ABSTRACT

Title of dissertation: GLOBAL PHENOMENA
FROM LOCAL RULES:
PEER-TO-PEER NETWORKS
AND CRYSTAL STEPS

Amy Finkbiner, Doctor of Philosophy, 2007

Dissertation directed by: Professor James Yorke
Department of Mathematics
Department of Physics
Institute for Physical Science and Technology

Professor Brian Hunt
Department of Mathematics
Institute for Physical Science and Technology

Professor Dionisios Margetis
Department of Mathematics
Institute for Physical Science and Technology

Even simple, deterministic rules can generate interesting behavior in dynamical systems. This dissertation examines some real world systems for which fairly simple, locally defined rules yield useful or interesting properties in the system as a whole. In particular, we study routing in peer-to-peer networks and the motion of crystal steps.

Peers can vary by three orders of magnitude in their capacities to process network traffic. This heterogeneity inspires our use of "proportionate load balancing," where each peer provides resources in proportion to its individual capacity. We provide an implementation that employs small, local adjustments to bring the entire network into a global balance. Analytically and through simulations, we demon-

strate the effectiveness of proportionate load balancing on two routing methods for de Bruijn graphs, introducing a new "reversed" routing method which performs better than standard forward routing in some cases.

The prevalence of peer-to-peer applications prompts companies to locate the hosts participating in these networks. We explore the use of supervised machine learning to identify peer-to-peer hosts, without using application-specific information. We introduce a model for "triples," which exploits information about nearly contemporaneous flows to give a statistical picture of a host's activities. We find that triples, together with measurements of inbound vs. outbound traffic, can capture most of the behavior of peer-to-peer hosts.

An understanding of crystal surface evolution is important for the development of modern nanoscale electronic devices. The most commonly studied surface features are steps, which form at low temperatures when the crystal is cut close to a plane of symmetry. Step bunching, when steps arrange into widely separated clusters of tightly packed steps, is one important step phenomenon. We analyze a discrete model for crystal steps, in which the motion of each step depends on the two steps on either side of it. We find an time-dependence term for the motion that does not appear in continuum models, and we determine an explicit dependence on step number.

GLOBAL PHENOMENA FROM LOCAL RULES: PEER-TO-PEER NETWORKS AND CRYSTAL STEPS

by

Amy Finkbiner

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2007

Advisory Committee:
Professor James Yorke, Chair/Advisor
Professor Brian Hunt, Co-Advisor
Professor Dionisios Margetis, Co-Advisor
Professor Edward Ott
Professor Neil Spring

Acknowledgements

I am indebted to many people for helping me through graduate school and my dissertation.

My co-advisor, Jim Yorke, is a source for many great ideas, but he has also always taken time to review my papers and presentations. Brian Hunt deserves just as much credit for his ideas and availability. I am also grateful to Ed Ott and Eric Harder for participating in the networks research group. I am thankful that Dio Margetis was willing to give me a fascinating project at a very busy time in my life. I am also grateful to Neil Spring for running an interesting class and serving on my committee.

I would not have made it through the daily routine without my series of excellent officemates, from Christian Zorn and Eric Errthum to Poorani Subramanian and Russell Halper. Special thanks go to Russ for suggesting the title of this dissertation.

The NPSC and AFOSR provided my financial support, and the math and IPST department staff were amazing in their ability to make that support run smoothly.

Finally, I would not be where I am today without my Swarthmore education and my family, so I give them all my thanks.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview of the dissertation

Even simple, deterministic rules can generate interesting behavior in dynamical systems. For example, the logistic map exhibits chaos for certain parameter values. This dissertation examines some real-world systems for which fairly simple, locally-defined rules yield useful or interesting properties in the system as a whole.

The remaining sections of this chapter provide an introduction to modern peer-to-peer networks and to crystal surfaces. We trace the history of peer-to-peer networks from completely centralized systems like Napster, to completely decentralized systems like Gnutella, to hybrid systems like FastTrack, to alternative systems like BitTorrent. We then give a detailed introduction to distributed hash tables, which are decentralized but structured peer-to-peer systems. Finally, we discuss the motivation behind the study of crystal steps and the step bunching phenomenon.

In Chapter 2, we address the issue of heterogeneity in peer-to-peer networks. By heterogeneity, we refer to the fact that peers can vary by three orders of magnitude in their capacities to process traffic. We define and analyze networks based on two routing methods for de Bruijn graphs, introducing a new "reversed" routing method which performs better than standard forward routing in some cases. We quantify how heterogeneity affects the average query path length under each routing

method, proving that the length decreases in most cases. We also provide an implementation of what we term "proportionate load balancing," which is applicable to a range of peer-to-peer networks. Proportionate load balancing employs small local adjustments to bring the entire network into a global balance, where each participant provides resources in proportion to its capacity. We demonstrate the effectiveness of proportionate load balancing on de Bruijn networks.

Chapter 3 explores the use of supervised machine learning to identify peer-to-peer hosts, without using port numbers or other application-specific information. Most studies in this area have attempted to classify TCP flows, rather than individual users. We introduce a model for "triples," which capture the composite behavior available when we study hosts rather than flows. Triples exploit information about nearly-contemporaneous flows to give a statistical picture of a host as a whole. We find that triples, together with measurements of inbound vs. outbound traffic, can capture most of the behavior of peer-to-peer hosts, while maintaining acceptable "false-positive" rates.

In Chapter 4, we analyze a discrete model for crystal steps. In this model, the motion of each step depends on the two steps on either side of it. We evaluate the effect of step-step interactions on the stability of the crystal surface, with a focus on step bunching.

1.2 Peer-to-peer networks

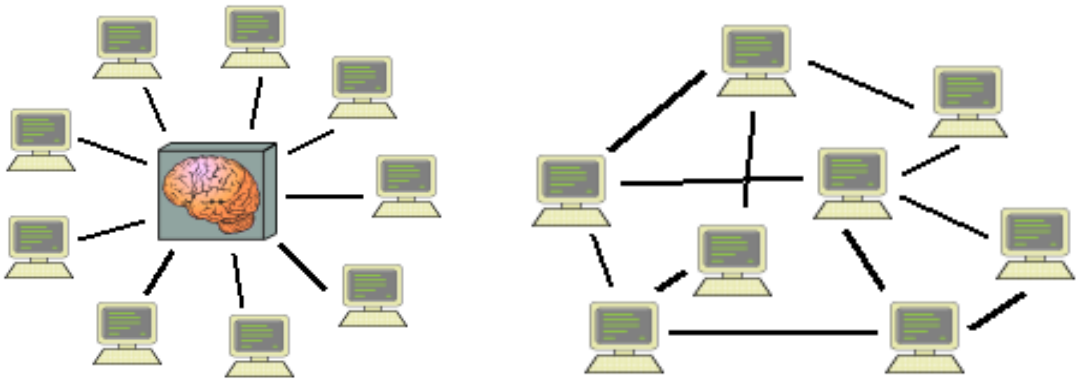
Classical internet activity fits the *client-server model*, where powerful central computers (the servers) provide resources to lower-capacity end users (the clients). Some examples of servers include `cnn.com`, which disseminates news; `microsoft.com`, which distributes Windows updates; `yahoo.com`, which provides forums for games and chatting; or the machines that handle all `umd.edu` email.

Peer-to-peer (P2P) applications, on the other hand, represent a distributed type of internet activity. A *peer-to-peer network* is a collection of machines that associate with one another to share files or other resources. A P2P network may have servers that facilitate these communications, but unlike in a client-server setup, the servers do not provide all of the content.

1.2.1 Statistics

Up to 60% of worldwide Internet traffic is peer-to-peer [42]. Most peer-to-peer applications are used for *file sharing* — sending files directly between end-users. By volume, P2P traffic is [42]

- 61% video content,
- 11% audio content,
- 11% compressed files,
- 17% other.



(a) A centralized P2P system.

(b) A decentralized P2P system.

Figure 1.1: Peer-to-peer applications may be divided broadly into two classes. Centralized systems have a server that organizes file sharing for a large number of subsidiary users; this server is a bottleneck and single point of failure for the system. Decentralized systems lack a central authority, so they avoid the weaknesses of centralized systems; however, searches can only follow a limited number of paths, so such systems can be highly inefficient.

$2.3 billion in copyrighted movies were downloaded worldwide in 2005 [36]. Only 13% of that loss came from users in the United States.

1.2.2 History

Broadly, there are two classes of peer-to-peer applications: centralized and decentralized. See Figure 1.1.

Centralized systems (Napster, DirectConnect, BitTorrent) have a computer that acts as a hub to organize file transmissions for a large number of client computers. Napster had a single hub for all its users; DirectConnect has many disjoint sets of

clients and a hub; BitTorrent has a hub for every individual file. Despite their centralized organization, these systems are peer-to-peer because shared files are transmitted directly between clients. The hubs act only as facilitators to help clients find other users who possess the information they want, not as content servers. The hubs also provide an obvious target for disabling the network; Napster was shut down by court order in July 2001 [53].

Decentralized systems (Gnutella, FastTrack (Kazaa), distributed hash tables (Kademilia)) have each peer connected to some small number of other peers, without a central authority, so searches for files must be distributed through their networks. Gnutella became popular with the demise of Napster. It employs flooding, where a peer asks its connected peers if they have a certain file, whereupon they ask their connected peers, etc. FastTrack automatically divides its peers into clients and supernodes, with supernodes performing searches in a Gnutella-like fashion for their connected clients. Distributed hash tables are structured to provide more efficient searches in exhange for the overhead of maintaining the structure. In all cases, once the requested file is located, it is transferred between the relevant clients.

Some systems (eDonkey2000) seem to lie between the two extremes. eDonkey2000 originally acted like DirectConnect with disjoint hub networks, but it now allows the hubs to pass searches among themselves.

1.3 Distributed hash tables

Distributed hash tables (DHTs) form completely decentralized P2P systems which, due to an imposed structure, have reliable and efficient search methods. See [32] for details on several popular protocols. Let $\mathcal{P}$ be the set of peers. In general, DHTs have three structural components:

1. A keyspace $\mathcal{K}$, with a publicly-known function $f : \{\text{data items}\} \to \mathcal{K}$ that assigns *keys* to data items.

2. A rule for assigning a portion, or *zone*, of the keyspace to each peer $i \in \mathcal{P}$, such that $\bigcup_{i \in \mathcal{P}} zone(i) = \mathcal{K}$.

3. A set of application-level connections, assigned between peers based on the keys in their zones.

In the following subsections, we formalize this definition, then briefly discuss the operation and maintenance of DHTs.

1.3.1 Continuous-discrete definition

For definiteness, we describe Naor and Wieder's continuous-discrete approach [37] to defining DHTs. In this framework,

1. $\mathcal{K}$ is a continuous space. There is a graph G_{cont} that has $\mathcal{K}$ as its vertex set. (f is unrestricted; it is often taken to be a uniform hash function.)

2. $\mathcal{K}$ is decomposed into possibly-overlapping, connected zones, one per peer.

3. If G_{cont} has an edge from $zone(i)$ to $zone(j)$, then there may be an application-level link from peer i to peer j. This yields a discrete graph G_{disc} on the vertex set $\mathcal{P}$ of peers.

In Chapter 2, we will define proportionate load balancing for DHTs that have a one-dimensional keyspace $\mathcal{K} = [0, 1) \mod 1$. With this type of DHT, peer zones are simply intervals. Peer i has label $id(i)$, and should keep track of (at least) its immediate predecessor $pred(i)$ and successor $succ(i)$, which are the peers such that

$$id(j) \in (id(pred(i)), id(succ(i))) \iff j = i. \tag{1.1}$$

We use the following definitions in this paper.

- Zone: $zone(i) = [id(i), id(succ(i))) \mod 1$
- Zone size: $z_i = id(succ(i)) - id(i) \mod 1$
- Relative zone size: $r_i = nz_i$, the ratio of z_i to the average zone size $\frac{1}{n}$

1.3.2 Data sharing

DHTs operate successfully because every peer knows to treat the location $id_x = f(x) \in \mathcal{K}$ as the authority on information about the data item x. Suppose $id_x \in zone(i)$. Fundamentally, DHTs need to provide just one operation: $lookup : \mathcal{K} \to \mathcal{P} : id_x \mapsto i$, which passes messages through the network to locate the peer in charge of item x.

Data items (files) enter the network by having a peer elect to share them. If peer o (the "owner") elects to share item x, it executes $lookup(f(x))$ to find the peer

i (the "intermediary") in charge of id_x. Peer o then gives its contact information to peer i.

When another peer s (the "seeker") wants to find item x, it also executes *lookup*($f(x)$) to find the peer i. Peer i reports peer o's contact information to peer s, so that s can obtain x from o. If no peer has elected to share the item x, peer i can definitively alert peer s to this fact.

In a network of n peers, many (though not all) DHTs provide search times of order $\mathcal{O}(\log_2(n))$ with $\mathcal{O}(\log_2(n))$ connections per peer. In Chapter 2, we will explain the *lookup* operation in detail for de Bruijn networks. These networks provide search times of order $\mathcal{O}(\log_2(n))$ with only $\mathcal{O}(k)$ connections per peer.

1.3.3 Maintenance of the network

New peers join the network by a process known as *bootstrapping*. A new peer must locate a peer already in the network, often via a list of IP addresses for computers known to have been connected in the past. The new peer locates a target through this intermediary, and consults with the target to divide that zone between them.

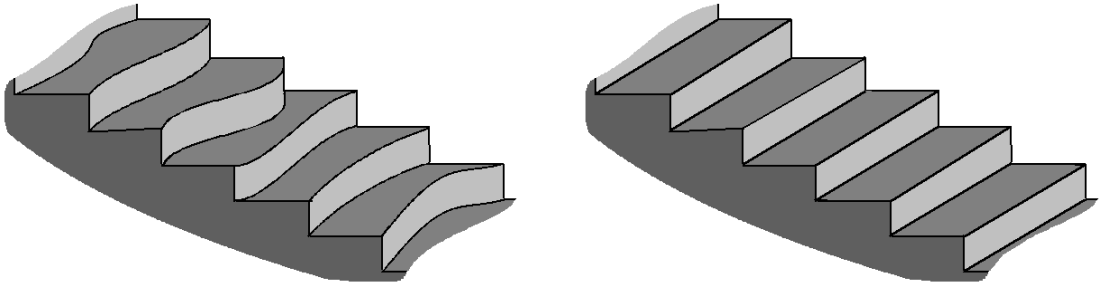
A peer can sign off of the network by transferring its zone to another peer with a consecutive region of the keyspace. If, instead, the peer simply drops out of the network without notifying any of its neighbors, a period of time is necessary for the network to completely recover. Ensuring satisfactory performance during this period is a subject of much study [44]. Typically, peers will re-advertise their available files periodically in order to assist in the recovery.

1.4 Crystal surfaces

An understanding of crystal surface evolution is important for modern nanoscale applications, such as nanowires and quantum dots [69]. A variety of surface features [81] can occur. For sufficiently low temperatures, crystal steps are the dominant feature on a crystal surface cut close to a plane of symmetry.

Specifically, a crystal lattice has certain planes of symmetry that depend on the lattice structure. A cubic lattice has three perpendicular planes of symmetry. The crystal can be cut at a "vicinal angle" $\phi << 1$ away from a plane of symmetry. At high temperatures, the surface will be statistically rough, but at temperatures T less than the "roughening temperature" T_R (which depends on the material), observable steps form. See Figure 1.2. The terraces have an average length $L \propto \cot(\phi)$.

Step bunching is a widely studied surface phenomenon. In the one-dimensional model, one equilibrium state is a train of uniformly-spaced steps. In contrast, under the step bunching instability, steps arrange themselves into widely-separated clusters of tightly-packed steps. This bunching has been observed in a variety of experimental systems, most commonly as a result of electromigration due to an applied direct current [70], but also as a result of material deposition onto the surface [65]. Such experimental observations have motivated interest in a theoretical understanding of the phenomenon.



(a) Crystal steps with curvature.

(b) Straight crystal steps.

Figure 1.2: At sufficiently low temperatures, steps are observable on a crystal surface cut close to a plane of symmetry. The steps are approximately one lattice constant high, and the terraces are typically tens or hundreds of lattice constants wide. Real crystal steps have kinks, islands, vacancies, and other imperfections, but they can be effectively modeled by smooth curved steps as in Figure 1.2(a). Under certain conditions, the steps can be considered to have zero curvature, as in Figure 1.2(b). This allows one-dimensional modeling of step positions.

Chapter 2

Exploiting heterogeneity

2.1 Introduction

Internet-enabled computers are heterogeneous, in that their capacities to process traffic vary widely [46, 29]. In a peer-to-peer situation, it can be considered equitable for the more capable peers to handle a larger fraction of the traffic [44]. We will equalize the *utilizations* of the peers, where utilization is defined in Section 2.3.1 as the fraction of a peer's capacity that is used by the P2P application.

We refer to the operation of equalizing utilizations as *proportionate load balancing* (PLB). This name distinguishes it from typical notions of load balancing, where the absolute amount of traffic to each peer is equalized. We will demonstrate analytically and through simulations that leveraging heterogeneity can improve the query path length and congestion in a particular class of distributed hash tables, the de Bruijn DHTs.

2.1.1 Related work

DHTs based on the de Bruijn architecture have been studied previously [16, 37, 25, 31, 17, 2]. We will present the basics of both *forward* and *reversed* de Bruijn networks. In particular, we offer an algorithm for routing in reversed de Bruijn networks ("reversed routing") that is novel for being provably correct, i.e., the destination is

always reached in absence of node failures. Broose [17] provided a reversed routing algorithm where the probability of failing to reach the desired destination was nonzero (but small under the assumption of homogeneous zone sizes). Distance Halving [37] defined its underlying architecture in a reversed manner, but only provided routing algorithms that relied (in whole or in part) on forward routing. D2B [16], Koorde [25], and ODRI [31] worked exclusively with forward networks. All of these employed an assumption of homogeneity in the partitioning of the keyspace.

Demand on the network's peers can differ for several reasons: (i) nonuniform numbers of data items may be assigned to the peers [21]; (ii) data items of varying popularities may receive highly nonuniform numbers of requests [28]; and (iii) the network structure may cause demand to vary inherently, as we will see in Section 2.3.2.2. Some prior approaches to coping with peer heterogeneity have divided the peers into a hierarchy of two or more levels based on their capacities [19]. Other approaches have employed a varying number of "virtual servers" [9, 22] for each peer. These approaches essentially use (i) to address heterogeneous peer capacities, but lack the real-time adaptability to deal with (ii) and (iii).

Our PLB implementation allows continuous variability in load distribution, and keeps the simplicity of a single-level DHT structure. Unlike a recent implementation [20], it allows continuous variability in peer capacities, and seeks to balance the ratio of these capacities to the actual traffic load at each peer, not simply to the amount of keyspace assigned to each peer. It also operates throughout the peers' lifetimes, not just during the process of joining or leaving the network.

Initially, proportionate load balancing seems similar to load balancing in het-

erogeneous processor networks [14]. While our implementation of PLB is similar to first-order diffusion schemes for processor networks, or the NbrAdjust operation for parallel databases [18], the problem setting is different. In P2P networks, traffic load is intimately tied to network structure. We modify the structure to balance queries at all stages along the search path, not just in the final stage of actually serving the response to a request.

2.1.2 DHTs and proportionate load balancing

As noted above, most protocols call for all peers to hold zones of approximately equal size [54], so that absolute traffic levels are nearly equal. However, by assigning more traffic to peers with higher capacity, PLB can balance their *relative* traffic levels. We will demonstrate how this is possible in certain DHTs, where the number of connections a peer has, and thus the amount of traffic it processes, depends on the size of its zone.

All DHTs that use the continuous-discrete framework of Section 1.3.1 satisfy the following assumption, which is critical for our implementation of proportionate load balancing.

(A1) The expected number of routing paths that pass through a given peer increases with the relative size of its zone.

Well-known examples where Assumption A1 holds include CAN [32] and ODRI [31].

The proofs in Section 2.2.2 will rely in part on the following assumption.

(A2) The peers occur in a random order in the keyspace, that is, $P(j = succ(i)) =$

$\frac{1}{n-1}$ for all $j \neq i$.

2.1.3 Overview

The remainder of this chapter is structured as follows. In Section 2.2 we define two classes of de Bruijn DHTs and analytically examine them under heterogeneous structure. In Section 2.3 we present a simple scheme for dynamically adapting DHTs (not necessarily de Bruijn) to exploit heterogeneity. In Section 2.4 we present simulation results.

2.2 Heterogeneous de Bruijn networks

Researchers have studied de Bruijn DHTs because they offer the potential for logarithmic diameter $\log_k(\#\text{peers})$ with constant degree k. We use them because they have node degree intimately tied to zone size [54], making them good candidates for proportionate load balancing.

Some arguments have been made against the practicality of de Bruijn DHTs [10]. First, heterogeneous partitioning of the keyspace prevents de Bruijn peers from having constant degree [10]. We embrace this disparity by providing the larger zones to peers with more capacity. Second, some peers in a DHT must have degree $\Omega(\log(\#\text{peers}))$ (unlike the peers in a classical de Bruijn network) if the network is to remain connected when half the peers fail [25]. By allowing varying peer degrees (Section 2.2.2.1) and using an average degree $k > 2$ ([10] footnote 1), we can achieve a balance between fault-tolerance and maintenance costs. Third, the path between two peers may be $\Omega(\#\text{peers})$ in the worst case. If the *ordering* of the peers in the keyspace is random, the probability of this is low; furthermore, heterogeneity improves the expected query path length (Section 2.2.2.2).

In this section, we will present the basics of both *forward* and *reversed* de Bruijn networks. The authors of Koorde and D2B provided more complete P2P protocols [25, 16] than we offer here. The authors of ODRI offered comparisons of de Bruijn with other DHT structures, including discussions of diameter, routing, and fault tolerance [31].

2.2.1 Forward and reversed de Bruijn networks

In this section, we first define continuous versions of forward and reversed de Bruijn graphs. We then show how to discretize these systems to obtain de Bruijn distributed hash tables.

2.2.1.1 de Bruijn graphs (continuous model)

The underlying space is the unit interval $[0, 1) \mod 1$. When considering graphs of degree k (integer ≥ 2), we will represent the values in base k, i.e.,

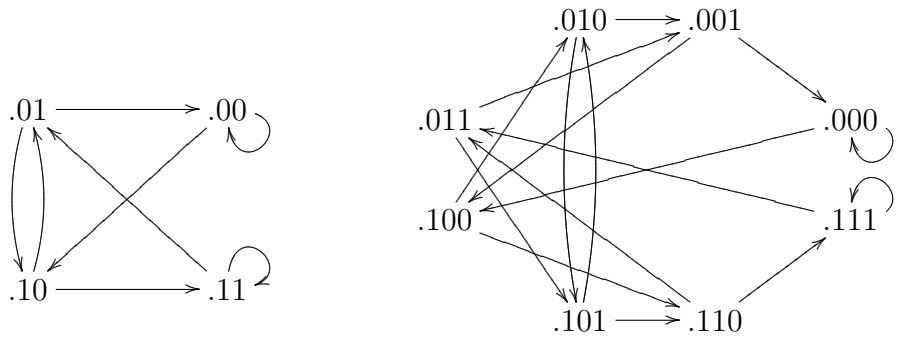
$$\boldsymbol{a} = .a_1a_2a_3\cdots = \sum_{i=1}^{\infty} \frac{a_i}{k^i} \in [0, 1) \tag{2.1}$$

where each $a_i \in \mathbb{Z}_k = \{0, 1, \ldots, k-1\}$. The value of k should be a reasonable number of connections for each peer to maintain, e.g., $k = 8$ or $k = 16$.

2.2.1.1.1 Forward de Bruijn graphs Algebraically, each point $\boldsymbol{a} \in [0, 1)$ has one outgoing edge, to the point $k\boldsymbol{a} \mod 1$. Symbolically, this is represented as an edge $.a_1a_2a_3\cdots \rightarrow .a_2a_3a_4\cdots$. We can thus refer to these as *left-shifting graphs*.

2.2.1.1.2 Reversed de Bruijn graphs Algebraically, each point $\boldsymbol{a} \in [0, 1)$ has k outgoing edges, to the points $(\boldsymbol{a} + x)/k \mod 1$, for $x \in \mathbb{Z}_k$. Symbolically, these are represented as edges $.a_1a_2a_3\cdots \rightarrow .xa_1a_2\cdots$. We can thus refer to these as *right-shifting graphs*.

2.2.1.1.3 Comments The outgoing neighbors of a vertex in a reversed de Bruijn graph are its incoming neighbors in the corresponding forward de Bruijn graph, and



(a) $k = D = 2$

(b) $k = 2$, $D = 3$

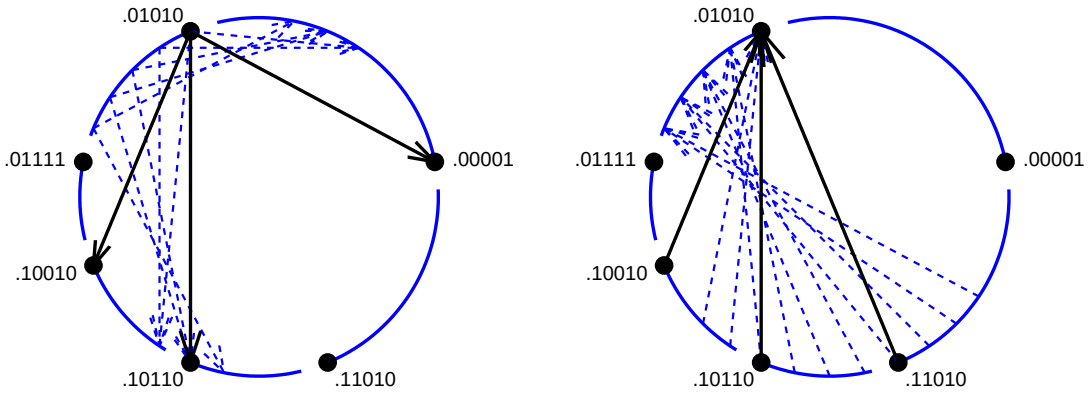
Figure 2.1: Two reversed de Bruijn graphs on finite spaces $\{.a_1a_2\cdots a_D : a_i \in \mathbb{Z}_k\}$. Each graph has k^D vertices; each vertex has in- and out-degree $k = 2$. The diameter of each graph is observably D.

vice versa.

In some applications, the underlying space may be taken as a collection of elements $\{.a_1a_2\cdots a_D : a_i \in \mathbb{Z}_k\}$ (D fixed) that is finite but still much larger (e.g., 2^{128}) than the set of peers might ever be. In this case, each vertex in the forward graph has k outgoing edges: $.a_1a_2\cdots a_{D-1}a_D \to .a_2a_3\cdots a_Dy$, $y \in \mathbb{Z}_k$. The *diameter* (the maximum length of any shortest path between vertices) of the resultant forward and reversed graphs is D. Fig. 2.1 shows two reversed de Bruijn graphs on small underlying spaces.

2.2.1.2 de Bruijn networks (discrete model)

As in Section 1.3, to create a distributed hash table, we assign each peer i an identifier $id(i)$ in accordance with Eq. (1.1). Peer i then controls the segment $zone(i) =$



(a) Outgoing edges

(b) Incoming edges

Figure 2.2: Here $k = 2$. The six large dots are the *id*s chosen by six peers, and the arcs depict the zone each peer is in charge of. Fig. 2.2(a) shows the outgoing edges for the peer with key $.010100\bar{0}$, while Fig. 2.2(b) shows the incoming edges for that same peer. The dashed lines are some of the relevant edges in the underlying reversed de Bruijn graph. The peer will have network-level connections to (or from) all zones matching at least one of these de Bruijn edges — the solid lines show these resulting connections in the P2P network.

$[id(i), id(succ(i)))$, and has links to all peers in control of zones where some underlying edge from $zone(i)$ terminates. See Fig. 2.2.

We now describe algorithms for routing in de Bruijn DHTs, and verify their correctness. We will suppose that peer i initiates a request for an item with key $\boldsymbol{b} = .b_1b_2b_3\cdots$, which lies in some other peer's $zone(j)$.

2.2.1.2.1 Forward routing Let d be such that $k^{-d} \leq z_i$. Then for any sequence $.b_1b_2b_3\cdots$, there exist $a_1, a_2, \ldots, a_d \in \mathbb{Z}_k$ such that $.a_1a_2\cdots a_db_1b_2b_3\cdots \in zone(i)$. At most two values of $(a_1, a_2, \ldots, a_d)$ are required, and peer i can calculate them. Following the sequence

$$\begin{array}{l} .a_1a_2\cdots a_{d-1}\ a_d \quad b_1 \quad b_2 \ \cdots \rightarrow \\ .a_2a_3\cdots \ \ a_d \quad b_1 \quad b_2 \quad b_3 \ \cdots \rightarrow \\ \qquad\quad \ddots \\ .a_db_1\cdots b_{d-2}b_{d-1}\ \ b_d\ \ b_{d+1}\cdots \rightarrow \\ .b_1\,b_2\cdots b_{d-1}\ \ b_d\ \ b_{d+1}b_{d+2}\cdots = \boldsymbol{b} \end{array} \tag{2.2}$$

of underlying edges yields a valid path of peer links through the network, starting at peer i and terminating at the appropriate peer j in d steps. The next hop is computable locally by each peer involved.

2.2.1.2.2 Reversed routing Let d be such that $(\boldsymbol{b} - k^{-d}, \boldsymbol{b} + k^{-d}) \subseteq zone(j)$.[1] (Peer i cannot determine d under our assumption of heterogeneity; we will deal with this difficulty soon.) Then any point $.b_1b_2\cdots b_dc_1c_2c_3\cdots$ lies in $(\boldsymbol{b} - k^{-d}, \boldsymbol{b} + k^{-d})$ and

[1] What happens if $\boldsymbol{b} = id(j)$? In the continuous case, this happens with probability 0, but can still be corrected by requiring peer $j-1$ to forward requests for $\boldsymbol{b}$ to peer j. In the case of a finite underlying space $\{.a_1a_2\cdots a_D : a_i \in \mathbb{Z}_k\}$, we may take $d = D$.

thus is covered by *zone*(j). Following any sequence

$$\begin{aligned}
&.c_1c_2\cdots\ c_d\ c_{d+1}c_{d+2}\cdots\rightarrow\\
&.b_dc_1\cdots c_{d-1}\ c_d\ c_{d+1}\cdots\rightarrow\\
&\qquad\ddots\\
&.b_2b_3\cdots\ c_1\quad c_2\quad c_3\ \cdots\rightarrow\\
&.b_1b_2\cdots\ b_d\quad c_1\quad c_2\ \cdots\in \mathit{zone}(j)
\end{aligned}\tag{2.3}$$

of underlying edges yields a valid path of peer links, starting at an arbitrary point $\boldsymbol{c}$ and terminating at the appropriate peer j in d steps.

Since the optimal value of d, $d_{\min}$, cannot be determined *a priori* by the source peer i, we must modify the procedure in Eq. (2.3). Peer i should perform *exponential polling* until $d_{\min}$ is exceeded, trying values $d = 1, 2, 4, 8, 16, \ldots$. The queries need not be returned to i between failed attempts. That is, using $\rightsquigarrow_d$ to denote the d-step procedure in Eq. (2.3), and starting at $\mathit{id}(i) = \boldsymbol{a} = .a_1a_2a_3\cdots$, the sequence

$$\begin{aligned}
\boldsymbol{a}\ &\rightsquigarrow_1\ .b_1\ \boldsymbol{a}\\
.b_1\boldsymbol{a}\ &\rightsquigarrow_2\ .b_1b_2\ b_1\boldsymbol{a}\\
.b_1b_2b_1\boldsymbol{a}\ &\rightsquigarrow_4\ .b_1b_2b_3b_4\ b_1b_2b_1\boldsymbol{a}\\
.b_1b_2b_3b_4b_1b_2b_1\boldsymbol{a}\ &\rightsquigarrow_8\ .b_1\cdots b_8\ b_1b_2b_3b_4b_1b_2b_1\boldsymbol{a}\\
&\ \ \vdots
\end{aligned}\tag{2.4}$$

reaches *zone*(j) in finite time. Letting $2^{h-1} < d_{\min} \leq 2^h$, the total number of steps is $\ell = 1 + 2 + 4 + \cdots + 2^h = 2^{h+1} - 1 \in [2d_{\min} - 1, 4d_{\min} - 1)$. The next hop is computable locally by each peer involved, as long as the current "stage" (1, 2, 4, 8, ...) is included with the message header.

Alternatively, peer i can send out multiple messages simultaneously, $\boldsymbol{a} \rightsquigarrow_1 .b_1\boldsymbol{a}$, $\boldsymbol{a} \rightsquigarrow_2 .b_1b_2\boldsymbol{a}$, ..., $\boldsymbol{a} \rightsquigarrow_{2^g} .b_1 \cdots b_{2^g}\boldsymbol{a}$. If no response is received, g should be increased. The value of g can be based on prior experience. The number of messages is the same as Eq. (2.4), but the response time is shorter because some are sent in parallel.

Of course, in either case, it is probably desirable to have a larger first attempt $\boldsymbol{a} \rightsquigarrow_{2^f} .b_1 \cdots b_{2^f}\boldsymbol{a}$, $2^f > 1$.

2.2.2 Heterogeneity in de Bruijn networks

We now offer analytical analyses of de Bruijn DHTs under heterogeneity. To our knowledge, this is the first such analysis for either forward or reversed networks. Section 2.2.2.1 shows that forward networks are much more liable to develop peers with only one outgoing neighbor, which are susceptible to disconnection from the graph. Section 2.2.2.2 demonstrates that high variance among zone sizes always reduces the query path length of reversed networks under reasonable assumptions, but that there are reasonable assumptions for forward networks that yield longer query path lengths. Finally, Section 2.2.2.3 discusses the use of caching as a complementary tool to PLB.

2.2.2.1 Number of neighbors

We begin with a simple property: node degree. If we were assuming homogeneity and roughly equal zone sizes, all peers would have in-degree and out-degree very near k [31]. However, as the zone sizes vary in our heterogeneous networks, the degrees

become non-uniform.

2.2.2.1.1 Forward and reversed neighbors

Theorem 2.1. *Suppose peer i has relative zone size r_i in either a forward or reversed de Bruijn DHT with n peers.*

- *Averaging over forward de Bruijn DHTs, peer i expects to have $k_i^{fwd} = 1 + kr_i$ outgoing links to other peers.*
- *Averaging over reversed de Bruijn DHTs, peer i expects to have $k_i^{rev} = k + r_i$ outgoing links to other peers.*

Note that k_i^{fwd} and k_i^{rev} are independent of n.

Proof. We begin with a general fact: If a DHT has n arbitrarily-ordered peers, and $[a, b)$ is an interval in the keyspace $[0, 1)$, then the expected number of distinct peers in charge of some or all of $[a, b)$ is $n(b - a) + 1$. Since the n peers are arranged in no particular order by Assumption A2, the expected number of peers that lie *inside* (a, b) is $n(b - a)$. Each of these peers is in charge of a portion of $[a, b)$. Additionally, there is another peer whose key either is a or immediately precedes a; this peer is in charge of the first portion of $[a, b)$.

Now recall that $zone(i) = [id(i), id(succ(i)))$.

In the forward case, $zone(i)$ maps to $[kid(i), kid(succ(i)))$, an interval of length kz_i. Then peer i expects to connect to $n \cdot kz_i + 1 = 1 + kr_i$ peers.

In the reversed case, $zone(i)$ maps to k evenly-spaced intervals $[\frac{id(i)}{k}, \frac{id(succ(i))}{k}) + \frac{x}{k}$, $x \in \mathbb{Z}_k$. Each of these intervals has length $\frac{1}{k} z_i$. Then peer i expects to connect to

$k(n \cdot \frac{1}{k} z_i + 1) = k + r_i$ peers. □

In the reversed case, even if r_i is very small, $k_i^{\text{rev}} \approx k$. Therefore, even a very weak peer will have alternative paths to route a query through if one of its neighbors fails. In the forward case, if r_i is very small, then $k_i^{\text{fwd}} \approx 1$ and the peer will be susceptible to disconnection from the graph ([54], Section 2).

The expected number of links for a peer with an average-sized zone ($r_i = 1$) is $k+1$, rather than k, which is the number of links in a uniform de Bruijn graph. This is because the uniform graph is unstable with respect to all of its zone boundaries lining up.

We note that the number of incoming links in a forward DHT is the number of outgoing links in the corresponding reversed DHT, and vice versa. However, incoming links do not directly help with routing around a failed neighbor.

2.2.2.1.2 Cost of PLB In Section 2.3.3, we will discuss the number of peers who must be notified when peer i participates in our implementation of PLB. In the worst case, all the in- and out-neighbors of either peer $pred(i)$ or i must be notified of a change in $id(i)$.

By Theorem 2.1, we expect the number of neighbors that must be contacted (in either the forward or reversed case) to be $\leq k^{\text{fwd}} + k^{\text{rev}} = (k+1)(\max\{r_i, r_{pred(i)}\} + 1)$.

2.2.2.2 Query path lengths

If we were assuming homogeneity, so all n peers had roughly equal zone sizes, then we would have $d \sim \log_k n$ for both the forward and reversed cases. Thus the number

of routing steps would be $\mathcal{O}(\log_k n)$, as in [37, 25, 16, 17, 31]. We note that linear polling in the reversed case (trying values $d = 1, 2, 3, 4, 5, \ldots$) would have resulted in $\mathcal{O}((\log_k n)^2)$ steps.

In this paper we assume heterogeneity of peers and zone sizes. The average number of routing steps will depend on these quantities:

- $\boldsymbol{r} = (r_1, \ldots, r_n)$, the relative zone sizes.
- $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_n)$, the relative rates at which the peers *request* items. Reasonable values: $\boldsymbol{\rho} = \mathbf{1}$ (uniform rates); $\boldsymbol{\rho} = (cap(1), \ldots, cap(n))$, where $cap(i)$ is the capacity[2] of the i^{th} peer (faster peers make more queries).
- $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n)$, the relative rates at which the peers *supply* items. Reasonable value: $\boldsymbol{\sigma} = \boldsymbol{r}$ (number of data items controlled is proportional to zone size). Hot spots (Section 2.2.2.3) may yield a $\boldsymbol{\sigma}$ that is essentially random if caching is not employed.

We will employ the fact that the weighted geometric mean

$$\text{WGM}(\boldsymbol{r}; \boldsymbol{\alpha}) = \left(\prod_{i=1}^{n} r_i^{\alpha_i} \right)^{1/\sum_{i=1}^{n} \alpha_i} \tag{2.5}$$

is always less than or equal to the weighted arithmetic mean

$$\text{WAM}(\boldsymbol{r}; \boldsymbol{\alpha}) = \frac{\sum_{i=1}^{n} \alpha_i r_i}{\sum_{i=1}^{n} \alpha_i} \tag{2.6}$$

with equality iff all r_i are equal. The proof is via Jensen's inequality.

It will be useful to have an estimate of $\log_k \text{WGM}(\boldsymbol{r}; \boldsymbol{r})$. Let $f(x) = x \log_k x$; f is infinitely differentiable and concave up on the interval $I = [r_{\min}, r_{\max}] \subset (0, \infty)$.

[2] A peer's capacity is the rate at which it can process P2P traffic; see Section 2.3.1.

Taylor's Theorem says that for any points $a, a+h \in I$ and some point ξ_h between a and $a+h$,

$$f(a+h) = f(a) + f'(a)h + \frac{1}{2}f''(\xi_h)h^2 \tag{2.7}$$

$$= a\log_k a + \log_k(ea)h + \frac{1}{2\ln k}\xi_h^{-1}h^2 \tag{2.8}$$

$$\geq a\log_k a + \log_k(ea)h + \frac{1}{2\ln k}r_{\max}^{-1}h^2. \tag{2.9}$$

The mean $\frac{1}{n}\sum_{i=1}^n r_i = 1$, so let $s_i = r_i - 1$. Then $\sum_{i=1}^n s_i = 0$. Finally,

$$\log_k \mathrm{wGM}(\boldsymbol{r};\boldsymbol{r}) = \frac{1}{n}\sum_{i=1}^n r_i \log_k r_i \tag{2.10}$$

$$= \frac{1}{n}\sum_{i=1}^n f(1+s_i) \tag{2.11}$$

$$\geq \frac{1}{n}\left(0 + \log_k(e)s_i + \frac{1}{2\ln k}r_{\max}^{-1}s_i^2\right) \tag{2.12}$$

$$= \frac{1}{2\ln k}r_{\max}^{-1}\frac{1}{n}s_i^2 \tag{2.13}$$

$$= \frac{1}{2\ln k}r_{\max}^{-1}\mathrm{Var}\{r_i\}. \tag{2.14}$$

2.2.2.2.1 Forward query path lengths

Theorem 2.2. *Consider a forward de Bruijn DHT with n peers.*

- *The length of the path from peer i to any given destination is*

$$\ell_j^{fwd} = \lceil \log_k n - \log_k r_i \rceil. \tag{2.15}$$

- *Taking a weighted average over all peers i in the network gives the the average query path length*

$$\bar{\ell}^{fwd} = \log_k n - \log_k \mathrm{wGM}(\boldsymbol{r};\boldsymbol{\rho}) + C \tag{2.16}$$

for some $0 < C < 1$.

Proof. From Section 2.2.1.2.1, the length of the path from i to any $\boldsymbol{b}$ is

$$d_i = \lceil \log_k \left(z_i^{-1} \right) \rceil = \log_k n - \log_k r_i + \epsilon_i, \tag{2.17}$$

for some $0 \leq \epsilon_i < 1$. For the second part of the theorem, we have

$$\bar{\ell}^{\text{fwd}} = \left(\sum_{i=1}^{n} \rho_i (\log_k n - \log_k r_i + \epsilon_i) \right) \left(\sum_{i=1}^{n} \rho_i \right)^{-1} \tag{2.18}$$

$$= \log_k n - \log_k \text{wGM}(\boldsymbol{r}; \boldsymbol{\rho}) + \text{wAM}(\boldsymbol{\epsilon}; \boldsymbol{\rho}) \tag{2.19}$$

yielding Eq. (2.16). □

If all peers have the same request rate ($\boldsymbol{\rho} = \mathbf{1}$), then

$$\log_k \text{wGM}(\boldsymbol{r}; \mathbf{1}) \leq \log_k \text{wAM}(\boldsymbol{r}; \mathbf{1}) = \log_k 1 = 0, \tag{2.20}$$

so the average query path length is at least $\log_k n$. On the other hand, if peers with larger zones also have larger ρ_i (e.g., $\boldsymbol{\rho} = (cap(1), \ldots, cap(n))$ and some form of PLB has been performed), then $\text{wAM}(\boldsymbol{r}; \boldsymbol{\rho}) > 1$ and the average query path length can be less than $\log_k n$.

2.2.2.2.2 Reversed query path lengths

We begin with a lemma.

Lemma 2.1. *Consider a peer j in a reversed de Bruijn DHT with n peers. If $\boldsymbol{b} \in zone(j)$, let $d_{\boldsymbol{b}}$ be minimal such that $.b_1 b_2 \cdots b_{d_{\boldsymbol{b}}} c_1 c_2 \cdots \in zone(j)$ for all $\boldsymbol{c}$. Then the expected value of $d_{\boldsymbol{b}}$ over $zone(j)$ is*

$$\bar{d}_j = \log_k n - \log_k r_j + C_k^{(j)} \tag{2.21}$$

where $0 < C_k^{(j)} < 2 + \frac{1}{\log_2 k} + \frac{1}{k-1} \leq 4$.

Proof. We will perform the computation assuming $\boldsymbol{b}$ is in the left half of *zone*(j); the argument for the right half is symmetric. Recall from Section 2.2.1.2.2 that $d_{\boldsymbol{b}}$ is minimal such that $(\boldsymbol{b} - k^{-d_{\boldsymbol{b}}}, \boldsymbol{b} + k^{-d_{\boldsymbol{b}}}) \subseteq$ *zone*(j). Thus $\boldsymbol{b} - id(j) \in [k^{-d_{\boldsymbol{b}}}, k^{-d_{\boldsymbol{b}}+1})$.

Let m be such that $id(j) + [k^{-m}, k^{-m+1})$ contains the midpoint of *zone*(j); we find $m = \lceil -\log_k \frac{z_j}{2} \rceil$. Then $d_{\boldsymbol{b}}$ is a step function on its half of *zone*(j), given by

$$d_{\boldsymbol{b}} = \begin{cases} d & \text{if } d > m \text{ and } \boldsymbol{b} - id(j) \in [k^{-d}, k^{-d+1}), \\ \\ m & \text{if } \boldsymbol{b} - id(j) \in [k^{-m}, \frac{z_j}{2}]. \end{cases} \tag{2.22}$$

The expected value of $d_{\boldsymbol{b}}$ is found by weighted average:

$$\bar{d}_j = \left[\sum_{d=m+1}^{\infty} d \left(k^{-d+1} - k^{-d} \right) \right. \tag{2.23}$$

$$\left. + m \left(\frac{z_j}{2} - k^{-m} \right) \right] / \frac{z_j}{2} \tag{2.24}$$

$$= \left[\frac{k^{-m+1}}{k-1} + m \frac{z_j}{2} \right] / \frac{z_j}{2} \tag{2.25}$$

$$= \lceil \log_k n - \log_k r_j + \log_k 2 \rceil + \frac{k^{-m+1}}{(k-1)\frac{z_j}{2}}. \tag{2.26}$$

We have used Gabriel's truncated staircase, $\sum_{i=m+1}^{\infty} ir^i = \frac{mr^{m+1}}{1-r} + \frac{r^{m+1}}{(1-r)^2}$ for $0 < r < 1$.

By definition of m, $k^{-m+1} \in (\frac{z_j}{2}, k\frac{z_j}{2}]$. Thus the final term satisfies $\frac{k^{-m+1}}{(k-1)\frac{z_j}{2}} \in (\frac{1}{k-1}, \frac{k}{k-1}] \subset (0, 2]$, yielding Eq. (2.21). □

We now proceed to the theorem that is the main result of this subsection.

Theorem 2.3. *Consider a reversed de Bruijn DHT with n peers.*

- *Averaging over all destinations* $\boldsymbol{b}$ *chosen uniformly at random from* zone(j)*, the expected length of the path from any given peer to* $\boldsymbol{b}$ *is*

$$\ell_i^{rev} \approx \mu \cdot \left(\log_k n - \log_k r_j + C_k^{(j)} \right) - 1, \tag{2.27}$$

where $\mu \leq 4$.

- *Taking a weighted average over all destination zones* j *in the network gives the the average query path length*

$$\bar{\ell}^{rev} \approx \mu \cdot (\log_k n - \log_k \text{wGM}(\boldsymbol{r}; \boldsymbol{\sigma}) + C_k)\,. \tag{2.28}$$

Here $C_k^{(j)}$ *and* C_k *are positive values bounded by* $2 + \frac{1}{\log_2 k} + \frac{1}{k-1} \leq 4$.

Proof. In Section 2.2.1.2.2, we found that the number of steps $\bar{\ell}_j^{\,\text{rev}} = 2^{\lceil \log_2 \bar{d}_j \rceil + 1} - 1 < 4\bar{d}_j - 1$, yielding the first part of the theorem with $\mu = 4$.

For the second part of the theorem, we average over all destination zones j,

$$\bar{\ell}^{\text{rev}} = \left(\sum_{j=1}^{n} \sigma_j \bar{\ell}_j^{\,\text{rev}} \right) / \sum_{j=1}^{n} \sigma_j \tag{2.29}$$

$$\approx \left(\sum_{j=1}^{n} \sigma_j \mu \cdot (\log_k n - \log_k r_j + C_k^{(j)}) \right) \left(\sum_{j=1}^{n} \sigma_j \right)^{-1} \tag{2.30}$$

$$= \mu \cdot \left(\log_k n - \log_k \text{wGM}(\boldsymbol{r}; \boldsymbol{\sigma}) + \text{wAM}(\boldsymbol{C}_{\boldsymbol{r}}^{(k)}; \boldsymbol{\sigma}) \right) \tag{2.31}$$

yielding Eq. (2.28). □

In practice, μ will depend on the distribution of the $\bar{d}_j$. Letting $h_j = \lceil \log_2 \bar{d}_j \rceil$, we have $\mu = 2^{h_j+1}/\bar{d}_j$. If $\bar{d}_j$ were uniformly distributed in $(2^{h_j-1}, 2^{h_j}]$, then $E(\mu) = 4 \ln 2 \approx 2.8$. If $\bar{d}_j$ were distributed as $1/x$ in $(2^{h_j-1}, 2^{h_j}]$ (such as by Benford's Law), then $E(\mu) = 2/\ln 2 \approx 2.9$.

If received requests are proportional to zone size ($\boldsymbol{\sigma} = \boldsymbol{r}$), then $\log_k \text{wGM}(\boldsymbol{r}; \boldsymbol{r}) > 0$, and the average query path length will be less than $\log_k n$. In particular, we saw above that $\log_k \text{wGM}(\boldsymbol{r}; \boldsymbol{r}) \geq \frac{1}{2 \ln k} r_{\max}^{-1} \text{Var}\{r_i\}$, so high variance among zone sizes has a beneficial effect on the average query path length.

2.2.2.3 Caching

Dynamic caching is an important tool for relieving *hot spots* [28] — items with extreme popularity that cause acute congestion for the peers that host them. Caching is a process for replicating such items at multiple nodes, to distribute the workload.

Caching is compatible with PLB and provides a complementary benefit. Since PLB is based on the actual amount of traffic processed by each peer, it is able to deal with hot spots to some degree, but caching can be faster and more effective. On the other hand, caching is not designed to alter the structure of the network in order to take advantage of heterogeneity.

2.2.2.3.1 Forward caching Forward de Bruijn DHTs permit a simple caching scheme [17, 37]. A hot spot at location $\boldsymbol{b} \in [0, 1)$ should be replicated at the k predecessor locations $\frac{\boldsymbol{b}+x}{k} = .xb_1b_2\cdots$, $x \in \mathbb{Z}_k$. Then a request for $\boldsymbol{b}$ will always pass through one of these caches, each with equal probability. The replication can be repeated as necessary.

2.2.2.3.2 Reversed caching The analogous scheme does not work for reversed de Bruijn DHTs, since there is only one predecessor location $k\boldsymbol{b} \mod 1$ to the hot spot $\boldsymbol{b}$. A routing-independent caching scheme such as [28] could be used. Alternatively, a peer that routes more than a certain number of requests for item $\boldsymbol{b}$ can begin caching the item, until some set of peers together controlling the interval $.b_{\text{crit}}\cdots b_{2^h}b_1\cdots b_{2^{h-1}}\cdots b_1b_2b_1\boldsymbol{c}$, for all $\boldsymbol{c} \in [0, 1)$, distributes the workload to a satisfactory level. (Here b_{crit} is the first value for which the workload is distributed

satisfactorily.)

2.3 Proportionate load balancing

In this section, we present an implementation of proportionate load balancing in the context of continuous-discrete DHTs (Section 1.3), not necessarily de Bruijn. Using only information about itself and the two peers adjacent to it in the keyspace $[0, 1)$, each peer will increase or decrease its zone size. Assumption 1 indicates that, over time, this should equalize all peers' utilizations. We stress that in simulations, we will have all peers alter their zone sizes simultaneously on clock ticks, but in reality a peer can alter its zone size more or less often depending on its individual needs.

2.3.1 Definitions

The *capacity* of a peer ($cap(i)$) is the rate at which it is able to process traffic for the P2P application. Generally, the outbound traffic rate will be the limiting factor [29]. Plots of estimated bottleneck bandwidths for peers in the Napster and Gnutella [32] networks are given in Section 3.1 of [46]; these plots show that there are frequently orders of magnitude differences between peer capacities.

Capacity is difficult to measure from the outside, but a peer can keep track of its own capacity with relative ease. To discourage under-reporting of capacity, peers might limit the rate at which they transfer files to peer i to be some multiple of i's published capacity. This does not affect the implementation of PLB. Over-reporting is a more complex issue, but not an insurmountable one. The predecessor and successor of a newly joined peer can restrict the growth of its zone by modifying the algorithm of Section 2.3.2.2. If a misbehaving peer does obtain a large zone and fails to pass on

all its messages, it can be treated like a failed peer as in Section 2.3.2.1.

The *load* of a peer ($load(i)$) is the rate at which the network asks it to process P2P traffic. A peer can keep track of this value, perhaps as an exponentially weighted moving average. Load is a dynamic quantity that can depend on relative zone size (Theorem 2.1), global network structure (Fig. 2.3), and key popularity (Section 2.2.2.3) in a complicated way.

The *utilization* of a peer ($util(i)$) is a dimensionless quantity equal to the ratio of its load and capacity: $util(i) = \frac{load(i)}{cap(i)}$. In order for a peer to keep up with its requests, its utilization should remain below 1.

2.3.2 Implementation

In our implementation, peer i is only required to keep track of limited information:

For the DHT: Its zone $zone(i) = [id(i), id(succ(i)))$ and the data items that map to it. Links to its neighbors in the underlying graph, for routing. Links to peers $pred(i)$ and $succ(i)$, for consistency.

For PLB: The capacity, load, and utilization of itself and peers $pred(i)$ and $succ(i)$.

2.3.2.1 Arriving and departing peers (churn)

When a new peer wants to join a P2P network, typically it must know how to contact some existing peer, who helps the new peer find its proper neighbors. Several methods for assigning a zone to a new peer were analyzed in [54]. In *single-point random-split*,

a randomly chosen point in the keyspace becomes the new peer's *id*. In *single-point center-split*, the peer that controls a randomly chosen point gives exactly half of its zone to the new peer. In *multi-point center-split* (random or semi-deterministic), a fixed number of points are sampled; the one belonging to the largest zone becomes the target of a center-split. As an example of the analyses, we have that under single-point center-split the largest and smallest zones are $\Omega(\log n)$ times larger and $\Omega\left(2^{\sqrt{2\log_2 n}}\right)$ times smaller than average, with high probability.

The center-split methods are readily generalizable to *proportional-split* methods, where the target zone is split, not in half, but in proportion to the capacities of the new and existing peers[3]. The intent is to equalize utilizations rather than zone sizes. In a multi-point method, the point belonging to the peer with the highest utilization becomes the target.

In our simulations, we will use *single-point proportional-split*, which works as follows: If the network has $n-1$ peers, then the n^{th} peer chooses a random point $id \in \mathcal{K}$, which will lie in the zone of some existing peer i. Peer n then receives the identifier $id(n) \in zone(i)$ such that $\frac{z_n}{z_i} = \frac{cap(n)}{cap(i)}$. (Alternatively, $id(n)$ could be chosen based on the load distribution function $g_i(x)$ discussed in Section 2.3.2.2.2.)

When peer i leaves the P2P network, its adjacent peers $pred(i)$ and $succ(i)$ should take control of its zone. If peer i announces its departure, the adjacent peers can split $zone(i)$ in proportion to their capacities, and take responsibility for the data items previously managed by i. If peer i fails or simply drops out, some information

[3]More generally, it may be split in proportion to some function of the capacities of the new and existing peers, depending on the nature of the relationship in Assumption 1.

may be temporarily lost. The Chord coping mechanism [49] is applicable to our DHTs; an analysis of coping mechanisms is beyond the scope of this paper. Typically, periodic re-publication of data items is required.

2.3.2.2 Existing peers

Proportional-split joining methods are insufficient to achieve balanced utilizations. Part of this is due to the unpredictable nature of peer departures, but at least as important is the fact that local DHT traffic load depends on the global structure of the network. Fig. 2.3 shows the distribution of load across a particular peer's zone for an actual simulation, a reversed de Bruijn network in which the request rates $\boldsymbol{\rho}$ (Section 2.2.2.2) were all equal and destinations $\boldsymbol{b} \in \mathcal{K}$ were chosen with uniform probability.

The dependence of load on network structure means that it is not practical (or perhaps even possible) to find an exact solution to the problem of utilization balancing, even if the set of peers is held constant. This is in contrast to processor networks or parallel databases, where an exact solution can be found as the result of a global linear equation, and the challenge is to find locally-computable approximations to this global solution.

Our implementation of PLB has similarities to the NbrAdjust operation of [18] and the first-order diffusion schemes described in [14], but we also address the non-uniformity depicted in Fig. 2.3, and we incorporate fail-safes to make sure that zone sizes do not change too quickly. Pseudocode for the local implementation of

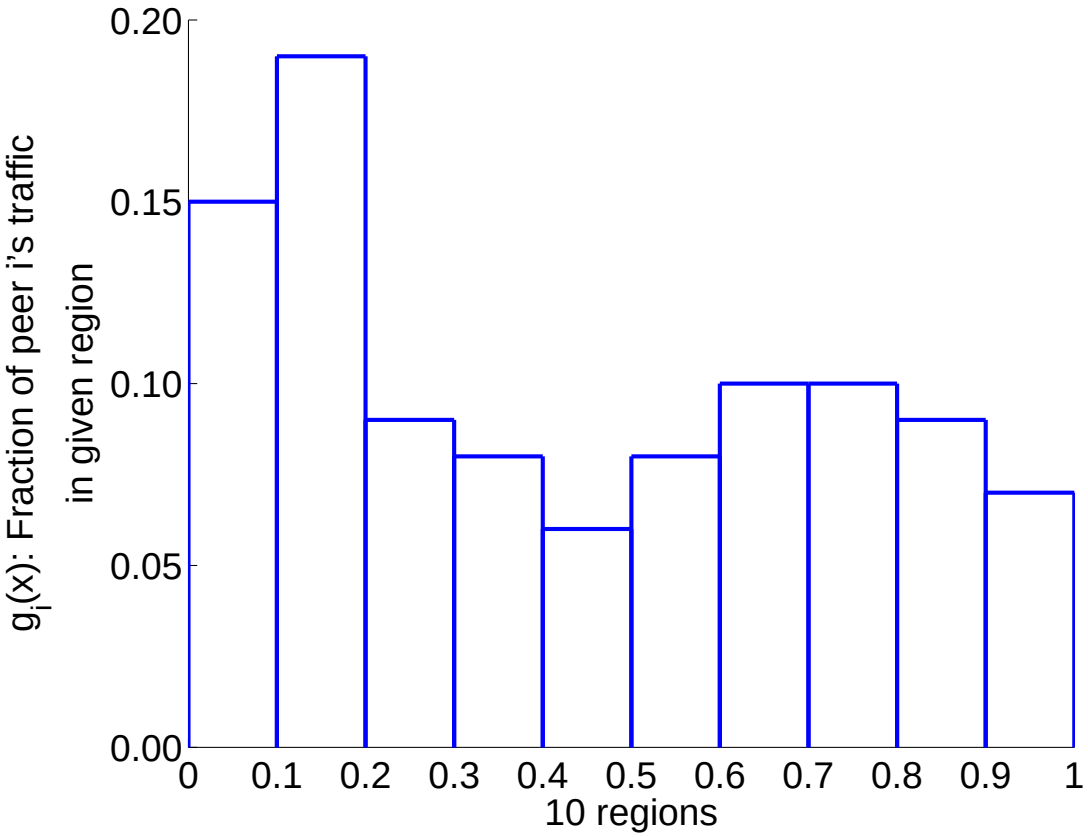


Figure 2.3: Traffic load is not distributed uniformly within zones, even under uniform selection of sources and destinations. This graph depicts the distribution of load across a particular peer's zone in a reversed de Bruijn network. In the simulation, 19% of the traffic went through the second portion of the zone, and only 6% through the fifth portion. Call this step function $g_i(x)$ for peer i.

PLB is given in Fig. 2.4.

2.3.2.2.1 The Balance operation Lines 1 and 2 of BALANCE assign h to be the peer with higher utilization, and ℓ the peer with lower utilization. Line 3 tests whether $util(h)$ is sufficiently higher than $util(\ell)$ for the GIVEZONE operation to be performed in line 4.

PLB requires two tolerance values, $t_{\min}, t_{\text{diff}} \in [0, 1)$. Their meaning is as follows. If $util(\ell) = 0$, we change $id(i)$ if $util(h) - util(\ell) \geq t_{\min}$. On the other hand, if the $util(\ell) = 1$, we change $id(i)$ if $util(h) - util(\ell) \geq t_{\min} + t_{\text{diff}}$. For other values of $util(\ell)$, we interpolate linearly. Smaller values of $t_{\min}$ and t_{diff} cause the network to take longer to settle down, but a more even distribution of utilizations is achieved.

2.3.2.2.2 The GiveZone operation Line 1 of GIVEZONE determines the fraction of peer h's load that should be transferred to peer ℓ so that $util(h)' = util(\ell)'$. Solving the equation $\frac{load(\ell)+f \cdot load(h)}{cap(\ell)} = \frac{(1-f) \cdot load(h)}{cap(h)}$ for f yields $f = \frac{1-util(\ell)/util(h)}{1+cap(h)/cap(\ell)}$. Because i communicates with both $pred(i)$ and $succ(i)$, $zone(i)$ may be expanded or contracted on both ends simultaneously, so we use the modified value $loadfrac(h, \ell) = \frac{1}{2}f$.

Recall that the distribution of traffic load is nonuniform within zones. In line 2, peer h determines the fraction $zonefrac(h, \ell)$ of its zone (adjacent to $id(i)$) that corresponds to $\alpha \cdot loadfrac(h, \ell)$ of its load. (The parameter $\alpha \in (0, 1]$ is tunable; we use $\alpha = 1$.) In our simulations, we had each peer keep a histogram of the activity in $s = 10$ equal sub-intervals of its zone, generating a step function g_h like in Fig. 2.3. If $h = i$, the integral is $G_{h,\ell}(x) = \int_0^x g_h(t)dt$. If $h = pred(i)$, the integral is $G_{h,\ell}(x) =$

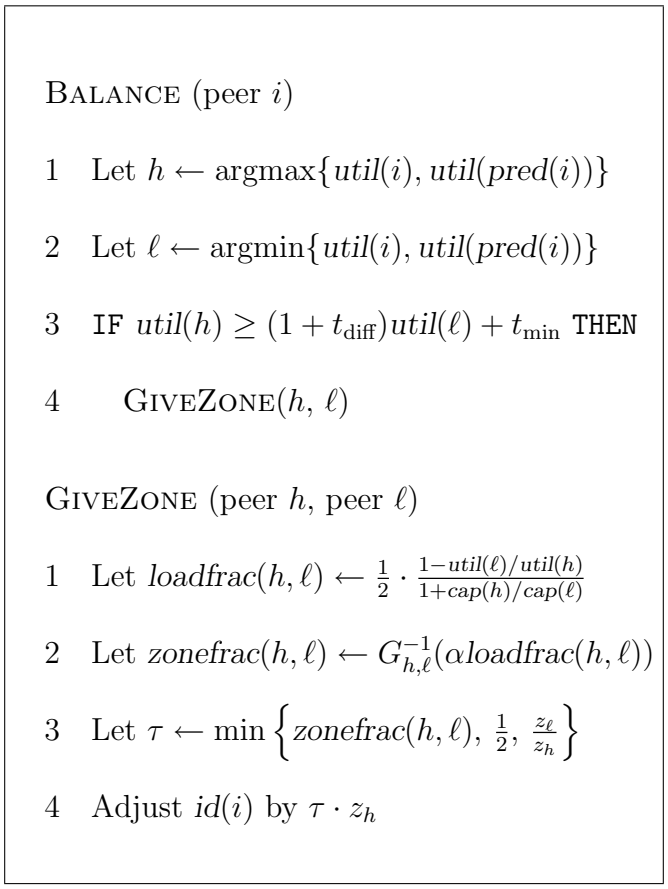
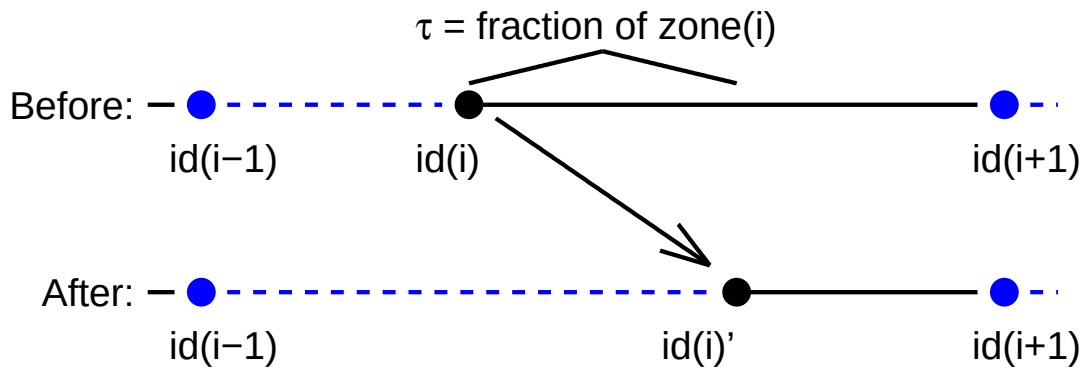
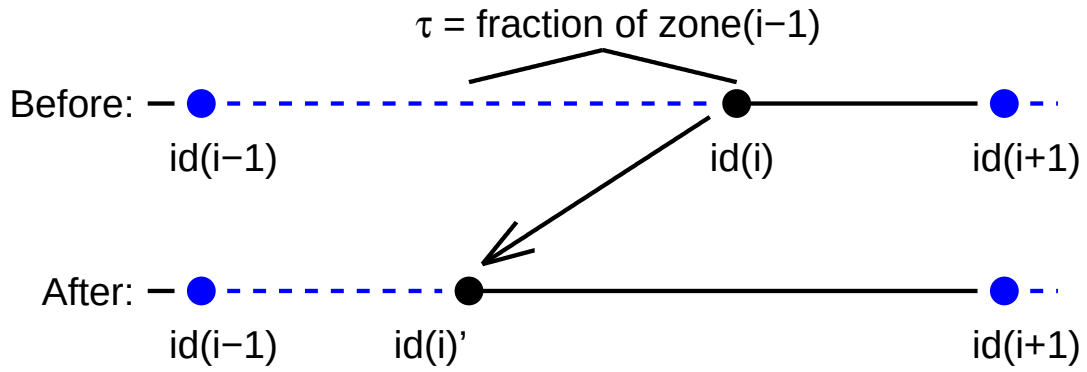


Figure 2.4: Pseudocode for PLB (Section 2.3.2.2). BALANCE describes the conditions under which PLB is performed. Here $t_{\min}, t_{\text{diff}} \in [0, 1)$. GIVEZONE determines the new boundary $id(i)'$ between $zone(i)$ and $zone(pred(i))$. In simulations, we have all peers execute BALANCE simultaneously on clock ticks, but in reality it can be executed at any time depending on each peer's needs.



(a) Contracting *zone*(i)



(b) Expanding *zone*(i)

Figure 2.5: In Fig. 2.5(a), peer i contracts its zone by increasing $id(i)$ to $id(i)'$.

In Fig. 2.5(b), peer i expands its zone by decreasing $id(i)$ to $id(i)'$.

$\int_{1-x}^{1} g_h(t)dt$. Notice that $G_{h,\ell}^{-1}$ is easy to calculate since g_h is a step function.

Line 3 ensures that $z'_h = z_h - \tau z_h \geq \frac{1}{2} z_h$ and $z'_\ell = z_\ell + \tau z_h \leq 2z_\ell$, so the zone sizes do not adjust too quickly. It is easy to repeat the BALANCE operation, but it is difficult to recover from a suddenly over-utilized peer.

In line 4, $id(i)$ is adjusted by the proper fraction of z_h. It is increased if $h = i$, and decreased if $h = pred(i)$. See Fig. 2.5.

2.3.3 Data and rewiring costs

When peer i changes its $id(i)$ based on PLB, the interval between the old $id(i)$ and the new $id(i)'$ changes ownership as discussed in the last section. The previous owner is h (i or $pred(i)$, the peer with higher utilization), while the new owner is the other peer ℓ. There is a *data cost* associated with moving information about data items from h to ℓ to maintain the proper definition of their zones. This cost is proportional to the relative size of the interval, τr_h, that was transferred.

There is also a *rewiring cost* for adjusting neighbor connections maintain the structure of the DHT. When $id(i)$ is changed, links to/from peer h may be removed, and links to/from peer ℓ may be added. For each link involving h, we check whether the link should be removed from h, and whether a corresponding link should be added for ℓ. (These are separate questions because both h and ℓ may simultaneously link to the same peer.) Any links not involving h will be unchanged. A special case arises if h has a (virtual) link to itself; then we must check all four combinations of old and new interval owners: $h \to h$, $\ell \to h$, $h \to \ell$, and $\ell \to \ell$.

We summarize these results in a theorem.

Theorem 2.4. *Consider continuous-discrete DHTs as in Section 1.3. Suppose* $id(i)$ *is adjusted by* $\tau \cdot z_h$ *in the* GiveZone *operation.*

- *Averaging over all distributions of data items, the expected data cost is*

$$\tau z_h \left(\sum D_j \right) = \tau r_h \bar{D}. \tag{2.32}$$

  Here D_j *is the number of data items stored by peer* j*, and* $\bar{D} = \frac{1}{n} \sum_{j=1}^{n} D_j$ *is the average number of stored (and thus of shared) data items per peer.*

- *The rewiring cost is at most* $2(\textit{out-degree of } h) + 2(\textit{in-degree of } h) + 4$.

2.4 Simulation experiments

We now use simulations to evaluate the performance of our proportionate load balancing implementation. These simulations are all of P2P networks based on various types of de Bruijn DHTs.

One key feature of a network, which we will report on for each experiment, is its maximum utilization. The maximum utilization is the greatest utilization experienced by any peer in the network. A large maximum utilization (near 1) implies that some peer is near capacity, and will start dropping queries if the request rates increase. A small maximum utilization (near 0) implies that all peers are experiencing relatively low load, and the request rate can be increased substantially without overloading the network.

2.4.1 Setup

We perform iterations of PLB on static and dynamic networks of various sizes, ranging from 64 to 8192 peers. During one *iteration*, all utilizations are computed, and then each peer performs the BALANCE operation (Section 2.3.2.2) once.

The capacity of each peer is assumed to be constant during the simulation. Capacities are selected from a probability distribution that approximates the estimated bottleneck upstream bandwidths of peers in the Gnutella network [46]. This distribution is described in Table 2.1. Capacities selected from this distribution will be highly variable, indicated by the standard deviation of F being larger than the mean.

The *request rate* ρ_i of each peer (Section 2.2.2.2) is also assumed to be constant

Table 2.1: Statistical properties of our capacity distribution.

Property	Value
CDF	$F(x) = \frac{1}{2.3} \log_{10} \frac{x}{25}$
Mean	940 Kbps
Standard deviation	1.5 Mbps
Median	350 Kbps
Range	25 Kbps $\leq x <$ 5 Mbps

during the simulation. We test two reasonable values of $\boldsymbol{\rho}$: $\mathbf{1}$ and $(cap(1), \ldots, cap(n))$ (Section 2.2.2.2). In the simulations presented here, the *supply rate* σ_i of each peer is taken to be proportional to its relative zone size r_i.[4]

We estimate the utilization of each peer by tracing paths from random source peers to random destination keys. The number of times a peer is traversed during this process allows us to estimate its relative traffic load. An average of 100 paths per source peer were traced; results were similar for different choices of the traced paths.

We tested various protocol parameters, and found no unpredictable results. The parameters used herein are as follows. de Bruijn graphs: $k = 8$, $D = 8$. BALANCE operation: $t_{\min} = 0.03$, $t_{\text{diff}} = 0.0\bar{7}$.

2.4.2 Static networks

In our static simulations, no peers joined or left the system. We assigned capacities to the n peers and fixed their order in the keyspace. We report on four stages of PLB. In each we measured network features such as maximum utilization, mean utilization,

[4]Simulations with smooth but very nonuniform distributions of data items showed that PLB copes well with different values of $\boldsymbol{\sigma}$; space prohibits their presentation here. Caching can be employed to level isolated hot spots, as described in Section 2.2.2.3.

and average query path length.

0. As a baseline, we tested the uniform network (all peers have $z_i = \frac{1}{n}$).

1. We then rebuilt the network with the single-point proportional-split method of Section 2.3.2.1.

2. We performed five iterations of PLB on this network (to evaluate realistic performance).

3. We continued performing 500 iterations for some simulations (to evaluate limiting behavior). The worst-case value of each feature between 450 and 500 iterations was used.

2.4.2.1 Uniform request rate

In these simulations, $\boldsymbol{\rho} \propto \mathbf{1}$. At each network size, 30 simulations were performed for stages 1 and 2. Because of time considerations, only 6 simulations were continued to stage 3.

Figs. 2.6 and 2.7 show how the maximum utilization drops under PLB. The inverse of the maximum utilization determines how much traffic the network can handle without any peer becoming over-utilized.

Relative to the corresponding uniform networks, forward de Bruijn DHTs saw a greater decrease in maximum utilization in stages 1 and 2 than did reversed de Bruijn DHTs. However, the starting values in stage 0 were about twice as great. The limiting behavior of both cases was about the same, with the maximum utilization

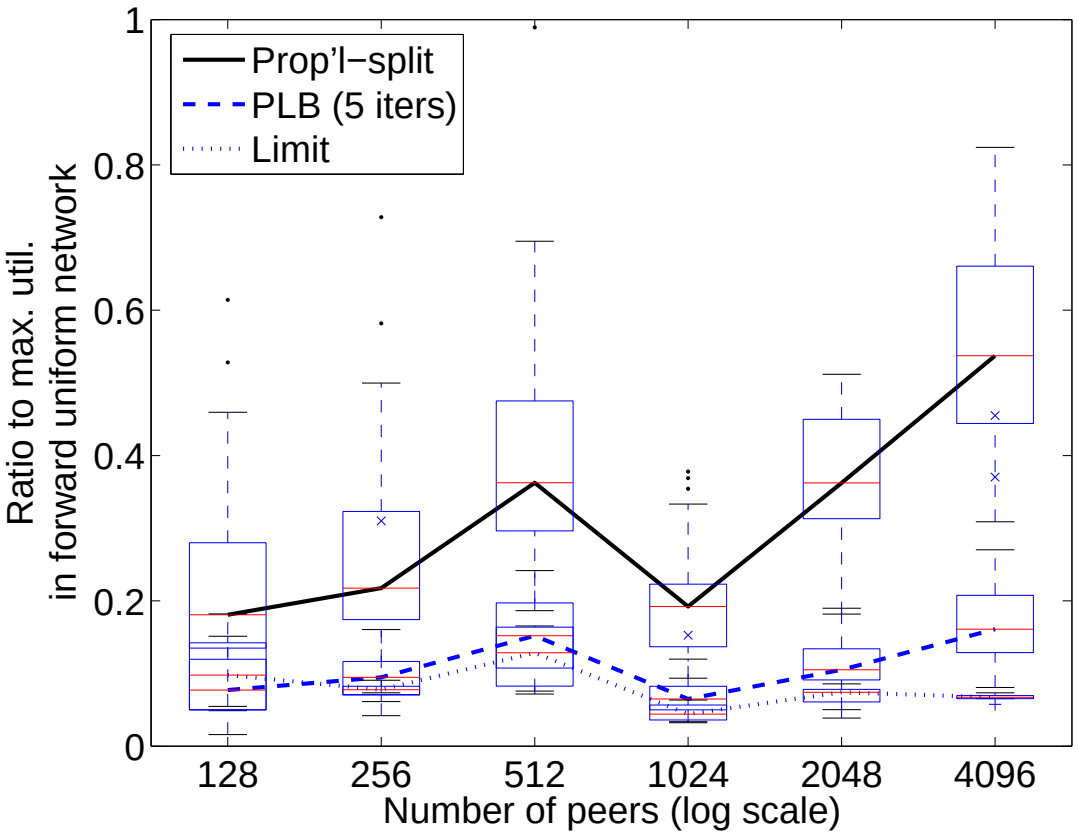


Figure 2.6: Results of the static simulations in Section 2.4.2.1, with uniform request rate $\boldsymbol{\rho} = \mathbf{1}$. The results for forward de Bruijn networks are depicted here. Multiple simulations were performed. These plots show median values connected by lines, with boxes delimiting the 25^{th} and 75^{th} percentiles, and whiskers showing the full extent of the data. Figs. 2.6 and 2.7 show how the maximum peer utilization decreases under PLB. All measurements are relative to the maximum utilization in the corresponding uniform network (peer capacities and ordering are the same as the original network, but all zones are given equal size). The solid line shows networks built with single-point proportional-split (Section 2.3.2.1). The dashed line shows the same proportional-split networks after 5 iterations of BALANCE. The dotted line shows the limiting behavior of PLB, after 450 iterations.

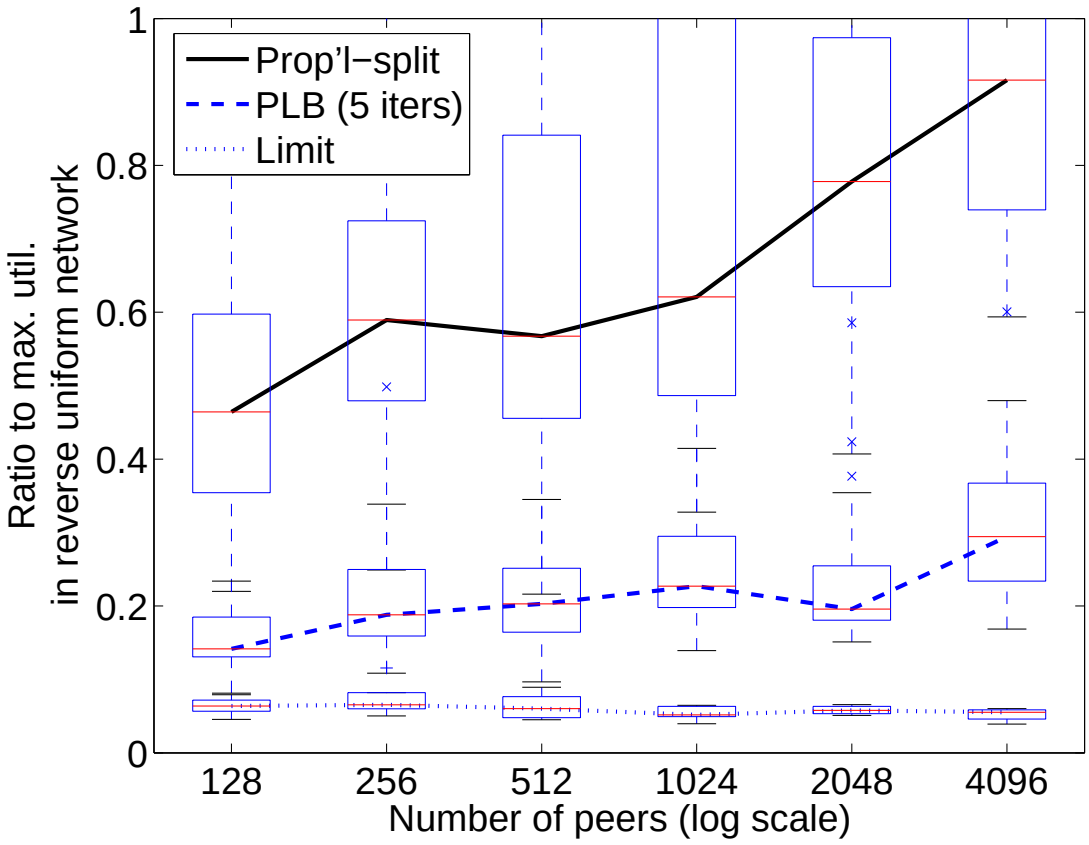


Figure 2.7: Results of the static simulations in Section 2.4.2.1, for reversed de Bruijn networks. This figure complements Figure 2.6, which shows the results for forward de Bruijn networks.

being 5% to 10% of its original value.

As predicted in Section 2.2.2.2, query path lengths decreased (by 10%) for reversed de Bruijn DHTs, and increased (by 10%) for forward de Bruijn DHTs. In stage 0, the reversed query path lengths were about 70% longer than the forward query path lengths.

2.4.2.1.1 Volatility Figure 2.8 compares the actual values of the maximum utilization for forward and reversed de Bruijn DHTs. In the limit of a large number of iterations, their best-case behavior is similar, but oscillations for the forward DHTs yield a higher worst-case behavior.

The sudden spikes in utilization seen in Figure 2.8 most often occur when a low-capacity peer becomes overwhelmed, usually for one of two reasons. (i) The peer might expand its zone and acquire more traffic than expected, due to imbalances in the keyspace. The imbalances are more profound in forward de Bruijn graphs, because of their more primitive routing procedure. (ii) A different peer might change the size of its zone, thus forcing it to connect to the peer in question. This is more problematic in the forward case because a small peer is likely to have only one outgoing connection, thus overwhelming its sole neighbor.

2.4.2.2 Varied request rate

In these simulations, $\boldsymbol{\rho} \propto (cap(1), \ldots, cap(n))$. At each network size, 30 simulations were performed for stages 1 and 2. Of these, 6 were continued to stage 3. See Figure 2.9.

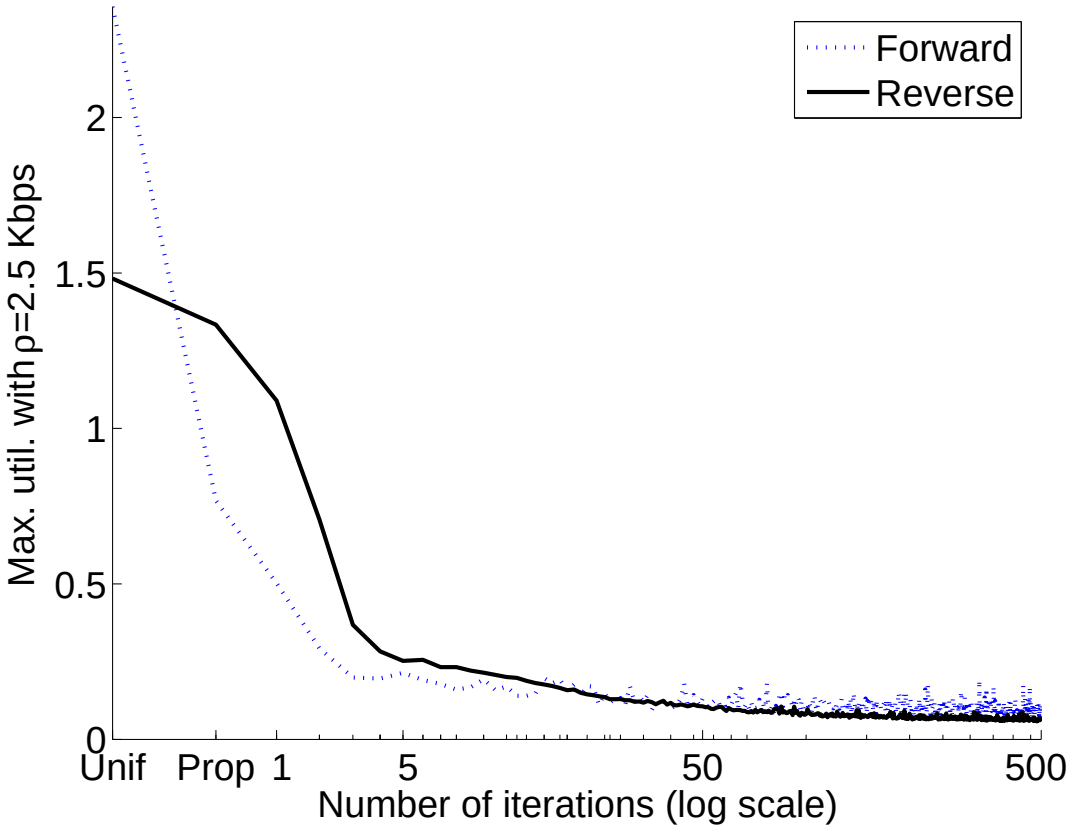
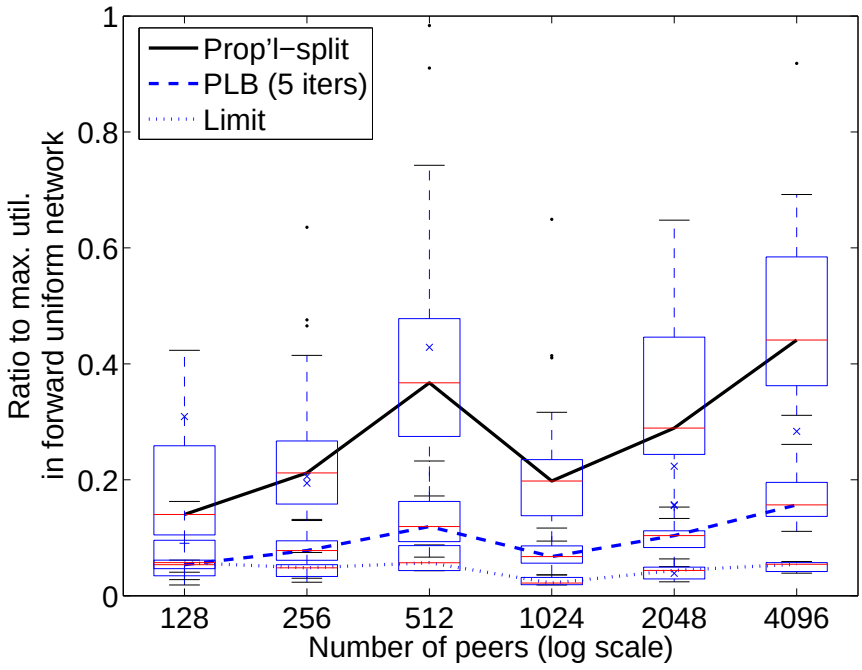
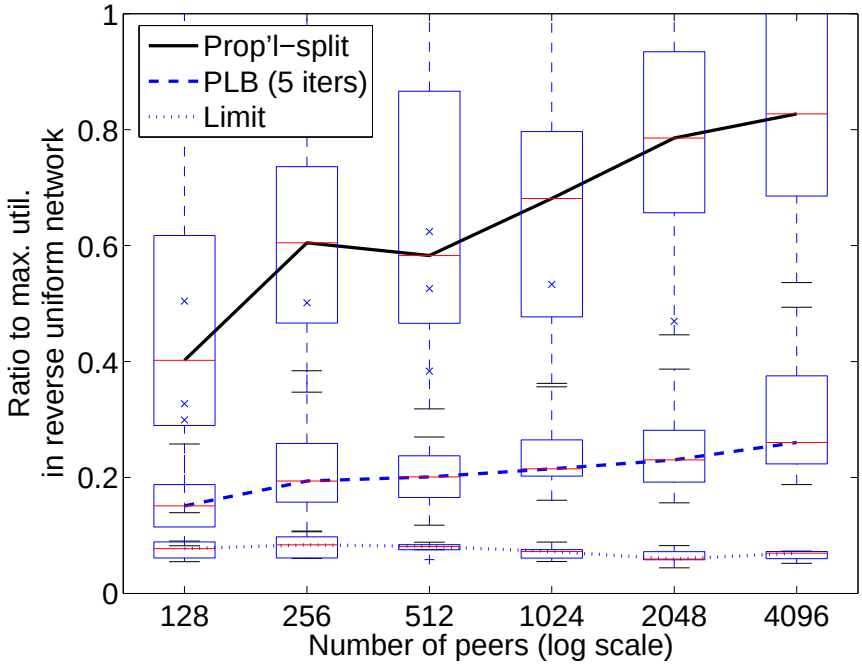


Figure 2.8: Sample result of the static simulations in Section 2.4.2.1. For a fixed injection rate $\boldsymbol{\rho}$, we compare the maximum utilization for forward and reversed de Bruijn DHTs as PLB is performed. The graph depicts the results of one simulation of each, with 2048 peers. Local fluctuations are much higher in the forward case.



(a) Forward de Bruijn



(b) Reversed de Bruijn

Figure 2.9: Results of the static simulations in Section 2.4.2.2, with capacity-proportional request rate. See Figure 2.6 for an explanation.

Results for maximum utilization were very similar to the simultations with uniform rerquest rate, as seen in Figure 2.9. Forward de Bruijn DHTs again saw a greater decrease in maximum utilization in stages 1 and 2 than did reversed de Bruijn DHTs. The starting values in stage 0 were again about twice as great. However, the limiting behavior in forward DHTs was slightly better than reversed.

Query path lengths again decreased for reversed de Bruijn DHTs. Forward de Bruijn DHTs saw a very slight decrease. In stage 0, the reversed query path lengths were about 65% longer than the forward query path lengths.

2.4.3 Dynamic networks

These simulations model networks that have reached a steady-state size, but with a fluctuating (dynamic) population of peers. We start with the average network size $n = 512$, and allow peers to join and leave the system according to the rules of Section 2.3.2.1.

Fix a *churn rate* χ. Between each iteration of PLB, we independently allow each peer to drop out of the network with probability χ, and we also insert 512χ new peers with capacities chosen from the distribution in Table 2.1. This process will tend to keep the network size close to 512.[5] How fast do peers come and go in reality? A detailed study of churn rates ([50], Figure 6) reveals that in the real-world DHT Kad, approximately 80% of the peers in the network at any given time have been present for at least one hour, while approximately 95% have been present for at least

[5]If the network size is n_i after i iterations, then $E\left(|n_{i+1} - 512| \mid n_i\right) = |(n_i - n_i\chi + 512\chi) - 512| = (1-\chi)|n_i - 512| < |n_i - 512|$.

ten minutes. Therefore, only a small percentage of the peers will have changed if we execute PLB every few minutes.

We executed 12 simulations of 500 iterations for forward and reversed networks for values of $\chi = 2^{-15}, 2^{-14}, \ldots, 2^{-3}$. The results for small χ were similar to Figure 2.8, as expected. A representative example for larger χ appears in Figure 2.10.

For definiteness, we define a *spike* as an increase in one step by at least a factor of two, and the *spike recovery time* as the number of steps to return to within 10% of the pre-spike value. First consider the maximum peer utilization, as we have in other experiments. As χ becomes large (we tested up to $\chi = 2^{-3} \approx 13\%$), the median distance between spikes and the median spike recovery time both approach approximately 5 steps. Now consider the 99^{th} percentile of the peer utilizations, depicted by the bottom line in Figure 2.10. Ignoring a very few highly-affected peers increases the median distance between spikes to approximately 75, without greatly affecting the spike recovery time.

Note that we are assuming no correlation between session length and capacity. It is plausible to think that in reality, peers with short session times also have low capacities. Such behavior would reduce the fluctuations in maximum utilization, since the removal of a peer with a small zone has less impact than the removal of one with a large zone.

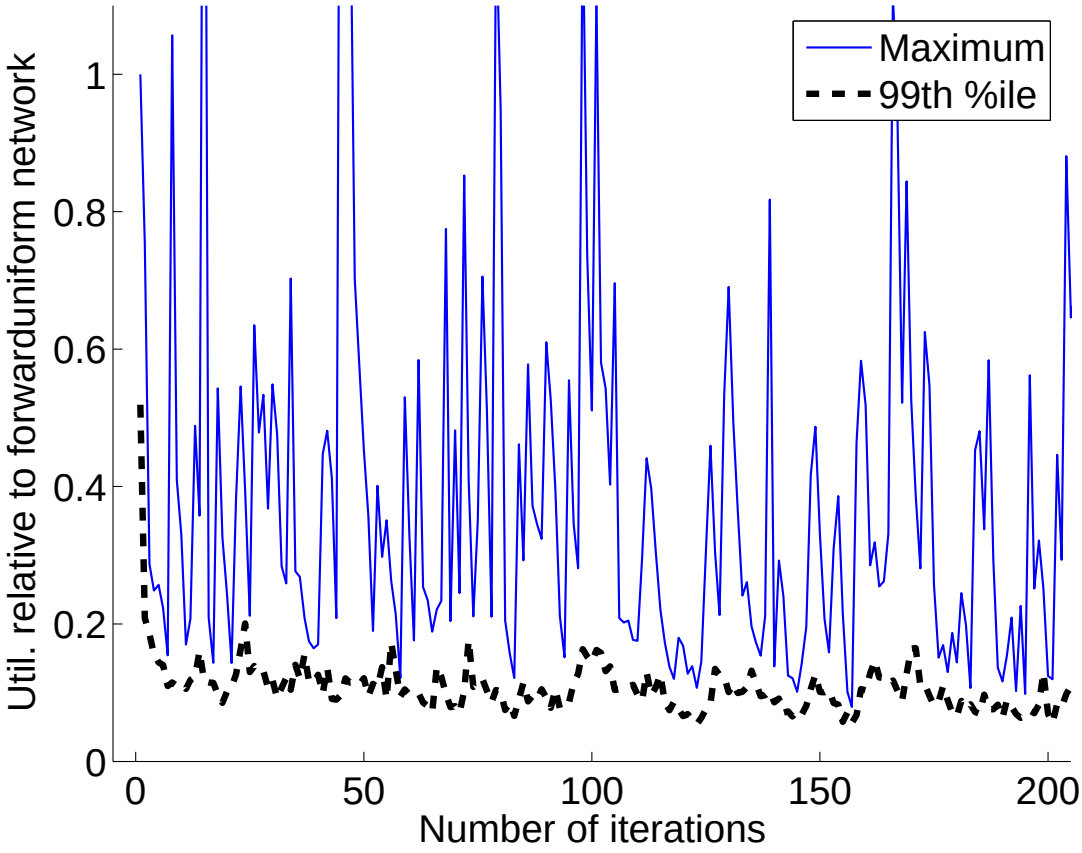


Figure 2.10: Sample result of the dynamic simulations in Section 2.4.3 (forward network pictured). The churn rate is $\chi = 2^{-5} \approx 3.1\%$. For clarity, only 200 iterations are displayed. Forward and reversed networks displayed statistically similar reactions to churn rate. We see that the maximum utilization is highly volatile, but the 99^{th} percentile of utilizations is relatively stable, indicating that the vast majority of peers in dynamic networks will benefit from PLB.

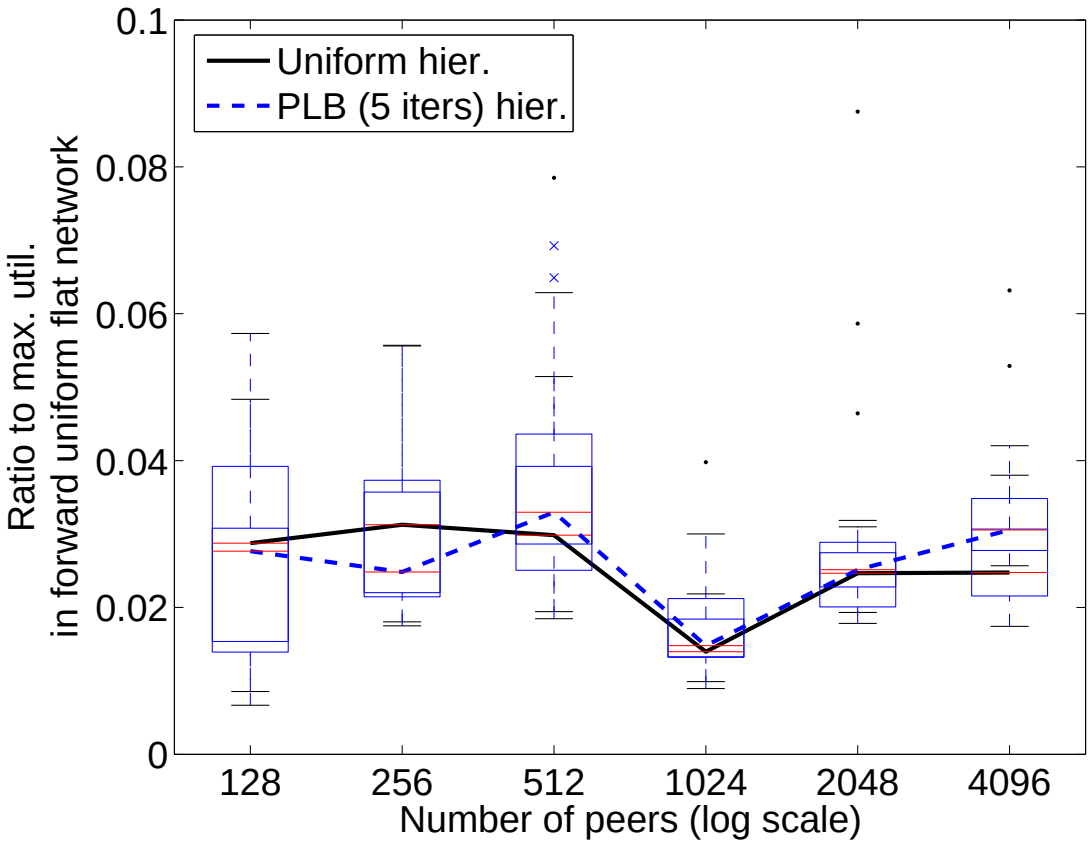


Figure 2.11: Results of the hierarchical simulations in Section 2.4.4. The results for forward de Bruijn neigworks are depicted here; see Figure 2.12 for the results for reversed de Bruijn networks. Peers were grouped by the threshold capacity $c_{\text{mean}} = 940$ Kbps, and results are shown relative to the corresponding flat uniform network (no hierarchy, all zones equal size). The solid line depicts a hierarchical uniform graph, while the dashed line shows a hierarchical graph to which PLB has been applied for 5 iterations.

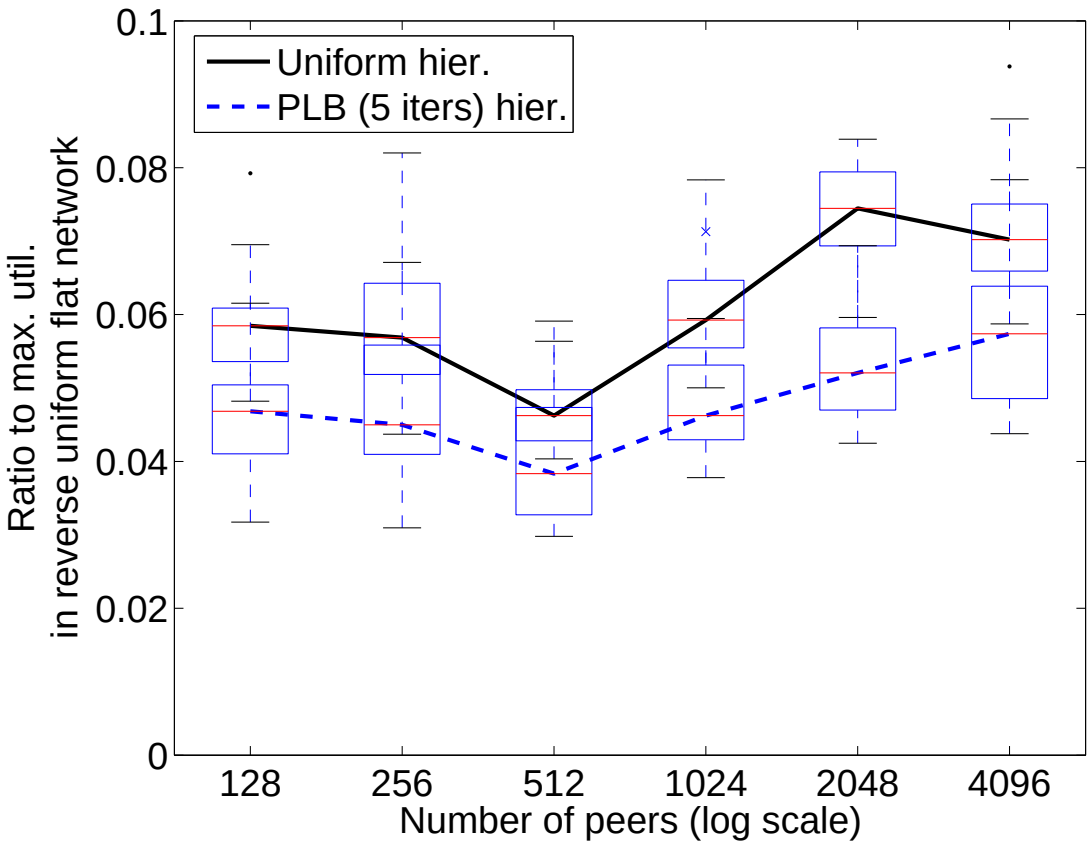


Figure 2.12: Results of the hierarchical simulations in Section 2.4.4, for reversed de Bruijn networks. This figure complements Figure 2.11, which shows the results for forward de Bruijn networks

.

2.4.4 Hierarchical DHTs

Hierarchical P2P systems [19] divide their peers into two (or sometimes more) classes. One class consists of *superpeers*, with high capacity and stability, and one consists of regular, transient peers. We implemented a simple hierarchical system based on reversed de Bruijn graphs, as follows.

Given a threshold capacity c, we label peer i a *superpeer* if $cap(i) \geq c$, or a regular peer otherwise. We form a de Bruijn DHT from the superpeers; then each of the regular peers is assigned to a superpeer at random. All peers share data, but a superpeer handles all of the routing of requests for its associated regular peers.

When building the network with the single-point proportional-split method (Section 2.3.2.1), we split zones in proportion to the square roots of the capacities of the new and existing superpeers. This is because the expected amount of traffic through a superpeer depends both on its zone size, and on the number of regular peers attached to it, which is proportional to its zone size. The BALANCE operation was unchanged.

Figures 2.11 and 2.12 show results for uniform hierarchical networks and PLB-iterated hierarchical networks, both compared to uniform non-hierarchical (flat) networks. We note two features of these plots. First, uniform hierarchical networks have maximum utilizations that are about 3% (forward) to 6% (reversed) of the value for a flat network, which is a large improvement. Second, executing PLB on these hierarchical networks yields maximum utilizations that are as good as or better than those for the uniform hierarchical networks.

2.5 Discussion

We have presented the mechanics of forward and reversed de Bruijn DHTs, under the continuous-discrete framework. We analyzed the effect of relative zone size on expected degree in each type of DHT. Furthermore, we investigated the effect of zone size distribution on the average query path length. Theorems 2.1, 2.2, and 2.3 of Section 2.2.2 describe the beneficial effect heterogeneity can have.

We have also presented a proportionate load balancing algorithm, which iteratively optimizes continuous-discrete DHTs by equalizing peer utilizations. After analyzing the data and rewiring costs involved, we performed simulations of static, dynamic, and hierarchical networks to verify the efficacy of PLB in reducing strain on peers. We found that the maximum relative load on any peer dropped to approximately 20% of the value for a uniformly-structured network after just five iterations of PLB, and to less than 10% in the limit of 450-500 iterations. A more complete practical algorithm could be a subject of future research.

Networks built with PLB offer a continuum of peer roles, from server-like to client-like, so each peer can choose an appropriate role for itself depending on the current makeup of the network. This differs from hierarchical schemes [19] which treat some peers almost exclusively as servers and the others only as clients.

We found that forward de Bruijn DHTs are more volatile in their response to PLB than reversed de Bruijn DHTs (Section 2.4.2.1.1). They also experienced larger starting values of maximum utilization, by approximately a factor of two. Although forward DHTs have shorter average query path length in uniformly-structured net-

works, under heterogeneity we have that the query path length may increase for forward DHTs, while for reversed DHTs it decreases by $\frac{\mu}{2 \ln k} r_{\max}^{-1} \mathrm{Var}\{r_i\}$. This is because in the forward case, a large zone size only benefits the owner, while in the reversed case, a large zone size benefits all the peers. We believe these features present a strong case for the consideration of reversed de Bruijn DHTs.

Chapter 3

Identification of peer-to-peer hosts

3.1 Introduction

Universities and corporations have a particular interest in identifying peer-to-peer traffic in their networks. Their concerns are twofold: bandwidth and security. With regard to bandwidth, peers in peer-to-peer networks typically upload more than clients in client-server systems [45], which puts strain on the networks they inhabit. With regard to security, sharing pirated content can result in costly legal action, or applications and content may contain viruses, spyware, or other malware [7].

Our focus on this chapter is on security concerns, so our goal is to identify the offending/vulnerable hosts. In contrast, when the major concern is for bandwidth, the goal is to identify individual traffic flows, or even packets. Higher accuracy can be achieved when identifying hosts, because more information is available: we can consider all the flows a host is involved in. Additionally, network monitoring is work-intensive. We are able to achieve acceptable results with NetFlow-style [8] data, while effective flow-identification schemes tend to require packet-level information [3, 55, 35].

We will use behavioral commonalities to identify peer-to-peer applications. Our data will be IP traffic passively observed at set monitoring locations. Examples of observable behaviors are average flow size, or number of IP addresses communicated

with in a certain period of time. Message passing, measured in "triples," will be a particularly important behavior. We will intentionally ignore other behaviors, such as port information [34], or the specific timing of requests, because they are application-specific. We will also attempt to make the identification as close to real-time as possible, rather than performing extensive post-processing analysis of network structure.

3.1.1 Related work

Most of the literature on identifying peer-to-peer traffic has focused on labeling flows (or IP pairs) rather than individual hosts. When applicable, payload examination can be a very effective method of classifying particular P2P applications [47, 1]. However, encryption and data volume are increasingly rendering payload methods inapplicable.

It is also possible to identify flows based only on features extracted from metadata. These features are typically processed by machine learning classification techniques. Some studies have attempted to identify specific applications, such as Kazaa or eDonkey. Two such studies achieved 80-90% accuracy, one [3] using only the sizes of its first several data packets in the flow, and another [55] using a wider variety of features, including packet interarrival times.

Other studies, which are more in line with our goals, have attempted to identify the categories of traffic that a flow belongs to, such as P2P or email. One such study [35] used features including packet interarrival times and TCP ports to achieve up to 95% accuracy for certain application categories, though only up to 55% for

P2P. Another [26] used features including port counts to identify 99% of P2P flows with a 10% false positive rate, but used a fixed algorithm rather than adaptable machine learning techniques. Though these studies did not rely on *prespecified* port information, the use of port numbers at all is increasingly unreliable, because it is possible for users to employ random port numbers.

A smaller number of studies have discussed identification of individual hosts, rather than flows. BLINC [27] used features including port counts to classify hosts into application categories, then used that information to classify the corresponding flows, achieving 95% accuracy on 80-90% of the flows. Some others focused on more narrow subsets of P2P applications: one [48] primarily investigated eDonkey, while another [52] discussed the identification of DHTs, without providing an implemenation, or suggesting the "triple" heuristic we discuss in Section 3.2.2. All these studies required post-processing on the graph of host connections.

Other work of interest includes the identification of compromised "stepping stone" computers [56] and the identification of relay nodes in Skype [51], because such hosts are qualitatively similar to the hosts in the middle of the "triples" we define in Section 3.2.2. The identification of email spamming machines by graph theory techniques [11] may also be of interest.

One might think of a "stepping stone" [56] as a host in the middle of a triple (see Section 3.2.2), but stepping stone analysis requires the flows to be very similar on the packet level. In the interest of general applicability, we only concern ourselves with the existence of the flows, no with any of their measurable properties (other than perhaps their overall size).

3.1.2 Trace data

We use publicly-available traces from the NLANR Passive Measurement and Analysis Project [38]. The PMA project provides packet header traces from a variety of monitoring locations. Most traces have a duration of 90 seconds, but special traces with durations of hours or days are also available.

Our main focus was on the Leipzig-I data set [39], a trace capturing all traffic of the University of Leipzig between 8pm Thursday, November 21 and 2pm Tuesday, November 26, 2002. This data was captured recently enough that peer-to-peer traffic is common, but sufficiently long ago that many peers were still using fixed TCP and UDP ports [34]. In addition to these advantages, the long time period of the trace allows us to study the behavior of a host in more detail.

We can also compare to 90-second traces from the University of Memphis [40], captured as recently as March 2006.

Because they each monitor the gateway to a university, both of these traces have a definite "inside" and "outside." For Leipzig-I, the inbound and outbound packets are stored in separate files (whose labels were inadvertently switched [33]). We place more focus on classifying the interior hosts, since the trace capture more information about their communications. The interior hosts may also be of more interest to a network administrator.

We may look at what happens when we use classifications of the exterior hosts to adjust our judgements of the interior hosts.

3.1.3 Overview

The remainder of this chapter is structured as follows. In Section 3.2 we discuss the measurable features of our data set, and derive a model for message passing. In Section 3.3 we discuss the use of port numbers, provide an overview of supervised learning techniques, and list the results of our classification experiments.

3.2 Feature set

The trace data we use provides packet-level records, but we convert them to NetFlow-style data [8] befpre performing our analyses. In particular, we use the following definition.

Definition 3.1. A *flow* is a sequence of packets sent back and forth between a pair of hosts, identified by the 5-tuple

$$f = (S, s;\ D, d;\ P), \tag{3.1}$$

where (S, s) are the source's IP address and port number, (D, d) are the destination's IP address and port number, and P is the protocol number. The "source" is the host that sends the first observed packet in the flow. If there is a gap of more than 60 seconds between consecutive packets, we consider there to be two distinct flows.

In this work, we only consider flows corresponding to the TCP and UDP protocols. Most other protocols could easily be included, but for the triples discussed in Section 3.2.2, it is important to ignore ICMP flows.

Using Definition 3.1, we divide packet traces into flows, keeping the following information about each flow:

- The 5-tuple $(S, s;\ D, d;\ P)$, with $s = d = 0$ for protocols without port numbers
- The timestamp of the first packet in the flow
- The timestamp of the last packet in the flow
- The number of packets sent from S to D

- The number of packets sent from D to S
- The number of bytes (header and payload) sent from S to D
- The number of bytes (header and payload) sent from D to S
- A flag identifying which IP address (S or D) is internal to the university

3.2.1 Elementary features

Whenever a new host X is observed, we create a database entry for it, noting the timestamp of the first observation and whether the host is interior or exterior to the university (if known). As X participates in more flows, we update the following values: the most recent timestamp; a list of the distict IP addresses X interacts with; the total number of flows initiated and accepted; the total number of packets initiated and accepted; the total number of payload bytes initiated and accepted; the total duration of individual flows; and counts of each protocol type.

From these values, we calculate nine acceptable features:

- The average number of new neighbors (IP addresses X interacts with) per second
- The average number of flows per neighbor
- The average number of packets per flow
- The average number of payload bytes per packet
- The average flow duration

- The fraction of flows that were initiated by X
- The fraction of packets that were initiated by X
- The fraction of payload bytes that were initiated by X
- The fraction of flows that used the TCP protocol

These features can be calculated at a specified clock time, or after seeing a specified amount of activity from host X.

3.2.2 Triples

We now present a feature that measures flow interactions.

3.2.2.1 Definition

For a decentralized peer-to-peer network to function, the peers must participate in message-passing. Whether in older systems such as Gnutella or newer systems based on distributed hash tables [32], the lack of a central authority implies that when a peer wants a file, he must make a lookup query of one or more of his neighbors, who must in turn make queries of one or more of their neighbors, until the file is located.

Therefore, peers in a decentralized peer-to-peer network will both initiate and accept flows with other peers, and in quick succession if they are not willing to introduce significant delay into the search process. This inspires the following definition, where $\textsc{Time}(f)$ is the timestamp of the first packet in flow f.

Definition 3.2. A *triple* for a host X is a pair of flows $f_1 = (A, a; X, x_1; p_1)$, $f_2 = (X, x_2; B, b; p_2)$ such that $A \neq B$ and $0 < \textsc{Time}(f_2) - \textsc{Time}(f_1) \leq \tau$. The triple is denoted $A \rightarrow X \rightarrow B$. The parameter τ is called the *window parameter*.

We use a window parameter of $\tau = 5$ seconds.

Note that either flow f_1 or f_2 may be involved in more than one triple. In fact, in the case of Gnutella, we would expect f_1 to be in multiple triples, since Gnutella lookup queries are passed to all neighbors.

3.2.2.2 Counting triples

Because we only observe traffic that takes place between a host inside the university and a host outside the university, we will be artificially unlikely to see triples for exterior hosts. We therefore focus on interior hosts. We will find that P2P hosts tend to have more triples, even though we cannot see the peer activity that takes place entirely within the university network. In the Leipzig-I dataset, we can easily limit our analysis to interior hosts.

Consider observing a interior host X for a time period $0 \leq t \leq T$. We keep track of the following statistics:

- N_1 = the number of flows X accepts
- N_2 = the number of flows X initiates
- S_1 = the number of accepted flows that are involved in a triple
- S_2 = the number of initiated flows that are involved in a triple

- H = the number of distinct hosts X communicates with (may be omitted — see below)

The flows counted by N_1 have the potential to be the first flow in a triple, while those counted by N_2 have the potential to be the second. We will model each flow's probability of being in a triple by assuming that the start times of the flows are uniformly distributed in the interval $[0, T]$, and then compare this expectation to the observations S_i.

We choose the uniform distribution for simplicity, and will find it to be effective. We tried the Poisson distribution with similar results, but it has been known for years [43] that Poisson models do not capture the burstiness in data networks. Non-Poisson arrival models [41] could also be considered.

Let f_1 be one of the N_1 inbound flows, and let $t_1 = \textsc{Time}(f_1)$. We define

$$p_1 = \mathrm{P}[f_1 \text{ is in a triple}] \tag{3.2}$$

$$= \mathrm{P}[\text{a flow } f_2 \text{ to a different host occurs with } \textsc{Time}(f_2) \in (t_1, t_1 + \tau]] \tag{3.3}$$

$$= 1 - \left(1 - \frac{\tau}{T}\right)^{\frac{H-1}{H}N_2} \tag{3.4}$$

where $\frac{H-1}{H}N_2$ is the average number of flows that X initiates to hosts, other than the host involved in f_1. Similarly, given an outbound flow f_2, we can define the probability

$$p_2 = 1 - \left(1 - \frac{\tau}{T}\right)^{\frac{H-1}{H}N_1} \tag{3.5}$$

that f_2 is involved in a triple.

If H is deemed to be to expensive to keep track of, we can use the approximations

$$p_1 \approx 1 - \left(1 - \frac{\tau}{T}\right)^{N_2}, \quad p_2 \approx 1 - \left(1 - \frac{\tau}{T}\right)^{N_1} \tag{3.6}$$

because $\frac{H-1}{H} \approx 1$ for large H.

For $i = 1, 2$, we can now model the number of flows that are involved in a triple using a Binomial(N_i, p_i) distribution, if we assume each flow has an independent, identically distributed probability of being involved in a triple. Recall that the binomial distribution, which has probability mass function

$$B_i(n) = \binom{N_i}{n} p_i^n (1 - p_i)^{N_i - n}, \tag{3.7}$$

has mean $\mu_i = N_i p_i$ and variance $\sigma_i^2 = N_i p_i (1 - p_i)$.

The Binomial(N_i, p_i) distribution can be approximated by a Normal(μ_i, σ_i^2) distribution. Therefore, to compare S_i to the expected μ_i, we may use the z-score

$$z_i = \frac{S_i - \mu_i}{\sigma_i} \tag{3.8}$$

The z-score represents the difference between S_i and the model mean, in units of the model standard deviation. The z-scores z_1 and z_2 will be our metrics for measuring triples.

3.3 Classification results

This section describes our testing methodology and results.

3.3.1 Determining the "true" classes

Although TCP/UDP port numbers are not generally an effective method for identifying peer-to-peer traffic [34], we can use them as a baseline measurement of truth in the Leipzig-I data set because of its age. Table 3.1 summarizes how much P2P traffic can be identified by port number; we see that port numbers cannot be used as a reliable means of identification in the Memphis data set.

Table 3.1: The fraction of TCP/UDP traffic, by number of flows and by amount of payload, that has known P2P port numbers.

	Leipzig-I[a]		**Memphis**[b]	
Application	**Flows**	**Bytes**	**Flows**	**Bytes**
eDonkey 2000	23.54%	12.38%	0.28%	0.01%
Soribada	1.52%	0.04%	—	—
WinMX	1.19%	3.58%	—	—
DirectConnect	1.05%	1.15%	—	—
KaZaA	1.04%	10.59%	0.04%	0.01%
Gnutella	0.48%	1.27%	0.75%	0.07%
Manolito	0.21%	0.01%	0.73%	0.01%
SoulSeek	0.10%	0.65%	—	—
Napster	0.04%	0.11%	0.08%	0.00%
iMesh	0.01%	0.01%	—	—
BitTorrent	0.00%	0.00%	0.10%	0.00%
Client/Server	48.51%	35.21%	50.94%	54.34%
Unknown	22.29%	34.99%	47.08%	45.55%

[a] From 20021121-200000.
[b] From MEM-1143763417-1.

Given an interior host X, we can assign one of four labels to each of its flows:

1. PEER-TO-PEER: At least one port is known to be associated with a peer-to-peer application (KaZaA, eDonkey 2000, etc.)

2. SERVER: At least one port associates the host with the *server* end of a specific non-P2P application (HTTP, games, instant messaging, trojans, etc.).

3. CLIENT: At least one port associates the host with the *client* end of a specific non-P2P application.

4. UNKNOWN: Neither port has a known association.

We use the port list in Appendix A.1, derived from a variety of online sources. For example, if X uses port 80 (HTTP), we label the flow SERVER; if the other host uses port 80, we label the flow CLIENT.

If at least 75% of the flows that host X participates in are UNKNOWN, then we also assign X the label UNKNOWN. Otherwise, we assign X one of the labels PEER-TO-PEER, SERVER, or CLIENT by a simple plurality vote of the relevant flows. Counts are shown in Table 3.2. Some time-of-day trends are apparent.

We conjecture that many of the UNKNOWN flows are peer-to-peer, because traditional client/server applications tend to use well-known ports. We also conjecture that some flows taking place over port 80 are also P2P [26], since hosts may try to hide their traffic to bypass firewalls or other security measures.

Table 3.2: The number of active hosts of each type, broken down by date and time. An "active" host is one that both sent and received one packet in the six-hour file window. Not all hosts were used in the experiments (Section 3.3.3), because the experiments employed a more limited time (or traffic volume) window.

Time	Date	P2P	Client	Server	Unk.	Total
8pm	Thu	189	2129	312	1120	3750
	Fri	144	1492	316	513	2465
	Sat	133	1205	312	301	1951
	Sun	190	1880	363	344	2777
	Mon	226	2364	324	454	3368
2am	Fri	102	788	169	246	1305
	Sat	100	415	256	309	1080
	Sun	104	646	236	258	1244
	Mon	93	768	780	98	1739
	Tue	122	872	203	398	1595
8am	Fri	167	3462	388	463	4480
	Sat	123	1482	214	419	2238
	Sun	131	1254	857	468	2710
	Mon	172	3929	561	453	5115
	Tue	171	3917	520	572	5180
2pm	Fri	184	3103	388	432	4107
	Sat	139	1683	400	352	2574
	Sun	169	1866	439	220	2694
	Mon	227	4326	437	535	5525

3.3.2 Supervised learning techniques

We will compare the results of a variety of supervised learning techniques. This section provides an overview of such techniques. Since we are only using the tools, not developing new techniques, we do not provide full details; see, e.g., [24] for more information.

3.3.2.1 Goals and definitions

The goal of supervised learning is to construct a function, based on known examples, that relates observable features to an outcome of interest. It can be used to answer questions such as: What is the probability of a customer defaulting on their mortgage? Will a patient's cancer recur? Is a computer access malicious?

Each event (item in the sample space) has:

- a *feature vector* $\overrightarrow{x} = (x_1, x_2, \ldots, x_p) \in \mathbb{R}^p$ of observable variables[1]
- an *outcome* variable $y \in Y$[2]

There is a body of known *training data* $(\overrightarrow{x_1}, y_1), \ldots, (\overrightarrow{x_N}, y_N)$. Supervised learning techniques attempt to build a function $f : \mathbb{R}^p \to Y$ that is close to having $f(\overrightarrow{x_j}) = y_j$ for all $1 \leq j \leq N$, without overfitting. Building f may be time-consuming, but once the setup process is complete, predicting the outcome $y = f(\overrightarrow{x})$ for a new observation $\overrightarrow{x}$ is fast.

3.3.2.2 Examples

We begin with a very simple example. Assume there are only two possible outcomes, $y = 0$ or $y = 1$. The training data divides into two classes, each with N_y members,

[1] It is possible to have *categorical variables* $x_i \in X_i$, where X_i is not $\mathbb{R}$, nor even a metric space; for example, $X_i = \{\text{female}, \text{male}\}$. Since none of our features are categorical, we use $X_i = \mathbb{R}$ throughout to simplify the discussion.

[2] If $|Y|$ is finite, building f is known as a "classification problem"; if $Y = \mathbb{R}$, it is known as a "regression problem." We focus on classification problems here.

and each with mean

$$\overrightarrow{\mu_y} = \frac{1}{N_y} \sum_{j=1}^{N_y} \overrightarrow{x_j}^{(y)}, \tag{3.9}$$

and covariance matrix

$$\Sigma_y = \frac{1}{N_y - 1} \sum_{j=1}^{N_y} (\overrightarrow{x_j}^{(y)} - \overrightarrow{\mu_y})^T (\overrightarrow{x_j}^{(y)} - \overrightarrow{\mu_y}). \tag{3.10}$$

If the true classes are normally distributed with equal covariances, then Fisher's linear discrminant [15] provides the Bayes optimal solution. This discriminant divides the feature space $\mathbb{R}^p$ with a hyperplane perpendicular to

$$\overrightarrow{w} = ((N_0 - 1)\Sigma_0 + (N_1 - 1)\Sigma_1)^{-1} (\overrightarrow{\mu_0} - \overrightarrow{\mu_1}). \tag{3.11}$$

That is, either

$$f_c(\overrightarrow{x}) = \begin{cases} 0 & \text{if } \overrightarrow{w} \cdot \overrightarrow{x} \geq c \\ \\ 1 & \text{if } \overrightarrow{w} \cdot \overrightarrow{x} < c, \end{cases} \qquad \text{or} \qquad f_c(\overrightarrow{x}) = \begin{cases} 1 & \text{if } \overrightarrow{w} \cdot \overrightarrow{x} \geq c \\ \\ 0 & \text{if } \overrightarrow{w} \cdot \overrightarrow{x} < c, \end{cases} \tag{3.12}$$

for some constant c.

We now briefly describe some of the most important supervised learning techniques. In a given problem, there may be more than two possible outcomes; these techniques can be generalized to multi-class versions with varying amounts of difficulty.

- Linear discriminant analysis [15]. The feature space is divided by a hyperplane. The model assumes the classes have equal covariances.

- Quadratic discriminant analysis [13]. The feature space is divided by a quadric surface. The model assumes the classes are normally distributed with unequal covariances.

- Naive Bayes classification [12]. All features are assumed to be independent.
- k-nearest neighbor algorithm [13]. A new observation $\overrightarrow{x}$ is assigned the most common y value of its k nearest neighbors $\overrightarrow{x_{j_1}}, \ldots, \overrightarrow{x_{j_k}}$ in the training set.
- Neural networks [30]. The function f is a compilaton of other functions, which fit a given model and are tuned by the training data.
- Classification trees [6]. The feature space is hierarchically divided along one variable at a time. At each point in the hierarchy, the split is chosen to reduce the node impurity $i = \frac{n_0 n_1}{(n_0+n_1)^2}$.
- Random forests [4]. A large number of classification trees are produced, each using a small random subset of the features and a randomly chosen (with replacement) subset of the training data.

We use Breiman's implementation for random forests [5], and LNKnet's implementation for all other techniques [23]. In LNKnet, we performed a simple mean-0, variance-1 normalization of the data; such normalization is unnecessary in random forests.

3.3.3 Results

Since the makeup of a university's active computer population varies over the course of a day, we grouped the input files by local start time: 8pm, 2am, 8am, and 2pm. For $n = 4, 16, 64, 256, 1024$, we observed the first n flows for each IP address. Again for $n = 4, 16, 64, 256, 1024$, we observed the first n seconds of activity for each IP

address. The features described in Section 3.2 were computed, then run through various supervised learning algorithms. To account for imbalances in class distribution, Peer-to-Peer hosts were given a weight five times as great as Client and Server hosts in the training process.

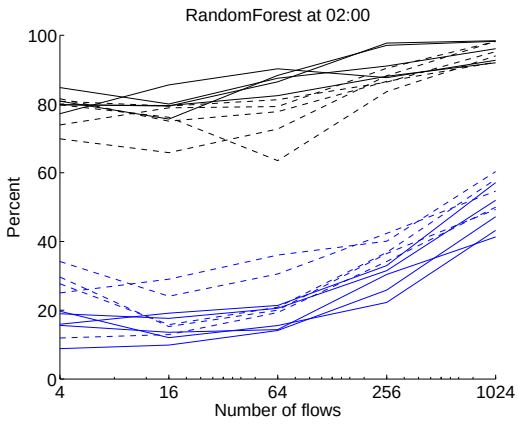
We estimated the success rate (percent of Peer-to-Peer hosts classified correctly) and false positive rate (percent of non-Peer-to-Peer (Client and Server) hosts classified as Peer-to-Peer) for each algorithm. With random forests, these come from the out of bag (OOB) error estimate. With the other algorithms, these come from 10-fold cross-validation (CV10).

A comparison of classification techniques is given in Tables 3.3 and 3.4. We see that the random forests technique generally has the highest success rates. It also gives larger false positive rates, but as discussed above, we believe many of these "false positives" are true Peer-to-Peer hosts disguising their traffic as Client, Server, or Unknown traffic.

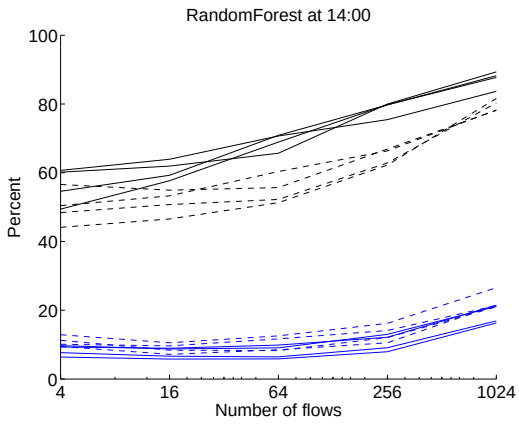
The dependence on n (the number of flows or seconds) is depicted in Figure 3.1, for random forests. We see that increasing the number of flows in the sample increases the success rate (and the false positive rate), while increasing the time scale of the sample has little effect.

3.3.4 Testing against the Memphis data set

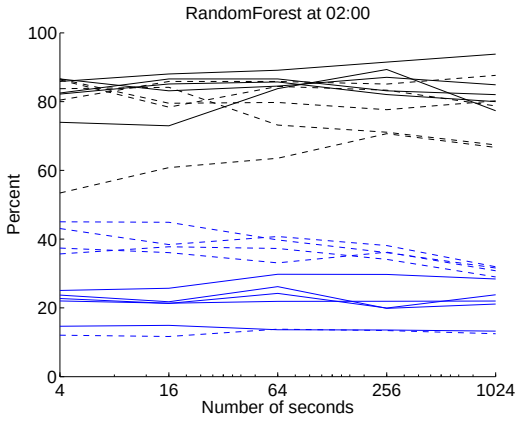
We compare Memphis data collected just after midnight on Saturday, March 31, 2006 with a Leipzig trace collected at 2am on Saturday, November 23, 2002. Because



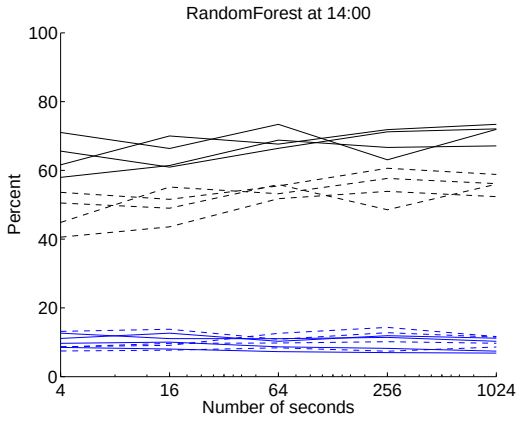
(a) Based on n flows, at 2am



(b) Based on n flows, at 2pm



(c) Based on n seconds of data, at 2am



(d) Based on n seconds of data, at 2pm

Figure 3.1: Sample of the results for the random forest classifier. Each line represents a different date. The upper (black) lines are success rates. The lower (blue) lines are false positive rates. The solid lines incorporate all features. The dashed lines incorporate only z_1, z_2, and the fraction of traffic initiated by the given host.

Table 3.3: The range of success and false positive rates, across different days, for a variety of classifiers, based on the first 16 flows from each host. The first line under each classifier is for the full feature set. The second line is for the limited feature set consisting of z_1, z_2, and the inbound/outbound traffic ratios.

	8pm		2am		8am		2pm	
Technique	**S%**	**FP%**	**S%**	**FP%**	**S%**	**FP%**	**S%**	**FP%**
Random forest	62-75	9-12	77-81	11-20	48-65	4-12	56-65	5-8
	50-67	13-15	71-79	14-29	41-55	4-13	43-57	7-11
Neural network	47-64	6-12	37-82	2-40	22-45	1-9	35-49	3-7
	37-54	16-19	44-78	18-55	15-37	3-17	16-33	4-11
Classification tree	45-56	5-7	58-66	5-12	27-43	2-7	32-46	4-5
	30-37	7-8	47-56	8-18	16-34	3-8	26-28	4-8
Linear discriminant	42-52	8-9	37-69	4-23	8-34	0-5	24-43	2-7
	33-42	10-12	30-59	7-33	1-27	0-6	13-28	4-7
Nearest neighbor	35-40	5-7	45-64	4-13	22-34	2-5	26-41	3-5
	19-31	7-9	30-53	7-18	13-20	2-7	15-21	4-7
Naive Bayes	27-41	4-4	24-57	4-8	16-29	1-3	23-35	2-4
	2-12	1-3	5-39	2-6	0-13	0-2	2-11	0-2

port number are not a reliable means of identification in the Memphis data set (see Section 3.3.1), we use Leipzig as the training data.

Specifically, we build a random forest on the Leipzig data just as in Section 3.3.3, then pass the Memphis data through the forest. We base the random forest on the first 64 seconds of data from each host. Less than 2% of the Memphis hosts had 16 or more flows, so we skip that comparative analysis.

Using the full feature set, 3% of the Memphis hosts were classified as Peer-to-Peer, 53% as Client, and 43% as Server.

Using the limited feature set consisting of z_1, z_2, and the inbound/outbound traffic ratios, which we consider to be less application-specific, 15% of the Memphis

Table 3.4: The range of success and false positive rates, across different days, for a variety of classifiers, based on the first 64 seconds of data from each host. The first line under each classifier is for the full feature set. The second line is for the limited feature set consisting of z_1, z_2, and the inbound/outbound traffic ratios.

	8pm		**2am**		**8am**		**2pm**	
Technique	**S%**	**FP%**	**S%**	**FP%**	**S%**	**FP%**	**S%**	**FP%**
Random forest	67-78	11-13	81-89	14-25	64-79	6-16	69-72	8-11
	44-65	11-13	66-84	14-40	51-75	7-24	49-62	8-12
Neural network	51-59	3-10	73-83	12-56	49-68	2-27	53-60	3-8
	55-64	25-31	62-93	16-60	57-70	10-27	48-63	16-27
Classification tree	38-51	6-9	56-71	7-16	41-59	2-9	45-50	5-7
	26-31	6-9	43-63	7-36	32-56	3-18	26-34	5-10
Linear discriminant	35-50	7-15	50-83	15-42	36-63	4-22	34-43	6-9
	34-47	17-28	47-88	17-65	35-49	9-23	20-48	11-23
Nearest neighbor	29-53	5-8	51-68	7-16	38-54	2-10	30-45	4-7
	23-30	10-12	39-59	17-24	27-37	6-13	25-33	7-13
Naive Bayes	18-39	3-4	37-53	4-9	36-56	3-6	33-38	3-4
	12-19	1-2	28-49	4-7	22-36	2-4	14-21	1-3

hosts were classified as Peer-to-Peer, 71% as Client, and 14% as Server.

We consider these results promising. The two data sets were collected more than three years apart, and we restrict attention to only interior hosts in the Leipzig data set, while this is not done in the Memphis data set. Nonetheless, a classifier built from the Leipzig hosts is sufficiently sensitive to distinguish different behaviors in the Memphis hosts, rather than classifying all Memphis hosts as, e.g., Clients.

3.4 Discussion

We have found that with minimal tweaking of the parameters, random forests can identify 80% or more of P2P hosts. One study [35] added a variety of adjustments to naive Bayes classifiers to improve performance from 65% to 95% on certain application categories; if they had started with a more powerful classifier such as random forests, their results might have been even greater.

Our success rates are comparable to those of related studies, given the limitations on our data. We must estimate true classes from port numbers, since we lack packet payloads. We also miss all peer activity taking place within the university; we see only communcations between interior and exterior hosts.

Chapter 4

Discrete analysis of crystal steps

4.1 Introduction

In Section 1.4, we discussed the importance of understanding crystal surface evolution, particularly the motion of steps. In this work, we develop an analytical description of step motion under a discrete model of the underlying physics.

Under continuum models, the crystal surface height is treated as a continuous function of horizontal position. Because physical steps have discrete heights, discrete models offer a more natural description of step behavior. Discrete models are still approximations, however, because the horizontal step positions are typically treated as continuous variables. In our discrete model analysis, we find an extra time dependence term ($t^{-1/2}$) in addition to the exponential dependence that emerges in continuum models.

We discuss the effect of the model's physical parameters on the step motion, and therefore the step bunching. Consider an equilibrium solution corresponding to equally spaced steps. When terrace width deviations tend to decay towards this solution, the behavior will remain stable. When the terrace widths exhibit growth away from equilibrium instead, step bunching can occur.

4.1.1 Related work

A discrete model for crystal steps was first formulated in the early 1950's [57], after Burton, Cabrera, and Frank noted the growth of crystals in solutions that were much less supersaturated than demanded by contemporary models. They concluded that the crystal surface holding the solution must not be smooth at the nanoscale. Such discrete models are now called "BCF models."

A variety of extensions to the original BCF model have been offered. Finite attachment-detachment rates at step edges play a large role in step behavior [66]. Step-step interactions based on the surface free energy were considered early on [63]; a variety of short- and long-range interactions have been considered since [67]. Modeling the crystal as a stack of concentric circular discs allows the introduction of curvature and step line tension [76]. Analyses of more general shapes have been limited to single-layer islands [58].

An electric field can also bias the terrace diffusion [80] by applying a direct current to the adatoms, which have a net charge due to their interactions with the crystal surface. The inclusion of an electric field in a BCF model [79, 77] is particularly important because application of a direct current is the most common method of inducing step bunching experimentally.

Step bunching [68] can appear in systems limited by the step edge attachment-detachment rates [66]. Experiments show that, under the application of a direct current, behavior of the bunching instability depends in a complex way on the system's temperature and on the direction of the applied current [70, 82]. Step bunching can

also be induced by deposition from the surrounding vapor, as shown experimentally [65] and numerically [72]. Finally, step curvature itself can also induce bunching in the model [61, 62].

In this work, we seek to understand step bunching of straight steps via analytical examination of the natural discrete equations, rather than by a continuum approximation [73] or numerically [72, 71, 78, 79]. We focus on the importance of the applied electric field.

Partial differential equations have been used to study step behavior, for example scaling laws for bunch widths. These PDEs can be derived directly from continuum versions of surface free energy and chemical potential [75, 76], or they can be derived as the continuum limit of discrete step equations [73]. The PDEs are highly nonlinear and tend to be of fourth order (e.g., equation (5) of [73]). Most analyses have employed linearized versions [74] for tractability. The linearization is performed about the solution corresponding to equally spaced steps (e.g., equation (6) of [62]). Growth of step bunches due to electromigration has been studied in a nonlinear context (e.g., equation (31) of [77]).

4.1.2 Overview

The remainder of this chapter is structured as follows. In Section 4.2, we construct a discrete model for the physics of step motion. In Section 4.3, we derive coupled equations for terrace widths based on this model. In Section 4.4, we find the linearization of this step motion equation. Finally, in Section 4.5, we find approximate solutions

to this linearized equation. A table of relevant parameters and variables appears in Appendix B.1.

4.2 Physical models of step motion

In this section, we derive a discrete model for step motion based on the underlying physics. We will analyze the behavior of a set of coupled linear ODEs, providing a contrast to the analyses of linear PDEs common in the literature.

4.2.1 Assumptions for the discrete model

We make the following assumptions about the geometry of our model:

(A3) The steps are straight (Figure 1.2(b))

(A4) The step sequence is monotone, downwards to the right

(A5) The step height a is constant

These assumptions permit a one-dimensional model of the crystal surface, with the position of the j^{th} step edge denoted $x_j(t)$. The j^{th} terrace has width $x_{j+1}(t) - x_j(t)$, or normalized width $u_j(t) = \frac{x_{j+1}(t)-x_j(t)}{L}$, where L is the average terrace width.

We also make the following assumptions about the kinetics.

(A6) Surface diffusion is faster than attachment-detachment (ADL kinetics)

(A7) Step-step interactions have energy $U \propto 1/(\text{terrace width})^2$ (e.g., entropic or elastic interactions)

(A8) There are no thermal fluctuations (low temperature)

Step motion proceeds from the motion of adsorbed atoms (adatoms) on the crystal surface. See Figure 4.1. Let $c_j(x,t)$ be the local concentration of adatoms on

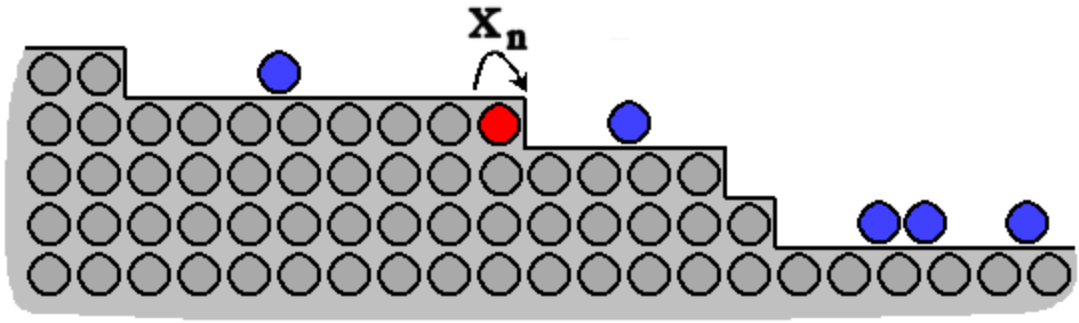


Figure 4.1: The crystal substrate is composed of bonded atoms. Adsorbed atoms (adatoms) interact with the surface, but move across it. If an adatom attaches at a step edge (x_n), the step edge advances by one lattice constant. Similarly, an adatom can detach from a step edge, causing the edge to retreat.

the j^{th} terrace, $x_j < x < x_{j+1}$. Also let $J_j(x,t)$ be the adatom current on the j^{th} terrace.

4.2.2 Discrete terrace equations

The model we derive here is very similar to the straight-step model in [62].

The step velocity law comes from mass conservation of adatoms near the step edge:

$$\frac{dx_j}{dt} = \frac{\Omega}{a}\left[-J_j(x_j^+) + J_{j-1}(x_j^-)\right], \tag{4.1}$$

where Ω is the atomic volume and a is the step height.

The adatom current obeys Fick's first law of diffusion and is also affected by electric drift with speed v_0:

$$J_j(x) = -D\frac{\partial c_j}{\partial x} + v_0 c_j. \tag{4.2}$$

The adatom density c_j satisfies Fick's second law and is also affected by the

desorption time τ and deposition rate R:

$$\frac{\partial c_j}{\partial t} = -\frac{\partial J_j}{\partial x} - \frac{1}{\tau}c_j + R \tag{4.3a}$$

$$= D\frac{\partial^2 c_j}{\partial x^2} - v_0\frac{\partial c_j}{\partial x} - \frac{1}{\tau}c_j + R. \tag{4.3b}$$

We will assume there is no deposition: $R = 0$.

For tractability, we make the quasi-steady approximation [57]

$$\frac{\partial c_j}{\partial t} \approx 0. \tag{4.4}$$

This approximation is not rigorous, but it agrees with physical observations. The justification is that step velocities (which depend on attachment-detachment) are much slower than terrace adatom diffusion, so the concentration c_j relaxes to steady-state between step adjustments.

4.2.3 Discrete step boundary conditions

We assume linear kinetics at both the left and right edges of the j^{th} terrace, that is,

$$J_j(x_j^+) = -k_+\left(c_j(x_j^+) - C_j^{\text{eq}}\right), \tag{4.5a}$$

$$J_j(x_{j+1}^-) = k_-\left(c_j(x_{j+1}^-) - C_{j+1}^{\text{eq}}\right), \tag{4.5b}$$

where k_+ and k_- are the attachment-detachment rate coefficients. If k_+,k_- are unequal, we say there is an "Ehrlich-Schwoebel barrier" at the step edge [78].

The equilibrium adatom density is given by the Arrhenius equation,

$$C_j^{\text{eq}} = c_0 e^{\mu_j/k_B T}. \tag{4.6}$$

Here μ_j is the step chemical potential, which is the change in surface free energy when an adatom attaches at the j^{th} step edge. Positive values of μ_j indicate a propensity

for a step edge to emit adatoms, while negative values indicate a propensity to accept adatoms.

Because we are ignoring step curvature (Assumption A3), the chemical potential depends only on step-step interactions. Assuming nearest-neighbor effects, we have

$$\mu_j \propto \frac{d}{dx_j}\left[U(x_j, x_{j+1}) + U(x_{j-1}, x_j)\right], \tag{4.7}$$

where, from Assumption A7,

$$U(x_j, x_{j+1}) \propto \frac{1}{(x_{j+1} - x_j)^2}. \tag{4.8}$$

Then, subsuming the constants of proportionality into g, we have

$$\frac{\mu_j}{k_B T} = g\left[\left(\frac{L}{x_{j+1} - x_j}\right)^3 - \left(\frac{L}{x_j - x_{j-1}}\right)^3\right]. \tag{4.9}$$

g is a dimensionless constant that describes the strength of the nearest-neighbor step-step interactions.

4.3 Step motion equation from the model

Here we find an equation for $\frac{du_j}{dt}$ that depends explicitly only on the terrace widths. We will see that when steps interact, $\frac{du_j}{dt}$ depends on the terraces two neighbors away, but when step-step interactions are eliminated, $\frac{du_j}{dt}$ depends only on the terraces one neighbor away.

4.3.1 Adatom density and adatom current

Under the quasi-steady approximation, the PDE in equation (4.3) becomes an ODE:

$$D\frac{\partial^2 c_j}{\partial x^2} - v_0\frac{\partial c_j}{\partial x} - \frac{1}{\tau}c_j = 0, \tag{4.10}$$

which has solutions

$$c_j(x) = B_j^+ e^{R_+(x-x_j)/L} + B_j^- e^{R_-(x-x_j)/L} \tag{4.11}$$

where $B_j^\pm$ will be determined by boundary conditions, and

$$R_\pm = \frac{Lv_0}{2D} \pm \frac{Lv_0}{2D}\sqrt{1+4\eta^2}, \tag{4.12}$$

$$\eta = \frac{\sqrt{D\tau}}{v_0\tau} \tag{4.13}$$

are dimensionless constants.

Substituting equation (4.11) into equation (4.2) gives

$$J_j(x) = -q_+ B_j^+ e^{R_+(x-x_j)/L} - q_- B_j^- e^{R_-(x-x_j)/L}, \tag{4.14}$$

where

$$q_\pm = \frac{D}{L}R_\pm - v_0 = -\frac{v_0}{2} \pm \frac{v_0}{2}\sqrt{1+4\eta^2}. \tag{4.15}$$

We will find it convenient to set

$$q_{\pm}^{+} = q_{\pm} + k_{-}, \quad q_{\pm}^{-} = q_{\pm} - k_{+}. \tag{4.16}$$

Substituting into equation (4.5) from equations (4.14) and (4.11) gives

$$B_j^{+} q_{+}^{-} + B_j^{-} q_{-}^{-} = -k_{+} C_j^{\mathrm{eq}}, \tag{4.17a}$$

$$B_j^{+} q_{+}^{+} e^{R_{+} u_j} + B_j^{-} q_{-}^{+} e^{R_{-} u_j} = k_{-} C_{j+1}^{\mathrm{eq}}, \tag{4.17b}$$

where

$$u_j = \frac{x_{j+1} - x_j}{L} \tag{4.18}$$

is the normalized width of step j. Now we can solve for $B_j^{\pm}$, finding

$$B_j^{\pm} = \pm d(u_j) \left[k_{-} q_{\mp}^{-} C_{j+1}^{\mathrm{eq}} + k_{+} q_{\mp}^{+} e^{R_{\mp} u_j} C_j^{\mathrm{eq}} \right], \tag{4.19}$$

where

$$d(u) = \frac{1}{q_{+}^{+} q_{-}^{-} e^{R_{+} u} - q_{+}^{-} q_{-}^{+} e^{R_{-} u}}. \tag{4.20}$$

Finally, from equation (4.6), we have

$$C_j^{\mathrm{eq}} = c_0 e^{g\left(u_j^{-3} - u_{j-1}^{-3}\right)} \tag{4.21}$$

where $g \ll 1$ is the strength of step interactions.

4.3.2 Terrace width

Via equation (4.1), the normalized terrace width satisfies

$$\dot{u}_j = \frac{\dot{x}_{j+1} - \dot{x}_j}{L} \tag{4.22a}$$

$$= \frac{\Omega}{La} \Big[-J_{j+1}(x_{j+1}) + \left(J_j(x_{j+1}) + J_j(x_j) \right) - J_{j-1}(x_j) \Big] \tag{4.22b}$$

$$= \frac{\Omega}{La}\Big[+q_+B^+_{j+1} + q_-B^-_{j+1}$$

$$- q_+B^+_j(1+e^{R_+u_j}) - q_-B^-_j(1+e^{R_-u_j})$$

$$+ q_+B^+_{j-1}e^{R_+u_{j-1}} + q_-B^-_{j-1}e^{R_-u_{j-1}}\Big] \tag{4.22c}$$

$$= \frac{\Omega}{La}\Big[+C^{\rm eq}_{j+2}\left[d(u_{j+1})k_-(q_+q^-_- - q_-q^-_+)\right]$$

$$+ C^{\rm eq}_{j+1}\left[d(u_{j+1})k_+(q_+q^+_-e^{R_-u_{j+1}} - q_-q^+_+e^{R_+u_{j+1}})\right.$$

$$\left.-d(u_j)k_-(q_+q^-_-(1+e^{R_+u_j}) - q_-q^-_+(1+e^{R_-u_j}))\right]$$

$$+ C^{\rm eq}_j\left[-d(u_j)k_+(q_+q^+_-e^{R_-u_j}(1+e^{R_+u_j}) - q_-q^+_+e^{R_+u_j}(1+e^{R_-u_j}))\right.$$

$$\left.+d(u_{j-1})k_-(q_+q^-_-e^{R_+u_{j-1}} - q_-q^-_+e^{R_-u_{j-1}})\right]$$

$$\left.+ C^{\rm eq}_{j-1}\left[d(u_{j-1})k_+(q_+q^+_-e^{R_-u_{j-1}}e^{R_+u_{j-1}} - q_-q^+_+e^{R_+u_{j-1}}e^{R_-u_{j-1}}))\right]\right] \tag{4.22d}$$

$$= \frac{\Omega}{La}c_0\Big[e^{g\left(u^{-3}_{j+2}-u^{-3}_{j+1}\right)}\left[-d(u_{j+1})k_+k_-(q_+-q_-)\right]$$

$$+ e^{g\left(u^{-3}_{j+1}-u^{-3}_j\right)}\left[+d(u_{j+1})k_+(-q_-q^+_+e^{R_+u_{j+1}} + q_+q^+_-e^{R_-u_{j+1}})\right.$$

$$\left.-d(u_j)k_-(q_+q^-_-(1+e^{R_+u_j}) - q_-q^-_+(1+e^{R_-u_j}))\right]$$

$$+ e^{g\left(u^{-3}_j-u^{-3}_{j-1}\right)}\left[-d(u_j)k_+(k_-(q_+-q_-)e^{(R_++R_-)u_j}\right.$$

$$\left.- q_-q^+_+e^{R_+u_j} + q_+q^+_-e^{R_-u_j})\right.$$

$$\left.+d(u_{j-1})k_-(q_+q^-_-e^{R_+u_{j-1}} - q_-q^-_+e^{R_-u_{j-1}})\right]$$

$$\left.+ e^{g\left(u^{-3}_{j-1}-u^{-3}_{j-2}\right)}\left[+d(u_{j-1})k_+k_-(q_+-q_-)e^{(R_++R_-)u_{j-1}}\right]\right]. \tag{4.22e}$$

Therefore, $\frac{du_j}{dt}$ depends on the two terraces to either side: u_j, u_{j+1}, u_{j-1}, and u_{j+2}, u_{j-2}. However, the dependence on u_{j+2} and u_{j-2} is only through the step-step interactions. If $g = 0$, $\frac{du_j}{dt}$ depends only on u_{j+1}, u_j, u_{j-1}.

4.4 Linearization

To make progress with the solution of equation 4.22e, we find the linearization. The coefficients of u_{j+2} and u_{j-2} will be $\mathcal{O}(g)$, reflecting that the two-away neighbors only affect the step motion through step-step interactions. We then investigate the dependence of all the coefficients on the physical parameters, in the limiting cases of very strong and very weak electric field.

4.4.1 Subordinate functions

We will linearize around the solution corresponding to equally spaced steps, that is, $u_j = 1$ for all j. We define the deviation

$$\tilde{u}_j = u_j - 1 \tag{4.23}$$

so we may consider $|\tilde{u}_j| \ll 1$. Several parts of equation (4.22e) must be linearized.

$$e^{Ku_j} \sim e^K + Ke^K\tilde{u}_j + \mathcal{O}\left(\tilde{u}_j^2\right), \tag{4.24}$$

$$u_j^{-3} - u_{j-1}^{-3} \sim 3\tilde{u}_{j-1} - 3\tilde{u}_j + \mathcal{O}\left(\tilde{u}_j^2 + \tilde{u}_{j-1}^2\right), \tag{4.25}$$

$$d(u_j) \sim d_0 + d_1\tilde{u}_j + \mathcal{O}\left(\tilde{u}_j^2\right), \tag{4.26}$$

where

$$d_0 = \frac{1}{q_+^+q_-^-e^{R_+} - q_+^-q_-^+e^{R_-}}, \quad d_1 = \frac{-q_+^+q_-^-R_+e^{R_+} + q_+^-q_-^+R_-e^{R_-}}{\left(q_+^+q_-^-e^{R_+} - q_+^-q_-^+e^{R_-}\right)^2}. \tag{4.27}$$

We note that

$$d_0R_+ + d_1 = -d_0^2(R_+ - R_-)q_+^-q_-^+e^{R_-}, \tag{4.28a}$$

$$d_0 R_- + d_1 = -d_0^2 (R_+ - R_-) q_+^+ q_-^- e^{R_+}, \tag{4.28b}$$

$$d_0 (R_+ + R_-) + d_1 = +d_0^2 (q_+^+ q_-^- R_- e^{R_+} - q_+^- q_-^+ R_+ e^{R_-}) \tag{4.28c}$$

4.4.2 Terrace width evolution

We now find the linearization of $\dot{\tilde{u}}_j = \dot{u}_j$. We require each of $|\tilde{u}_j| \ll 1$, $|g\tilde{u}_j| \ll 1$, and $|R_\pm \tilde{u}_j| \ll 1$ for all j.

$$\begin{aligned}
\dot{\tilde{u}}_j \sim \frac{\Omega}{La} c_0 \Big[& (1 - 3g\tilde{u}_{j+2} + 3g\tilde{u}_{j+1}) \left[-(d_0 + d_1 \tilde{u}_{j+1})(k_+ k_- (q_+ - q_-)) \right] \\
& + (1 - 3g\tilde{u}_{j+1} + 3g\tilde{u}_j) \big[+(d_0 + d_1 \tilde{u}_{j+1})(k_+(-q_- q_+^+ e^{R_+} + q_+ q_-^+ e^{R_-}) \\
& \qquad + k_+(-q_- q_+^+ R_+ e^{R_+} + q_+ q_-^+ R_- e^{R_-}) \tilde{u}_{j+1}) \\
& \qquad - (d_0 + d_1 \tilde{u}_j)(k_-(q_+ q_-^- (1 + e^{R_+}) - q_- q_+^- (1 + e^{R_-})) \\
& \qquad + k_-(q_+ q_-^- R_+ e^{R_+} - q_- q_+^- R_- e^{R_-}) \tilde{u}_j) \big] \\
& + (1 - 3g\tilde{u}_j + 3g\tilde{u}_{j-1}) \big[-(d_0 + d_1 \tilde{u}_j)(k_+(k_-(q_+ - q_-) e^{R_+ + R_-} \\
& \qquad - q_- q_+^+ e^{R_+} + q_+ q_-^+ e^{R_-}) \\
& \qquad + k_+(k_-(R_+ + R_-)(q_+ - q_-) e^{R_+ + R_-} \\
& \qquad \qquad - q_- q_+^+ R_+ e^{R_+} + q_+ q_-^+ R_- e^{R_-}) \tilde{u}_j) \\
& \qquad + (d_0 + d_1 \tilde{u}_{j-1})(k_-(q_+ q_-^- e^{R_+} - q_- q_+^- e^{R_-}) \\
& \qquad + k_-(q_+ q_-^- R_+ e^{R_+} - q_- q_+^- R_- e^{R_-}) \tilde{u}_{j-1}) \big] \\
& + (1 - 3g\tilde{u}_{j-1} + 3g\tilde{u}_{j-2}) \big[+(d_0 + d_1 \tilde{u}_{j-1})(k_+ k_-(q_+ - q_-) e^{R_+ + R_-} \\
& \qquad + k_+ k_-(R_+ + R_-)(q_+ - q_-) e^{R_+ + R_-} \tilde{u}_{j-1}) \big] \Big] + \mathcal{O}\left(\tilde{u}_\cdot^2\right)
\end{aligned} \tag{4.29a}$$

$$\sim \frac{\Omega}{La} c_0 \Big[(1 - 3g\tilde{u}_{j+2} + 3g\tilde{u}_{j+1})$$

$$
\begin{aligned}
&\big[-d_0 k_+ k_-(q_+ - q_-) \\
&\quad - d_1 k_+ k_-(q_+ - q_-)\tilde{u}_{j+1}\big] \\
&+ (1 - 3g\tilde{u}_{j+1} + 3g\tilde{u}_j) \\
&\quad \big[+d_0 \left(k_+ k_-(q_+ - q_-)\left(1 + e^{R_+} + e^{R_-}\right)\right. \\
&\qquad \left. -(k_+ + k_-)q_+ q_- \left(e^{R_+} - e^{R_-}\right)\right) \\
&\quad + d_0^2 k_+^2 (R_+ - R_-)(q_+ - q_-) q_+^+ q_-^+ e^{R_+ + R_-}\tilde{u}_{j+1} \\
&\quad + k_-(q_+ - q_-)\left(d_0^2 k_-(R_+ - R_-) q_+^- q_-^- e^{R_+ + R_-} + d_1 k_+\right)\tilde{u}_j\big] \\
&+ (1 - 3g\tilde{u}_j + 3g\tilde{u}_{j-1}) \\
&\quad \big[-d_0 \left(k_+ k_-(q_+ - q_-)\left(e^{R_+ + R_-} + e^{R_+} + e^{R_-}\right)\right. \\
&\qquad \left. -(k_+ + k_-)q_+ q_- \left(e^{R_+} - e^{R_-}\right)\right) \\
&\quad - k_+(q_+ - q_-)\left(d_0^2 k_+(R_+ - R_-) q_+^+ q_-^+\right. \\
&\qquad \left. + k_-(d_0(R_+ + R_-) + d_1)\right) e^{R_+ + R_-}\tilde{u}_j \\
&\quad - d_0^2 k_-^2 (R_+ - R_-)(q_+ - q_-) q_+^- q_-^- e^{R_+ + R_-}\tilde{u}_{j-1}\big] \\
&+ (1 - 3g\tilde{u}_{j-1} + 3g\tilde{u}_{j-2}) \\
&\quad \Big[\big[+d_0 k_+ k_-(q_+ - q_-) e^{R_+ + R_-} \\
&\quad + (d_0(R_+ + R_-) + d_1) k_+ k_-(q_+ - q_-) e^{R_+ + R_-}\tilde{u}_{j-1}\big]\Big] + \mathcal{O}\left(\tilde{u}_\cdot^2\right)
\end{aligned}
\tag{4.29b}
$$

$$
\begin{aligned}
\sim \frac{\Omega}{La} c_0 \Big[& \tilde{u}_{j+2}\big[+3g d_0 k_+ k_-(q_+ - q_-)\big] \\
& + \tilde{u}_{j+1}\big[(q_+ - q_-) d_0^2 (k_+^2 (R_+ - R_-) q_+^+ q_-^+ e^{R_+ + R_-} \\
& \qquad + k_+ k_-(q_+^+ q_-^- R_+ e^{R_+} - q_+^- q_-^+ R_- e^{R_-}))
\end{aligned}
$$

$$
\begin{aligned}
&- 3gd_0(k_+k_-(q_+ - q_-)(e^{R_+} + e^{R_-} + 2) \\
&\qquad - (k_+ + k_-)q_+q_-(e^{R_+} - e^{R_-}))\big] \\
&+ \tilde{u}_j\big[(q_+ - q_-)d_0^2\big(-(R_+ - R_-)(k_+^2 q_+^+ q_-^+ - k_-^2 q_+^- q_-^-)e^{R_+ + R_-} \\
&\qquad - k_+k_-((q_+^+ q_-^- R_- e^{R_+} - q_+^- q_-^+ R_+ e^{R_-})e^{R_+ + R_-} \\
&\qquad\qquad + (q_+^+ q_-^- R_+ e^{R_+} - q_+^- q_-^+ R_- e^{R_-}))\big) \\
&\qquad + 3gd_0(k_+k_-(q_+ - q_-)(e^{R_+ + R_-} + 2e^{R_+} + 2e^{R_-} + 1) \\
&\qquad - 2(k_+ + k_-)q_+q_-(e^{R_+} - e^{R_-}))\big] \\
&+ \tilde{u}_{j-1}\big[(q_+ - q_-)d_0^2 e^{R_+ + R_-}(-k_-^2(R_+ - R_-)q_+^- q_-^- \\
&\qquad + k_+k_-(q_+^+ q_-^- R_- e^{R_+} - q_+^- q_-^+ R_+ e^{R_-})) \\
&\qquad - 3gd_0(k_+k_-(q_+ - q_-)(2e^{R_+ + R_-} + e^{R_+} + e^{R_-}) \\
&\qquad - (k_+ + k_-)q_+q_-(e^{R_+} - e^{R_-}))\big] \\
&+ \tilde{u}_{j-2}\big[+3gd_0k_+k_-(q_+ - q_-)e^{R_+ + R_-}\big]\Big] + \mathcal{O}\left(\tilde{u}_{\cdot}^2\right). \qquad (4.29c)
\end{aligned}
$$

Note that all affine terms have cancelled out.

We will typically use the more compact notation

$$
\begin{aligned}
\dot{\tilde{u}}_j \sim\ & (g\beta_2)\,\tilde{u}_{j+2} + (\alpha_1 + g\beta_1)\,\tilde{u}_{j+1} + (\alpha_0 + g\beta_0)\,\tilde{u}_j \\
& + (\alpha_{-1} + g\beta_{-1})\,\tilde{u}_{j-1} + (g\beta_{-2})\,\tilde{u}_{j-2}, \qquad (4.30)
\end{aligned}
$$

where

$$
\begin{aligned}
\alpha_1 = \frac{\Omega c_0}{La}&(q_+ - q_-)d_0^2(k_+^2(R_+ - R_-)q_+^+ q_-^+ e^{R_+ + R_-} \\
&+ k_+k_-(q_+^+ q_-^- R_+ e^{R_+} - q_+^- q_-^+ R_- e^{R_-})), \qquad (4.31a) \\
\alpha_{-1} = \frac{\Omega c_0}{La}&(q_+ - q_-)d_0^2 e^{R_+ + R_-}(-k_-^2(R_+ - R_-)q_+^- q_-^-
\end{aligned}
$$

$$+ k_+k_-(q_+^+q_-^-R_-e^{R_+} - q_+^-q_-^+R_+e^{R_-})), \tag{4.31b}$$

$$\alpha_0 = -(\alpha_1 + \alpha_{-1}); \tag{4.31c}$$

$$\beta_2 = 3\frac{\Omega c_0}{La}d_0k_+k_-(q_+ - q_-), \tag{4.31d}$$

$$\beta_{-2} = 3\frac{\Omega c_0}{La}d_0k_+k_-(q_+ - q_-)e^{R_+ + R_-}, \tag{4.31e}$$

$$\begin{aligned}\beta_1 = -3\frac{\Omega c_0}{La}d_0(&k_+k_-(q_+ - q_-)(e^{R_+} + e^{R_-} + 2)\\ &- (k_+ + k_-)q_+q_-(e^{R_+} - e^{R_-})),\end{aligned} \tag{4.31f}$$

$$\begin{aligned}\beta_{-1} = -3\frac{\Omega c_0}{La}d_0(&k_+k_-(q_+ - q_-)(2e^{R_+ + R_-} + e^{R_+} + e^{R_-})\\ &- (k_+ + k_-)q_+q_-(e^{R_+} - e^{R_-})),\end{aligned} \tag{4.31g}$$

$$\beta_0 = -\left(\beta_2 + \beta_1 + \beta_{-1} + \beta_{-2}\right). \tag{4.31h}$$

4.4.3 Dependence on the physical parameters

Here we examine the behavior of each of the coefficients α_n, β_n. A summary of the physical parameters appears in Appendix B.1. We should always have

(A9) $L > 0$, $a > 0$, $\Omega > 0$, $c_0 > 0$, $\tau > 0$, $D > 0$, $k_+ > 0$, $k_- > 0$

We will also assume

(A10) $v_0 > 0$, $g \geq 0$

Then we have $q_+ > 0 > q_-$ and $R_+ > 0 > R_-$.

From these assumptions, we first note that

$$d_0 = \left[q_+^+q_-^-e^{R_+} - q_+^-q_-^+e^{R_-}\right]^{-1} \tag{4.32a}$$

$$= \big[(q_+q_- - k_+q_+ + k_-q_- - k_+k_-)e^{R_+}$$

$$- (q_+q_- + k_-q_+ - k_+q_- - k_+k_-)e^{R_-}\big]^{-1} \tag{4.32b}$$

$$= \big[(q_+q_- - k_+k_-)\left(e^{R_+} - e^{R_-}\right)$$

$$- q_+\left(k_+e^{R_+} + k_-e^{R_-}\right) + q_-\left(k_-e^{R_+} + k_+e^{R_-}\right)\big]^{-1} \tag{4.32c}$$

$$< 0. \tag{4.32d}$$

Holding all the parameters $L, a, \Omega, c_0, \tau, D, k_+, k_-$ constant, and letting v_0 vary in $(0, \infty)$, we have

$$R_+ \sim \begin{cases} Lv_0/D & \text{for } v_0 \gg 1 \\ L/\sqrt{D\tau} & \text{for } v_0 \ll 1, \end{cases} \qquad R_- \sim \begin{cases} -L/(v_0\tau) & \text{for } v_0 \gg 1 \\ -L/\sqrt{D\tau} & \text{for } v_0 \ll 1, \end{cases} \tag{4.33a}$$

$$q_+ \sim \begin{cases} D/(v_0\tau) & \text{for } v_0 \gg 1 \\ \sqrt{D/\tau} & \text{for } v_0 \ll 1, \end{cases} \qquad q_- \sim \begin{cases} -v_0 & \text{for } v_0 \gg 1 \\ -\sqrt{D/\tau} & \text{for } v_0 \ll 1. \end{cases} \tag{4.33b}$$

Further assuming that $\delta = L/\sqrt{D\tau} \ll 1$, we have

$$d_0 \sim \begin{cases} -\left[k_-v_0e^{Lv_0/D}\right]^{-1} & \text{for } v_0 \gg 1 \\ -\left[2(k_+ + k_-)\sqrt{D/\tau}\right]^{-1} + \mathcal{O}(\delta) & \text{for } v_0 \ll 1. \end{cases} \tag{4.34}$$

4.4.3.1 Non-interaction terms ($\mathcal{O}(1)$)

From equation (4.31a), we can calculate

$$\alpha_1 \sim \frac{\Omega c_0}{La} \cdot \begin{cases} -\frac{k_+(k_+ + k_-)}{k_-}\frac{Lv_0}{D}e^{-Lv_0/D} & \text{for } v_0 \gg 1 \\ -\delta\sqrt{\frac{D}{\tau}}\frac{k_+}{k_+ + k_-} & \text{for } v_0 \ll 1. \end{cases} \tag{4.35}$$

From equation (4.31b), we can calculate

$$\alpha_{-1} \sim \frac{\Omega c_0}{La} \cdot \begin{cases} k_+ \frac{L}{v_0 \tau} & \text{for } v_0 \gg 1 \\ \delta \sqrt{\frac{D}{\tau}} \frac{k_-}{k_+ + k_-} & \text{for } v_0 \ll 1. \end{cases} \tag{4.36}$$

When the electric field is very strong, $\alpha_1 \to 0^-$ and $\alpha_{-1} \to 0^+$, with α_1 decaying faster in v_0 than α_{-1}.

In the case where there is no Ehrlich-Schwoebel effect, that is, $k_+ = k_-$, then, when the electric field is very weak, we have $\alpha_1 \approx -\alpha_{-1}$, and $\alpha_0 \approx 0$.

4.4.3.2 Step-step interaction terms ($\mathcal{O}(g)$)

From equation (4.31d), we know $\beta_2 < 0$ for all valid choices of the physical parameters. We can calculate

$$\beta_2 \sim \frac{\Omega c_0}{La} \cdot \begin{cases} -3k_+ e^{-Lv_0/D} & \text{for } v_0 \gg 1 \\ -3\frac{k_+ k_-}{k_+ + k_-} & \text{for } v_0 \ll 1. \end{cases} \tag{4.37}$$

From equation (4.31e), we know it is also true that $\beta_{-2} < 0$ for all valid choices of the physical parameters. We can calculate

$$\beta_{-2} \sim \frac{\Omega c_0}{La} \cdot \begin{cases} -3k_+ & \text{for } v_0 \gg 1 \\ -3\frac{k_+ k_-}{k_+ + k_-} & \text{for } v_0 \ll 1. \end{cases} \tag{4.38}$$

From equation (4.31f), we know $\beta_1 > 0$ for all valid choices of the physical parameters. We can calculate

$$\beta_1 \sim \frac{\Omega c_0}{La} \cdot \begin{cases} 3k_+ & \text{for } v_0 \gg 1 \\ 12\frac{k_+ k_-}{k_+ + k_-} & \text{for } v_0 \ll 1. \end{cases} \tag{4.39}$$

From equation (4.31g), we know $\beta_{-1} > 0$ for all valid choices of the physical parameters as well. We can calculate

$$\beta_{-1} \sim \frac{\Omega c_0}{La} \cdot \begin{cases} 9k_+ & \text{for } v_0 \gg 1 \\ 12\frac{k_+k_-}{k_++k_-} & \text{for } v_0 \ll 1. \end{cases} \tag{4.40}$$

When the electric field is very strong, we have $\beta_2 \approx \beta_{-2}$, while $\beta_{-1} \approx 3\beta_1$.

When the electric field is very weak, we have $\beta_2 \approx \beta_{-2}$, and $\beta_1 \approx \beta_{-1}$.

4.5 Solutions to linearized equation

We transform the linearized step motion equation to simplify the coupling, thus finding an integral expression for $\tilde{u}_j(t)$. We then use the method of steepest descents to estimate this integral for long times t.

4.5.1 Transformation

Emulating the solution to a difference equation in [59], we assume that $\tilde{u}_j(t)$ takes the form

$$\tilde{u}_j(t) = \int_{C(t)} e^{ijz} f(z,t) dz \tag{4.41}$$

for all j, t, for some fixed function $f(z,t)$ and contours $C(t)$. Writing the endpoints of $C(t)$ as $C_0(t), C_1(t)$, and assuming we may interchange integration and differentiation, the Leibniz integral rule gives

$$\dot{\tilde{u}}_j(t) = \int_{C(t)} e^{ijz} \frac{\partial f}{\partial t}(z,t) dz + C_1'(t) e^{ijC_1(t)} f(C_1(t),t) - C_0'(t) e^{ijC_0(t)} f(C_0(t),t). \tag{4.42}$$

To eliminate the last two terms, we require the endpoint contribution of the integrand to vanish, so $e^{ijz} f(z,t) \to 0$ in measure as $z \to a(t)$, $z \to b(t)$.

For convenience, we define

$$h(z) = (g\beta_2)e^{2iz} + (\alpha_1 + g\beta_1)e^{iz} + (\alpha_0 + g\beta_0) + (\alpha_{-1} + g\beta_{-1})e^{-iz} + (g\beta_{-2})e^{-2iz}. \tag{4.43}$$

Now, substituting equation (4.41) into equation (4.30) gives

$$\int_{C(t)} e^{ijz} \frac{\partial f}{\partial t}(z,t) dz \sim \int_{C(t)} e^{ijz} h(z) f(z,t) dz. \tag{4.44}$$

For this equation to hold for all j, $f(z,t)$ should satisfy the differential equation

$$\frac{\partial f}{\partial t}(z,t) = h(z) f(z,t), \tag{4.45}$$

which has solutions

$$f(z,t) = \mathcal{U}(z)e^{th(z)}. \tag{4.46}$$

We will use the requirement $\mathcal{U}(x) \to 0$ as $x \to \pm\infty$ to ensure the vanishing endpoint contributions in equation 4.42. Then we can take $C(t) = \mathbb{R}$ for all t, so that equation 4.41 is a simple Fourier transform. The other factors e^{ijz} and $e^{th(z)}$ are oscillatory along this contour, confirming decay at the endpoints.

4.5.2 Initial data

$\mathcal{U}(z)$ can be determined by the initial data of the problem. We have $\tilde{u}_j(0) = \int_{\mathbb{R}} e^{ijz} f(z,0)dz = \int_{\mathbb{R}} e^{ijz}\mathcal{U}(z)dz$. Treating j as a real (rather than integer) parameter, we see that $\mathcal{U}(z)$ is the inverse Fourier transform of $\tilde{u}_j(0)$:

$$\mathcal{U}(z) = \frac{1}{2\pi}\int_{\mathbb{R}} e^{-izj}\tilde{u}_j(0)dj. \tag{4.47}$$

To represent a valid physical system, $\tilde{u}_j(0)$ must be continuous in j, the average of the $\tilde{u}_j(0)$ over all $j \in \mathbb{Z}$ must equal 0, and we must have $\tilde{u}_j(0) \geq -1$ for all j. We also restrict ourselves to $\tilde{u}_j(0)$ such that $\mathcal{U}(z) \to 0$ as $z \to \pm\infty$ in a neighborhood of the real axis.

Later, we will need to use the fact that that $\mathcal{U}(-\overline{z}) = \overline{\mathcal{U}(z)}$, which is easily verified:

$$\mathcal{U}(-\overline{z}) = \frac{1}{2\pi}\int_{\mathbb{R}} e^{ij\overline{z}}\tilde{u}_j(0)dj \tag{4.48a}$$

$$= \frac{1}{2\pi}\int_{\mathbb{R}} \overline{e^{-ijz}}\tilde{u}_j(0)dj \tag{4.48b}$$

$$= \frac{1}{2\pi}\int_{\mathbb{R}} \overline{e^{-ijz}\tilde{u}_j(0)}dj \tag{4.48c}$$

$$= \overline{\mathcal{U}(z)} \tag{4.48d}$$

As a consequence, $\mathcal{U}$ maps the imaginary axis to the real axis, since $\mathcal{U}(iy) = \mathcal{U}(-\overline{iy}) = \overline{\mathcal{U}(iy)}$.

As an example, we might choose $\tilde{u}_j(0;\epsilon) = je^{-|j|/\epsilon}$, which yields $\mathcal{U}(z;\epsilon) = -\frac{2i}{\pi}\frac{\epsilon^3 z}{(1+(\epsilon z)^2)^2}$. Since $\tilde{u}_j(0;\epsilon)$ is an odd function of j, we are guaranteed that

$$\frac{1}{2N+1}\sum_{j=-N}^{N}\tilde{u}_j(0;\epsilon) = 0 \tag{4.49}$$

for all N, including $N \to \infty$.

4.5.3 Steepest descents

From equations (4.41) and (4.46), we have

$$\tilde{u}_j(t) = \int_{\mathbb{R}} \mathcal{U}(z)e^{ijz}e^{th(z)}dz, \tag{4.50}$$

where $\mathcal{U}(z)$ is given by equation (4.47). In each case below, we will consider the complex plane restricted to $x \in (-\pi, \pi]$ to determine the relevant critical point(s) and contour of integration. This region should be repeated at intervals of $2n\pi$ ($n \in \mathbb{Z}$) to deform the entire real line $\mathbb{R}$. e^{ijz} and $e^{th(z)}$ are periodic in Real (z) with period 2π, so only the value of $\mathcal{U}(z)$ will differ between regions.

We are interested in the long-term behavior of the terrace widths, so we assume $t \gg 1$. We consider steps "in the bulk" of the crystal, where $\left|\frac{j}{t}\right| \ll 1$. To approximate the above integral using Riemann's method of steepest descents, we must locate the critical points where $h'(z) = 0$. From $h(z)$ in equation (4.43), we have

$$h'(z) = i\left(2(g\beta_2)e^{2iz} + (\alpha_1 + g\beta_1)e^{iz} - (\alpha_{-1} + g\beta_{-1})e^{-iz} - 2(g\beta_{-2})e^{-2iz}\right). \tag{4.51}$$

Letting $w = e^{iz}$, we know that w cannot equal 0. Therefore, $h'(z) = 0$ if and only if w is a nonzero root of the polynomial

$$p_g(w) = (2g\beta_2)w^4 + (\alpha_1 + g\beta_1)w^3 - (\alpha_{-1} + g\beta_{-1})w - (2g\beta_{-2}). \tag{4.52}$$

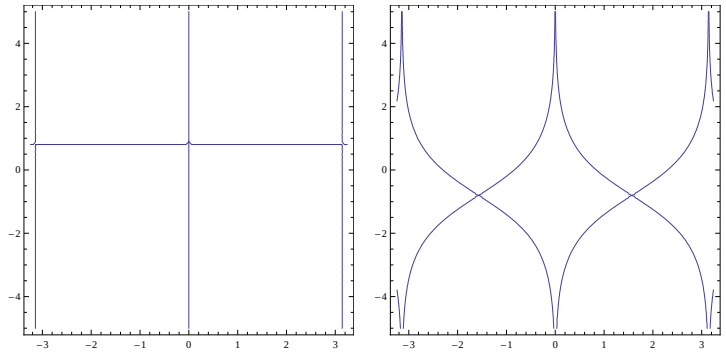
Suppose we have determined a contour C passing through a unique maximum critical point z_0, where $h'(z_0) = 0$. We can approximate equation (4.50) by making the change of variable

$$\zeta^2 = h(z_0) - h(z) \sim -\frac{1}{2}h''(z_0)(z - z_0)^2 + \mathcal{O}\left((z - z_0)^3\right) \tag{4.53}$$

so that $z \sim z_0 + \left[-\frac{1}{2}h''(z_0)\right]^{-1/2}\zeta + \mathcal{O}(\zeta^2)$ and $\frac{dz}{d\zeta} \sim \left[-\frac{1}{2}h''(z_0)\right]^{-1/2} + \mathcal{O}(\zeta)$. (Here we must fix a branch of the square root function. We choose it such that $\sqrt{\overline{z}} = \overline{\sqrt{z}}$ for all z not on the negative real axis.) Then

$$\begin{aligned}
\tilde{u}_j(t) &= \int_C \mathcal{U}(z)e^{ijz}e^{th(z)}dz && \text{(4.54a)}\\
&\sim \int_{\mathbb{R}} \mathcal{U}(z(\zeta))e^{ijz(\zeta)}e^{t(h(z_0)-\zeta^2)}\frac{dz}{d\zeta}d\zeta && \text{(4.54b)}\\
&\sim \mathcal{U}(z_0)e^{ijz_0}e^{th(z_0)}\left[-\frac{1}{2}h''(z_0)\right]^{-1/2}\int_{\mathbb{R}} e^{-t\zeta^2}d\zeta && \text{(4.54c)}\\
&= \mathcal{U}(z_0)\cdot\sqrt{\pi}\left[-\frac{1}{2}h''(z_0)\right]^{-1/2}\cdot(e^{iz_0})^j\cdot e^{th(z_0)}t^{-1/2}. && \text{(4.54d)}
\end{aligned}$$

The factor $t^{-1/2}$ is a departure from continuum estimations. It is the fastest decay we could expect to see in an asymptotic expansion. If the first non-zero derivative of h were of higher order ($h'(z_0) = h''(z_0) = \cdots = h^{(m-1)}(z_0) = 0$, $h^{(m)}(z_0) \neq 0$, $m > 2$), the decay would be $t^{-1/m}$.



(a) Both real (b) Both complex

Figure 4.2: The case $g = 0$. Contours of constant imaginary part equal to the imaginary part of a critical point, shown for $x \in (-\pi, \pi]$.

4.5.3.1 No step interactions ($g = 0$)

In this case, the polynomial (4.52) is simply

$$p_0(w) = \alpha_1 w^3 - \alpha_{-1} w. \tag{4.55}$$

We must ignore the root $w = 0$, because $0 = w = e^{iz}$ has no solutions for finite z. The other roots are $\pm w_0$, where $w_0 = \sqrt{\frac{\alpha_{-1}}{\alpha_1}}$. Depending on the physical parameters, w_0 may be real, pure imaginary, or zero. $w_0 = 0$ is a bifurcation point, so we analyze the other two cases in depth. They are generically pictured in Figure 4.2.

The deformed integration path should be chosen so that Imag $(h(z))$ is constant and Real $(h(z))$ reaches a maximum at a critical point. We have

$$\text{Imag}\,(h(x+iy)) = \sin(x)\left(\alpha_1 e^{-y} - \alpha_{-1} e^{y}\right), \tag{4.56a}$$

$$\text{Real}\,(h(x+iy)) = \cos(x)\left(\alpha_1 e^{-y} + \alpha_{-1} e^{y}\right) + \alpha_0. \tag{4.56b}$$

4.5.3.1.1 Real roots ($\boldsymbol{sign}(\alpha_1) = \boldsymbol{sign}(\alpha_{-1})$). When w_0 is real, $h(z_0)$ is real as well, so we must find the contours where $\text{Imag}\,(h(z)) = 0$. We immediately see that $\text{Imag}\,(h(x+iy)) = 0$ if and only if $x = n\pi$ (for some $n \in \mathbb{Z}$) or $e^{-y} = \sqrt{\frac{\alpha_{-1}}{\alpha_1}}$.

When $x = 2n\pi$, we have $\cos(x) = 1$, so $\text{Real}\,(h(x+iy)) \to -\infty$ as $y \to \pm\infty$. Similarly, when $x = (2n+1)\pi$, $\text{Real}\,(h(x+iy)) \to +\infty$. Therefore, along the horizontal contour $y = -\ln\sqrt{\frac{\alpha_{-1}}{\alpha_1}}$ in Figure 4.2(a), we know that $\text{Real}\,(h(z))$ reaches a maximum at the points $z_n = (2n+1)\pi - i\ln\sqrt{\frac{\alpha_{-1}}{\alpha_1}}$. We have $z_0 = \pi - i\ln\sqrt{\frac{\alpha_{-1}}{\alpha_1}}$.

For the final calculation, note that

$$e^{iz_0} = -w_0 = -\sqrt{\frac{\alpha_{-1}}{\alpha_1}} < 0, \tag{4.57a}$$

$$h(z_0) = \left(\sqrt{|\alpha_1|} + \sqrt{|\alpha_{-1}|}\right)^2 > 0, \tag{4.57b}$$

$$h''(z_0) = -2\sqrt{\alpha_1\alpha_{-1}} < 0. \tag{4.57c}$$

Also recall that $\mathcal{U}(-\overline{z}) = \overline{\mathcal{U}(z)}$. Now, by summing equation (4.54) over all adjacent regions of width 2π, we have

$$\tilde{u}_j(t) \sim \sum_{n\in\mathbb{Z}} \mathcal{U}(z_0+2n\pi)\cdot\sqrt{\pi}\left[-\frac{1}{2}h''(z_0+2n\pi)\right]^{-1/2}\cdot(e^{i(z_0+2n\pi)})^j\cdot e^{th(z_0+2n\pi)}t^{-1/2} \tag{4.58a}$$

$$= \sqrt{\pi}\left[-\frac{1}{2}h''(z_0)\right]^{-1/2}\cdot(-w_0)^j\cdot e^{th(z_0)}t^{-1/2}\cdot\sum_{n\in\mathbb{Z}}\mathcal{U}\left((2n+1)\pi - i\ln\sqrt{\frac{\alpha_{-1}}{\alpha_1}}\right) \tag{4.58b}$$

$$= \sqrt{\pi}\,(\alpha_1\alpha_{-1})^{-1/4}\cdot\left(-\sqrt{\frac{\alpha_{-1}}{\alpha_1}}\right)^j\cdot e^{t\left(\sqrt{|\alpha_1|}+\sqrt{|\alpha_{-1}|}\right)^2}t^{-1/2}$$
$$\cdot\, 2\sum_{n=0}^{\infty}\text{Real}\left(\mathcal{U}\left((2n+1)\pi - i\ln\sqrt{\frac{\alpha_{-1}}{\alpha_1}}\right)\right). \tag{4.58c}$$

4.5.3.1.2 Pure imaginary roots ($\boldsymbol{sign}(\alpha_1) \neq \boldsymbol{sign}(\alpha_{-1})$). Here $w_0 = i\sqrt{\left|\frac{\alpha_{-1}}{\alpha_1}\right|}$, so $z_n = (n+\frac{1}{2})\pi - i\ln\sqrt{\left|\frac{\alpha_{-1}}{\alpha_1}\right|}$ for $n \in \mathbb{Z}$ are all the critical points.

When w_0 is imaginary, $h(z_0)$ is complex. Furthermore, we have the imaginary part $\text{Imag}\,(h(x+iy)) = 0$ only when $x = n\pi$ $(n \in \mathbb{Z})$, and not along the horizontal contour in Figure 4.2(a). Because the contours where $\text{Imag}\,(h(x+iy)) = \text{Imag}\,(h(z_0))$, a nonzero constant, may not cross the contours where $\text{Imag}\,(h(z)) = 0$, they are restricted to regions where $x \in (n\pi, (n+1)\pi)$, so we have $y \to \pm\infty$. Therefore, to keep $\text{Imag}\,(h(x+iy))$ from growing, we must have $\sin(x) \to 0$ with exponential decay. This implies that the contours are asymptotic to $x = n\pi$ $(n \in \mathbb{Z})$.

When $x \approx 2n\pi$, we have $\cos(x) > 0$, so $\text{Real}\,(h(x+iy)) \to -\infty$ as $y \to -\infty$ and $\text{Real}\,(h(x+iy)) \to +\infty$ as $y \to +\infty$. Similarly, when $x \approx (2n+1)\pi$, $\text{Real}\,(h(x+iy)) \to +\infty$ as $y \to -\infty$ and $\text{Real}\,(h(x+iy)) \to -\infty$ as $y \to +\infty$. Therefore, along the contour joining $(-\pi, +\infty)$ to $(0, -\infty)$ to $(\pi, +\infty)$ in Figure 4.2(b), $\text{Real}\,(h(z))$ reaches maxima at the critical points $z_0, -\overline{z_0}$.

For the final calculation, note that

$$e^{iz_0} = w_0 = i\sqrt{\left|\frac{\alpha_{-1}}{\alpha_1}\right|}, \tag{4.59a}$$

$$h(z_0) = |\alpha_1| - |\alpha_{-1}| - 2i\sqrt{|\alpha_1\alpha_{-1}|}, \tag{4.59b}$$

$$h''(z_0) = 2i\sqrt{|\alpha_1\alpha_{-1}|}. \tag{4.59c}$$

Also note that $h(-\overline{z}) = \overline{h(z)}$ as well. Now we have

$$\tilde{u}_j(t) \sim \sum_{n\in\mathbb{Z}} \mathcal{U}(z_0+2n\pi)\cdot\sqrt{\pi}\left[-\frac{1}{2}h''(z_0+2n\pi)\right]^{-1/2}\cdot(e^{i(z_0+2n\pi)})^j\cdot e^{th(z_0+2n\pi)}t^{-1/2}$$
$$+\sum_{n\in\mathbb{Z}} \mathcal{U}(-\overline{z_0}-2n\pi)\cdot\sqrt{\pi}\left[-\frac{1}{2}h''(-\overline{z_0}-2n\pi)\right]^{-1/2}$$

$$\cdot \left(e^{i(-\overline{z_0}-2n\pi)}\right)^j \cdot e^{th(-\overline{z_0}-2n\pi)} t^{-1/2} \tag{4.60a}$$

$$= \sqrt{\pi} t^{-1/2} \sum_{n\in\mathbb{Z}} \Bigg(\mathcal{U}(z_0+2n\pi) \cdot \left[-\frac{1}{2}h''(z_0)\right]^{-1/2} w_0^j e^{th(z_0)}$$

$$+ \overline{\mathcal{U}(z_0+2n\pi)} \cdot \left[-\frac{1}{2}\overline{h''(z_0)}\right]^{-1/2} \overline{w_0}^j e^{\overline{th(z_0)}} \Bigg) \tag{4.60b}$$

$$= 2\sqrt{\pi} t^{-1/2} \sum_{n\in\mathbb{Z}} \text{Real} \left(\mathcal{U}(z_0+2n\pi) \cdot \left[-\frac{1}{2}h''(z_0)\right]^{-1/2} w_0^j e^{th(z_0)} \right) \tag{4.60c}$$

$$= 2\sqrt{\pi} t^{-1/2} \Bigg[\text{Real} \left(\left(-i\sqrt{|\alpha_1\alpha_{-1}|}\right)^{-1/2} \left(i\sqrt{\left|\frac{\alpha_{-1}}{\alpha_1}\right|}\right)^j e^{th(z_0)} \right)$$

$$\cdot \sum_{n\in\mathbb{Z}} \text{Real}\left(\mathcal{U}(z_0+2n\pi)\right)$$

$$- \text{Imag} \left(\left(-i\sqrt{|\alpha_1\alpha_{-1}|}\right)^{-1/2} \left(i\sqrt{\left|\frac{\alpha_{-1}}{\alpha_1}\right|}\right)^j e^{th(z_0)} \right)$$

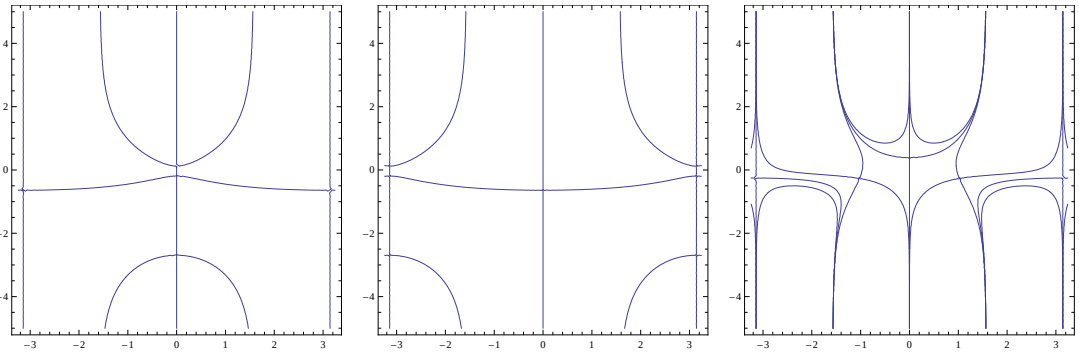
$$\cdot \sum_{n\in\mathbb{Z}} \text{Imag}\left(\mathcal{U}(z_0+2n\pi)\right) \Bigg]. \tag{4.60d}$$

4.5.3.2 Step repulsions ($g > 0$)

In this case, $p_g(w)$ is given by the full equation (4.52). Note that $p_g(w)$ has real coefficients, so if w_0 is a root, then $\overline{w_0}$ is also a root. This implies that if z_0 is a critical point, then $-\overline{z_0}$ is also a critical point.

We know that $\beta_{\pm 2}$ are both negative. Therefore, $p_g(w) \to -\infty$ as $w \to \pm\infty$, and $p_g(0) > 0$. The intermediate value theorem implies that $p_g(w)$ has at least two real roots, one positive and one negative. Furthermore, if $p_g(w)$ has four real roots, it is either the case that three are positive and one is negative, or one is positive and three are negative. The cases are generically pictured in Figure 4.3.

It is possible to determine whether $p_g(w)$ has four real roots, or two real and



(a) Three positive (b) Three negative (c) One positive, one negative, two complex conjugates

Figure 4.3: The case $g > 0$. Contours of constant imaginary part equal to the imaginary part of a critical point, shown for $x \in (-\pi, \pi]$.

two complex roots, by finding the sign of the discriminant of $p_g(w)$ (positive in the former case, negative in the latter).

4.5.3.2.1 Constant imaginary part. The deformed integration path should be chosen so that $\mathrm{Imag}\,(h(z))$ is constant. We have

$$\mathrm{Imag}\,(h(x+iy)) = \sin(x)\big(2\cos(x)\left[(g\beta_2)e^{-2y} - (g\beta_{-2})e^{2y}\right] + \left[(\alpha_1 + g\beta_1)e^{-y} - (\alpha_{-1} + g\beta_{-1})e^{y}\right]\big) \tag{4.61}$$

First, consider a real root w_0 of $p_g(w)$. Since w_0 is real, $h(z_0)$ is also real, so we must find the contours where $\mathrm{Imag}\,(h(x+iy)) = 0$. This equality holds when $\sin(x) = 0$, or when $\cos(x) = -\frac{(\alpha_1+g\beta_1)e^{-y}+(\alpha_{-1}+g\beta_{-1})e^{y}}{2((g\beta_2)e^{-2y}-(g\beta_{-2})e^{2y})}$, which approaches 0 as $y \to \pm\infty$. Therefore, for all $n \in \mathbb{Z}$, the contours $x = n\pi$ satisfy $\mathrm{Imag}\,(h(z)) = 0$, and the lines $x = \frac{\pi}{2} + n\pi$ are asymptotes of the remaining contours with $\mathrm{Imag}\,(h(z)) = 0$.[1]

[1]It is not possible to choose, e.g., $x = 0$ as the deformed contour, because the deformation would

Now, consider a possible complex root w_0 of $p_g(w)$. We must find the contours where $\text{Imag}\,(h(x+iy)) = \text{Imag}\,(h(z_0))$, a nonzero constant. Since these may not cross the contours where $\text{Imag}\,(h(z)) = 0$, they are restricted to regions where $x \in (n\pi, (n+1)\pi)$, so $y \to \pm\infty$. Since the term $(g\beta_2)e^{-2y} - (g\beta_{-2})e^{2y}$ dominates the growth in y, we must have either $\sin(x) \to 0$ or $\cos(x) \to -\frac{(\alpha_1+g\beta_1)e^{-y}+(\alpha_{-1}+g\beta_{-1})e^{y}}{2((g\beta_2)e^{-2y}-(g\beta_{-2})e^{2y})}$ fast enough to keep $\text{Imag}\,(h(z))$ from growing. Once again, we find that the contours are asymptotic to $x = n\pi$ and $x = \frac{\pi}{2} + n\pi$ $(n \in \mathbb{Z})$.

4.5.3.2.2 Steepest descent paths. We will integrate over contours where the real part $\text{Real}\,(h(z))$ reaches a maximum at one of the critical points. We have

$$\begin{aligned}\text{Real}\,(h(x+iy)) = \cos(2x)\left[(g\beta_2)e^{-2y} + (g\beta_{-2})e^{2y}\right] \\ + \cos(x)\left[(\alpha_1 + g\beta_1)e^{-y} + (\alpha_{-1} + g\beta_{-1})e^{y}\right] + (\alpha_0 + g\beta_0). \quad (4.62)\end{aligned}$$

Since $e^{\pm 2y}$ dominate the growth of $\text{Real}\,(h(z))$ as $y \to \pm\infty$, and $\beta_{\pm 2} < 0$, we have $\text{Real}\,(h(z)) \to +\infty$ along the lines $x = \frac{\pi}{2} + n\pi$ $(n \in \mathbb{Z})$, and $\text{Real}\,(h(z)) \to -\infty$ along the lines $x = n\pi$ $(n \in \mathbb{Z})$. We now have three cases to consider.

In the first case, $p_g(w)$ has three positive roots and one negative root. Let w_0 be the middle positive root, and $z_0 = -i\ln(w_0)$. The necessary contour in Figure 4.3(a) joins z_0 to the negative root and its reflection. By summing equation (4.54) over all adjacent regions of width 2π, we have

$$\tilde{u}_j(t) \sim \sum_{n\in\mathbb{Z}} \mathcal{U}(z_0 + 2n\pi) \cdot \sqrt{\pi}\left[-\frac{1}{2}h''(z_0 + 2n\pi)\right]^{-1/2} \cdot w_0^{-j} \cdot e^{th(z_0+2n\pi)}t^{-1/2} \quad (4.63a)$$

$$= \sqrt{\pi}\left[-\frac{1}{2}h''(z_0)\right]^{-1/2} \cdot w_0^{-j} \cdot e^{th(z_0)}t^{-1/2} \cdot \sum_{n\in\mathbb{Z}} \mathcal{U}(i\ln(w_0) + 2n\pi), \quad (4.63b)$$

have to pass through regions of x-values where the integral does not converge.

and, because $\mathcal{U}(-\overline{z}) = \overline{\mathcal{U}(z)}$,

$$\sum_{n\in\mathbb{Z}} \mathcal{U}(i\ln(w_0) + 2n\pi) = \mathcal{U}(i\ln(w_0)) + \sum_{n=1}^{\infty} \big(\mathcal{U}(2n\pi + i\ln(w_0)) + \mathcal{U}(-2n\pi + i\ln(w_0))\big) \tag{4.64a}$$

$$= \mathcal{U}(i\ln(w_0)) + 2\sum_{n=1}^{\infty} \text{Real}\left(\mathcal{U}(2n\pi + i\ln(w_0))\right) \in \mathbb{R}. \tag{4.64b}$$

In the second case, $p_g(w)$ has one positive root and three negative roots. Let w_0 be the middle negative root, and $z_0 = \pi - i\ln|w_0|$. The necessary contour in Figure 4.3(b) joins z_0 and its reflection to the positive root. As above, we have

$$\tilde{u}_j(t) \sim \sqrt{\pi}\left[-\frac{1}{2}h''(z_0)\right]^{-1/2} \cdot w_0^{-j} \cdot e^{th(z_0)}t^{-1/2} \cdot \sum_{n\in\mathbb{Z}} \mathcal{U}(i\ln|w_0| + (2n+1)\pi), \tag{4.65}$$

and

$$\sum_{n\in\mathbb{Z}} \mathcal{U}(i\ln|w_0| + (2n+1)\pi) = \sum_{n=0}^{\infty} \big(\mathcal{U}((2n+1)\pi + i\ln|w_0|) + \mathcal{U}(-(2n+1)\pi + i\ln|w_0|)\big) \tag{4.66a}$$

$$= 2\sum_{n=0}^{\infty} \text{Real}\left(\mathcal{U}((2n+1)\pi + i\ln|w_0|)\right) \in \mathbb{R}. \tag{4.66b}$$

In the third case, $p_g(w)$ has one positive root, one negative root, and two complex conjugate roots. Let $w_0 = e^{iz_0}$ be the root with positive imaginary part, and $\overline{w_0} = e^{i(-\overline{z_0})}$ the root with negative imaginary part. In Figure 4.3(c), the necessary contour for z_0 proceeds from the asymptote $x = -\pi$ to the asympote $x = 0$, while the contour for $-\overline{z_0}$ proceeds from the asymptote $x = 0$ to the asympote $x = +\pi$. We have

$$\tilde{u}_j(t) \sim \sum_{n\in\mathbb{Z}} \mathcal{U}(z_0 + 2n\pi) \cdot \sqrt{\pi}\left[-\frac{1}{2}h''(z_0 + 2n\pi)\right]^{-1/2} \cdot w_0^{-j} \cdot e^{th(z_0+2n\pi)}t^{-1/2}$$

$$+\sum_{n\in\mathbb{Z}}\mathcal{U}(-\overline{z_0}-2n\pi)\cdot\sqrt{\pi}\left[-\frac{1}{2}h''(-\overline{z_0}-2n\pi)\right]^{-1/2}$$

$$\cdot\,\overline{w_0}^{-j}\cdot e^{th(-\overline{z_0}-2n\pi)}t^{-1/2} \tag{4.67a}$$

$$=\sqrt{\pi}t^{-1/2}\sum_{n\in\mathbb{Z}}\Bigg(\mathcal{U}(z_0+2n\pi)\cdot\left[-\frac{1}{2}h''(z_0)\right]^{-1/2}w_0^{-j}e^{th(z_0)}$$

$$+\overline{\mathcal{U}(z_0+2n\pi)}\cdot\left[-\frac{1}{2}\overline{h''(z_0)}\right]^{-1/2}\overline{w_0}^{-j}e^{\overline{th(z_0)}}\Bigg) \tag{4.67b}$$

$$=2\sqrt{\pi}t^{-1/2}\sum_{n\in\mathbb{Z}}\text{Real}\left(\mathcal{U}(z_0+2n\pi)\cdot\left[-\frac{1}{2}h''(z_0)\right]^{-1/2}w_0^{-j}e^{th(z_0)}\right), \tag{4.67c}$$

using the fact that $h(-\overline{z})=\overline{h(z)}$ as well.

4.5.3.3 Relationship between the solutions ($g\ll 1$)

Assume that a root w of $p_g(w)$ can be expanded as

$$w=w_0+g^{\alpha}w_1+g^{2\alpha}w_2+\cdots \tag{4.68}$$

for some $\alpha\neq 0$.

First, assume that $w_0=0$, and $w\sim g^{\alpha}w_1$. Then

$$\underbrace{g^{4\alpha+1}(2\beta_2w_1^4)}_{①}+\underbrace{g^{3\alpha}(\alpha_1w_1^3)}_{②}+\underbrace{g^{\alpha}(-\alpha_{-1}w_1)}_{③}+\underbrace{g^{1}(-2\beta_{-2})}_{④}\sim 0. \tag{4.69}$$

The balance between ① and ② yields $\alpha=-1$, and thus $w_1=-\frac{\alpha_1}{2\beta_2}$. The balance between ③ and ④ yields $\alpha=1$, and thus $w_1=-\frac{2\beta_{-2}}{\alpha_{-1}}$. So two of the roots are

$$w\sim-\frac{\alpha_1}{2\beta_2}g^{-1}\qquad\text{and}\qquad w\sim-\frac{2\beta_{-2}}{\alpha_{-1}}g. \tag{4.70}$$

Now, assume that $w_0\neq 0$. Then

$$\underbrace{g^{1}(2\beta_2w_0^4-2\beta_{-2})}_{①}+\underbrace{g^{0}(\alpha_1w_0^3-\alpha_{-1}w_0)}_{②}+\underbrace{g^{\alpha}(3\alpha_1w_0^2w_1-\alpha_{-1}w_1)}_{③}\sim 0. \tag{4.71}$$

The $\mathcal{O}(1)$ term yields $w_0 = \pm\sqrt{\frac{\alpha_{-1}}{\alpha_1}}$. The balance between (1) and (3) yields $\alpha = 1$, and thus $w_1 = -2\frac{\beta_2 w_0^4 - \beta_{-2}}{3\alpha_1 w_0^2 - \alpha_{-1}} = \frac{\beta_{-2}\alpha_1^2 - \beta_2\alpha_{-1}^2}{\alpha_1^2\alpha_{-1}}$. So the other two roots are

$$w \sim \sqrt{\frac{\alpha_{-1}}{\alpha_1}} + gw_1 \qquad \text{and} \qquad w \sim -\sqrt{\frac{\alpha_{-1}}{\alpha_1}} + gw_1. \tag{4.72}$$

The $\mathcal{O}(1)$ roots correspond to the roots in the deformed contours we chose for the integration — they are either the two complex roots, or one is the middle positive/negative root, since the remaining roots are $\mathcal{O}(g) \to 0$ and $\mathcal{O}(g^{-1}) \to \pm\infty$.

4.5.3.3.1 Dependence on time and step number. The step-step interaction case can now be considered as a regular perturbation of the no-interaction case, for steps in the bulk of the crystal.

When α_1 and α_{-1} have equal signs, which is possible for some intermediate values of v_0, we have equation (4.58c):

$$\tilde{u}_j(t) \sim \sqrt{\pi}\,(\alpha_1\alpha_{-1})^{-1/4} \cdot \left(-\sqrt{\frac{\alpha_{-1}}{\alpha_1}}\right)^j \cdot e^{t\left(\sqrt{|\alpha_1|}+\sqrt{|\alpha_{-1}|}\right)^2} t^{-1/2}$$
$$\cdot\, 2\sum_{n=0}^{\infty} \text{Real}\left(\mathcal{U}\left((2n+1)\pi - i\ln\sqrt{\frac{\alpha_{-1}}{\alpha_1}}\right)\right)$$

The time dependence is proportional to $e^{t\left(\sqrt{|\alpha_1|}+\sqrt{|\alpha_{-1}|}\right)^2} t^{-1/2}$, which is exponential growth with a polynomial decay term. The dependence on j is proportional to $\left(-\sqrt{\frac{\alpha_{-1}}{\alpha_1}}\right)^j$, which is growth or decay depending on the sign of j. Note that the sign of $\tilde{u}_j(t)$ alternates with odd and even values of j.

When α_1 and α_{-1} have opposite signs, which holds for very strong and very weak electric fields, we have equation (4.60c):

$$\tilde{u}_j(t) \sim 2\sqrt{\pi}t^{-1/2} \sum_{n\in\mathbb{Z}} \text{Real}\left(\mathcal{U}(z_0 + 2n\pi) \cdot \left[-\frac{1}{2}h''(z_0)\right]^{-1/2} w_0^j e^{th(z_0)}\right)$$

First consider the time dependence. Recall from equation (4.59) that $\text{Real}\,(h(z_0)) = |\alpha_1| - |\alpha_{-1}|$. We found in Section 4.4.3 that α_1 decays to 0 faster than α_{-1} when the electric field is very strong, implying that $\text{Real}\,(h(z_0)) < 0$. Now consider the dependence on step number. Recall that $w_0 = i\sqrt{\left|\frac{\alpha_{-1}}{\alpha_1}\right|}$, so again we have growth or decay depending on the sign of j.

4.6 Discussion

We have analyzed the motion of crystal steps as a discrete system of coupled linear ODEs. Much of the literature has focused on continuum approximations of surface height, which can provide reliable indications of the stability of uniform step trains; step bunching is one form of instability. However, these continuum approximations miss a time dependence term that appears in the discrete analysis, and they do not give a full picture of the dependence on step number.

Recall our measure of the deviation of terrace width from the average value, $\tilde{u}_j(t) = \frac{x_{j+1}(t)-x_j(t)}{L} - 1$. We found that the time dependence of $\tilde{u}_j(t)$ is proportional to $e^{th(z_0)}t^{-1/2}$. The $t^{-1/2}$ factor does not appear in continuum approximations, and is the fastest decay we can expect from a first-order asymptotic analysis of the discrete model. The $h(z_0)$ exponent depends on the physical parameters of the problem. Stability is indicated when $\tilde{u}_j(t)$ decays with time, while instabilities such as step bunching are possible when $\tilde{u}_j(t)$ grows with time.

We also found that the step number dependence of $\tilde{u}_j(t)$ is proportional to $(e^{iz_0})^j$. Therefore, $\tilde{u}_j(t)$ grows or decays in j depending on the sign of j. We have identified parameter regions where the base e^{iz_0} is a negative real number, indicating alternating signs of the terrace width deviation. Such alternation could lead to the "double steps" that have been observed in silicon [60] and ruthenium [64].

Appendix A

Addendum to Chapter 3

A.1 Standard ports

In 2003, many peer-to-peer applications used constant port numbers. In addition, many client-server applications use IANA-assigned ports, or other constant ports. The following port values were used for determining the "true" classes in Section 3.3.1.

Table A.1: Peer-to-peer ports, protocols, and their usages. Compiled from various online sources.

Port	Protocol	Application
412	TCP/UDP	DirectConnect
1214	TCP	KaZaA
1412	TCP/UDP	DirectConnect
2000	TCP	eDonkey 2000
2004	UDP	eDonkey 2000
2234	TCP	SoulSeek
3415	TCP	KaZaA
3531	TCP/UDP	KaZaA
4329	TCP	iMesh
4444	TCP	Napster
4661	TCP	eDonkey 2000
4662	TCP	eDonkey 2000
4663	TCP	eDonkey 2000
4665	UDP	eDonkey 2000
4666	UDP	eDonkey 2000
4670	TCP	eDonkey 2000

Table A.2: Table A.1 continued.

Port	Protocol	Application
5498	TCP	Hotline Connect
5499	UDP	Hotline Connect
5500	TCP	Hotline Connect
5501	TCP	Hotline Connect
5502	TCP	Hotline Connect
5503	TCP	Hotline Connect
5534	TCP	SoulSeek
5555	TCP	Napster
6257	UDP	WinMX
6346	TCP	Gnutella
6347	UDP	Gnutella
6348	TCP	Gnutella
6662	TCP	eDonkey 2000
6666	TCP	Napster
6699	TCP	WinMX
6700	TCP	Napster
6701	TCP	Napster
6881	TCP	BitTorrent
7674	UDP	Soribada
7675	TCP	Soribada
7676	TCP	Soribada
7677	TCP	Soribada
7788	TCP/UDP	BuddyShare
8311	TCP	Scour
8875	TCP	Napster
8888	TCP	Napster
8889	TCP	Napster
22321	TCP/UDP	Soribada
22322	TCP	Soribada
41170	TCP/UDP	Blubster, Piolet, Manolito

Table A.3: Client/server ports, protocols, and their usages. Compiled from various online sources.

Port	Protocol	Application
20	TCP	FTP
21	TCP	FTP
22	TCP	SSH
23	TCP	telnet
25	TCP	SMTP
53	TCP/UDP	DNS
80	TCP	HTTP
110	TCP	POP3
123	UDP	network time protocol
143	TCP	IMAP
389	TCP/UDP	LDAP
443	TCP	HTTPS
993	TCP	IMAP over SSL
1024	TCP/UDP	Windows browsing
1025	TCP/UDP	Windows browsing
1026	TCP/UDP	Windows browsing
1080	TCP	proxy server
1490	TCP	vocaltec videoconferencing
1755	UDP	windows media streaming
1863	TCP/UDP	msn messenger
2703	TCP/UDP	SpamAssassin
4099	TCP	AIM
5050	TCP	Yahoo IM
5190	TCP	aim
6277	TCP/UDP	DCC anti-spam
6665	TCP	IRC
6667	TCP	IRC
6670	TCP	vocaltec videoconferencing
7070	TCP/UDP	realaudio
8000	TCP	webserver
8080	TCP	HTTP2
22555	UDP	vocaltec videoconferencing
25793	TCP	vocaltec videoconferencing
32768	TCP/UDP	*nix browsing
32769	TCP/UDP	*nix browsing
32770	TCP/UDP	*nix browsing

Appendix B

Addendum to Chapter 4

B.1 Parameters and variables of the crystal problem

Table B.1: Overview of parameters and variables.

Symbol	**Dimensions**	**Brief description**
L	length	Typical terrace width
a	length	Step height
Ω	length^2	Atomic volume
$x_j(t)$	length	Position of j^{th} step edge at time t
$u_j(t)$	$\emptyset$	$(x_{j+1}(t) - x_j(t))/L$; normalized length of j^{th} terrace
$\tilde{u}_j(t)$	$\emptyset$	$u_j(t) - 1$; deviation from uniform terrace width
c_0	1/length	Adatom density at an isolated step
$c_j(x)$	1/length	Adatom density at position x on the j^{th} terrace
$J_j(x)$	1/time	Adatom current (flux) at position x on the j^{th} terrace
τ	time	Desorption time
D	length^2/time	Terrace diffusivity
v_0	length/time	Down-step velocity due to electric field
k_+, k_-	length/time	Rate coefficients for attachment-detachment
g	$\emptyset$	Strength of step interactions
δ	$\emptyset$	$L/\sqrt{D\tau}$
η	$\emptyset$	$\sqrt{D\tau}/(v_0\tau)$

Bibliography: Peer-to-peer networks

[1] Abiola Abimbola, Qi Shi, and Madjid Merabti. Using intrusion detection to detect malicious peer-to-peer network traffic. In *PostGraduate Networking Conference (PGNet)*, Manchester, UK, June 2003.

[2] Ittai Abraham, Baruch Awerbuch, Yossi Azar, Yair Bartal, Dahlia Malkhi, and Elan Pavlov. A generic scheme for building overlay networks in adversarial scenarios. In *International Parallel and Distributed Processing Symposium (IPDPS)*, Nice, France, April 2003.

[3] Laurent Bernaille, Renata Teixeira, Ismael Akodkenou, Augustin Soule, and Kave Salamatian. Traffic classification on the fly. *ACM SIGCOMM Computer Communication Review*, 36(2):23–26, April 2006.

[4] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, October 2001.

[5] Leo Breiman and Adele Cutler. Random forests (classification/clustering). `http://www.stat.berkeley.edu/~breiman/RandomForests`, June 2004.

[6] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth, Inc., Belmont, California, USA, 1984.

[7] Multi-State Information Sharing & Analysis Center. Monthly cyber security tips newsletter: Security concerns — peer to peer (P2P) file sharing. `http://www.msisac.org/awareness/news/2007-04.cfm`, April 2007.

[8] Cisco. Cisco IOS NetFlow version 9 flow-record format. `http://www.cisco.com`, February 2007.

[9] Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Wide-area cooperative storage with CFS. In *ACM Symposium on Operating Systems Principles (SOSP)*, Banff, Canada, October 2001.

[10] Anwitaman Datta, Sarunas Girdzijauskas, and Karl Aberer. On de Bruijn routing in distributed hash tables: There and back again. In *IEEE International Conference on Peer-to-Peer Computing (P2P)*, Zurich, Switzerland, August 2004.

[11] Prasanna Desikan and Jaideep Srivastava. Analyzing network traffic to detect e-mail spamming machines. In *ICDM Workshop on Privacy and Security Aspects of Data Mining*, Brighton, UK, November 2004.

[12] Pedro Domingos and Michael Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29(2):103–130, November 1997.

[13] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, New York, USA, 1973.

[14] Robert Elsässer, Burkhard Monien, and Robert Preis. Diffusion schemes for load balancing on heterogeneous networks. *Theory of Computing Systems*, 35(3):305–320, May 2002.

[15] Ronald Aylmer Fisher. The use of multiple measures in taxonomic problems. *Annals of Eugenics*, 7:179–188, September 1936.

[16] Pierre Fraigniaud and Philippe Gauron. D2B: A de Bruijn based content-addressable network. *Theoretical Computer Science*, 355(1):65–79, April 2006.

[17] Anh-Tuan Gai and Laurent Viennot. Broose: A practical distributed hashtable based on the de-Bruijn topology. In *IEEE International Conference on Peer-to-Peer Computing (P2P)*, Zurich, Switzerland, August 2004.

[18] Prasanna Ganesan, Mayank Bawa, and Hector Garcia-Molina. Online balancing of range-partitioned data with applications to peer-to-peer systems. In *International Conference on Very Large Data Bases (VLDB)*, Toronto, Canada, September 2004.

[19] Luis Garcés-Erice, Ernst W. Biersack, Keith W. Ross, Pascal Felber, and Guillaume Urvoy-Keller. Hierarchical peer-to-peer systems. *Parallel Processing Letters*, 13(4):643–657, March 2003.

[20] George Giakkoupis and Vassos Hadzilacos. A scheme for load balancing in heterogeneous distributed hash tables. In *ACM Symposium on Principles of Distributed Computing (PODC)*, Las Vegas, Nevada, USA, July 2005.

[21] Sarunas Girdzijauskas, Anwitaman Datta, and Karl Aberer. On small world graphs in non-uniformly distributed key spaces. In *IEEE International Workshop on Networking Meets Databases (NetDB)*, Tokyo, Japan, April 2005.

[22] P. Brighten Godfrey and Ion Stoica. Heterogeneity and load balance in distributed hash tables. In *IEEE Conference on Computer Communications (INFOCOM)*, Miami, Florida, USA, March 2005.

[23] MIT Lincoln Laboratory; Information Systems Technology Group. LNKnet. `http://www.ll.mit.edu/IST/lnknet`, February 2004.

[24] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer Series in Statistics. Springer, New York, New York, USA, 2001.

[25] M. Frans Kaashoek and David R. Karger. Koorde: A simple degree-optimal distributed hash table. In *International Workshop on Peer-to-Peer Systems (IPTPS)*, Berkeley, California, USA, February 2003.

[26] Thomas Karagiannis, Andre Broido, Michalis Faloutsos, and Kc claffy. Transport layer identification of P2P traffic. In *ACM SIGCOMM Internet Measurement Conference (IMC)*, Taormina, Sicily, Italy, October 2004.

[27] Thomas Karagiannis, Konstantina Papagiannaki, and Michalis Faloutsos. Blinc: Multilevel traffic classification in the dark. *ACM SIGCOMM Computer Communication Review*, 35(4):229–240, October 2005.

[28] David Karger, Eric Lehman, Tom Leighton, Matthew Levine, Daniel Lewin, and Rina Panigrahy. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *ACM Symposium on Theory of Computing*, El Paso, Texas, USA, May 1997.

[29] Karthik Lakshminarayanan and Venkata N. Padmanabhan. Some findings on the network performance of broadband hosts. In *ACM/USENIX Internet Measurement Conference (IMC)*, Miami, Florida, USA, October 2003.

[30] Yuchun Lee. Classifiers: Adaptive modules in pattern recognition systems. Master's thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, May 1989.

[31] Dmitri Loguinov, Anuj Kumar, Vivek Rai, and Sai Ganesh. Graph-theoretic analysis of structured peer-to-peer systems: Routing distances and fault resilience. In *ACM SIGCOMM Conference*, Karlsruhe, Germany, August 2003.

[32] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communication Surveys and Tutorials*, 7(2):72–93, April 2005.

[33] Klaus Mochalski. Personal communication, December 2006.

[34] Andrew W. Moore and Konstantina Papagiannaki. Toward the accurate identification of network applications. In *Passive and Active Network Measurement*, volume 3431 of *Lecture Notes in Computer Science*, pages 41–54. Springer Berlin/Heidelberg, March 2005.

[35] Andrew W. Moore and Denis Zuev. Internet traffic classification using bayesian analysis techniques. In *ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, Banff, Alberta, Canada, June 2005.

[36] Motion Picture Association and L.E.K. The cost of movie piracy. `http://www.mpaa.org/researchStatistics.asp`, May 2006.

[37] Moni Naor and Udi Wieder. Novel architectures for P2P applications: The continuous-discrete approach. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, San Diego, California, June 2003.

[38] National laboratory for applied network research — passive measurement and analysis. `http://pma.nlanr.net`, 2006. Thanks to the National Science Foundation cooperative agreement nos. ANI-0129677 (2002) and ANI-9807479 (1998).

[39] University of Leipzig internet access link. Leipzig-I data set. `http://pma.nlanr.net/Special/leip1.html`, November 2002.

[40] University of Memphis internet access link. Memphis data set. `http://pma.nlanr.net/PMA/Sites/MEM.html`, March 2006.

[41] Philippe Olivier and Nabil Benameur. Flow level IP traffic characterization. In *International Teletraffic Congress (ITC)*, Salvador da Bahia, Brazil, December 2001.

[42] Andrew Parker. P2P in 2005. `http://www.cachelogic.com`, August 2005.

[43] Vern Paxson and Sally Floyd. Wide area traffic: The failure of poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, June 1995.

[44] Sylvia Ratnasamy, Scott Shenker, and Ion Stoica. Routing algorithms for DHTs: Some open questions. In *International Workshop on Peer-to-Peer Systems (IPTPS)*, Cambridge, Massachusetts, USA, March 2002.

[45] Stefan Saroiu, Krishna P. Gummadi, Richard J. Dunn, Steven D. Gribble, and Henry M. Levy. An analysis of internet content delivery systems. In *Symposium on Operating Systems Design and Implementation (ODSI)*, Boston, Massachusetts, USA, December 2002.

[46] Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Conference on Multimedia Computing and Networking (MMCN)*, San Jose, California, USA, January 2002.

[47] Subhabrata Sen, Oliver Spatscheck, and Dongmei Wang. Accurate, scalable in-network identification of P2P traffic using application signatures. In *International Conference on World Wide Web*, New York, New York, USA, May 2004.

[48] Cyril Soldani. *Peer-to-Peer Behaviour Detection by TCP Flows Analysis*. PhD thesis, University of Liège, Liége, Wallonia, Belgium, May 2004.

[49] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *ACM SIGCOMM Conference*, San Diego, California, USA, August 2001.

[50] Daniel Stutzbach and Reza Rejaie. Understanding churn in peer-to-peer networks. In *ACM SIGCOMM Internet Measurement Conference (IMC)*, Rio de Janeriro, Brazil, October 2006.

[51] Kyoungwon Suh, Daniel R. Figueiredo, Jim Kurose, and Don Towsley. Characterizing and detecting relayed traffic: A case study using Skype. In *IEEE Conference on Computer Communications (INFOCOM)*, Barcelona, Catalunyaa, Spain, April 2006.

[52] Anssi Tauriainen. A self-learning system for P2P traffic classification. Seminar on Internetworking, at the Telecommunications Software and Multimedia Laboratory, Helsinki University of Technology, Otaniemi, Espoo, April 2005.

[53] United States District Court; Northern District of California. A&M Records, et al. v. Napster, Inc., March 2001.

[54] Xiaoming Wang, Yueping Zhang, Xiafeng Li, and Dmitri Loguinov. On zone-balancing of peer-to-peer networks: Analysis of random node join. In *ACM SIGMETRICS Conference*, New York, New York, USA, June 2004.

[55] Sebastian Zander, Nigel Williams, and Grenville Armitage. Internet archeology: Estimating individual application trends in incomplete historic traffic traces. In *Passive and Active Measurement Workshop (PAM)*, Adelaide, Australia, March 2006.

[56] Yin Zhang and Vern Paxson. Detecting stepping stones. In *USENIX Security Symposium*, San Diego, California, USA, June 2000.

Bibliography: Crystal steps

[57] W.K. Burton, N. Cabrera, and F.C. Frank. The growth of crystals and the equilibrium structure of their surfaces. *Philosophical Transactions of the Royal Society of London, Series A*, 243(866):299–358, June 1951.

[58] Russel E. Caflisch and Bo Li. Analysis of island dynamics in epitaxial growth of thin films. *Multiscale Modeling and Simulation*, 1(1):150–171, January 2003.

[59] George F. Carrier, Max Krook, and Carl E. Pearson. *Functions of a Complex Variable: Theory and Technique*. McGraw-Hill, Inc., New York, St. Louis, San Francisco, Toronto, London, Sydney, 1966.

[60] D. J. Chadi. Stabilities of single-layer and bilayer steps on Si(001) surfaces. *Physical Review Letters*, 59(15):1691–1694, October 1987.

[61] Pak-Wing Fok. *Simulations of Axisymmetric Stepped Surfaces with a Facet*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, June 2007.

[62] Pak-Wing Fok, Rodolfo R. Rosales, and Dionisios Margetis. Unification of step bunching phenomena on vicinal surfaces. *Physical Review B*, 76(3):033408, July 2007.

[63] E.E. Gruber and W.W. Mullins. On the theory of anisotropy of crystalline surface tension. *Journal of Physics and Chemistry of Solids*, 28(5):875–887, May 1967.

[64] G. Held, S. Uremović, and D. Menzel. Rearrangement of stepped Ru(001) surfaces upon oxygen adsorption. *Physical Review Letters*, 331–333(2):1122–1128, July 1995.

[65] Wei Hong, Ho Nyung Lee, Mina Yoon, Hans M. Christen, Douglas H. Lowndes, Zhigang Suo, and Zhenyu Zhang. Persistent step-flow growth of strained films on vicinal substrates. *Physical Review Letters*, 95(9):095501, August 2005.

[66] Navot Israeli and Daniel Kandel. Profile of a decaying crystalline cone. *Physical Review B*, 60(8):13707–13717, August 1999.

[67] C. Jayaprakash, Craig Rottman, and W. F. Saam. Simple model for crystal shapes: Step-step interactions and facet edges. *Physical Review B*, 30(11):6549–6554, December 1984.

[68] Daniel Kandel and John D. Weeks. Step bunching as a chaotic pattern formation process. *Physical Review Letters*, 69(26):3758–3761, December 1992.

[69] Kavli Institute of Nanoscience; Delft University of Technology. Research in the quantum transport group. `http://qt.tn.tudelft.nl/research/`.

[70] A.V. Latyshev, A.L. Aseev, A.B. Krasilnikov, and S.I. Stenin. Transformations on clean Si(111) stepped surface during sublimation. *Surface Science*, 213(1):157–169, 1989.

[71] Da-Jiang Liu and John D. Weeks. Quantitative theory of current-induced step bunching on Si(111). *Physical Review B*, 57(23):14891–14900, June 1998.

[72] Feng Liu, J. Tersoff, and M.G. Lagally. Self-organization of steps in growth of strained films on vicinal substrates. *Physical Review Letters*, 89(6):1268–1271, February 1998.

[73] Dionisios Margetis, Michael J. Aziz, and Howard A. Stone. Continuum approach to profile scaling in nanostructure decay. *Physical Review B*, 71(16):165432, April 2005.

[74] C. Misbah and O. Pierre-Louis. Pulses and disorder in a continuum version of step-bunching dynamics. *Physical Review E*, 53(5):R4318, May 1996.

[75] William W. Mullins. Flattening of a nearly plane solid surface due to capillarity. *Journal of Applied Physics*, 30(1):77–83, January 1959.

[76] A. Rettori and J. Villain. Flattening of grooves on a crystal surface: A method of investigation of surface roughness. *Journal de Physique (Paris, France)*, 49(2):257–267, 1988.

[77] Masahide Sato and Makio Uwaha. Growth of step bunches formed by the drift of adatoms. *Surface Science*, 442(2):318–328, November 1999.

[78] Massahide Sato and Makio Uwaha. Growth law of step bunches induced by the Ehrlich-Schwoebel effect in growth. *Surface Science*, 493(1):494–498, November 2001.

[79] S. Stoyanov and V. Tonchev. Properties and dynamic interaction of step density waves at a crystal surface during electromigration affected sublimation. *Physical Review B*, 58(3):1590–1600, July 1998.

[80] Stoyan Stoyanov. Electromigration induced step bunching on Si surfaces: How does it depend on the temperature and heating current direction? *Japanese Journal of Applied Physics*, 30(1):1–6, January 1991.

[81] Brian S. Swartzentruber. Scanning tunneling microscope / Atom-tracking lab. `http://www.sandia.gov/surface_science/stm/`, August 2002.

[82] Katsumichi Yagi, Hiroki Minoda, and Masashi Degawa. Step bunching, step wandering and faceting: Self-organization at Si surfaces. *Surface Science Reports*, 43(2–4):45–126, July 2001.