the effect of increasing $\mathcal{P}(n)$ by a factor of

$$\frac{kw^{d+1}}{kw^d} = w.$$

Since $w \geq 2$, increasing d by one increases n by at least a factor of 2, while increasing $\mathcal{P}(n)$ by one.

Since k and w have have little effect on n and d has a large effect on n, $\mathcal{P}(n)$ must be minimized when w and k are minimized. Hence $\mathcal{P}(n)$ is minimzed when $k = 1$, $w = 2$, and $d = \lg n$ resulting in

$$\mathcal{P}(n) \geq (k-1) + (w-1)\, d$$
$$\mathcal{P}(n) \geq \lg n.$$

□

Approximation 1. *In a GHN with n nodes, d is minimized when*

$$k \approx -e\mathcal{W}_{-1}\left(-\frac{ne^{-\frac{\mathcal{P}(n)}{e}}}{e}\right).$$

Derivation. This derivation is simplified by using the approximate expression for $\mathcal{P}(n)$ from equation (3.3.2). Rearranging for d yields

$$\mathcal{P}(n) \approx k + wd$$
$$\mathcal{P}(n) - k \approx wd$$
$$d \approx \frac{\mathcal{P}(n) - k}{w}.$$

Similarly, from equation (3.3.1),

$$\begin{aligned} n &= kw^d \\ \frac{n}{k} &= w^d \\ \ln n - \ln k &= d \ln w \\ d &= \frac{\ln n - \ln k}{\ln w}. \end{aligned}$$

Equating gives

$$\begin{aligned} \frac{\ln n - \ln k}{\ln w} &\approx \frac{\mathcal{P}(n) - k}{w} \\ \frac{w}{\ln w} &\approx \frac{\mathcal{P}(n) - k}{\ln n - \ln k} \\ e^{\ln w} (\ln w)^{-1} &\approx \frac{\mathcal{P}(n) - k}{\ln n - \ln k} \\ e^{-\ln w} (\ln w) &\approx \frac{\ln n - \ln k}{\mathcal{P}(n) - k} \\ (-\ln w) e^{-\ln w} &\approx \frac{\ln k - \ln n}{\mathcal{P}(n) - k} \\ -\ln w &\approx \mathcal{W}_{-1}\left(\frac{\ln k - \ln n}{\mathcal{P}(n) - k}\right) \\ w &\approx e^{-\mathcal{W}_{-1}\left(\frac{\ln k - \ln n}{\mathcal{P}(n) - k}\right)}. \end{aligned}$$

Thus

$$\begin{aligned} d &= \frac{\ln n - \ln k}{\ln w} \\ &\approx \frac{\ln n - \ln k}{-\mathcal{W}_{-1}\left(\frac{\ln k - \ln n}{\mathcal{P}(n) - k}\right)} \\ &\approx \frac{\ln k - \ln n}{\mathcal{W}_{-1}\left(\frac{\ln k - \ln n}{\mathcal{P}(n) - k}\right)}. \end{aligned} \tag{3.3.8}$$

Let

$$z = \frac{\ln k - \ln n}{\mathcal{P}(n) - k}.$$

Then

$$d = -\frac{\ln n - \ln k}{\mathcal{W}_{-1}(z)}.$$

Differentiating with respect to k yields

$$\begin{aligned} \frac{\partial d}{\partial k} &= \frac{\partial}{\partial k}\left(-\frac{\ln n - \ln k}{\mathcal{W}_{-1}(z)}\right) \\ &= -\frac{\left(0 - \frac{1}{k}\right)\mathcal{W}_{-1}(z) - (\ln n - \ln k)\frac{\mathcal{W}_{-1}(z)}{z(\mathcal{W}_{-1}(z)+1)}\frac{\partial z}{\partial k}}{\mathcal{W}_{-1}(z)^2} \\ &= \frac{1}{k\mathcal{W}_{-1}(z)} + \frac{\ln n - \ln k}{z\mathcal{W}_{-1}(z)\left(\mathcal{W}_{-1}(z)+1\right)}\frac{\partial z}{\partial k}. \end{aligned}$$

The derivative of z with respect to k is

$$\begin{aligned} \frac{\partial z}{\partial k} &= \frac{\partial}{\partial k}\left(-\frac{\ln n - \ln k}{\mathcal{P}(n) - k}\right) \\ &= -\frac{-\frac{1}{k}\left(\mathcal{P}(n) - k\right) - (-1)(\ln n - \ln k)}{\left(\mathcal{P}(n) - k\right)^2} \\ &= -\frac{\mathcal{P}(n) + k\ln k - k - k\ln n}{k\left(\mathcal{P}(n) - k\right)^2}. \end{aligned}$$

Substituting in z and $\frac{\partial z}{\partial k}$ and simplifying gives

$$\frac{\partial d}{\partial k} = \frac{1 - e^{\mathcal{W}_{-1}\left(\frac{\ln n - \ln k}{k - \mathcal{P}(n)}\right)}}{k\left(\mathcal{W}_{-1}\left(\frac{\ln n - \ln k}{k - \mathcal{P}(n)}\right) + 1\right)}. \tag{3.3.9}$$

Critical points occur when $\frac{\partial d}{\partial k} = 0$ or $\frac{\partial d}{\partial k}$ is undefined. The former occurs when $n = k$, while the latter occurs when $k = 0$, or

$$\begin{aligned}
\mathcal{W}_{-1}\left(\frac{\ln n - \ln k}{k - \mathcal{P}(n)}\right) + 1 &= 0 \\
\mathcal{W}_{-1}\left(\frac{\ln n - \ln k}{k - \mathcal{P}(n)}\right) &= -1 \\
\frac{\ln n - \ln k}{k - \mathcal{P}(n)} &= (-1)\, e^{-1} \\
\frac{\ln n - \ln k}{k - \mathcal{P}(n)} &= -\frac{1}{e} \\
e \ln n - e \ln k &= \mathcal{P}(n) - k \\
k - e \ln k &= \mathcal{P}(n) - e \ln n \\
e^{k} e^{-e \ln k} &= e^{\mathcal{P}(n)} e^{-e \ln n} \\
e^{k} k^{-e} &= e^{\mathcal{P}(n)} n^{-e} \\
k e^{-\frac{k}{e}} &= n e^{-\frac{\mathcal{P}(n)}{e}} \\
\left(-\frac{k}{e}\right) e^{-\frac{k}{e}} &= -\frac{n e^{-\frac{\mathcal{P}(n)}{e}}}{e} \\
-\frac{k}{e} &= \mathcal{W}_{-1}\left(-\frac{n e^{-\frac{\mathcal{P}(n)}{e}}}{e}\right) \\
k &= -e\mathcal{W}_{-1}\left(-\frac{n e^{-\frac{\mathcal{P}(n)}{e}}}{e}\right).
\end{aligned}$$

This is valid when

$$\begin{aligned} k &\geq 1 \\ -e\mathcal{W}_{-1}\left(-\frac{ne^{-\frac{\mathcal{P}(n)}{e}}}{e}\right) &\geq 1 \\ \mathcal{W}_{-1}\left(-\frac{ne^{-\frac{\mathcal{P}(n)}{e}}}{e}\right) &\leq -\frac{1}{e}. \end{aligned}$$

The inequality holds when

$$-\frac{1}{e} \leq -\frac{ne^{-\frac{\mathcal{P}(n)}{e}}}{e} < 0.$$

Since the upper bound holds trivially, the expression is valid when

$$\begin{aligned} -\frac{ne^{-\frac{\mathcal{P}(n)}{e}}}{e} &\geq -\frac{1}{e} \\ e^{-\frac{\mathcal{P}(n)}{e}} &\leq \frac{1}{n} \\ -\frac{\mathcal{P}(n)}{e} &\leq -\ln n \\ \mathcal{P}(n) &\geq e\ln n. \end{aligned}$$

The first derivative test is used to determine if the critical point is a maximum or minimum. Equation (3.3.9) is negative when the numerator and denominator have different signs. Since the exponent in the numerator is between 0 and $-\frac{1}{e}$, the entire numerator must always be positive. Thus the expression is negative if and only if

$$\mathcal{W}_{-1}\left(\frac{\ln n - \ln k}{k - \mathcal{P}(n)}\right) < 0.$$

Following the derivation in theorem (1) yields

$$k < -e\mathcal{W}_{-1}\left(-\frac{ne^{-\frac{\mathcal{P}(n)}{e}}}{e}\right).$$

Therefore the critical point is a minimum because the first derivative changes from negative to positive through the point and, since there are no other valid critical points, this must be a global minimum. □

Chapter 4

Routing

All overlay networks have some non-zero probability of failure that can be a result of, among other things, hardware failure, network congestion, or malicious users. This section addresses the probability of failure of three different routing schemes in a full GHN where $n = kw^d$ and $n, k, w, d \in \mathbb{Z}$. All of the routing schemes operate by routing through cliques, rather than through specific nodes, and attempt to make forward progress at each iteration. If forward progress is not possible then the message is dropped.

A message is said to be routed successfully if it reaches any node in the target clique. Formally, if node N_i sends a message m to a clique C then it is said to be routed successfully if and only if a node $N_j \in C$ receives m. This section assumes that the node sending the message is alive, but makes no assumptions about any other nodes in the network. Thus with probability q^k all k nodes in the target clique may have failed and, regardless of the routing scheme used, the message will not be routed successfully.

Only the longest path in the GHN is considered when calculating the probability of success. The longest possible path occurs when the source ID differs from the destination

ID in every component. Specifically the longest path is one from a node $N_i \in C_x$ to $N_j \in C_y$ where $\mathcal{H}(C_x, C_y) = d$.

4.1 The Probability of a Path Existing

The probability of any path existing between $N_i \in C_x$ and C_y is addressed before considering the probability of success of a particular algorithm. Much theoretical work has been done with respect to Q_d, the special case where $w = 2$ and $k = 1$. As a result, this section discusses the connectivity results of Q_d and suggests that, since each node in a GHN has at least as many pointers as Q_d, a GHN has *at least* the same probability of a path existing.

Ideally it would be possible to determine the exact probability of a path existing between N_i and N_j in Q_d, however the lack of research addressing this problem directly suggests it is rather difficult. Indeed, section 4.2 confirms that the expression for the exact probability of failure for even the simplest of routing algorithms is unwieldly. Thus the problem is addressed in two different manners: finding the largest connected component after a set of f nodes have failed from Q_d, and finding the number of paths between N_i and N_j given that f nodes have failed. Although neither address the problem directly, they do provide a strong indication of the resilancy of hypercubes.

In the first case, if a set of f, $|f| \leq 2d - 3$, nodes fail then the largest remaining component contains at least $2^d - |f| - 1$ nodes [23]. This leads to the following theorem.

Theorem 2. *If N_i is sending a message to N_j in Q_d and $2 \lg n - 3$ nodes fail then a path from N_i to N_j exists with probability*

$$1 - \frac{2 \lg n - 3}{n - 1}.$$

Proof. Since the the initial node N_i is not allowed to fail, there are therefore

$$\binom{2^d - 1}{2d - 3}$$

ways that the f nodes could fail. Of these,

$$\binom{2^d - 2}{2d - 4}$$

involve the destination node N_j failing and are therefore unrouteable. Thus, assuming that each configuration of node failures occurs with equal probability, the probability of being unable to route to the target is

$$\begin{aligned}
&\binom{2^d - 2}{2d - 4} / \binom{2^d - 1}{2d - 3} \\
&= \frac{\left(2^d - 2\right)!}{(2d-4)!\,(2^d - 2 - 2d + 4)!} \cdot \frac{(2d-3)!\,\left(2^d - 1 - 2d + 3\right)!}{(2^d - 1)!} \\
&= \frac{\left(2^d - 2\right)!}{(2^d - 1)!} \cdot \frac{(2d-3)!}{(2d-4)!} \cdot \frac{\left(2^d - 2d + 2\right)!}{(2^d - 2d + 2)!} \\
&= \frac{2d-3}{2^d - 1},
\end{aligned}$$

but since this is in Q_d where $d = \lg n$,

$$\frac{2d-3}{2^d - 1} = \frac{2\lg n - 3}{n - 1}.$$

Thus as n gets large up to $2\lg n - 3$ nodes can fail and a message can be sent succesfully with probability

$$1 - \frac{2\lg n - 3}{n - 1}.$$

□

In the second case the number of fault-free paths between N_i and N_j in Q_d are counted. If $f \leq d-2$ nodes fail in Q_d then a fault-free path of length l exists for each l satisfying $\mathcal{H}(N_i, N_j) + 2 \leq l \leq 2^d - 2f - 1$, where l and $\mathcal{H}(N_i, N_j)$ have the same parity [10].

Theorem 3. *If $d = \lg n$ nodes fail in Q_d then there are at least $\frac{n}{2} - \lg n - 2$ paths between any two remaining nodes.*

Proof. In the worst case $\mathcal{H}(N_i, N_j) = d$. Thus there is a path of length l for each l that satisfies $d + 2 \leq l \leq 2^d - 2f - 1$, where l and d have the same parity. Since d nodes have failed, $f = d$ and $d + 2 \leq l \leq 2^d - 2d - 1$.

Without loss of generality, assume that d is even. Therefore $2^d - 2d - 1$ is odd and since d and l must have the same parity, $d + 2 \leq l \leq 2^d - 2d - 2$. Thus there are

$$\begin{aligned} &\frac{2^d - 2d - 2 - (d+2)}{2} \\ &= \frac{2^d - 2d - 4}{2} \\ &= 2^{d-1} - d - 2 \end{aligned}$$

paths. Since $d = \lg n$, there are therefore at least $\frac{n}{2} - \lg n - 2$ paths between any two remaining nodes. □

4.2 The Naïve Algorithm

The naïve algorithm is the simplest possible routing scheme in any Hamming-like network. Messages are routed by matching the components of the destination ID in consecutive order and if the message cannot be routed to the desired clique then it is dropped. Thus

at any step in the routing protocol the sending node has exactly one clique to send the message to.

Algorithm 1 provides pseudocode for the naïve algorithm. Lines 1 to 3 check if the message is for the current node and consumes the message if it is. Otherwise lines 5 to 11 find the first component that differs from its own ID, b. The ID of the next hop is found by taking the current node's ID and switching component b to match the destination ID.

Algorithm 1 Naïve Routing

```
 1: if message.destination = self.address then
 2:    consume_message
 3:    return
 4: else
 5:    for b = 0 to message.destination.length do
 6:       if message.destination[b] ≠ self.address[b] then
 7:          next_hop = message.destination[0 : b + 1] + self.address[b + 1 :]
 8:          message.forward_to(next_hop)
 9:          return
10:       end if
11:    end for
12: end if
```

4.2.1 Calculating the Probability of Failure

Two different methods of calculating the probability of failure are given in this section. First the exact probability is derived, resulting in a complex summation. Second, a close upper bound is found. Although it overestimates the probability of failure, chapter 5 shows that it is close enough to use as an approximation.

The Exact Probability of Failure

A path through three cliques is first considered then it is extrapolated into a path of length d. Suppose a message is being routed from $N_i \in C_x$ to C_z via clique C_y. Additionally, assume that of the k nodes in each clique, i_0 are alive in C_x, i_1 are alive in C_y, and i_2 are alive in C_z. The probability of this configuration occuring is

$$p^{i_0+i_1+i_2-1}\,(1-p)^{3k-i_0-i_1-i_2}\,.$$

Note that the one is subtracted in the first exponent because N_i, the node sending the message, is guaranteed to be alive. Next the number of ways these nodes can be arranged such that the message can be sent successfully sent is calculated.

Since N_i is guaranteed to be alive, there are $i_0 - 1$ live nodes that can be placed in $k-1$ possible locations; this can be done in

$$\binom{k-1}{i_0-1}$$

ways. In order for the message to be routed successfully, at least one of the i_1 alive nodes in C_y must be adjacent to a live node in C_x.

Suppose $i_1 \geq j \geq 1$ nodes in C_y are adjacent to live nodes in C_x. The j live nodes in C_y that are adjacent to live nodes in C_x can be arranged in $\binom{i_0}{j}$ different ways. This leaves $i_1 - j$ live nodes in C_y that are adjacent to the $k - i_0$ failed nodes in C_x; these can be placed in $\binom{k-i_0}{i_1-j}$ ways. Thus the total number of ways to place the i_1 live nodes in C_y

such that at least one is adjacent to a live node in C_x is given by the expression

$$\sum_{j=1}^{i_1} \binom{i_0}{j} \binom{k-i_0}{i_1-j}.$$

Following the same derivation, the number of ways that the i_2 live nodes in C_z can be arranged such that at least one is adjacent to one of the i_1 live nodes in C_y is

$$\sum_{j=1}^{i_2} \binom{i_1}{j} \binom{k-i_1}{i_2-j}.$$

Thus, given i_0 nodes alive in C_x, i_1 nodes alive in C_y, and i_2 nodes alive in C_z, there are

$$\binom{k-1}{i_0-1} \sum_{j=1}^{i_1} \binom{i_0}{j} \binom{k-i_0}{i_1-j} \sum_{j=1}^{i_2} \binom{i_1}{j} \binom{k-i_1}{i_2-j}$$

ways to arrange them such that a message can be sent from $N_i \in C_x$ to C_z.

Multiplying this by the probability of this configuration occuring and summing over all i_0, i_1, and i_2 gives the total probability of a path existing. Hence

$$P\left(\text{path}|d=2\right) =$$
$$\sum_{i_0}^{k} \sum_{i_1}^{k} \sum_{i_2}^{k} p^{i_0+i_1+i_2-1} \left(1-p\right)^{3k-i_0-i_1-i_2} \binom{k-1}{i_0-1} \sum_{j=1}^{i_1} \binom{i_0}{j} \binom{k-i_0}{i_1-j} \sum_{j=1}^{i_2} \binom{i_1}{j} \binom{k-i_1}{i_2-j}.$$

Generalizing to a path of length d through $d+1$ cliques gives

$$\sum_{i_0,i_1,\ldots,i_d=1}^{k} p^{\sum_{j=0}^{d} i_j - 1} \left(1-p\right)^{(d+1)k-\sum_{j=0}^{d} i_j} \binom{k-1}{i_0-1} \prod_{j=1}^{d} \sum_{m=1}^{i_j} \binom{i_{j-1}}{m} \binom{k-i_{j-1}}{i_j-m}. \tag{4.2.1}$$

Approximating the Probability of Failure

This section derives the probability of routing successfully using the naïve algorithm by first considering the probability of routing directly or indirectly to a neighbouring clique, then using these probabilities to determine the probability of routing along a path.

The probability that $N_i \in C_x$ is not able to send a message directly to $N_k \in C_y$ is the probability that N_k has failed. Hence

$$Q\left(\text{directly}\right) = q. \tag{4.2.2}$$

Next the probability that a node $N_i \in C_x$ is unable to route to a node $N_k \in C_y$ indirectly via a clique-mate $N_j \in C_x$ is calculated. This is equal to the probability that either N_j or N_k has failed. Thus

$$\begin{aligned} Q\left(\text{via clique mate}\right) &= q + q - q^2 \\ &= q\left(2 - q\right). \end{aligned}$$

The probability that a node $N_i \in C_x$ is not able to route indirectly to any node in C_y via *any* cliquemate in C_x is the probability that all $k-1$ cliquemates are unable to route the message. Hence

$$\begin{aligned} Q\left(\text{indirectly}\right) &= Q\left(\text{via clique-mate}\right)^{k-1} \\ &= \left(q\left(2 - q\right)\right)^{k-1}. \end{aligned} \tag{4.2.3}$$

The probability that a node $N_i \in C_x$ is unable to route to any node in a neighbouring

clique C_y is the probability that it is unable to do so either directly or indirectly. Hence, from equations (4.2.2) and (4.2.3),

$$\begin{aligned} Q\left(\text{neighbouring clique}\right) &= Q\left(\text{directly}\right) Q\left(\text{indirectly}\right) \\ &= q\left(q\left(2-q\right)\right)^{k-1}. \end{aligned} \tag{4.2.4}$$

The longest path in the naïve routing algorithm is equal to the number of dimensions. Assuming that the probability of at each step is independent, then the probability of being able to route along a path is the probability that the message is able to be passed forward at each step. Thus

$$\begin{aligned} P\left(\text{forward progress}\right) &= 1 - Q\left(\text{neighbouring clique}\right) \\ &= 1 - q\left(q\left(2-q\right)\right)^{k-1}, \end{aligned}$$

and the probability of being able to route through the entire path is that forward progress can be made at each of d steps. Hence

$$\begin{aligned} P\left(\text{path}\right) &= P\left(\text{forward progress}\right)^{d} \\ &= \left(1 - q\left(q\left(2-q\right)\right)^{k-1}\right)^{d}. \end{aligned} \tag{4.2.5}$$

The independence assumption made results in equation (4.2.5) being a slight overestimate of the actual probability of success. Chapter 5 shows that the overestimate is small in practice and still makes for a good approximation of the actual value. The following example illustrates the overestimate for when $k = 2$, $d = 2$, and $q = p = 0.5$.

Figure 4.1 illustrates a path of length two with $k = 2$. The nodes in this figure are

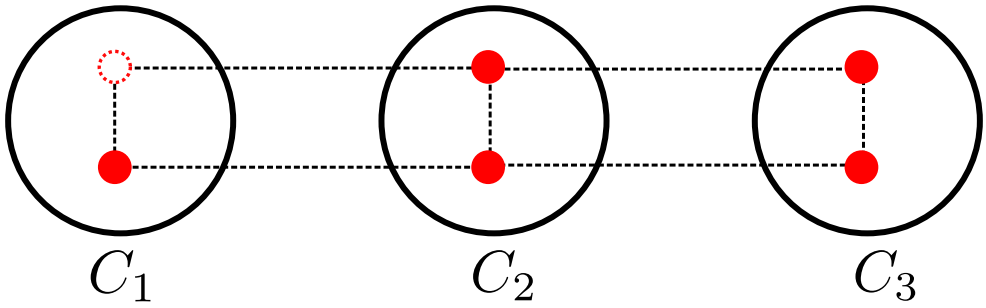


Figure 4.1: A sample path through a GHN with $k = 2$ and $d = 2$ where the message originates at the dashed circle in C_1 and is destined for C_3.

referred to by their clique name and whether they are at the top of the clique or at the bottom. For example, the dashed circle at the top of C_1 is referred to as $C_{1,T}$. Suppose a message is being routed from $C_{1,T}$ to C_3. Since it is assumed that the starting node is alive, there are $2^5 = 32$ possible ways the remaining nodes could fail. Each of these combinations and whether it is possible to route through it is shown in figure 4.2; A or D indicates whether the node is alive or dead, respectively, and the *Route?* column indicates whether it is possible to route through this configuration. If it is assumed that $q = p = 0.5$ then each configuration occurs with probability $0.5^5 = 0.03125$. Since there are 12 configurations that route successfully, the total probability of routing successfully is $12 \cdot 0.03125 = 0.375$.

The predicted probability from equation (4.2.5) is

$$\begin{aligned}
&\left(1 - q\left(q\left(2 - q\right)\right)^{k-1}\right)^d \\
&\left(1 - 0.5\left(0.5\left(2 - 0.5\right)\right)^{2-1}\right)^2 \\
&= \frac{25}{64} = 0.390625,
\end{aligned}$$

resulting in an error of $0.390625 - 0.375 = 0.015625$.

$C_{1,T}$	$C_{1,B}$	$C_{2,T}$	$C_{2,B}$	$C_{3,T}$	$C_{3,B}$	Route?
A	A	A	A	A	A	Y
A	A	A	A	A	D	Y
A	A	A	A	D	A	Y
A	A	A	A	D	D	N
A	A	A	D	A	A	Y
A	A	A	D	A	D	Y
A	A	A	D	D	A	N
A	A	A	D	D	D	N
A	A	D	A	A	A	Y
A	A	D	A	A	D	N
A	A	D	A	D	A	Y
A	A	D	A	D	D	N
A	A	D	D	A	A	N
A	A	D	D	A	D	N
A	A	D	D	D	A	N
A	A	D	D	D	D	N
A	D	A	A	A	A	Y
A	D	A	A	A	D	Y
A	D	A	A	D	A	Y
A	D	A	A	D	D	N
A	D	A	D	A	A	Y
A	D	A	D	A	D	Y
A	D	A	D	D	A	N
A	D	A	D	D	D	N
A	D	D	A	A	A	N
A	D	D	A	A	D	N
A	D	D	A	D	A	N
A	D	D	A	D	D	N
A	D	D	D	A	A	N
A	D	D	D	A	D	N
A	D	D	D	D	A	N
A	D	D	D	D	D	N
						Total: 12

Figure 4.2: A summary of all configurations of node failures in a path of length three with $k = 2$, where A means the node is alive and D means the node is dead. The *Route?* column indicates whether it is possible to route a message through this configuration; Y indicates that it is possible to route a message, while N indicates it is not.

4.2.2 Minimizing the Probability of Failure

Now that the probability of failure has been calculated, the natural question to ask is how to minimize this rate of failure. Specifically, given $\mathcal{P}(n)$ and n, what configuration of w, k and d maximizes equation (4.2.5)? This question is answered in the following theorem.

Approximation 2. *Given $\mathcal{P}(n)$ and n, equation (4.2.5) is maximized when*

$$\begin{aligned} k &\approx -\mathcal{W}_{-1}\left(-ne^{-\mathcal{P}(n)}\right) \\ d &\approx \frac{\ln n - \ln k}{\ln\left(\frac{\mathcal{P}(n)-k}{\ln n - \ln k}\right)} \\ w &\approx e^{-\mathcal{W}_{-1}\left(\frac{\ln k - \ln n}{\mathcal{P}(n)-k}\right)}. \end{aligned}$$

Derivation. Differentiating with respect to k gives

$$\begin{aligned} \frac{\partial}{\partial k} P(\text{path}) &= -\left(1-q\alpha^{k-1}\right)^d \frac{\partial}{\partial k}\left(d\ln\left(1-q\alpha^{k-1}\right)\right) \\ &= -\left(1-q\alpha^{k-1}\right)^d \left(\frac{\partial d}{\partial k}\ln\left(1-q\alpha^{k-1}\right) - \frac{dq\alpha^{k-1}\ln\alpha}{1-q\alpha^{k-1}}\right) \\ &= \left(1-q\alpha^{k-1}\right)^d \left(\frac{dq\alpha^{k-1}\ln\alpha}{1-q\alpha^{k-1}} - \ln\left(1-q\alpha^{k-1}\right)\frac{\partial d}{\partial k}\right). \end{aligned}$$

Since $0 < q < 1$, $0 < \alpha < 1$ and k is a function of n the term $\ln\left(1-q\alpha^{k-1}\right)$ will approach $\ln 1$ as n grows. This can be approximated by taking the first term of the Taylor series around the point $q\alpha^{k-1} = 0$ to get

$$\ln\left(1-q\alpha^{k-1}\right) \approx -q\alpha^{k-1}.$$

Thus

$$\begin{aligned}\frac{\partial}{\partial k}P\left(\text{path}\right) &\approx \left(1-q\alpha^{k-1}\right)^{d}\left(\frac{dq\alpha^{k-1}\ln\alpha}{1-q\alpha^{k-1}}-\left(-q\alpha^{k-1}\right)\frac{\partial d}{\partial k}\right)\\ &\approx q\alpha^{k-1}\left(1-q\alpha^{k-1}\right)^{d}\left(\frac{d\ln\alpha}{1-q\alpha^{k-1}}+\frac{\partial d}{\partial k}\right). \qquad (4.2.6)\end{aligned}$$

The derivative of d with respect to k is found by solving equations (3.3.1) and (3.3.2) for w, equating, differentiating with respect to k and solving for $\frac{\partial d}{\partial k}$. Thus

$$\begin{aligned}\frac{\mathcal{P}\left(n\right)-k}{d} &= \left(\frac{n}{k}\right)^{\frac{1}{d}}\\ \frac{\partial}{\partial k}\frac{\mathcal{P}\left(n\right)-k}{d} &= \frac{\partial}{\partial k}\left(\frac{n}{k}\right)^{\frac{1}{d}}\\ \frac{-d-\left(\mathcal{P}\left(n\right)-k\right)\frac{\partial d}{\partial k}}{d^{2}} &= \left(\frac{n}{k}\right)^{\frac{1}{d}}\left(-\frac{\ln\frac{n}{k}}{d^{2}}\frac{\partial d}{\partial k}-\frac{1}{dk}\right)\\ \frac{\partial d}{\partial k} &= \frac{d\left(k-\left(\frac{n}{k}\right)^{\frac{1}{d}}\right)}{k\left(\left(\frac{n}{k}\right)^{\frac{1}{d}}\ln\frac{n}{k}+k-\mathcal{P}\left(n\right)\right)}.\end{aligned}$$

Substituting this back into equation (4.2.6) and simplifying gives

$$\frac{\partial}{\partial k}P\left(\text{path}\right)\approx dq\alpha^{k-1}\left(1-q\alpha^{k-1}\right)\left(\frac{k-\left(\frac{n}{k}\right)^{\frac{1}{d}}}{k\left(\frac{n}{k}\right)^{\frac{1}{d}}\ln\frac{n}{k}+k^{2}-k\mathcal{P}\left(n\right)}+\frac{\ln\alpha}{1-q\alpha^{k-1}}\right).$$

This has critical points at

$$\begin{aligned} d &= 0, \\ q &= 0, \\ \alpha &= 0, \\ 1 - q\alpha^k &= 0, \text{and} \\ \frac{k - \left(\frac{n}{k}\right)^{\frac{1}{d}}}{k\left(\frac{n}{k}\right)^{\frac{1}{d}} \ln \frac{n}{k} + k^2 - k\mathcal{P}(n)} + \frac{\ln \alpha}{1 - q\alpha^{k-1}} &= 0. \end{aligned}$$

Of these critical points, only the last is valid. Solving for d gives

$$d = \frac{\ln n - \ln k}{\ln\left(\frac{k\left((\mathcal{P}(n)-k)\ln\alpha + q\alpha^{k-1} - 1\right)}{k\ln\alpha\ln\frac{n}{k} + q\alpha^{k-1} - 1}\right)}.$$

Since the additive terms in the denominator are small in comparison to the multiplicative and they are found within a logarithm, it is fair to approximate this term by dropping the additive terms. Hence

$$\begin{aligned} \frac{k\left((\mathcal{P}(n) - k)\ln\alpha + q\alpha^{k-1} - 1\right)}{k\ln\alpha\ln\frac{n}{k} + q\alpha^{k-1} - 1} &\approx \frac{k\left((\mathcal{P}(n) - k)\ln\alpha\right)}{k\ln\alpha\ln\frac{n}{k}} \\ &\approx \frac{\mathcal{P}(n) - k}{\ln n - \ln k}, \end{aligned}$$

and therefore

$$d \approx \frac{\ln n - \ln k}{\ln\left(\frac{\mathcal{P}(n)-k}{\ln n - \ln k}\right)}.$$

From equation (3.3.8)

$$\begin{aligned}
d &\approx -\frac{\ln n - \ln k}{\mathcal{W}_{-1}\left(-\frac{\ln n - \ln k}{\mathcal{P}(n)-k}\right)} \\
\frac{\ln n - \ln k}{\ln\left(\frac{\mathcal{P}(n)-k}{\ln n - \ln k}\right)} &\approx -\frac{\ln n - \ln k}{\mathcal{W}_{-1}\left(-\frac{\ln n - \ln k}{\mathcal{P}(n)-k}\right)} \\
\mathcal{W}_{-1}\left(-\frac{\ln n - \ln k}{\mathcal{P}(n)-k}\right) &\approx -\ln\left(\frac{\mathcal{P}(n)-k}{\ln n - \ln k}\right) \\
-\frac{\ln n - \ln k}{\mathcal{P}(n)-k} &\approx -\ln\left(\frac{\mathcal{P}(n)-k}{\ln n - \ln k}\right) e^{-\ln\left(\frac{\mathcal{P}(n)-k}{\ln n - \ln k}\right)} \\
\frac{\ln n - \ln k}{\mathcal{P}(n)-k} &\approx \ln\left(\frac{\mathcal{P}(n)-k}{\ln n - \ln k}\right) \frac{\ln n - \ln k}{\mathcal{P}(n)-k} \\
1 &\approx \ln\left(\frac{\mathcal{P}(n)-k}{\ln n - \ln k}\right) \\
k &\approx -\mathcal{W}_{-1}\left(-ne^{-\mathcal{P}(n)}\right).
\end{aligned}$$

This expression for k is larger then the maximum k from lemma 1 and is therefore invalid. Since there is no critical point between

$$-e\mathcal{W}_{-1}\left(-\frac{ne^{-\frac{\mathcal{P}(n)}{e}}}{e}\right)$$

from lemma 1 and the solution

$$-\mathcal{W}_{-1}\left(-ne^{-\mathcal{P}(n)}\right)$$

the function must therefore be increasing through the endpoint. Thus the maximum is

found at the endpoint, giving the solution

$$
\begin{aligned}
k &\approx -\mathcal{W}_{-1}\left(-ne^{-\mathcal{P}(n)}\right) \\
d &\approx \frac{\ln n - \ln k}{\ln\left(\frac{\mathcal{P}(n)-k}{\ln n - \ln k}\right)} \\
w &\approx e^{-\mathcal{W}_{-1}\left(\frac{\ln k - \ln n}{\mathcal{P}(n)-k}\right)}.
\end{aligned}
$$

□

4.3 The Always-Forwards Algorithm

The previous section described the naïve algorithm in which messages are routed by matching the components of the destination ID in a specific order. The *always-forwards* algorithm is similar in that messages are passed by matching components of the destination ID, however the node passing the message can choose to send it to any clique that matches the destination ID in one more component. Thus, if a message being sent to C_y is at some node $N_i \in C_x$ then N_i can choose to send it to any of its $\mathcal{H}(C_x, C_y)$ neighbours that match the destination ID in an additional component.

Algorithm 2 describes the always-forwards algorithm. Lines 1 to 3 check if the message is intended for the current node and consume the message if it is. Line 4 initializes a flag that indicates the message has not yet been sent to its destination. Lines 5 to 10 try forwarding the message to each clique that is closer to the destination than the current node. Lines 6 to 9 determine the next hop address by first checking if the destination address differs from the current address at position i. If so, then the next hop address is calculated by taking the current address and changing component i to match the destination address.

Algorithm 2 Always-Forwards Routing

```
1: if message.destination = self.address then
2:    consume_message
3: end if
4: message_sent = False
5: for i = 0 to message.destination.length ∧¬message_sent ∧  do
6:    if message.destination[i] ≠ self.address[i] then
7:       next_hop = self.address[0:i] + message.destination[i] + self.address[i + 1 :]
8:       message_sent = message.forward_to(next_hop)
9:    end if
10: end for
```

4.3.1 Calculating the Probability of Success

This section calculates the probability of a message being routed successfully. The probability of success, as opposed to the probability of failure, is used because it significantly simplifies the calculations.

The probability that a node $N_i \in C_x$ is able to route directly to a neighbouring node $N_k \in C_y$ is the probability that N_k is alive. Thus

$$P\,(\text{directly}) = p.$$

The probability that a node $N_i \in C_x$ is able to route indirectly to any node $N_k \in C_y$ via a specific cliquemate $N_j \in C_x$ is the probability that both N_j and N_k are alive. Hence

$$P\,(\text{indirectly}) = p^2.$$

The probability that a node $N_i \in C_x$ is able to route to any $N_j \in C_y$ via any of its $k-1$

cliquemates is the probability that not all of them are unable to route the message. Hence

$$\begin{aligned} P\left(\text{any}\right) &= 1 - \left(1 - P\left(\text{indirectly}\right)\right)^{k-1} \\ &= 1 - \left(1 - p^2\right)^{k-1}. \end{aligned}$$

Thus the probability that a message is routed from $N_i \in C_x$ to any $N_j \in C_y$ either directly or via a cliquemate is

$$\begin{aligned} P\left(\text{specific neighbour}\right) &= p + P\left(\text{any}\right) - p \cdot P\left(\text{any}\right) \\ &= 1 - \left(1 - p\right)^k \left(p + 1\right)^{k-1}. \end{aligned}$$

Suppose a message being sent to C_y is at node $N_i \in C_x$ such that $\mathcal{H}\left(C_x, C_y\right) = h$; node N_i therefore has h possible cliques to forward the message to. The probability that N_i is able to forward the message to at least one of h cliques is the probability that it is not the case that it can't forward the message to any of them. Thus

$$\begin{aligned} P\left(\text{at least one of} \mid h\right) &= 1 - \left(1 - P\left(\text{specific neighbour}\right)\right)^h \\ &= 1 - \left(\left(1 - p\right)^k \left(p + 1\right)^{k-1}\right)^h \end{aligned}$$

Finally, the probability that a message can be passed along a path from node $N_i \in C_x$ to $N_j \in C_y$ where $\mathcal{H}\left(C_x, C_y\right) = d$ is

$$\begin{aligned} P\left(\text{path}\right) &= \prod_{i=1}^{d} P\left(\text{at least one of} \mid h = i\right) \\ &= \prod_{i=1}^{d} \left(1 - \left(\left(1 - p\right)^k \left(p + 1\right)^{k-1}\right)^i\right). \end{aligned} \tag{4.3.1}$$

4.3.2 Maximizing the Probability of Success

Ideally this section would find an exact configuration for d, w, and k that maximizes equation (4.3.1), however this equation proves rather unwieldy. The following lemma gives an approximation to equation (4.3.1).

Approximation 3. *Equation* (4.3.1) *is approximated by*

$$exp\left(1+\frac{1}{(1-p)^k\,(p+1)^{k-1}-1}\right)$$

for large d and small to moderate p.

Derivation. Let

$$f\,(i)=\left((1-p)^k\,(p+1)^{k-1}\right)^i.$$

Therefore, from equation (4.3.1),

$$\begin{aligned}P\,(\text{path}) &= \prod_{i=1}^{d}\left(1-f\,(i)\right)\\ &= \exp\left(\ln\left(\prod_{i=1}^{d}\left(1-f\,(i)\right)\right)\right)\\ &= \exp\left(\sum_{i=1}^{d}\ln\left(1-f\,(i)\right)\right).\end{aligned}$$

Since

$$\lim_{i\to\infty} f\,(i)=0$$

then a reasonable approximation of $\ln\left(1-f\,(i)\right)$ is the first term of the Taylor series around

$f(i) = 0$. Thus

$$\ln(1 - f(i)) \approx -f(i)$$

and

$$\begin{aligned} P(\text{path}) &= \exp\left(\sum_{i=1}^{d} \ln(1 - f(i))\right) \\ &\approx \exp\left(-\sum_{i=1}^{d} f(i)\right). \end{aligned}$$

Simplifying gives

$$P(\text{path}) \approx \exp\left(-\frac{(1-p^2)^k \left(\left((1-p)^k (p+1)^{k-1}\right)^d - 1\right)}{(1-p^2)^k - p - 1}\right).$$

Since $f(i)$ quickly approaches zero, a lower bound on $P(\text{path})$ comes from the infinite sum

$$\begin{aligned} P(\text{path}) &\leq \exp\left(-\sum_{i=1}^{\infty} f(i)\right) \\ &\leq \exp\left(1 + \frac{1}{(1-p)^k (p+1)^{k-1} - 1}\right). \end{aligned} \tag{4.3.2}$$

This provides an approximation that is accurate for small $f(i)$ and large d (see section 5.3 for simulation results) . Since $f(i)$ decreases exponentially with respect to k and d, and d and k are functions of n this approximation becomes more accurate as n gets large. □

Theorem 4. *As the network grows, equation* (4.3.2)*is maximized when*

$$k = -e\mathcal{W}_{-1}\left(-\frac{ne^{-\frac{\mathcal{P}(n)}{e}}}{e}\right)$$

Proof. Equation (4.3.2) grows as

$$(1-p)^k (p+1)^{k-1}$$

approaches 0. Since $0 < p < 1$ this occurs when k is maximized. This occurs when d is minimized which, from lemma 1, is when

$$k = -e\mathcal{W}_{-1}\left(-\frac{ne^{-\frac{\mathcal{P}(n)}{e}}}{e}\right)$$

□

Corollary 1. *If a GHN is using the always-forwards routing algorithm and requires a failure rate of* $\epsilon, 0 < \epsilon < 1$, *then*

$$\mathcal{P}(n) = -e\left(\ln\left(\frac{\gamma e^{-\frac{\gamma}{e}-1}}{n}\right)+1\right),$$

where

$$\gamma = \frac{\ln\left(\frac{1}{\ln(1-\epsilon)-1}+1\right)+\ln(p+1)}{\ln(1-p)+\ln(p+1)}.$$

Proof. Equation (4.3.2) gives the probability of success. If ϵ gives a maximum failure rate

then

$$
\begin{aligned}
1-\epsilon &= \exp\left(1+\frac{1}{(1-p)^k(p+1)^{k-1}-1}\right)\\
\ln(1-\epsilon) &= 1+\frac{1}{(1-p)^k(p+1)^{k-1}-1}\\
\ln(1-\epsilon)-1 &= \left((1-p)^k(p+1)^{k-1}-1\right)^{-1}\\
(\ln(1-\epsilon)-1)^{-1} &= (1-p)^k(p+1)^{k-1}-1\\
(\ln(1-\epsilon)-1)^{-1}+1 &= (1-p)^k(p+1)^{k-1}\\
\ln\left((\ln(1-\epsilon)-1)^{-1}+1\right) &= k\ln(1-p)+(k-1)\ln(p+1)\\
\ln\left((\ln(1-\epsilon)-1)^{-1}+1\right)+\ln(p+1) &= k\left(\ln(1-p)+\ln(p+1)\right)\\
k &= \frac{\ln\left(\frac{1}{\ln(1-\epsilon)-1}+1\right)+\ln(p+1)}{\ln(1-p)+\ln(p+1)}.
\end{aligned}
$$

This is the smallest k that will give an error rate of ϵ. Let this value of k be γ. Then, from theorem 4, $k=-e\mathcal{W}_{-1}\left(-\frac{ne^{-\frac{\mathcal{P}(n)}{e}}}{e}\right)$. Therefore in order for ϵ to be the failure rate

$$\gamma=-e\mathcal{W}_{-1}\left(-\frac{ne^{-\frac{\mathcal{P}(n)}{e}}}{e}\right).$$

Solving for the number of pointers gives

$$
\begin{aligned}
-e\mathcal{W}_{-1}\left(-ne^{-\frac{\mathcal{P}(n)}{e}-1}\right) &= \gamma \\
\mathcal{W}_{-1}\left(-ne^{-\frac{\mathcal{P}(n)}{e}-1}\right) &= -\frac{\gamma}{e} \\
-ne^{-\frac{\mathcal{P}(n)}{e}-1} &= \left(-\frac{\gamma}{e}\right)e^{-\frac{\gamma}{e}} \\
-ne^{-\frac{\mathcal{P}(n)}{e}-1} &= -\gamma e^{-\frac{\gamma}{e}-1} \\
e^{-\frac{\mathcal{P}(n)}{e}-1} &= \frac{\gamma e^{-\frac{\gamma}{e}-1}}{n} \\
-\frac{\mathcal{P}(n)}{e}-1 &= \ln\left(\frac{\gamma e^{-\frac{\gamma}{e}-1}}{n}\right) \\
-\frac{\mathcal{P}(n)}{e} &= \ln\left(\frac{\gamma e^{-\frac{\gamma}{e}-1}}{n}\right)+1 \\
\mathcal{P}(n) &= -e\left(\ln\left(\frac{\gamma e^{-\frac{\gamma}{e}-1}}{n}\right)+1\right).
\end{aligned}
$$

□

4.4 The Never-Backwards Algorithm

The previous two algorithms have passed messages by forcing them to make forward progress. This, however, does not take advantage of the variable width of a GHN. With a width greater than two, a message can be passed along a row – keeping the Hamming distance to the destination the same – before reaching a node that is closer to the target. In practice the number of hops along a row is unbounded but, to simplify the analysis, this section considers a model where the message is allowed at most one hop along a row. Section 5.4 gives the results of simulating the never-backwards protocol and shows that this assumption does not result in much error.

Suppose message m is Hamming distance h from its target. The Never-Backwards algorithm routes by first checking if it is possible to route directly to a node that is closer to the target than itself. If such a node exists then the message is sent to it, exactly like the Always-Forwards algorithm. If such a node does not exist then the message is routed to one of the $h\,(w-2)$ cliques that have the same Hamming distance as the current clique. Although the issue is not addressed directly in this thesis, it is assumed that there is a mechanism to prevent messages from being routed to the same clique more than once.

Algorithm 3 describes the never backwards algorithm. Lines 1 to 10 are identical to the always-forwards algorithm. If the message could not be sent directly to a closer node then it tries sending it to each of its rowmates. Line 11 iterates over all the dimensions and lines 12 to 17 try each rowmate in dimension i.

Algorithm 3 Never-Backwards Routing

```
 1: if message.destination = self.address then
 2:    consume_message
 3: end if
 4: message_sent = False
 5: for i = 0 to message.destination.length ∧¬message_sent do
 6:    if message.destination[i] ≠ self.address[i] then
 7:       next_hop = self.address[0:i] + message.destination[i] + self.address[i + 1 :]
 8:       message_sent = message.forward_to(next_hop)
 9:    end if
10: end for
11: for i = 0 to message.destination.length ∧¬message_sent do
12:    if message.destination[i] ≠ self.address[i] then
13:       for w = 0 to width ∧¬message_sent do
14:          next_hop = self.address[0 : i] + w + self.address[i + 1 :]
15:          message_sent = message.forward_to(next_hop)
16:       end for
17:    end if
18: end for
```

4.4.1 Calculating the Probability of Failure

This section calculates an estimate on the probability of a message being successfully routed through a GHN using the Never-Backwards algorithm. Although an exact formulation of the probability of failure is ideal, there are numerous statistical dependencies that are difficult to overcome. These dependencies are ignored and result in a good estimate of the probability of failure, rather than an exact calculation of it.

The first simplifying assumption made is that a message goes through at most one rowmate before reaching its destination. This clearly overestimates the probability of failure as it ignores numerous potential paths, however the probability of a path via two or more rowmates existing while a path via one or fewer rowmates does not exist is extremely low.

The probability of a node N_i not being able to route to one of its $w-2$ rowmates directly is the probability that all $w-2$ rowmates have failed. Thus

$$Q\,(\text{to row mate}) = q^{w-2}.$$

Thus the probability of the initial node $N_i \in C_x$ not being able to route to a rowmate $N_k \in C_y$ via a specific cliquemate $N_j \in C_x$ is the probability that either N_j has failed or that N_j has not failed, but all of its $w-2$ rowmates have failed. Thus

$$\begin{aligned} Q\,(\text{to any row mate via clique mate}) &= q + Q\,(\text{to row mate}) - q \cdot Q\,(\text{to row mate}) \\ &= q - (q-1)\,q^{w-1}. \end{aligned}$$

Since there are $k-1$ cliquemates, the probability of not being able to route to a specific

rowmate via *any* cliquemate is the probability that all $k-1$ them are unable to route to a rowmate. Thus

$$\begin{aligned} Q\left(\text{to row mate indirectly}\right) &= Q\left(\text{to any row mate via clique mate}\right)^{k-1} \\ &= \left(q-\left(q-1\right)q^{w-1}\right)^{k-1}. \end{aligned}$$

The probability of not being able to route to a rowmate must also include the probability of the initial node not being able to route to a row mate directly, which is q^{w-2}. To simplify the final expression for the probability of failure, however, the previous expression is raised to the power of k instead of $k-1$. This results in a slight overestimate of the probability of failure as the calculation assumes an additional point of failure. Note that the error because of this simplification shrinks as k grows large and, since k is a function of n, declines as n increases. Thus

$$Q\left(\text{to row mate}\right) \leq \left(q-\left(q-1\right)q^{w-1}\right)^{k}.$$

This calculates the probability of not being able to make the first hop along a row, but the second hop still remains. The second hop is the probability of not being able to route to a neighbouring clique and is calculated as

$$Q\left(\text{neighbouring clique}\right) = q\left(q\left(2-q\right)\right)^{k-1}.$$

in equation (4.2.4). Thus the probability of not being able to route across a row via a

single row mate is the probability that either the first or second hop fails. Hence

$$\begin{aligned} Q\left(\text{via row mate}\right) =& Q\left(\text{to row mate}\right) + Q\left(\text{neighbouring clique}\right) \\ & - Q\left(\text{to row mate}\right) \cdot Q\left(\text{neighbouring clique}\right) \\ & = \frac{\left(\left(\left(2-q\right)q\right)^k + q - 2\right)\left(q - \left(q-1\right)q^{w-2}\right)^k - \left(\left(2-q\right)q\right)^k}{q-2}. \end{aligned}$$

The probability of not being able to route to the row mate that is closer the destination than the current node is the probability that it can't be reached via a row mate and that the current clique can't route it directly. Thus

$$\begin{aligned} Q\left(\text{to closer clique}\right) &= Q\left(\text{via row mate}\right) \cdot Q\left(\text{neighbouring clique}\right) \\ &= \frac{\left(\left(2-q\right)q\right)^k \left(\left(\left(2-q\right)q\right)^k - \left(\left(\left(2-q\right)q\right)^k + q - 2\right)\left(q - \left(q-1\right)q^{w-2}\right)^k\right)}{\left(q-2\right)^2}. \end{aligned}$$

If a message is at Hamming distance h from its destination then the probability of it making forward progress is the probability that it can successfully route a message down any of the h rows that get it closer to the target. Ignoring some independence issues, the probability of this occurring is approximately

$$P\left(\text{forward progress}|h\right) \approx 1 - Q\left(\text{to closer clique}\right)^h.$$

Finally, the probability of being able to successfully route a message along an entire

path is the probability that it makes forward progress at each step. Hence

$$\begin{aligned} P\left(\text{path}\right) &= \prod_{i=1}^{d} P\left(\text{forward progress}|h=i\right) \\ &= \exp\left(\ln \prod_{i=1}^{d} P\left(\text{forward progress}|h=i\right)\right) \\ &= \exp\left(\sum_{i=1}^{d} \ln P\left(\text{forward progress}|h=i\right)\right) \\ &= \exp\left(\sum_{i=1}^{d} \ln\left(1-Q\left(\text{to closer clique}\right)^{h}\right)\right). \end{aligned}$$

Since $Q\left(\text{to closer clique}\right)^{h}$ approaches zero quickly,

$$\ln\left(1-Q\left(\text{to closer clique}\right)^{h}\right)$$

can be approximated by the first term of the Taylor expansion around $Q\left(\text{to closer clique}\right)^{h} = 0$ to get

$$\ln\left(1-Q\left(\text{to closer clique}\right)^{h}\right) \approx -Q\left(\text{to closer clique}\right)^{h}.$$

Thus

$$\begin{aligned} P\left(\text{path}\right) &\approx \exp\left(\sum_{i=1}^{d} -Q\left(\text{to closer clique}\right)\right) \\ &\approx \exp\left(-\sum_{i=1}^{d} Q\left(\text{to closer clique}\right)\right). \end{aligned}$$

A natural lower bound on the probability of success comes from the infinite sum

$$P\left(\text{path}\right) \leq \exp\left(-\sum_{i=1}^{\infty} Q\left(\text{to closer clique}\right)\right).$$

Using the Mathematica equation solver, this expression simplifies to

$$P\left(\text{path}\right) \leq \exp\left(\frac{\left(\left(2-q\right)q\right)^{k}\left(\frac{(q-2)\left(\left(q-(q-1)q^{w-2}\right)^{k}-1\right)}{\left(\left(2-q\right)q\left(q-(q-1)q^{w-2}\right)\right)^{k}-\left(\left(2-q\right)q\right)^{k}+q-2}+1\right)}{\left(\left(2-q\right)q\right)^{k}+q-2}\right). \tag{4.4.1}$$

The following corollary determines $P\left(\text{path}\right)$ as w gets large.

Corollary 2.

$$\lim_{w\to\infty} P\left(path\right) = \exp\left(\frac{\left(\left(2-q\right)q\right)^{k}\left(\frac{(q-2)\left(q^{k}-1\right)}{q^{2k}\left(2-q\right)^{k}-\left(\left(2-q\right)q\right)^{k}+q-2}+1\right)}{\left(\left(2-q\right)q\right)^{k}+q-2}\right). \tag{4.4.2}$$

4.4.2 Maximizing the Probability of Success

Unsurprisingly, the derivative of equation (4.4.1) is large and difficult to find the roots of. As a result, this section does not minimize the equation by means of finding the roots of the derivative. Rather, it relies on the intuition formed in the previous two sections: the probability of success occurs when d is minimized and k is maximized. The following theorem considers the probability of success when either w or k is maximized.

Theorem 5. *In the limit, equation* (4.4.1) *is maximized when k is maximized.*

Proof. Consider the limit of equation (4.4.1) as k grows without bound:

$$\lim_{k\to\infty} P\left(\text{path}\right) = \lim_{k\to\infty} \exp\left(\frac{\left(\left(2-q\right)q\right)^{k}\left(\frac{(q-2)\left(\left(q-(q-1)q^{w-2}\right)^{k}-1\right)}{\left(\left(2-q\right)q\left(q-(q-1)q^{w-2}\right)\right)^{k}-\left(\left(2-q\right)q\right)^{k}+q-2}+1\right)}{\left(\left(2-q\right)q\right)^{k}+q-2}\right).$$

Since

$$\lim_{k\to\infty} \left((2-q)\,q\right)^k = 0$$

the entire numerator goes to zero and

$$\lim_{k\to\infty} P\,(\text{path}) = 1.$$

Corollary 2 shows that maximizing w results in $P\,(\text{path}) < 1$ and therefore, in the limit, equation (4.4.1) is maximized when k is maximized. □

Chapter 5

Simulations

Simulations were run in order to provide empirical evidence in support of the theoretical results of this thesis. The simulations did not create an actual network; rather a graph was created in the same structure of a GHN of a particular configuration, nodes were randomly removed with probability q, and each routing algorithm was run to determine if it was possible to route between $N_i \in C_{\langle 0,0,\ldots,0\rangle}$ to $C_{\langle 1,1,\ldots,1\rangle}$.

Each configuration is defined by five parameters: n, the number of nodes, $\mathcal{P}(n)$, the number of pointers per node, q, the node failure rate, k, the clique size, and the routing algorithm used. The number of nodes varied from 2^{10} to 2^{20} in powers of two, the number of pointers varied from $3 \ln n$ to $\sqrt{n}$ in powers of two, the clique size varied from 1 to $-e\mathcal{W}_{-1}\left(-\frac{ne^{-\frac{p}{e}}}{e}\right)$ in powers of 2 (see lemma 1), and q was varied from 0.1 to 0.9 in increments of 0.1. Each configuration was run 1000 times.

In order to simplify the simulations, only *full* GHNs were used, where a full GHN has exactly kw^d nodes and $k, w, d \in \mathbb{Z}$. This limited the possible configurations as it is rarely possible to get exactly n nodes with integer values for k, w, and d. The configuration to

use was determined by calculating the exact values for d and w as a function of $\mathcal{P}(n)$, k and n (see lemma 3.1), then the combination of $(\lceil d \rceil, \lceil w \rceil), (\lfloor d \rfloor, \lceil w \rceil), (\lceil d \rceil, \lfloor w \rfloor)$, and $(\lfloor d \rfloor, \lfloor w \rfloor)$ that minimized $\left| n - kw^d \right|$ was selected. Since both w and d have to be rounded this can result in some large discrepancies between the desired n and the actual n used.

Some error occurs when predicting the success rate of the always-forwards and never-backwards algorithms for large q. This is largely a result of using the first term of the Taylor series to approximate the expression $\ln(1 - f(x))$ around the point $f(x) = 0$. Although $f(x)$ goes to zero quickly, it does so much more slowly as q approaches 1. The observed error should be considered admissible for two reasons. First, the error tends towards underestimating the probability of success. Thus the predicted values are actually lower bounds on the probability of success which, since they are fairly tight, make them almost as useful as exact values.

Second, the error occurs at unrealistically high values of q. Research focusing on the properties of churn in P2P networks has shown that the distribution of session-length times is given by the complimentary cumulative distribution function of a Weibull distribution with shape parameter $0.34 \leq k \leq 0.65$ and scale parameter $20 \leq \lambda \leq 165$ [19, 17]. Suppose that nodes in a GHN are configured to synchronously check if their neighbours are alive every 20 minutes. Thus the worst time to route would be immediately before nodes update their pointers. Since the cumulative distribution function of a Weibull distribution is given by $1 - e^{-(x/\lambda)^k}$ [21] the greatest failure rate at $x = 20$ is 0.63, which occurs when $k = 0.65$ and $\lambda = 20$. Thus in the worst case of nodes infrequently and synchronously checking their pointers the failure rate is 0.63. This suggests that failure rates greater than 0.7 are unlikely and that lower bounds for such failure rates are acceptable.

The remainder of this chapter is divided into four sections. Section 5.2 addresess the

probability of a path existing in a GHN, whuile the remaining sections cover specific results obtained by simulating each of the routing algorithms described in chapter 4.

5.1 The Probability of Any Path Existing

Before considering the results of specific routing algorithms, this section deals with the probability of *any* path existing from $C_{\langle 0,0,\ldots,0\rangle}$ to $C_{\langle 1,1,\ldots,1\rangle}$. This gives a reference point for the efficacy of the routing algorithms presented in chapter 4. The results in this section were obtained using the same parameters previously described and the existance of a path was found using depth-first search.

Figure 5.1 shows the probabiltiy of a path existing as a function of q, the probability of a node failing, where each chart has a different k. The remaining parameters, w and d, are ignored, thus producing columns of dots at some locations. Larger values of d and smaller values of w are found towards the bottom of the columns, while small values of d and large values of w appear towards the top.

The results show that k has a large influence not only on the probability of a path existing, but also on how the GHN reacts to increasing node failure rates. Consider figures 5.1(a) and 5.1(f) where $k = 1$ and $k = 32$, respectively. When $k = 1$ the probability of a path existing decreases linearly as q increases, however when $k = 32$ the probability of a path existing drops below one only for the most extreme values of q; in fact, when $k = 32$ and $q = 0.9$ the probability of a path existing is greater than 0.8.

Figure 5.2 shows the probability of finding a path in a GHN as a function of k when $d = 3$, where each chart shows a different node failure rate, q. The probability of a path existing clearly increases as a function of k, even for large values of q, although this

relationship is not nearly as obvious for small values of q. This should be expected as the probability of a route existing is high when q is small, therefore making the increase less apparent.

Figure 5.3 shows the probability of a path existing in a GHN as a function of q when $k = 4$ where each chart is a different width, w. With the excpetion of $w = 4$ in figure 5.3(a), it appears that the width of the GHN makes little different in the probability of a path existing. This is at least partly due to the relationship between w and d; d decreases as w increases with n held constant. Thus the path length decreases as w increases, making it more probable that a path exists.

In summary, the probability of a path existing in a GHN is dominated by the clique size, k, while the width, w, has little impact on the existance of a path. These results are consistent with the theoretical results from chapter 4 that found the probability of routing successfully is maximized when k is maximized.

5.2 Simulating the Naïve Algorithm

Simulation results indicate that equation (4.2.5) is a good estimate of the probability of routing successfully. Figure 5.4(a) shows the probability of success as a function of the node failure rate for $k = 1$ and various values of d. The markers, representing the observed values, are mostly on the prediction curve, indicating that the predicted values were accurate. The small discrepancies can be attributed to the independence assumption discussed in section 4.2.1. Notice that the highest rate of success occurs when $d = 1$ which supports the claim made in theorem 2.

Figure 5.4(b) shows the routing success rate as a function of the node failure rate when

$d = 3$, with each series representing a different clique size. Note that clique sizes up to 2^9 were tested, however they resulted in $d < 3$ and are therefore not shown in this figure. This plot makes the error due to the independence assumption made in section 4.2.1 more prominent, however this error diminishes as k increases. The reason for this is two-fold: first, as k increases the contribution of the independence assumption becomes smaller. Secondly, when n is held constant, d decreases as k increases. Since equation (4.2.5) is raised to the power of d, this significantly reduces the compounding effect of the error.

Figure 5.4(c) shows the routing success rate as a function of the number of dimensions when $q = 0.5$ and with each series representing a different clique size. Note that $q = 0.5$ was selected because it shows the largest error between the predicted and observed values. The plot illustrates that the predicted value is still fairly close but, more importantly, that it follows the same shape as the observed values. The latter is important as it supports the claims of minimization made in theorem 2.

It should be clear from the simulations that the naïve algorithm is not particularly effective. For example, consider the points $q > 0.6$ and $k = 32$ from figure 5.4(b). Assume that $w = 2$ since the width is irrelevant to the naïve algorithm. There are therefore $32 \cdot 2^3 = 256$ nodes in the network maintaining $32 + 2 \cdot 3 = 38$ pointers each. Since $\sqrt{256} = 16$, each node is responsible for more than $2\sqrt{n}$ pointers and the probability of success drops dramatically when $q \geq 0.6$. Since each node is responsible for more than $2\sqrt{n}$ pointers, they could be arranged in one dimension with $\sqrt{n}$ cliques of size $\sqrt{n}$ and obtain a much better probability of success.

5.3 Simulating the Always-Forwards Algorithm

Simulation results indicate that equation (4.3.2) is a tight lower bound on the probability of routing success for the always forwards algorithm and that the probability of routing is maximized when k is maximized. Figure 5.5 shows the observed and predicted probability of success as a function of q and k. The columns of dots occur when trials were performed with the same q and k, but with varying d; larger values of d occur closer to the curve. In some cases the predicted lower bound is slightly higher than the observed values. This is a result of using the Taylor series to approximate $\ln 1 - x$ around the point $x = 0$ and only occurs as q approaches 1.

Figure 5.6 shows the probability of routing success as a function of k when $d = 3$ for varying values of q. Multiple markers indicate that several trials were run with the same d, k and q, but w was different. Since the width of the GHN doesn't influence the always-forwards algorithm any variation between the markers is purely by chance. The results indicate that equation (4.3.2) is accurate for small d even though it is calculated by summing over d from 1 to infinity. This suggests that, aside from q, the largest factor influencing the performance of the always-forwards algorithm is the clique size.

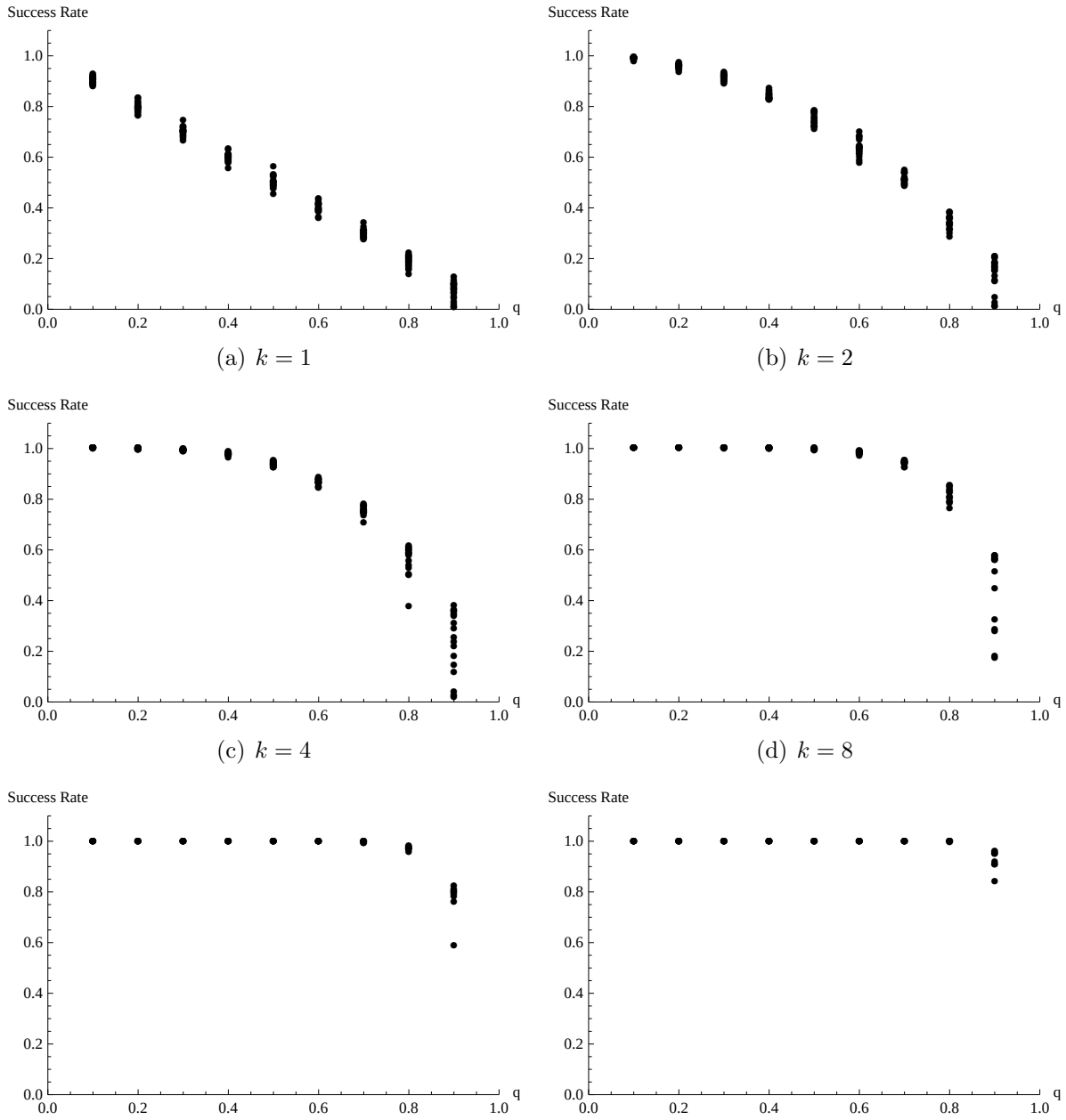
The always-forwards algorithm shows a dramatic improvement over the naïve algorithm in terms of probability of success. Consider the observed probability at $q = 0.8$ in figure 5.5(f) where $k = 32$ and d is approximately 5. Since the width is irrelevant, assume it is 2. There are therefore at least $32 \cdot 2^5 = 1024$ nodes in the network maintaining $32 + 2 \cdot 5 = 42$ nodes each, or slightly more than $\sqrt{n}$. Even considering the largest d, the observed probability of success is approximately 0.75, or nearly double that of the naïve algorithm with the same k and q, but a *smaller* d and *fewer* nodes.

5.4 Simulating the Never-Backwards Algorithm

Simulations of the never-backwards algorithm indicate that equation (4.4.1) is a good estimate of the success rate for small to moderate q and is a reasonable lower bound for larger q. Figure 5.7 shows the simulation results for $k = 4$ and various values of w as a function of q. In all cases the predicted value appears to be a good estimate $q \leq 0.7$ and forms a lower bound for $q \geq 0.8$.

Figure 5.8 shows the probability of routing successfully as a function of k for particular values of q. The solid line shows the lower bound on the probability of success that occurs when $w = 2$ and the dashed line shows the probability of success as w gets large. The figures suggest that equation (4.4.1) is a close lower bound for the expected probability of success when w is maximized.

As expected, the never-backwards algorithm shows a large improvement over the always-forwards algorithm. Figure 5.8(d) shows that when $q = 0.8$ and $k = 32$ the probability of successfully routing is near one. The probability of success using the always-forwards algorithm with the same configuration is approximately 0.75, as shown in figure 5.5(f). Thus the probability of success is significantly improved by allowing messages to be routed across rows before sending to a node that is closer to the target.

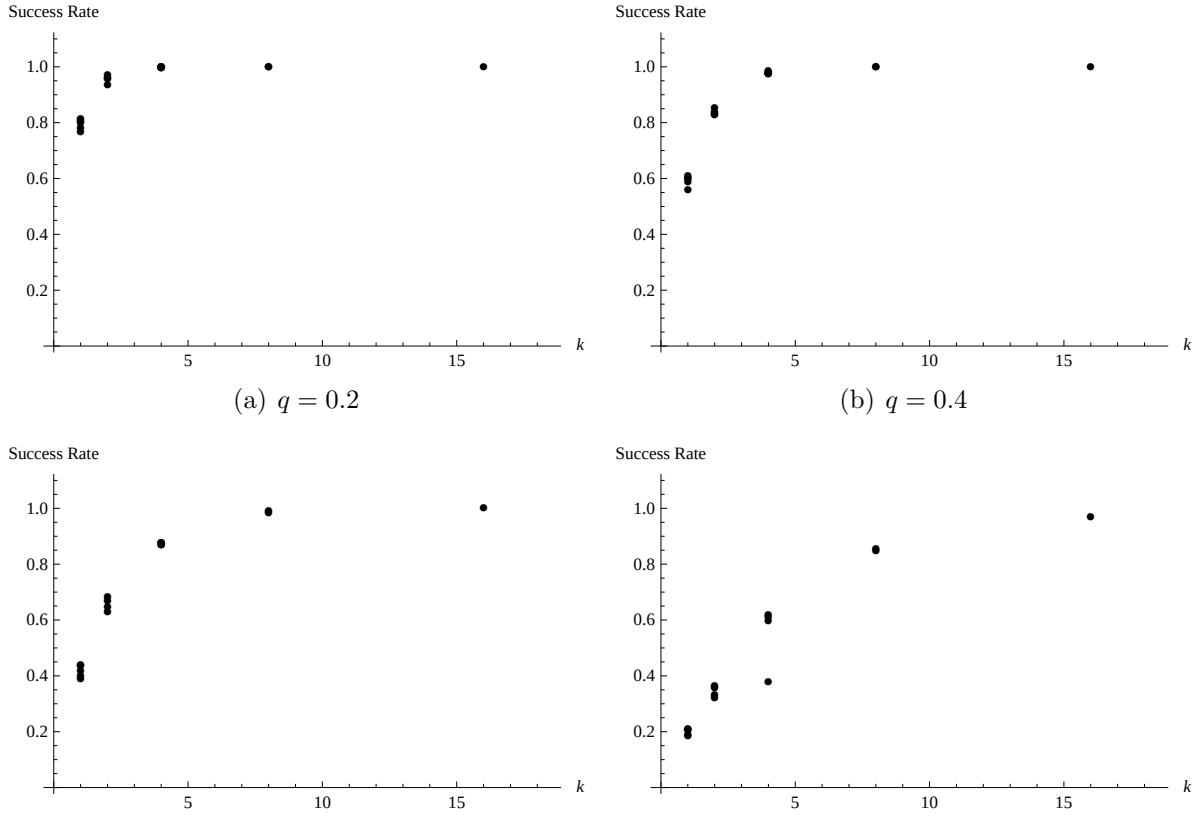


(a) $k = 1$ (b) $k = 2$

(c) $k = 4$ (d) $k = 8$

(e) $k = 16$ (f) $k = 32$

Figure 5.1: Simulation results for finding the probability of a path existing in a GHN. Each chart shows the probability of routing successfully as a function of q and k with the line representing the predicted lower bound on the probability of success and the dots representing the mean of 500 trials. The columns of dots occur when numerous trials were performed with the same q and k, but different d; trials with a larger d value are found at the bottom of the column, while trials with low d values occur at the top.



(a) $q = 0.2$ (b) $q = 0.4$

(c) $q = 0.6$ (d) $q = 0.8$

Figure 5.2: Simulation results for finding the probability of a path existing in a GHN. Each chart shows the probability of routing as a function of k for $d = 3$ and varying values of q. Each marker represents the success rate observed over 500 trials. Multiple markers occur when their values of d, k and q are the same, but w is different.

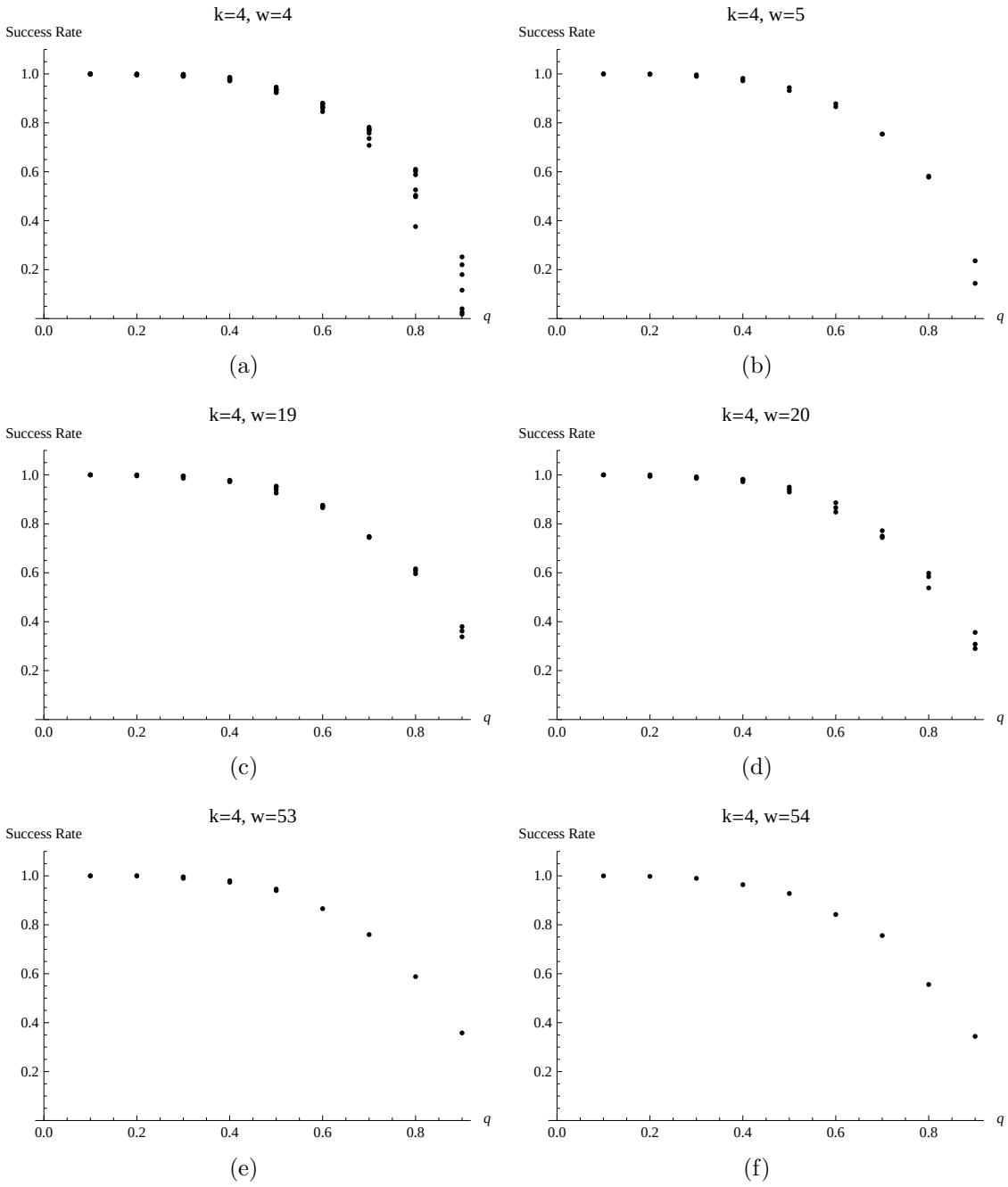
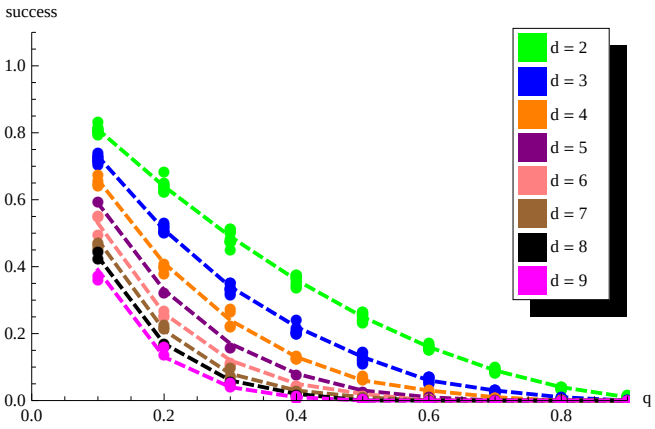
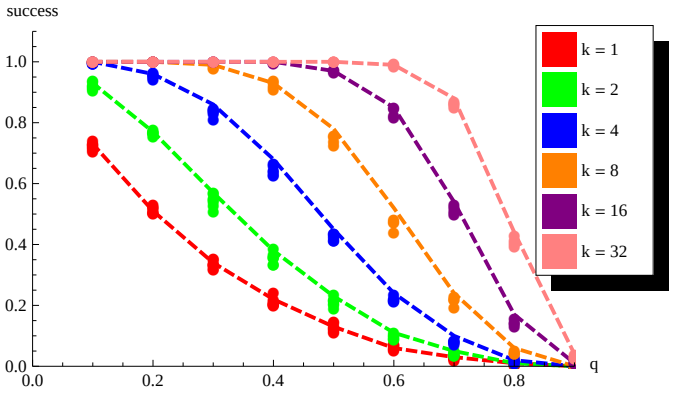


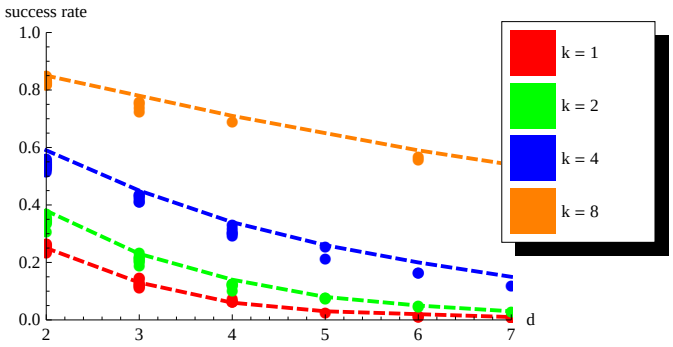
Figure 5.3: Simulation results for finding the probability of a path existing in a GHN. The solid line shows the predicted success rate as a function of the width, clique size, and node failure rate, while the markers represent the success rate over 1000 trials. Note that multiple markers in a column occur when when w, k, and q are the same, but d is different. Higher d values occur towards the bottom of the column.



(a) $k = 1$

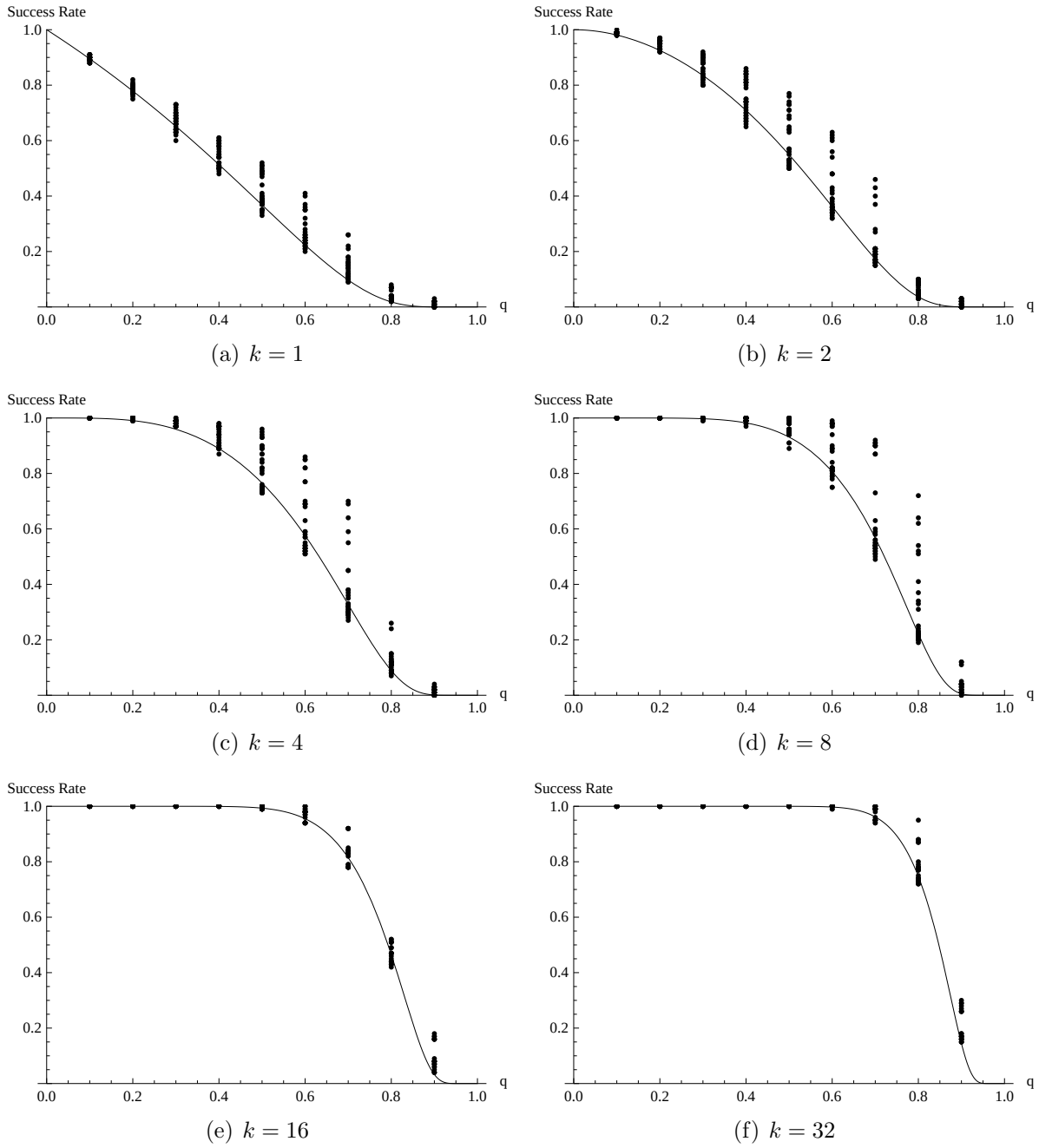


(b) $d = 3$



(c) $q = 0.5$

Figure 5.4: Simulation results for the naïve algorithm. Markers represent the success rate over 1000 trials and the dotted line is the expected success rate from equation (4.2.1). Multiple dots occur when a trial was conducted with d, k, and q were the same, but w was different. Since the naïve algorithm is not impacted by w the observed variation is purely by chance.

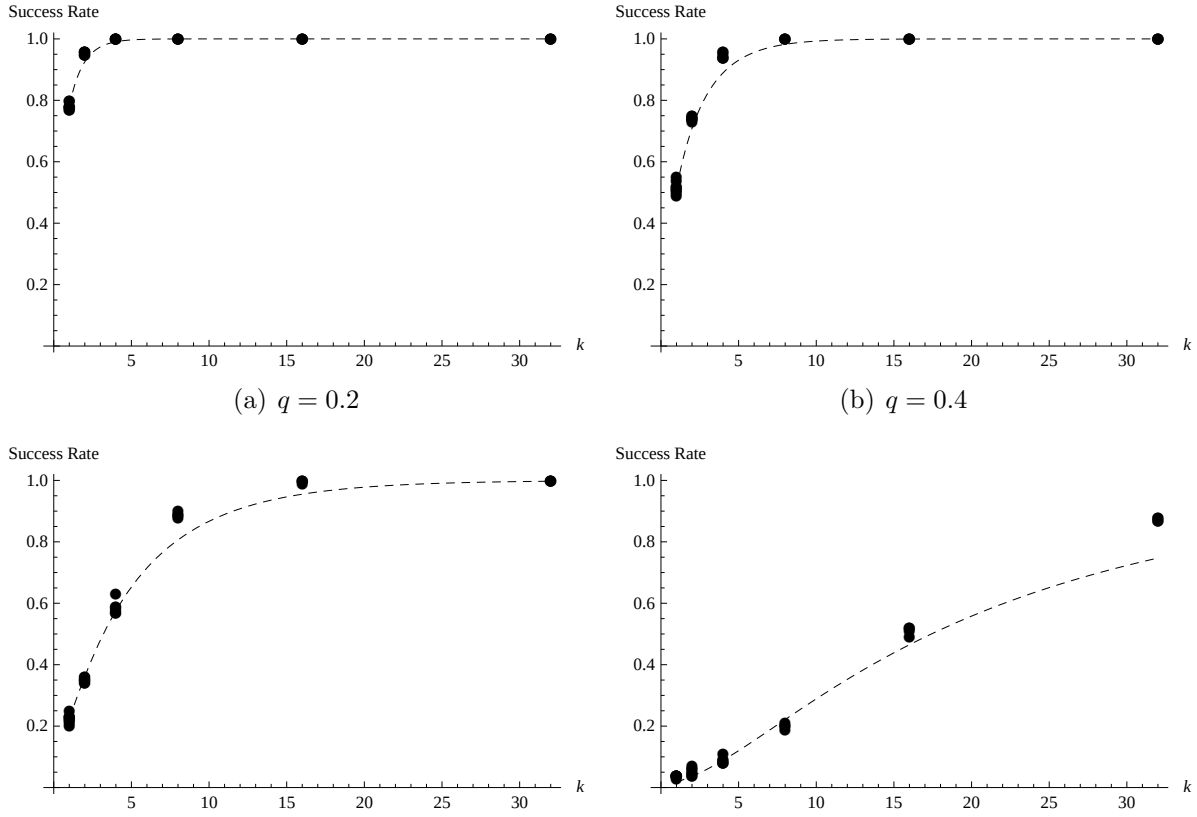


(a) $k = 1$ (b) $k = 2$

(c) $k = 4$ (d) $k = 8$

(e) $k = 16$ (f) $k = 32$

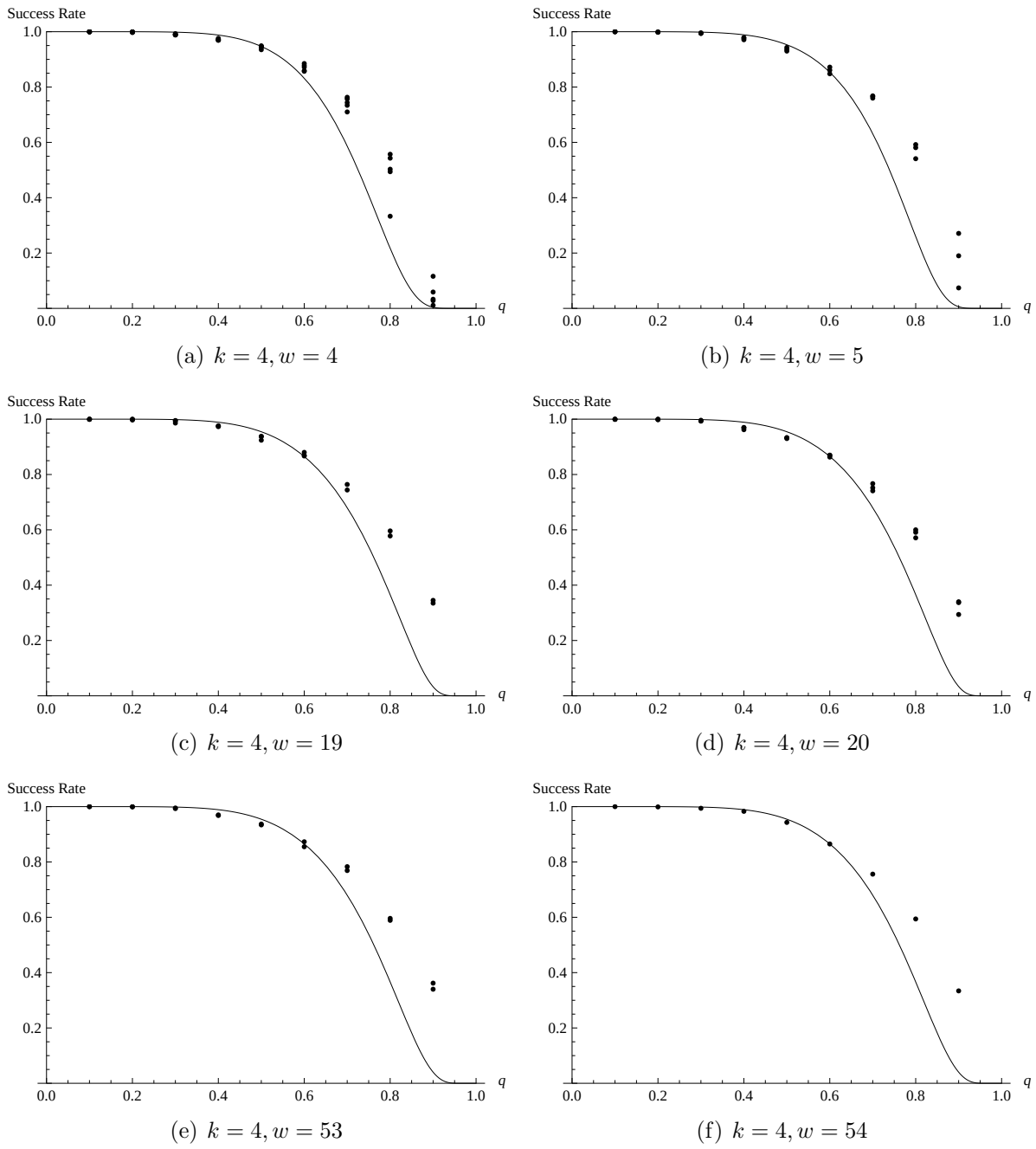
Figure 5.5: Simulation results for the always-forwards algorithm. Each chart shows the probability of routing successfully as a function of q and k with the line representing the predicted lower bound on the probability of success and the dots representing the mean of 1000 trials. The columns of dots occur when numerous trials were performed with the same q and k, but different d; trials with a larger d value are found at the bottom of the column, while trials with low d values occur at the top.



(a) $q = 0.2$ (b) $q = 0.4$

(c) $q = 0.6$ (d) $q = 0.8$

Figure 5.6: Simulation results for the always-forwards algorithm. Each chart shows the probability of routing as a function of k for $d = 3$ and varying values of q. Each marker represents the success rate observed over 1000 trials. Multiple markers occur when their values of d, k and q are the same, but w is different.

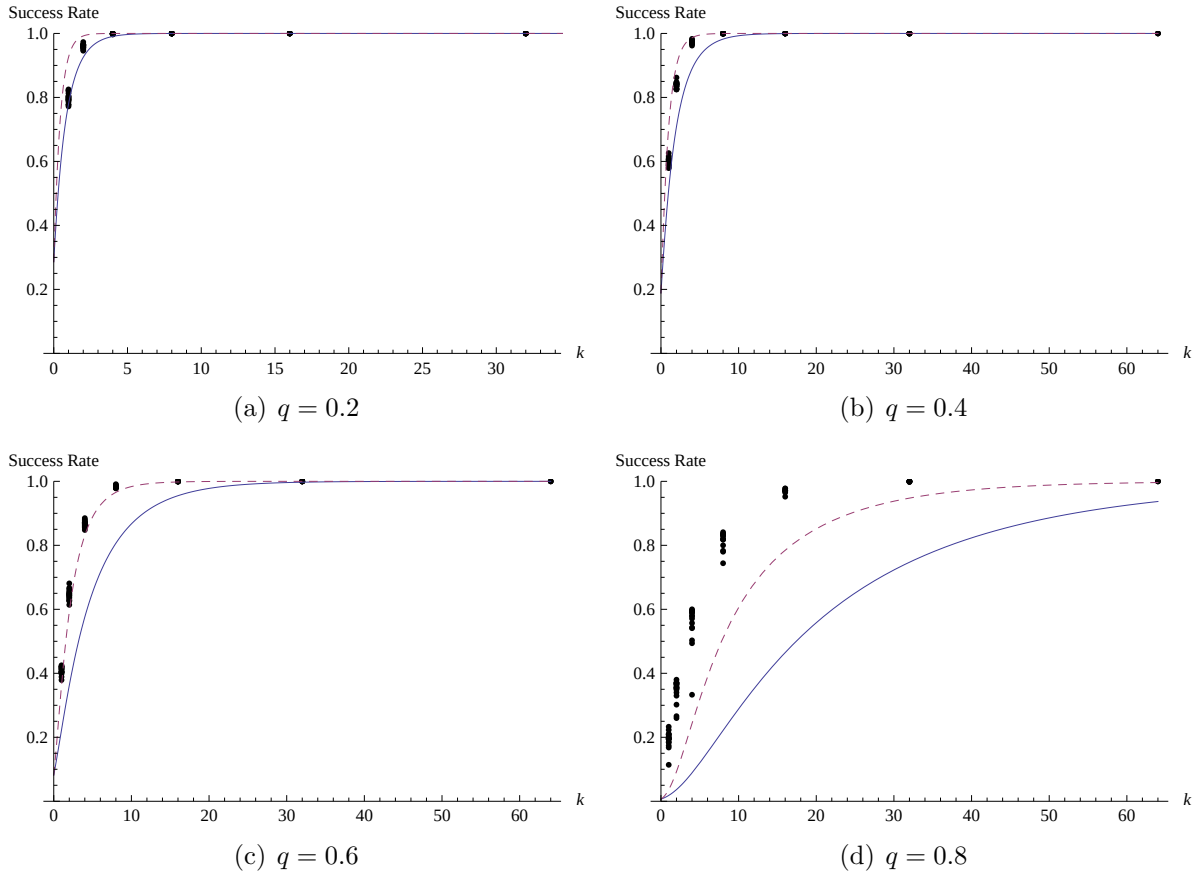


(a) $k = 4, w = 4$ (b) $k = 4, w = 5$

(c) $k = 4, w = 19$ (d) $k = 4, w = 20$

(e) $k = 4, w = 53$ (f) $k = 4, w = 54$

Figure 5.7: Simulation results for the never-backwards algorithm. The solid line shows the predicted success rate as a function of the width, clique size, and node failure rate, while the markers represent the success rate over 1000 trials. Note that multiple markers in a column occur when when w, k, and q are the same, but d is different. Higher d values occur towards the bottom of the column.



(a) $q = 0.2$ (b) $q = 0.4$

(c) $q = 0.6$ (d) $q = 0.8$

Figure 5.8: Simulation results for the never-backwards algorithm as a function of k. Each chart shows the predicted success plotted as a function of k for a particular value of q. The markers represent the success rate over 1000 trials. Note that multiple markers in a column occur when k, and q are the same, but d is different. Higher d and w values occur towards the bottom of the column. The solid line represents the predicted value from equation (4.4.1) with $w = 2$ while the dashed line represents the predicted value for large w from equation (4.4.2).

Chapter 6

Conclusion

6.1 Future Work

This thesis presented a new type of service-based overlay network that can be tuned to a varying number of pointers per node. Although this work presents good approximations of the probability of success and minimization of those functions, there is still much work to be done. For example, it would be useful to tighten approximations presented in chapter 4, particularly for large values of q, as well as to find tight upper bounds on these expressions. This work would also benefit from mathematically bounding the error in the given approximations.

Since the probability of failure is given as a function of q, n, and $\mathcal{P}(n)$, it would be useful to find a minimum $\mathcal{P}(n)$ that guarantees a particular error rate, ϵ. Finding such a function was done for the always-forwards algorithm in corollary 1, however solving this problem in the general case for specific values of k, w, and d yields difficult non-linear partial differential equations.

6.2 Summary

This thesis presented a new service-based overlay network called a Grouped Hamming Network. Originally intended as an ancillary distributed data structure to select a random node in a P2P network, this network can be used as a general purpose service-based network. The key new feature of the GHN compared to previous networks is its ability to easily adjust the overhead associated with maintaining the network by means of changing the number of pointers maintained per node. This allows the probability of successfully routing under random node failures to increase as the available bandwidth increases. Previously the overhead associated with a particular P2P network was a property of that network and often critical when considering which P2P network to use. As a result of the structure presented in this thesis it is now possible to implement a P2P network as a GHN and fine-tune the overhead associated with the network.

Three different routing schemes were presented and then analyzed under random uniform node failures. Theoretical analysis of each routing scheme gave predictions for the expected probability of routing success as well as the network configuration that maximizes the probability of success. The optimal configurations were given in terms of n, the number of nodes in the network, and $\mathcal{P}(n)$, the number of pointers maintained per node. Thus the optimal configuration of the network is entirely defined by the only two parameters the user would be concerned about and the remaining parameters are left "under the hood".

Simulation results support the theoretical analysis both in terms of their predicted values and in terms of their minimizations. Furthermore the results support previous work that shows Hamming networks – in particular, hypercubes – are extremely resilient to node failures. Unlike previous research on hypercubes, the GHN presented in this thesis gives a

high probability of success even for the simplest of routing algorithms. Thus they can be easily implemented, yet still have good theoretical behaviour.

References

[1] Corless, R. M., Gonnet, G. H., Hare, D. E. G., Jeffrey, D. J., and Knuth, D. E. On the Lambert W function. *Advances in Computational Mathematics 5*, 1 (1996), 329–359.

[2] Dabek, F., Li, J., Sit, E., Robertson, J., Kaashoek, M. F., and Morris, R. Designing a DHT for low latency and high throughput.

[3] Dally, W. Performance analysis of k-ary n-cube interconnection networks. *IEEE Transactions on Computers 39* (1990), 775–785.

[4] Gupta, A., Liskov, B., and Rodrigues, R. Efficient routing for peer-to-peer overlays. In *NSDI'04: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation* (Berkeley, CA, USA, 2004), USENIX Association, pp. 9–9.

[5] Gupta, I., Birman, K., Linga, P., Demers, A., and van Renesse, R. Kelips: Building an efficient and stable P2P DHT through increased memory and background overhead. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)* (2003).

[6] Kaklamanis, C., Krizanc, D., and Tsantilas, T. Tight bounds for oblivious routing in the hypercube. In *SPAA '90: Proceedings of the second annual ACM symposium on Parallel algorithms and architectures* (New York, NY, USA, 1990), ACM, pp. 31–36.

[7] Kang, H., Chan-Tin, E., Hopper, N., and Kim, Y. Why Kad lookup fails. Tech. rep., University of Minnesota, 2009.

[8] Li, J., Stribling, J., Morris, R., and Kaashoek, M. F. Bandwidth-efficient management of dht routing tables. In *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation* (Berkeley, CA, USA, 2005), USENIX Association, pp. 99–114.

[9] Liben-Nowell, D., Balakrishnan, H., and Karger, D. Analysis of the evolution of peer-to-peer systems. In *PODC '02: Proceedings of the twenty-first annual symposium on Principles of distributed computing* (New York, NY, USA, 2002), ACM, pp. 233–242.

[10] Ma, M., Liu, G., and Pan, X. Path embedding in faulty hypercubes. *Applied Mathematics and Computation 192*, 1 (2007), 233 – 238.

[11] Manku, G. S., Bawa, M., and Raghavan, P. Symphony: Distributed hashing in a small world. In *USITS'03: Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems* (Berkeley, CA, USA, 2003), USENIX Association, pp. 10–10.

[12] Maymounkov, P., and Mazières, D. Kademlia: A peer-to-peer information system based on the XOR metric. In *IPTPS '01: Revised Papers from the First In-*

ternational Workshop on Peer-to-Peer Systems (London, UK, 2002), Springer-Verlag, pp. 53–65.

[13] Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Shenker, S. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications* (New York, NY, USA, 2001), ACM, pp. 161–172.

[14] Rhea, S., Geels, D., Roscoe, T., and Kubiatowicz, J. Handling churn in a DHT. In *ATEC '04: Proceedings of the annual conference on USENIX Annual Technical Conference* (Berkeley, CA, USA, 2004), USENIX Association, pp. 10–10.

[15] Rowstron, A. I. T., and Druschel, P. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware '01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg* (London, UK, 2001), Springer-Verlag, pp. 329–350.

[16] Schlosser, M., Sintek, M., Decker, S., and Nejdl, W. *Agents and Peer-to-Peer Computing*. Springer Berlin / Heidelberg, 2003, ch. HyperCuP Hypercubes, Ontologies, and Efficient Search on Peer-to-Peer Networks.

[17] Steiner, M., En-Najjary, T., and Biersack, E. W. Long term study of peer behavior in the KAD DHT. *IEEE/ACM Trans. Netw. 17*, 5 (2009), 1371–1384.

[18] Stoica, I., Morris, R., Liben-Nowell, D., Karger, D. R., Kaashoek, M. F., Dabek, F., and Balakrishnan, H. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. Netw. 11*, 1 (2003), 17–32.

[19] Stutzbach, D., and Rejaie, R. Understanding churn in peer-to-peer networks. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement* (New York, NY, USA, 2006), ACM, pp. 189–202.

[20] Urdaneta, G., Pierre, G., and Steen, M. V. A survey of DHT security techniques. *ACM Computing Surveys* (2009).

[21] Weibull, W. A statistical distribution function of wide applicability. *Journal of Applied Mechanics* (1951), 293–297.

[22] Wu, D., Tian, Y., and Ng, K.-W. Analytical study on improving DHT lookup performance under churn. In *P2P '06: Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing* (Washington, DC, USA, 2006), IEEE Computer Society, pp. 249–258.

[23] Yang, X., Evans, D. J., Chen, B., Megson, G. M., and Lai, H. On the maximal connected component of hypercube with faulty vertices. *International Journal of Computer Mathematics 81*, 5 (2004), 515–525.