Huge Networks, Tiny Faulty Nodes

by

Enoch Peserico

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

March 2007
[June 2007]

Author .....
Department of Electrical Engineering and Computer Science
March 19, 2007

Certified by .....
Larry Rudolph
Principal Research Scientist
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Huge Networks, Tiny Faulty Nodes

by

Enoch Peserico

Submitted to the Department of Electrical Engineering and Computer Science
on March 19, 2007, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Can one build, and efficiently use, networks of arbitrary size and topology using a "standard" node whose resources, in terms of memory and reliability, do not need to scale up with the complexity and size of the network? This thesis addresses two important aspects of this question.

The first is whether one can achieve efficient connectivity despite the presence of a constant probability of faults per node/link. Efficient connectivity means (informally) having every pair of regions connected by a constant fraction of the independent, entirely non-faulty paths that would be present if the entire network were fault free - even at distances where each path has only a vanishingly small probability of being fault-free. The answer is yes, as long as some very mild topological conditions on the high level structure of the network are met - informally, if the network is not too "thin" and if it does not contain too many large "holes". The results go against some established "empyrical wisdom" in the networking community.

The second issue addressed by this thesis is whether one can route efficiently on a network of arbitrarly size and topology using only a constant number c of bits/node (even if c is less than the logarithm of the network's size!). Routing efficiently means (informally) that message delivery should only stretch the delivery path by a constant factor. The answer again is yes, as long as the volume of the network grows only polynomially with its radius (otherwise, we run into established lower bounds). This effectively captures every network one may build in a universe (like our own) with finite dimensionality using links of a fixed, maximum length and nodes with a fixed, minimum volume. The results extend the current results for compact routing, allowing one to route efficiently on a much larger class of networks than had previously been known, with many fewer bits.

Thesis Supervisor: Larry Rudolph
Title: Principal Research Scientist

Acknowledgments

I have to thank a vast number of people for their encouragement, support, and advice in writing this thesis, among them its readers (Tom Knight and Piotr Indyk), my fellow students at MIT's computer science lab, and, last but not least, my parents. But above them all I have to thank my advisor, Larry Rudolph, for all these years of guidance and freedom.

Contents

List of Figures

Chapter 1

Introduction

Can one build, from nodes with little reliability, small memory and low computational speed, networks that efficiently scale to arbitrary size and topology? This thesis attempts to address some aspects of this question. This first chapter explores the general question in greater depth; it explains its relevance, both theoretical and practical; it describes the state of the art in the field; and, placing them in the context of the greater whole, it introduces the two issues, robust connectivity and compact routing, that form the core of the thesis.

1.1 A theoretical issue: scalable computing and scalable networking

This thesis focuses on issues of *scalability*, but *in the context of networking rather than in that of computing.* There is a fundamental difference between the two. The function of a computing device, or a network used as a computing device, is to receive an input and produce an output in a "black box" fashion. In order to achieve this result one might have to carefully orchestrate the flow of information between different parts of the computing device; but there is no specific requirement that each component of the device should be individually controllable and capable of communicating efficiently with every other component. In contrast, the fundamental function of a network is

to allow each individual component node to (efficiently) send information to (and receive information from) every other node; whatever computational ability nodes have is purely ancillary to this task.

This thesis explores the issue of the existence of a "universal" node, with a *fixed* amount of resources, that can be used to assemble efficient networks of arbitrary size and complexity. A parallel from the computing world would be that of universal cellular automata that, using lattices of cells with a finite amount of state, can carry out computations of arbitrary complexity (once again, note that cellular automata are typically treated as computing ensembles, rather than networks - one is interested in their global behavior rather than in the ability of having arbitrary pairs of cells communicate). In this regard, cellular automata sharply contrast with other parallel computing models, such as the PRAM [27], where the complexity of the individual CPU, and in particular the size of its registers, must grow with the number of CPUs involved in the system and with the amount of memory. It is natural then to ask whether one can build a network where each node can route information to any other node, even if the memory of each node is a constant independent of the size of the network - and in particular less then the logarithm of the size of the network. Or must the resolution at which we can address the individual components of a system necessarily worsen as the system increases in size?

Note that scalability is not only about memory size - but also, for example, about reliability. Engineering considerations, but also fundamental physical limits, make it impossible to create devices that operate with 100% reliability. Instead, in any given time interval, every elementary device - whether a register storing 1 bit, a half adder adding 2 bits, or a physical link transmitting 1 bit, has a certain probability of malfunctioning. As the space and time complexity of a computation increases, there are more points in space and time where a computing device carrying it out may fail; similarly, as the size and diameter of a network increases, there are more points where information in transit may be lost or corrupted. It is then natural to ask whether, as the complexity and diameter of a network grow, one can achieve communication between arbitrary pairs of nodes with non-vanishing probability, even

at distances considerably larger than the mean distance between failures. Or can one achieve reliable communication only at bounded distances in any system subject to a constant density of noise?

In addition to whether scalable networking is feasible, it is natural to ask how efficient it can be made. For example, even if it is feasible to route between nodes with relatively little memory, how much longer are the resulting paths compared to a network where every node had an unbounded amount of memory available? Are there tradeoffs between resources, so that one can achieve communication with little memory, but only with high reliability, or viceversa?

1.2 Practical applications

It might seem that the questions we raise are only of theoretical relevance. For example, one might argue that a few hundred bits are in any case sufficient to address every elementary particle in our galaxy - and today even the smallest processors easily exceed those memory requirements. Yet, we believe the questions raised by this thesis can actually have practical impact. Some immediate, practical consequences of our results will become apparent during their exposition. But we can immediately begin to argue about practical impact of this line of research in the long term.

Advances in nanotechnology and biological engineering promise to bring us, over the next decade or two, a wealth of "smart materials", composed by myriads of minuscule sensor/actuators units. Each unit would be constantly interacting with the environment it is immersed in on an extremely fine scale: acquiring information, exchanging it with its neighbors (turning the whole material into an extremely large adhoc network) and potentially reacting (e.g. exerting mechanical force). A number of prototypes have already been built, the most well known probably being the Berkeley motes [18] and the Intel motes [23]; all these employ silicon based chips for their computational needs. While these chips have indeed a memory capacity of several kilobytes at least, the prototypes are already pushing the limits of electronic miniaturization, while each node is still fairly bulky: even shrinking the size of each gate to

one atomic radius, maintaining the current node-to-gate size ratio individual nodes would still be visible to the naked eye. In order to achieve true nanoscale interaction with reality, the logic used by these sensors/actuators must be made simpler.

In addition, because in smart materials computing functions are secondary to sensor/actuator functions, researchers are looking to substrates other than silicon to base individual nodes on - substrates that, although offering other advantages in terms of interaction with the environment, offer much weaker computational potential. One prime example would be living cells whose protein transcription logic has been hijacked to provide computing/networking functionality (for preliminary examples of what can already be achieved through this approach see [7]). In the case of protein transcription logic, energy bounds on the maximum protein transcription load that a cell can bear limit the total number of logic gates that can simultaneously have a positive output to perhaps a hundred - more than an order of magnitude less than even the very earliest (and smallest) silicon chips.

True scalability in terms of reliability would also be a boon. As it stands, silicon is sufficiently reliable to guarantee functionality on chips of a few hundred million transistors (though safety margins are not particularly large). But just like silicon is the cutting edge for density, it is the cutting edge for reliability. No other substrate matches it, from living cells to nanomechanical devices. Thus, turning to other substrates for smart materials, composed by billions or even trillions of units, would definitely mean dealing with adhoc networks where faults on any given path connecting two nodes are almost a certainty.

In fact, even some current systems based on silicon experience serious difficulties due to architectures that do not scale well in terms of reliability; and would benefit from any approach that guaranteed scalable reliability. A prime example are sensor networks deployed in "extreme environments". High temperatures, mechanical stress, and even simply environmental radiation found in space or at very high altitudes often compromise their computational abilities of individual units - or simply disable them altogether. In these networks the mean distance between failures can become considerably shorter than the diameter of the network, compromising long range

connectivity. Peer to peer networks, where virtual paths between nodes often fail because of hardware and software heterogeneity, ISP traffic shaping policies, and most of all selfish user behavior, are another area that would benefit from architectural approaches aimed at guaranteeing scalable reliability.

1.3 Towards ultimate network scalability

The ultimate goal of this line of research would be the development of all the layers from the schematics of a simple "universal" computing/networking element (ideally less than 100 gates in terms of circuit complexity), to a truly distributed operating system and expressive programming language capable of efficiently controlling in an arbitrary fashion an arbitrarily large number of such elements arbitrarily connected together, even when a sizable fraction of them (e.g., up to 20%) are subject to failures. Researchers of new nanotechnology "hardware" - whether modified bacterial cells, nanomechanical machines, etc. - who managed to implement the simple schematics would then know that their "platform" would be able to support any desired functionality, without having to worry about how to code that functionality into their hardware. At the same time, application developers would be able to code sophisticated system behaviors in an expressive language, with the knowledge that their work would be portable to any platform that implemented the basic schematics. Such a grand goal is beyond the scope of this thesis; we do, however, take two important, preliminary steps towards it.

Chapter 2 addresses the following question: *in networks affected by a small, but positive rate of"topological noise" - noise due to imprecise node positioning and/or a small probability of failure of nodes and links - (when) can one still have good connectivity at arbitrarily long range?* Good connectivity means that arbitrary pairs of nodes or regions are connected by nearly as many disjoint, fault-free paths as if the entire network were fault-free and nodes were optimally placed. Clearly, good connectivity in the presence of a strictly positive fault rate is not always achievable: a $1-$dimensional line of nodes becomes fragmented into a number of small, disconnected

"islands" if every link has a small probability of failure.

We give a simple topological condition that guarantees good connectivity at arbitrary range and is satisfied in many cases of practical interest. A rigorous formalization of the intuitive notion of "hole" in a (not necessarily planar) graph is at the heart of our result and our proof. It turns out that "holes" effectively erode connectivity in the region "around" them, to a distance that grows logarithmically with the "circumference" of the hole itself, and proportionally to the probability of link/node failure. Extensive simulations refine our theoretical analysis. This result essentially characterizes networks where connectivity depends on the "big picture" structure of the network, and not on the local noise caused by faulty or imprecisely positioned nodes and links. It also nicely complements a recent result [19] by Ho et al. that shows how to automatically exploit good connectivity when it is available, using a simple randomized network coding scheme that uses only local information at every node.

Chapter 3 wields new results in the well studied problem of *compact routing*: is it possible to efficiently route messages in a network of arbitrary topology and size with only a constant number of bits per cell? "Efficiently" means that the total number of nodes involved in the transmission, the total number of bits transmitted, and the total time required for the message to arrive to destination are all within a constant factor of the optimal, i.e. of what would be achievable if every node had an oracle telling it to which of its neighbors route any incoming bits.

It is known [34] that, in general, to achieve efficient routing one needs a number of bits per node *polynomial* in the size of the network. However, we show how a constant number of bits per node is sufficient for efficient routing in any network with polynomial growth - i.e. where the total number of nodes at a distance of at most h hops from any given point is at most polynomial in h. We note that polynomial growth is a fairly mild condition satisfied, for example, by any physically implementable network having only "local" connectivity (it is a much milder condition than low doubling dimension [13], [41]). As a side result, we also obtain an efficient routing scheme for trees that uses only constant bits per node (improving on the

previous logarithmic bound by Thorup and Zwick [42]).

Chapter 2

Robust network connectivity

Can networks scale in terms of connectivity/bandwidth without their nodes having to become more and more reliable as the diameter of the network grows and paths lengthen? And in particular when the average distance between source and destination becomes much longer than the average distance between faults? In this chapter we show that the answer to this question can be positive as long as the network meets some very mild topological conditions: namely, it should not be too "thin" and it should not sport too many large "holes". The notion of hole is at the core of our result, and has a number of implications of immediate, practical applicability to networking, from sensor network simulations to peer to peer networks.

2.1 Introduction

This work analyzes the connectivity of large diameter networks where every link has a probability p of failure. We derive a simple and yet widely applicable condition on the topology of the network that guarantees good connectivity, i.e. a number of edge disjoint, non faulty paths between any two regions of the network almost as high as if no faults where present. Our condition can easily hold even if the two regions are at a distance much larger than the expected "distance between faults", $1/p$, and thus, even if any single path between the two is faulty with high probability. We then extend our result to give a simple condition on the topography of the deployment area of large

diameter radio networks that, despite random positioning of nodes and faults, again guarantees connectivity between regions of the deployment area almost as good as if no faults occurred and if all nodes were placed optimally. Our result characterizes networks where connectivity depends essentially on the "big picture" structure of the network and not on the local "noise" due to imprecise node positioning or to faulty nodes and links.

The connectivity between distant regions in large diameter networks depends on two main factors: large scale structure and small scale noise. The large scale structure of a network is typically linked to the topography of the area in which the network is deployed. For example, a city-wide adhoc radio network is essentially two dimensional (but could have an elongated shape in the case of a city along a river or in a narrow mountain valley) and necessarily has a large number of "holes" in areas, such as ponds or electromagnetically shielded buildings, that nodes cannot populate and through which information cannot flow. As another example, a SmartDust [20] large sensor network deployed on the surface of a building, of an aircraft or of a ship to detect signs of impending structural failure has a topology essentially determined by the nature of the building - a topology that could be particularly intricate in cases such as that of the Eiffel tower. Of course, there are many examples of large scale adhoc distributed peer to peer networks whose "large scale" topology depends not on physical factors, but on the algorithm used to form the links.

Connectivity is also affected by a level of "small scale noise": in large networks a fraction of the nodes is almost inevitably faulty, the deployment process is often imprecise, and sometimes mobility can alter the position of nodes over time. The effects of this noise, even when relatively limited on short range communications, can accumulate and seriously compromise long range connectivity: if every link has a probability p of failure, then the probability that a path of h hops is entirely fault free is $\approx e^{-hp}$, which becomes vanishingly small when h is much larger than the average distance between faults, $1/p$. This can make long range communication problematic in large diameter networks with even a moderate level of small scale noise, such as sensor networks, large mobile radio networks, or nanotechnological ensembles.

This work characterizes networks whose connectivity essentially depends only on their large scale structure and is independent from small scale noise. In these "robustly linked" networks connectivity is always, with high probability, within a small factor of the optimal.

2.1.1 Related work

Most of the (vast) literature on connectivity in the presence of small scale noise studies how different types and levels of small scale noise affect networks with a single, specific large scale structure - rather than how variations in large scale structure affect connectivity in the presence of a given level or range of small scale noise. [1]

Resilience to faults was widely studied in the 1960-80s for now classic networks such as the mesh, the butterfly, the hypercube and many others (for an excellent review see [27]). More recently, the surge of interest in structured peer to peer networks (such as Tapestry [46], Pastry [37], Kademlia [31], Chord [40] or CAN [36]) has been generating a growing literature on the effects of small scale noise caused by individual nodes joining and leaving the network - but again, the focus is on the effects of this noise on a specific large scale structure.

The bulk of the literature on networks without an a priori well specified large scale structure, such as ad hoc mobile networks and sensor networks, seems to study "for simplicity" deployment areas that are either circles or squares. We remark that square and circle enjoy this vast popularity in studies of mobility, routing, energy efficiency etc. even though, due to space limitations, we have to focus this brief review on studies of connectivity. With a "balls and bins" argument, [16] shows that in the unit square populated at random with nodes of communication radius $r << 1$, a population of $\Theta((1/r)^2 \log(1/r))$ nodes, i.e. a $\log(1/r)$ node density, is necessary and sufficient to guarantee that with high probability every node is connected to every other, and that $\Theta(1/r)$ connections can be maintained simultaneously between

[1] "Small world" phenomena [2] [43] [22] [30] [26] - where a few long range random links can radically alter the large scale structure of a network, are a somewhat orthogonal problem; also, most of the related literature focuses on how these long range links shorten path length rather than how they affect the number of disjoint, non-faulty paths between regions.

randomly chosen pairs of nodes. This result actually holds even in the presence of interference under some very mild assumptions on the nature of interference [4].

On the square or circle, a constant node density is sufficient to guarantee that the majority of nodes (rather than all nodes) are connected to each other with high probability, even if they are positioned at random and/or a constant fraction of them and/or of the links is faulty. This is a well known percolation theory result ([8], refined by a long sequence of papers - see [32] for an excellent review). Booth et al. [3] and Gupta and Kumar [15] exploit this property to estimate optimum power ranges for connectivity. Haas et al. [28], Krishnamachari et al. [25], and Sasson et al. [39] exploit it to save on the energy cost of network broadcast: every node fails to retransmit the broadcast (and therefore saves on energy) with a constant probability, without seriously compromising the reach of the broadcast itself. Several other papers investigate the effects on long range connectivity of different small scale connectivity situations: for example Franceschetti et al. [9] consider the case of nodes whose local connectivity area is not a disk but a ring, and Dubhashi et al. [6] consider the effects on long range connectivity of increasing node density, while reducing the fraction of the (growing number) of neighbors with which a node communicates.

Once again, we remark that all the results in the previous paragraph are proved for the square or the circle; there is no evidence that they would hold on deployment areas of different topologies - e.g. long and narrow strips, or even "Sierpinski" circles and squares with many holes. Among the few works that consider areas of different topologies, Dousse et al. [5] point out that, with nodes and faults placed uniformly at random in a strip of constant width, the probability of two nodes being connected becomes exponentially small with their distance; and therefore argue for a reliable, correctly placed infrastructure of base stations to ensure connectivity in the long and narrow valleys of the Swiss Alps. Li et al. [29] investigate experimentally connectivity in ellipse shaped regions between the foci of the ellipse, when the area is populated with a constant density of randomly placed nodes and faults; although no clear asymptotic behavior emerges from their analysis.

The general thrust seems to be that long range connectivity is possible, even with

randomly placed nodes and/or faults, in regions whose width is "sufficient" compared to their length. But there is no clear analytical notion of just what is "sufficient". And there is no study of how connectivity might be affected by the lack of a simply connected topology (informally speaking, what happens if the network is peppered with holes).

2.1.2 Our contribution

We provide a detailed theoretical analysis of the effects of the large scale structure of a network on its connectivity, supported by extensive experimentation. Informally, we show that even extremely narrow strips (of width logarithmic in their length) are sufficient to guarantee good connectivity in a network with imprecisely positioned/faulty nodes and links, *provided there are not too many large holes in the network.* Large holes require larger width, since, in the presence of small scale noise, they can effectively erode connectivity in the region surrounding them to a distance that is logarithmic in the girth (informally, the "circumference") of the hole, and grows with the level of small scale noise (see fig. 2-1).

We model the network as a multigraph whose vertices correspond to cells of the deployment area, and where a cell may contain many network nodes. A link between two cells indicates that those two cells have the potential to communicate, even though that potential has some probability p of not being achieved because of faults and other small scale noise. This approach enables a unified treatment of a large number of cases of practical interest: random and optimal placement of nodes, random faults either in links or in the nodes themselves, with either failstop faults (where faulty nodes/links are simply silent) or Byzantine faults (where faulty nodes jam all communication in their neighborhood). We consider connectivity between pairs of regions (which we term *zones*), rather than between pairs of individual vertices, as this allows much stronger connectivity guarantees. Section 2.2 presents our model, and formally introduces some graph notions crucial to our subsequent analysis. In particular, we formalize the notion of *hole* in a multigraph, which is intuitive in multigraphs representing two dimensional networks yet far more subtle in the case of

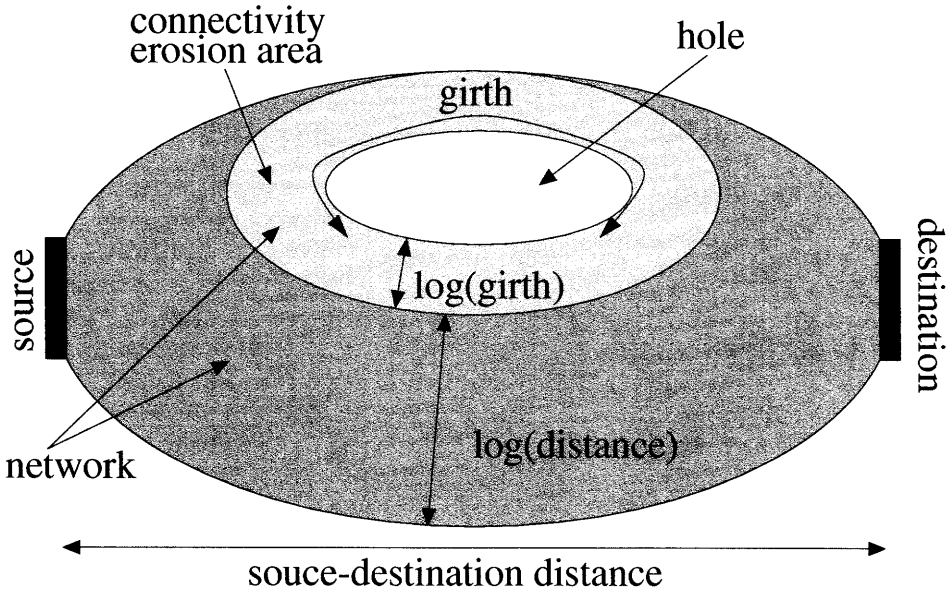

Figure 2-1: Source and destination are robustly linked if connected by a strip/double cone of width logarithmic in its length. Holes effectively "erode" part of the connecting strip width: every hole erodes connectivity up to a distance logarithmic in its width.

complex topologies such as those created by the presence of long range links. Equally important is the notion of *robustly linked zones.* Informally speaking these are zones connected by a strip/cone of width at least logarithmic in their distance even when the region around each hole (to a distance logarithmic in its circumference) is not counted.

Section 2.3 analytically proves our main result - namely, that robustly linked zones enjoy a level of connectivity (measured in terms of edge disjoint fault free paths between them) almost as high as if there was no small scale noise due to faulty or imprecisely positioned nodes or links. Our proof hinges on a novel analysis technique that provides an upper bound to the number of distinct cut sets of a given size between two sets of vertices in a multigraph, paired with Menger's theorem and a Chernoff bound.

Section 2.4 shows how our analysis can be easily extended, not only to point-to-point networks with randomly failing links, but for a much wider variety of networks. It can encompass the effects of faults on both links or nodes, either failstop (where

faulty nodes or links are simply silent) or Byzantine (where faulty nodes maliciously jam all communication in their neighborhood), as well as geometric radio networks where nodes are randomly placed in a deployment area with a given topology.

Simulations, involving as many as a hundred million cells, validate the analysis and give a feel for the constant factors in Section 2.5.

Section 2.6 summarizes our results and analyzes their significance. Some of the implications are counterintuitive and surprising. For example, when designing a multipath connection between two areas to provide fault tolerant connectivity, the widely favored choice of completely disjoint paths turns out to be, in the light of our findings, a strongly suboptimal choice at distances much larger than the average distance between faults.

2.2 A simple network model

This section presents our model (in subsection 2.2.1) and reviews and introduces several concepts central to both the theoretical and the experimental analysis carried out in the subsequent sections; in particular the notion of *holes* (subsection 2.2.2) and that of *robustly linked zones* (subsection 2.2.3).

2.2.1 Networks as multigraphs.

We represent networks as undirected *multigraphs* - informally speaking graphs where the same pair of vertices can be linked by multiple edges. Vertices of the multigraph can represent nodes of the network; they can also represent, instead, "cells" of the deployment area. The latter interpretation allows our model to easily extend to a large number of cases of practical interest, presented in section 2.4. An edge between vertices of the multigraph represents the *potential* of a direct link between two nodes in the first case, or two cells of the deployment area in the second case.

Every edge in the multigraph has an i.i.d. probability p of being *inactive*: inactive edges represent "faulty" links that provide no connectivity. Note that there is a fundamental difference between pairs of nodes/cells that have no direct connectivity

because there is no link between them whatsoever, and pairs nodes/cells that have no direct connectivity because all direct links between them are inactive. The presence or absence of an edge in the multigraph models the topology of the deployment area: it is a fixed, known aspect of the network. For example, an impassable mountain range may prevent connectivity between pairs of antennas/cells on the opposite sides of it; we model this as an absence of edges between the corresponding vertices of the multigraph. The probability p that each individual edge may be inactive represents instead the inevitable uncertainty in the exact positioning/behavior of nodes in the network, a divergence from the "ideal" situation that can be known a priori only in probabilistic terms. For example, two adjacent nodes/cells that are not directly connected only because of a hardware or software malfunction are modeled by two multigraph vertices connected by an inactive edge.

We evaluate the connectivity of the network in terms of the number of edge-disjoint paths that are active (i.e. formed entirely by active edges) between two nodes of the multigraph, or, more in general, between two sets of vertices in the multigraph. For brevity we call such sets *zones*.

We acknowledge that the hypothesis that links may fail with an i.i.d. probability p is a simplification - some simplification in a large network model is inevitable to make it tractable. Our work can be viewed as using a simpler model of local effects in order to analyze with greater accuracy the effects of large scale network structure - as opposed to other studies that make the opposite sacrifice, and use a greatly simplified model of the large scale structure to analyze with greater accuracy local effects. The two approaches can easily complement each other. Remembering that nodes in our multigraph model can represent cells of the deployment area rather than nodes of the network, one can choose a cell size larger than the scale of the local phenomena causing fault correlation. In this way one can study the large scale structure of a network through our model, deriving the parameters that drive it from a model that can deal with the intricacies of intra-cell interactions at a scale where the topology of the network is still quite simple.

2.2.2 Holes, cuts and phases

This subsection formalizes the intuitive notion of *hole* and introduces the related concepts of *girth*, *cut* and *phase*.

Roughly speaking, we represent a hole as the set of all cycles "around" it - cycles that cannot be shrunk smoothly to a single point, but instead "tighten" around the hole itself. This corresponds to the intuitive notion of a hole on a two dimensional deployment region. The intuition is somewhat more fragile when we try to extend it to a multigraph corresponding to a three dimensional deployment region. In this case, the "hole" in the middle of a doughnut fits our definition, but the intuitive notion of "spherical hole" in a piece of swiss cheese does not, since a cycle can "slip to one side" of the surface of the spherical hole and smoothly shrink into nothingness. It is important to note, however, that our formal notion of hole is well defined on *any* multigraph topology, even particularly intricate ones on which visual intuition tends to fail.

In order to give a formal definition of holes, we first have to formally define cycles and what it means to "smoothly" transform one into another. We capture the latter concept of small *topological distance* between two cycles by measuring how many edges must be added/removed at a time to transform one cycle into the other without ever "opening" it.

Definition 1. *A* path *of length ℓ in a multigraph G is a sequence of ℓ edges of G, $(u_1, v_1), \ldots, (u_\ell, v_\ell)$, such that $v_i = u_{i+1}$ for $1 \leq i < \ell$. If $v_\ell = u_1$ the path is a* cycle.

Definition 2. *The* topological distance *between two cycles γ and γ' in a multigraph G is the least d such that there exists a sequence of cycles starting with γ and ending with γ' with no two consecutive cycles in the sequence having symmetric difference* [2] *larger than d.*

Note that the symmetric difference of two sets of cycles (and, in particular, the difference of two cycles) is always a set of cycles, since a set of edges is a set of cycles

[2]The symmetric difference between two sets A and B is $(A \cup B) \ominus (A \cap B)$, i.e. the set of elements in one but not both of A and B.

if and only if every vertex appears in it an even number of times. We are now in a position to formally define holes and hole cuts - set of edges that "cut" every cycle in a hole.

Definition 3. *Given a cycle γ of length g, the set of all cycles at topological distance less than g from γ is a* hole of girth g *if it contains no cycle shorter than γ.* [3]

Definition 4. *A* cut *of a hole h is a set of edges that intersects every cycle of h. A cut of a set of holes $H = \{h_1, \ldots, h_n\}$ is a set of edges that intersects every cycle in every hole in H. A cut of a set of holes H is* minimal *if none of its proper subsets is also a cut of H.*

Note that some cycles may be elements of no hole. These are cycles that "go around" multiple holes of the same girth; and thus are always the symmetric difference of two or more cycles belonging to holes. While the girth of a hole h is in some sense its "circumference", the size of its minimal cuts (those cuts with no "unnecessary" edges) represents the thickness of its "walls" separating h from other holes of equal or greater girth. These walls can obviously contain holes of lesser girth than h, whose cycles are in the symmetric difference between different cycles of h. In the next sections we shall see how the presence of these lesser holes can substantially deteriorate the connectivity in the region surrounding h if they occur at many points where the cuts of h are small. This motivates the notion of phase of a cut, with which we end this subsection and whose significance will become clearer in the next subsection and in the following section.

Definition 5. *A* phase assignment *to a hole h is a cycle γ_h of h whose edges are labeled with distinct positive integers. Given γ_h, the* phase $\phi_{\gamma_h}(C)$ *at which a cut C of h intersects h is equal to the largest label of an edge of γ_h in C.*

Note that for every hole of girth g there is a trivial phase assignment where every cut of the hole has phase at most g - one need only number the g edges of a minimal

[3] This is a slightly different notion than that of hole often used in graph theory (that of a non-chordal cycle). Our definition allows one to model the "thickness of the wall around the hole" - a parameter of crucial importance for this work.

length cycle from 1 to g. In fact, in most situations simply using the girth of a hole instead of the phase of a cut under a particular phase assignment makes the analysis considerably simpler without affecting too much its accuracy.

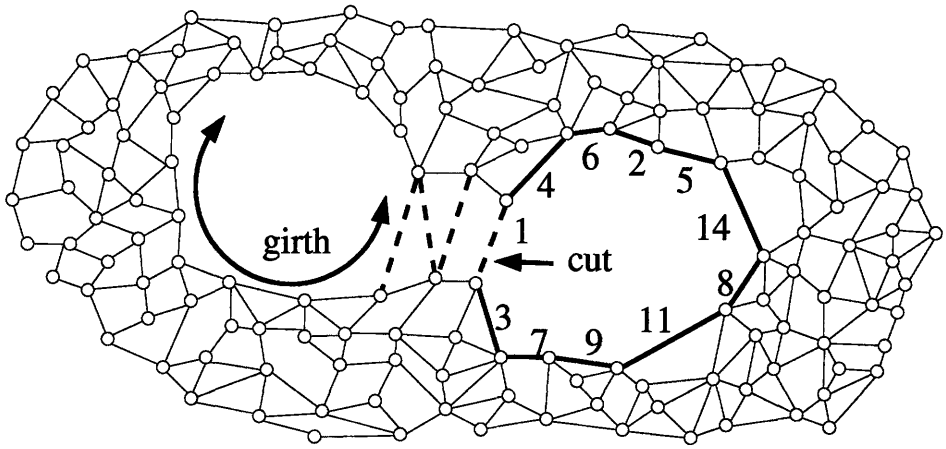

Figure 2-2: A network with two large holes (and many smaller ones). The dotted edges form a minimal cut (of both holes). The numbered edges form a phase assignment to the hole on the right. Note that the only requirement on the labels is that they are distinct positive integers.

2.2.3 Robustly linked zones

In this last subsection we introduce the notion *r-robustly ℓ-linked zones* - which will be central to the remaining sections of the chapter and is, indeed, its cornerstone. Informally speaking, two zones are robustly linked if there are not too many small cuts separating them, and if those cuts do not intersect too many large holes. This guarantees - as shown in section 2.3 - that the number of small cut sets separating the two zones is not too large, limiting the number of points where a few faults might entirely compromise connectivity. Two robustly linked zones are then connected, even in the presence of faults, by almost as many disjoint, fault-free paths as if no faults were present.

More formally:

Definition 6. *Let A and B be two zones of a multigraph G connected by ℓ (edge) disjoint paths. Consider the multigraph $G(A, B)$ obtained by collapsing A and B each*

into a single point, and joining the two points with an edge $e(A, B)$. *A and B are* r-robustly ℓ-linked *if for every hole h of* $G(A, B)$ *there exists a phase assignment* γ_h *such that, for every minimal cut C of all paths from A to B in G, the set of holes of* $G(A, B)$ *cut by C,* $H_{G(A,B)}(C)$, *satisfies:*

$$|C| \geq r \cdot \sum_{h \in H_{G(A,B)}(C)} \log_2(\phi_{\gamma_h}(C)) \tag{2.1}$$

Note that $H_{G(A,B)}(C)$ can be much larger than the set of all holes containing cycles with $e(A, B)$, since in order to cut those holes C might have to intersect holes of lesser girth "embedded" in their "walls". Each of these lesser holes contributes the logarithm of its phase to the right hand term of the inequality above, and therefore diminishes the robustness r with which the two zones are linked.

Equation 2.1 may seem daunting - one may well wonder how easy it is to check its validity for a given multigraph. It is written to be as widely applicable as possible, at the cost of being rather unwieldy. Most of the times, it can be considerably simplified without compromising its applicability. The main way to simplify it is to just use the girth of a hole as an upper bound for the phase term $\phi_{\gamma_h}(C)$ - without having to deal with the intricacies of phase assignment. A further simplification can often be obtained grouping all holes in a region in a *small* set of classes $C_1, \ldots, C_k$ comprising holes of roughly the same girth - typically with each class comprising holes arising from the same "aspect" of the network. For example, a sensor network deployed in the streets of a city with an elongated topology might produce a multigraph with three classes of holes: the main hole obtained from joining together the extremities of the city, secondary holes corresponding to city blocks, and tertiary holes arising from the local geometry of the sensor network (e.g. whether it is deployed in a hexagonal or square lattice). For each class C_i, one can bound the logarithm of the girth of every node with the logarithm of the largest girth g_i of a node in the class, and the size of the cut of each hole with the size c_i of the cut of the smallest hole in the class. Then, equation 2.1 can be weakened to become simply:

$$1 \geq r \cdot \sum_{i=1...k} \frac{\log_2(g_i)}{c_i} \tag{2.2}$$

The rest of this subsection attempts to give a more precise intuition of what it means for two zones to be $r-$robustly $\ell-$linked, an why, for many networks of practical interest, the "robustness factor" r between any two zones is a constant that depends only on the "local" geometry of the network, and not on the distance between the two zones.

Let G be a rectangular mesh of length L and width W (with $L >> W$), and let us see when the two short sides of the mesh (which are connected by W disjoint edge disjoint paths running parallel to the long sides) are $r-$robustly W-linked. The holes of G are essentially all the little squares of side 1 that form the mesh - holes of girth 4 and cut 1.

Collapsing the two short sides of mesh into two points A and B, and connecting them with an additional edge $e(A, B)$, we obtain the graph $G(A, B)$. $G(A, B)$ has all the holes of G, and in addition it has one large hole - that formed by all paths from A to B that have been turned into cycles by the addition of $e(A, B)$. All these cycles, the shortest of which have length $L + 1$, form a single hole h of girth $L + 1$, since they can be "smoothly" changed into each other by adding or removing, one at a time, little square cycles of length $4 < L + 1$. The singleton $\{e(A, B)\}$ is a minimal cut of h, since it intersects all its cycles. Note that every other minimal cut of h has size at least W, since it necessarily intersects every path from A to B.

In this case we can easily show that the two short sides of the mesh are 0.2-robustly W-linked if $W \geq \log_2(L + 1)$ - in fact, they are $r-$robustly $W-$linked if $(\frac{1}{r} - 4)W \geq \log_2(L+1)$. Every minimal cut C of h intersects at most $2|C|$ little square holes of G (since every edge belongs to at most 2 squares), both with phase no larger than their girth, 4. Similarly, it cuts h with a phase no larger than its girth, $L+1$. Then the sum of the logarithms of the phases of all holes intersected by C ($1/r$ of the right-hand term in equation 2.1) is no more than $2|C| \log_2(4) + \log_2(L+1) = |C| \cdot 4 + \log_2(L+1)$, and equation 2.1 is satisfied as long as these two terms add up to at most r times the

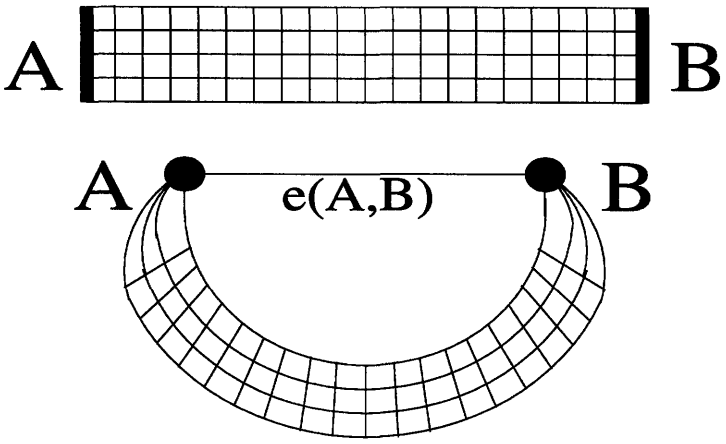

Figure 2-3: The graph G (top), a mesh, and the graph G(A,B) (bottom) obtained by collapsing the two regions A and B into single points and connecting them with an additional edge e(A,B).

size of the cut itself - i.e. as long as $W \geq \frac{r}{1-4r} \cdot \log_2(L+1)$.

This example should give the intuition of why we are considering the graph $G(A,B)$ instead of G itself. For any two zones A and B in an arbitrary multigraph G, the phases of the holes in $G(A,B)$ add together the logarithm of the distance between A and B with the logarithms of the girths of the holes across a section of G itself, and compare the sum with the width of the strip connecting A and B. Two zones are then $r-$robustly linked if they are connected by a strip whose width is at least as large as r times the logarithm of its length, even when every hole "eats up" the surrounding area of the graph up to a distance equal to r times the logarithm of the hole's girth, effectively removing this eroded area from contributing to the width of the strip connecting A and B.

It is natural to then ask why we are introducing the notion of phase, and adding up logarithms of phases, rather than directly adding up logarithms of girths in equation 2.1. The reason is that, for our proof in the following section, we do not need the strips between A and B and around every hole to have a width logarithmic in their length *at every point* in the strip. All we need is for the width to be proportional to the length's logarithm at only $1/2$ the points of the strip; proportional to the length's logarithm minus 1 at only $1/4$ more of the points of the strip; proportional to the

length's logarithm minus 2 at only 1/8 more of the points of the strip; and so on. This weaker condition allows us to consider robust connectivity between single points or small zones with limited fan-out at arbitrarily large distance - not just at distance exponential in the fan-out. For example, two points are robustly linked if they are connected by two conical regions tipped by the two points and joined at the bases, if the section at distance d from the tip of each cone is at least logarithmic with d.

As a somewhat more complex example, consider the d-dimensional hypercube of 2^d nodes - a particularly popular network that is the basis for the networking substrate of both supercomputers (e.g. the Intel HyperCube [33]) and peer to peer networks (e.g. CAN [36]). The holes of a hypercube coincide with its faces - and have therefore girth 4 and minimal cuts of size 1. Consider two points A and B at the opposite sides of the hypercube: $A = 0, \ldots, 0$ and $B = 1, \ldots, 1$. Consider the d (edge disjoint) paths of length d between A and B, $p_0, \ldots p_{d-1}$, where the j^{th} edge of p_i crosses dimension $(i+j)_{mod(d)}$. Every two paths p_i and p_{i+1} run "at one square of distance" from each other. It is then easy to see that all these paths lie in a subgraph of the hypercube (still with all holes of girth 4) where every edge insists on at most two squares. By the same argument used for the mesh, A and B are then e.g. $0.2-$robustly $d-$linked as long as $d \geq 0.2(2d\log_2(4)) + \log_2(d+1))$, which is satisfied for all $d \geq 1$ (again, the first of the two terms on the right side of the inequality comes from the holes of the hypercube proper, and the second from the hole formed when connecting A and B with $e(A, B)$).

We can easily extend this to prove that any pair of points on the $d-$dimensional hypercube are $0.2-$robustly $d-$linked: if the two points differ on $\delta < d$ dimensions, we simply consider the δ paths of length δ on the sub-cube where A and B differ on all dimensions, and $d - \delta$ additional paths of length $\delta + 2$ obtained by first moving away from A on one of the $d - \delta$ dimensions on which A and B coincide, running parallel to one of the "short" paths, and the crossing back to B over the first dimension crossed. By virtue of the theorem proved in the next section, this means that a hypercube can tolerate a constant failure rate independent of its size and still guarantee a number of disjoint paths between two points almost as high as if no faults were present.

A simple example of two regions that are *not* robustly linked, even though they are connected by a very "fat" (in fact, square!) strip, underscores the importance of holes. Consider the graph G formed by two points A, B, and d disjoint paths of length d between them. This graph is full of holes of large girth and small cut! Each contains a single cycle of size $2d$ formed by two of the disjoint paths connecting A and B - the fact that the paths are disjoint ensures that the cycle cannot be "shrunk". A and B cannot then be $r-$robustly d linked with a robustness r that remains bounded away from 0 as d grows, since every edge in a minimal cut intersects at least one such hole. It is indeed easy to see that for any given probability p of link failure, as d grows A and B become almost certainly disconnected. The probability that any particular path between them is entirely fault free is at most e^{-dp} - meaning that the probability of at least one entirely fault free path between the two points is no larger than de^{-dp}, which quickly converges to 0 as d grows much larger than $1/p$ (i.e. when A and B are at a distance much larger than the "expected distance between faults").

2.3 Robustly linked zones are connected by active paths w.h.p.

This section is devoted to proving that, if two zones A and B are $r-$robustly ℓ-linked, then with high probability they are linked by $\frac{2}{3}\ell$ disjoint active paths even if links are inactive with some probability p that depends solely on r. More formally:

Theorem 1. *If A and B are r-robustly ℓ-linked in a multigraph G (or in a supergraph of G) in which every link is independently inactive with probability $p < 2^{-\frac{6}{r}+9}$, then the probability that at less then $\frac{2}{3}\ell$ disjoint and fully active paths connect them is $2^{-\Omega(\ell)}$.*[4]

Note that the constants involved in theorem 1 (and, later, in theorem 2) are rather large, due to a number of simplifications in the proof. We refine the constants in section 2.5 through extensive simulations.

[4]With the $\Omega(\ell)$ here and in theorem 2 we mean "at least $k\ell$ for some constant k that does not depend on the multigraph but only by the margin by which p satisfies the inequality."

Proof. Our proof proceeds in two steps. First, we obtain an upper bound on the number of minimal edge cuts of a given size intersecting all paths between A and B. Then we apply Chernoff's inequality to bound the probability that in any of them less than $\frac{2}{3}\ell$ links are going to be active, thereby proving the thesis by virtue of Menger's theorem.

To obtain an upper bound on the number of minimal edge cuts of a given size intersecting all paths between A and B, we consider the multigraph $G(A, B)$ obtained from G by collapsing A and B each into a single point, and adding a single edge $e(A, B)$ between the two. Note that any cycle of $G(A, B)$ containing $e(A, B)$ corresponds to a path connecting A and B in G, and that if and only if E is an edge cut intersecting all paths from A to B in G then $E \cup \{e(A, B)\}$ is an edge cut intersecting all paths from A to B in $G(A, B)$. We then construct and analyze a *cut tree* $T(A, B)$, a tree whose edges are each labeled with an edge of $G(A, B)$. More precisely, all the edges connecting any node of $T(A, B)$ to its children are labeled with distinct edges from a phase assignment of a hole in $G(A, B)$. The edge labels implicitly associate to each node v of $T(A, B)$ the set of edges of $G(A, B)$ labeling the path that connects v to the root of $T(A, B)$. We prove that every minimal edge cut intersecting all paths from A to B is associated to at least one leaf of $T(A, B)$ - obviously at a height equal to the number of edges in the edge cut. Then, an upper bound on the number of leaves of $T(A, B)$ at height h translates into an upper bound on the number of distinct minimal edge cuts of size h intersecting all paths from A to B. We prove such a bound leveraging the fact that A and B are $r-$robustly $\ell-$linked: intuitively, if this is the case, then there cannot be too many holes of large girth intersected by any path connecting A and B, and therefore the branching factor of $T(A, B)$ cannot be too large.

Let us analyze the process through which the algorithm builds the cut tree $T(A, B)$. For any node v of $T(A, B)$, let $E(v)$ be the set of all edges of G labeling the path from v to the root of $T(A, B)$, plus the edge $e(A, B)$ connecting A and B in $G(A, B)$ (but not in G). The cut tree $T(A, B)$ is iteratively created as follows:

1. Create the root of $T(A, B)$.

2. While there exists a current leaf v of $T(A, B)$ (possibly the root if yet without children) such that there exists a cycle with an odd number of edges of $E(v)$:

   (a) Let h be a hole of minimum girth containing one such cycle.

   (b) For all edges $e_1, \ldots, e_n$ in γ_h such that $E(v) \cup \{e_i\}$ is a subset of a minimal cut of all paths from A to B add a child v_i to v and label the edge connecting v and v_i with e_i.

Note that throughout the construction process, for every node v of the cut tree $T(A, B)$, $E(v)$ is a (not necessarily proper) subset of a minimal cut of all paths from A to B in $G(A, B)$: this property holds when the root of $T(A, B)$ is created, since every edge cut of all paths from A to B must include $e(A, B)$, and, by construction, it also holds whenever a new node is added to the cut tree.

The key to proving that the construction process terminates, with every minimal edge cut intersecting all paths from A to B associated to at least one leaf of $T(A, B)$, is the following lemma:

Lemma 1. *Every node v of $T(A, B)$ such that $E(v)$ is a* proper *subset of some minimal edge cut C intersecting all paths from A to B in $G(A, B)$ will eventually have a child v' such that $E(v')$ is also a (larger) subset of C.*

Proof. We first prove that throughout the construction process, as long as there exists a node v of the (partially constructed) cut tree that is currently childless when $E(v)$ is not an edge cut of all paths from A to B, at least one more iteration of the while loop will take place. In this case, there is certainly a path p connecting in $G(A, B)$ A to B without edges in $E(v)$. Also, since $E(v)$ is by construction a subset of a *minimal* edge cut of all paths from A to B, for every edge in $E(v)$ there must be a path connecting A to B containing only that edge of those in $E(v)$ - otherwise that edge could be removed without compromising any cut that is a superset of $E(v)$, and $E(v)$ would no longer be a subset of a *minimal cut.* One such path and p then form a cycle in $G(A, B)$ with exactly one edge in $E(v)$, making the condition of the while loop hold.

Let h be a hole of least girth g among those containing a cycle with an odd number of edges in $E(v)$. Such a hole certainly exists, since a cycle that satisfies the condition of the while loop either belongs to a hole, or is the symmetric difference of two or more cycles - at least one of which must contain an odd number of edges in $E(v)$ - that all belong to holes. We now prove that *all* cycles of h, and in particular γ_h, contain an odd number of edges in $E(v)$. Suppose this is not the case. Then h contains two cycles, the first with an odd and the second with an even number of edges in $E(v)$, and there is a sequence of cycles starting with one and ending with the other, such that every pair of consecutive cycles has a symmetric difference smaller than g. Of the set S_g of all such sequences of cycles, consider the (nonempty) subset S_{g-1} consisting of all those sequences in S_g minimizing the number of pairs of consecutive cycles with a symmetric difference of size $g-1$. Iteratively, of every set S_i, $0 < i < g-1$, consider the (nonempty) subset S_{i-1} consisting of those sequences minimizing the number of pairs of consecutive cycles with a symmetric difference of size $i-1$. Consider an arbitrary sequence of cycles in S_0; since the sequence would begin with a cycle with an odd number and end with a cycle with an even number of edges of $E(v)$, there is at least a pair of consecutive cycles in the sequence, γ and γ', the first with an odd and the second with an even number of edges of $E(v)$.

All that is left to prove is that the symmetric difference between the two cycles, $\gamma \ominus \gamma'$, which must obviously contain an odd number of edges in $E(v)$, consists of a single cycle δ belonging to a hole of girth $|\delta| < g$ - contradicting the hypothesis that g is currently the minimal girth of a hole containing an odd number of edges of $E(v)$. If $\gamma \ominus \gamma'$ could be partitioned into $n \geq 2$ cycles $\delta_1, \ldots, \delta_n$, each smaller than $\gamma \ominus \gamma'$ then, by inserting between the γ and γ' the $n-1$ cycles $\gamma \ominus \delta_1, \ldots, \gamma \ominus \delta_1 \ominus \cdots \ominus \delta_{n-1}$, one would remove from the sequence a pair of consecutive cycles with symmetric difference of size $\gamma \ominus \gamma'$, introducing $n-1$ pairs of cycles with smaller symmetric differences - violating the hypothesis that the sequence under consideration is in S_0 and therefore also in $S_{\gamma \ominus \gamma'}$. Then $\gamma \ominus \gamma'$ consists of a single cycle δ. If δ were at topological distance less than $|\delta|$ from a cycle δ_1 such that $|\delta_1| < |\delta|$, i.e. there existed a sequence of n cycles $\delta_1, \ldots, \delta_n = \delta$ starting with δ_1 and ending with δ such that

$|\delta_i \ominus \delta_{i+1}| < |\delta|$ for all positive $i < n$, then, by inserting $\gamma \ominus \delta_1, \ldots, \gamma \ominus \delta_{n-1}$ between γ and γ', we would replace in the sequence chosen from S_0 a pair of consecutive cycles whose symmetric difference has size $|\delta|$ with several pairs of consecutive cycles with smaller symmetric differences, violating again the hypothesis that the sequence of cycles under consideration belongs to S_0 and therefore also to S_δ. Then $\gamma \ominus \gamma'$ contains a single cycle δ with an odd number of edges in $E(v)$ belonging to a hole of smaller girth than h, contradicting the hypothesis. This proves that indeed, if h is a hole currently of least girth containing a cycle with an odd number of edges of $E(v)$, then all cycles of h, and in particular γ_h, must also contain an odd number of edges of $E(v)$.

Finally we can prove that every minimal cut C of all paths from A to B that is a superset of $E(v)$ must also contain at least one edge from γ_h not already in $E(v)$, thereby proving the lemma, since by construction on step 2.*b* we add one child to v for every such edge. Since C is a *minimal* cut of all paths from A to B, each edge of C must have one vertex that is either in A or connected to A by a path with no edges of C, and one vertex that is either in B or connected to B by a path with no edges in C; let us call them the A-vertex and the B-vertex of that edge. If, while following γ_h, we encounter first the A-vertex and then the B-vertex of an edge in C, then, of the next edge in C we encounter, we must first encounter the B-edge and then the A-edge - or there would be a path with no edges of C connecting A to B. Symmetrically, whenever we encounter of an edge of C first the B vertex, of the next edge of C we must encounter first the A vertex. Then, γ_h must contain an even number of edges of C, and therefore at least one that is not in $E(v)$. □

As a consequence of Lemma 1 we can immediately prove the following:

Lemma 2. *The construction process of $T(A, B)$ eventually terminates, and for every edge cut C intersecting all paths from A to B in G there exists at least one distinct leaf v at height $|C|$ in $T(A, B)$ such that $E(v) = C \uplus \{e(A, B)\}$.*

Proof. The construction of $T(A, B)$ eventually terminates, since by Lemma 1 at every iteration of the while loop the cut tree grows by one edge and the cut tree is limited

in degree and height by the number of edges in $G(A, B)$.

We now prove that, for each minimal edge cut $C(A, B)$ intersecting all paths from A to B in $G(A, B)$, there is at least one node v at height $|C|$ in $T(A, B)$ such that $E(v) = C(A, B)$. This follows from the observation that, if we denote with ρ the root of the cut tree $T(A, B)$, $E(\rho) = \{e(A, B)\}$ is a subset of every minimal cut intersecting all paths from A to B in $G(A, B)$. Then, by Lemma 1, for every minimal cut $C'(A, B)$ of all paths from A to B in $G(A, B)$ there is a path from the root to a leaf of $T(A, B)$ such that $C'(A, B)$ is a superset of each of the (growing) edge sets $E(v)$ associated to each node v encountered along the path, the last of which must coincide with $C'(A, B)$ itself.

The thesis follows immediately remembering a set of edges C of G is a minimal edge cut intersecting all paths from A to B in G if and only if $C \uplus \{e(A, B)\}$ is a minimal edge cut intersecting all paths from A to B in $G(A, B)$, and that, for any node v at height h in $T(A, B)$, $|E(v) \ominus \{e(A, B)\}| = h$. □

Our next goal is to obtain an upper bound on the number of leaves of $T(A, B)$ at a given height - which by virtue of Lemma 2 immediately translates into an upper bound on the number of minimal edge cuts of a given size intersecting all paths from A to B in G. We prove the following:

Lemma 3. *If A and B are $r-$robustly $\ell-$linked the number $n(h)$ of leaves at height h in the cut tree $T(A, B)$ satisfies:*

$n(h) = 0$ *for* $h < \ell$ *and*

$n(h) \leq 2^{2\lceil \frac{h}{r} \rceil + 3h - 2}$ *for* $h \geq \ell$.

Proof. Since a leaf at height h corresponds, by virtue of Lemma 2, to an edge cut of size h intersecting all paths from A to B in G, and no such edge cut of size less than ℓ exists since A and B are ℓ-linked, we immediately have that $n(h) = 0$ for $h < \ell$.

To bound $n(h)$ for $h \geq \ell$, consider a generic leaf v of $T(A, B)$, and let $C(v) = E(v) \ominus \{e(A, B)\}$ be the set of edges labeling the path from v to the root of $T(A, B)$. Obviously the height of v is equal to $|C(v)|$. Note that every edge e_i , $1 \leq i \leq |C(v)|$, of $C(v)$ belongs to the phase assignment γ_{h_i} of some hole h_i, cut by $C(v)$ since

when h_i was taken under consideration in the first step of the while loop all its cycles contained an odd, and therefore positive, number of edges from a subset of $C(v)$. For every edge e_i let ψ_i be its label in γ_{h_i}, which by definition is equal or less than the phase of $C(v)$ on h_i. Let us now focus our attention on the sequence $\lceil \log_2(\psi_1(v)) \rceil, \ldots, \lceil \log_2(\psi_{|C(v)|}(v)) \rceil$. It is easy to see that the leaves u of $T(A,B)$ at the same height $|C(v)|$ as v that satisfy $\lceil \log_2(\psi_i(u)) \rceil = \lceil \log_2(\psi_i(v)) \rceil$ for all positive $i \leq |C(v)|$ are at most $\prod_{i=1}^{|C(v)|} 2^{\lceil \log_2(\psi_i(v)) \rceil} = 2^{(\sum_{i=1}^{|E(v)|} \lceil \log_2(\psi_i(v)) \rceil)}$, since if u and v are distinct $\psi_i(v) \neq \psi_i(u)$ for some i and there are at most 2^k distinct positive integers whose base 2 logarithm, rounded up to the nearest integer, is k. Since A and B are r-robustly ℓ-linked, we have that $r \sum_{i=1}^{|C(v)|} \log_2(\psi_i) \leq |C(v)|$, i.e. $\sum_{i=1}^{|C(v)|} \log_2(\psi_i) \leq \frac{|C(v)|}{r}$ and therefore $\sum_{i=1}^{|C(v)|} \lceil \log_2(\psi_i) \rceil \leq \lceil \frac{|C(v)|}{r} \rceil + |C(v)| - 1$. Since A and B are r-robustly ℓ-linked, we have that $r \sum_{i=1}^{|C(v)|} \log_2(\psi_i) \leq |C(v)|$, i.e. $\sum_{i=1}^{|C(v)|} \log_2(\psi_i) \leq \frac{|C(v)|}{r}$ and therefore $\sum_{i=1}^{|C(v)|} \lceil \log_2(\psi_i) \rceil \leq \lceil \frac{|C(v)|}{r} \rceil + |C(v)| - 1$.

The number of distinct sequences of $|C(v)|$ non-negative integers $n_1, \ldots, n_{|C(v)|}$ such that $\sum_{i=1}^{|C(v)|} n_i \leq k$ is at most $\binom{\lfloor k \rfloor + |C(v)|}{|C(v)|}$. Then, the total number of leaves at the same height h in $T(A,B)$ is:

$$
\begin{aligned}
n(h) &\leq \binom{\lfloor \lceil \frac{h}{r} \rceil + |h| - 1 \rfloor + h}{h} \cdot 2^{\lceil \frac{h}{r} \rceil + h - 1} \\
&\leq \binom{\lceil \frac{h}{r} \rceil + 2h - 1}{h} \cdot 2^{\lceil \frac{h}{r} \rceil + 2h - 1} \\
&\leq 2^{2\lceil \frac{h}{r} \rceil + 3h - 2}
\end{aligned}
$$

□

We now derive a simple upper tail Chernoff bound for the probability that, of a set of $n \geq \ell$ edges, each inactive with probability p, more than $n - \frac{2}{3}\ell$ are inactive, i.e. less than $\frac{2}{3}\ell$ are active. The probability that, of n trials, each negative with probability p, more than $(1+\delta)np$ are negative is less than $(\frac{e^\delta}{(1+\delta)^{(1+\delta)}})^{np}$, which, letting $(1+\delta)np = n - \frac{2}{3}\ell$, becomes $e^{-np} \cdot (\frac{enp}{n - \frac{2}{3}\ell})^{n - \frac{2}{3}\ell}$.

This upper bound can be combined with that of Lemma 3 to give an upper bound to the probability that any edge cut set of G will have less than $\frac{2}{3}\ell$ active nodes:

$$
\begin{aligned}
P(A,B) &< \sum_{i=\ell}^{\infty} e^{-pi} \cdot (\frac{epi}{i - \frac{2}{3}\ell})^{i - \frac{2}{3}\ell} \cdot 2^{2\lceil \frac{i}{r} \rceil + 3i - 2} \\
&\leq \sum_{i=\ell}^{\infty} (3ep)^{\frac{i}{3}} \cdot 2^{i(\frac{2}{r} + 3)} \\
&= \sum_{i=\ell}^{\infty} 2^i (\frac{\log_2(p)}{3} + \frac{2}{r} + \frac{\log_2(3e) + 9}{3})
\end{aligned}
$$

where the last term is $2^{-\Omega(\ell)}$ as long as $\log_2(p) <$
$-\left(\frac{6}{r} + \log_2(3e) + 9\right)$.

By Menger's theorem, this is also an upper bound to the probability that less than $\frac{2}{3}\ell$ active disjoint paths exist in G between A and B, completing the proof of theorem 1. □

2.4 Extending the applicability

Our model can be applied directly to networks with point-to-point connectivity where the position of nodes is well known a priori, and the only source of divergence from the ideal situation are faults in the network links. In this section we show how it can be applied to many other cases of practical interest at the cost of some small measure of approximation in the parameters that define the network. In subsection 2.4.1 we consider a "node-centric" model, where nodes (rather than edges) of a multigraph may become inactive with some probability p, and one is interested in the number of node disjoint (rather than edge disjoint) active paths between two zones. In subsection 2.4.2 we show how to leverage the results of subsection 2.4.1 to model large adhoc radio networks with random placement of nodes in the deployment area (both in the absence and in the presence of node faults): in this case connectivity depends on the topography of the deployment area.

2.4.1 A node-centric model

In the previous section we have analyzed networks as multigraphs where links between nodes may become inactive, evaluating their connectivity in terms of active edge disjoint paths between two zones. In this subsection, we show how our model can be easily adapted to be "node-centric", considering faulty nodes instead of faulty edges, and evaluating the number of node disjoint rather than edge disjoint paths between two zones.

We can easily adapt the techniques of Theorem 1 considering, instead of edge cuts, *node cuts* between two zones, i.e. sets of nodes such that every path between the two

zones contains at least one edge incident on a node from the set; and in particular *minimal* node cuts between two zones, i.e. node cuts without any proper subsets that are themselves node cuts between those two zones. We can then prove the following:

Theorem 2. *Let A and B be two zones of a multigraph G. If G can be extended by adding a set of nodes B' in such a way that:*

1. *every node of B' is connected only to nodes of B and every node of B is connected to at least one node of B',*
2. *the resulting graph $G(A, B)$ has degree at most d,*
3. *A and B' are $r-$robustly linked in $G(A, B)$,*
4. *there exist at least ℓ node-disjoint paths between A and B',*

then, even if every node is inactive with independent probability $p \leq 2^{-(\frac{6d}{r}+(\log_2(3e)+9d))}$, the probability that less than $\frac{2}{3}\ell$ node disjoint active paths exist between A and B in G is $2^{-\Omega(\ell)}$.

The "addition" of the set B' and condition 1 are essentially technicalities to deal with the degenerate case of the source and destination sharing some nodes; the only substantial condition imposed in addition to those of theorem 1 is the upper limit on the degree of nodes in the region between source and destination (to ensure that node cuts between them are of sufficient size).

Proof. Note that the set of all node cuts between A and B in G coincides with the set of all node cuts between A to B' in $G(A, B)$ that do not contain nodes of B' - this follows from the fact that one can reach a node of B' only through some node(s) of B and that from any node of B one can always reach some node(s) of B'.

The major effort in the proof is proving that for any minimal node cut C between A and B in G there exists a *distinct* minimal edge cut of size at most $d|C|$ between A and B' in $G(A, B)$. For any (not necessarily minimal) node cut C between A to B consider the set of all paths from some node of B' to some node of C with only one edge (the last) incident on a node of C; denote with $E(C)$ the set of all such "last"

path edges incident on C. Obviously $E(C)$ is a (not necessarily minimal) edge cut between A and B'; and if C is minimal, for every node $v \in C$, every subset of $E(C)$ that is also an edge cut between A and B' must contain at least one edge incident in v. We associate to every minimal node cut C between A and B an arbitrary subset $E_m(C)$ of $E(C)$ that is a minimal edge cut between A and B', and prove that, for any distinct pair of minimal node cuts C_1 and C_2 between A and B, there is at least one edge in one but not both of $E_m(C_1)$ and $E_m(C_2)$.

We prove that in $E(C_1 \cup C_2)$ there is at least one edge in one but not both of $E_m(C_1)$ and $E_m(C_2)$. In order to do so, we first prove that at least one edge $e \in E(C_1 \cup C_2)$ is incident on a node v in one but not both of C_1 and C_2. Suppose this is not the case. Then every edge in $E(C_1 \cup C_2)$ is incident on some node belonging to a set of nodes $C_3 \in E(C_1) \cap E(C_2)$; meaning that every path from $C_1 \cup C_2$ to B', and therefore every path from A to B, passes through C_3 - against the hypothesis that C_1 and C_2 are both minimal and distinct. Assume without loss of generality that v belongs to C_1 and not to C_2. Then e cannot belong to $E_m(C_2)$ (having no vertices in C_2), and all we have left to prove is that it belongs to $E_m(C_1)$.

By definition of $E(C_1 \cup C_2)$, there exists a path p starting with the vertex of e in C_1, and reaching B' without ever touching any other vertex of either C_1 or C_2. Since C_1 is minimal, then either $v \in A$, or there exists a path p' from A to v where the only edge incident on C_1 is the last. In the first case, p is a path from A to B' and its only edge incident on a vertex of $E(C_1)$ is e, so e must belong to every minimal edge cut between A and B' that is a subset of $E(C_1)$. In the second case, consider the path $< p', p >$ from A to B': only two of its edges are incident on some vertex of C_1: the last edge e' of p' and the first edge e of p. Every minimal edge cut between A and B' that is a subset of $E(C_1)$ must then contain at least one of them. It is easy to see that $e' \notin E(C_1)$, or there would be a path from its vertex not in C_1 to some node of B' without any edges incident on nodes of C_1, which, juxtaposed to $p \ominus e'$, would form a path from A to B' that never crosses C_1. Then e must belong to any minimal edge cut that is a subset of $E(C_1)$; and then $E_m(C_1) \neq E_m(C_2)$, proving that to every minimal node cut C between A and B in G we can associate a distinct

minimal edge cut $E_m(C)$ between A and B' in $G(A, B)$.

We can then apply Lemma 3 used in the proof of Theorem 1 in the previous section to obtain an upper bound of $2^{2\lceil \frac{hd}{r} \rceil + 3hd - 2}$ on the number of node cut sets of size h intersecting all paths from A to B in G for $h \geq \ell$ - obviously, by virtue of Menger's theorem, no node cuts of size less than ℓ intersect all paths from A to B in G. Employing the same Chernoff bound used in Theorem 1 we then have that the probability that less than $\frac{2}{3}\ell$ node disjoint active paths exist between A and B in G is no more than:

$\sum_{i=\ell}^{\infty} e^{-pi} \cdot (\frac{epi}{i-\frac{2}{3}\ell})^{i-\frac{2}{3}\ell} \cdot 2^{2\lceil \frac{id}{r} \rceil + 3id - 2}$

$\leq \sum_{i=\ell}^{\infty} (3ep)^{\frac{i}{3}} \cdot 2^{i(\frac{2}{r}+3)}$

$= \sum_{i=\ell}^{\infty} 2^{i(\frac{\log_2(p)}{3} + \frac{2d}{r} + \frac{\log_2(3e)+9d}{3})}$

where the last term is $2^{-\Omega(\ell)}$ if $p < 2^{-(\frac{6d}{r} + (\log_2(3e)+9d))}$.

□

2.4.2 Geometric Radio networks

In this subsection we leverage the results of subsection 2.4.1 to analyze the connectivity of large adhoc radio networks where individual nodes (of which a small fraction may be faulty) are placed uniformly at random in the deployment area. We derive a condition on the topography of the deployment area that guarantees that the "noise" due to random placement of nodes and to a small fraction of faulty nodes produces only a small reduction of connectivity compared to the case of optimal placement of entirely non-faulty nodes.

The fundamental idea is to interpret the nodes of our multigraph model as cells of the deployment area rather than as nodes of the network itself. For example, tiling a 2-dimensional deployment area with hexagonal cells of side at most $1/\sqrt{13} \approx 27\%$ of the communication radius of individual radios, a node in a cell is always within range of any node in any of the 6 adjacent cells. We can then obtain an upper bound on the probability p that each cell of the multigraph is "faulty" as a function of active node density n measured in terms of average active nodes/cells, $p \leq e^{-n}$; and exploit the analysis of the previous section and subsection to obtain a condition, on

the topography of the area in which a radio network is deployed, sufficient for almost optimal connectivity between two regions despite a constant fraction of silently faulty nodes, and despite random placement of all nodes. All that is required is a thin strip of width (measured in terms of cells) logarithmic in the distance between the two regions, with either no holes, or enough extra width to compensate for every hole "eroding" a strip around it of thickness logarithmic in the hole's "circumference" - the base of the logarithm being a function of r.

In fact, with this cell oriented approach we can model not only failstop failures that simply silence a node, but also maliciously active nodes which jam all communication within their communication radius. The key idea is to evaluate upper bounds to the probability p_{jam} that any given cell falls within the communication radius of a malicious node and is jammed, and to the maximum number of cells max_{mal} in a minimal cut that a single malicious node can affect. For most "reasonable" tilings - tetrahedral, cubic, hexagonal, square or triangular - max_{mal} is a relatively small constant that depends only on the *local* geometry of the tiling, while $p_{jam} \leq knp_{mal}$, where p_{mal} is the probability that any given node is malicious, n is the node/cell density introduced above, and k is, like max_{mal}, a relatively small constant depending only on the *local* geometry of the tiling.

2.5 Simulations

Simulations confirm our theoretical analysis as to how connectivity is affected by faults in robustly linked networks. In particular, they show that even with relatively high link failure probabilities, thin strips of a few dozen nodes can maintain good connectivity at extremely long ranges (millions of hops). Furthermore, they confirm the logarithmic relationship between the length of the connecting strip and minimum width to guarantee connectivity. Finally, they confirm the drastic reduction of connectivity caused by holes.

Consider a long rectangular network, tiled with a hexagonal lattice, where the two "short" sides of the rectangle constitute the source and destination zones and are

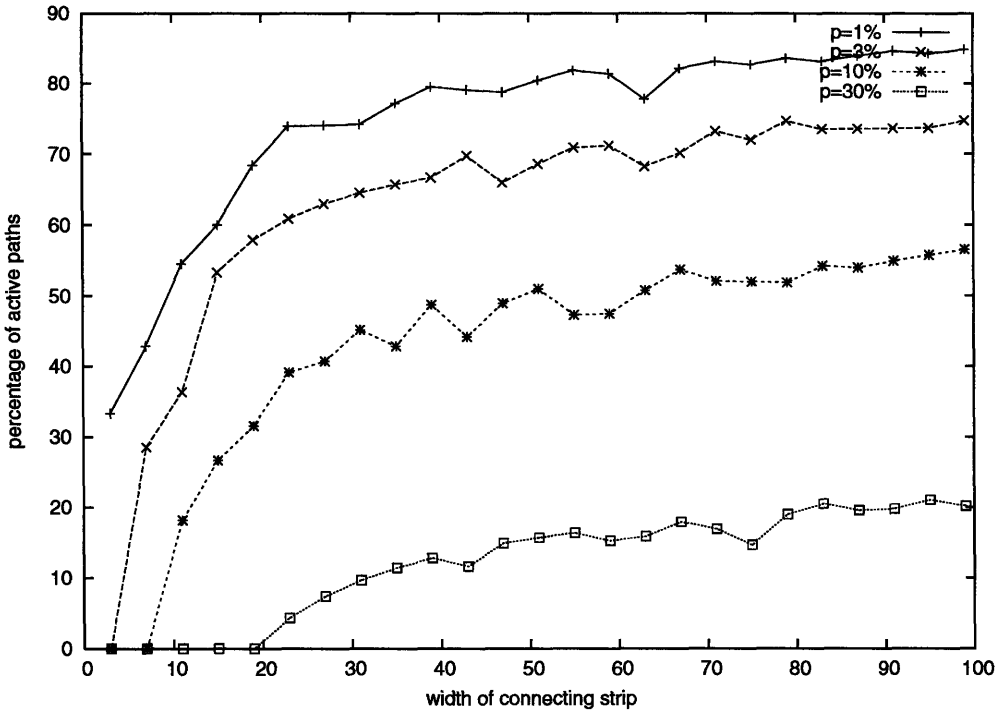

Figure 2-4: Efficiency of usage of a 10^6 hexagon long strip, as a function of its width (expressed as a number of disjoint paths), for different failure probabilities.

parallel to one side of the hexagons. The strip is 10^6 hexagons long, and we vary its width from 1 to 50 hexagons, i.e. from 1 to 99 disjoint paths (a width of h hexagons translates into a width of $2h - 1$ paths).

Figure 2-4 shows the number of active paths between source and destination, as a percentage of the number that would be active in the absence of faults, as a function of the width of the connecting strip, for various fault probabilities ("1%", "3%", "10%" and "30%"). The percentage, which we can interpret as the efficiency of usage of the strip, increases with the width of the strip; although there is relatively little increase above a width of $30 - 40$ paths, reflecting the fact that this is indeed the critical width range needed to support, in the presence of faults, efficient communication at a distance of up $1,000,000$ hops.

Our theoretical analysis predicts that this "critical width", at which efficiency peaks, should grow as the logarithm of the length of the connecting strip. Our simulations validate this prediction. In Figure 2-5 we plot the minimum width (again, in

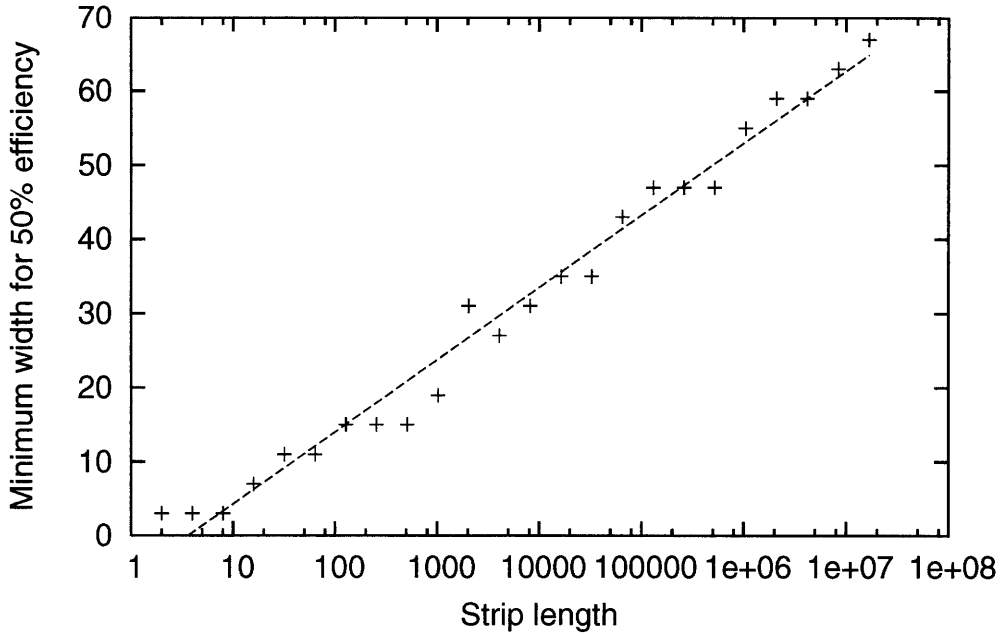

Figure 2-5: The minimum width required to reach 50% efficiency as a function of the length of the connecting strip (with a constant link failure rate of 10%).

terms of paths) required to reach a 50% efficiency as a function of the connecting strip length, assuming a link failure rate of 10%. Note that the x axis is on a logarithmic scale, from 2 to 20 million hops; the datapoints lie very close to a line, as expected.

Figure 2-7 underscores the crucial role of holes in degrading a network's resilience to failures. We "tear" a connecting strip 100000 hops long and 99 paths wide with long series of "slashes" running equidistant and parallel to its length, effectively creating holes of girth equal to twice the length of the slashes and cut size equal to the distance between the slashes (see figure 2-6). *Note that these slashes do not reduce the number of paths connecting source and destination in the absence of link failures!.* However, as predicted by our theoretical analysis, in the presence of a 10% failure rate a large number of long slashes (corresponding to a large number of holes of high girth) can seriously affect the network's connectivity. Figure 2-7 shows how efficiency of usage of the strip degrades with the length of the slashes, for 2, 10 and 50 parallel sequences of slashes respectively running at a distance of 33, 9 and 2 paths from each other.

Our theoretical analysis predicts that holes effectively "erode" connectivity in the

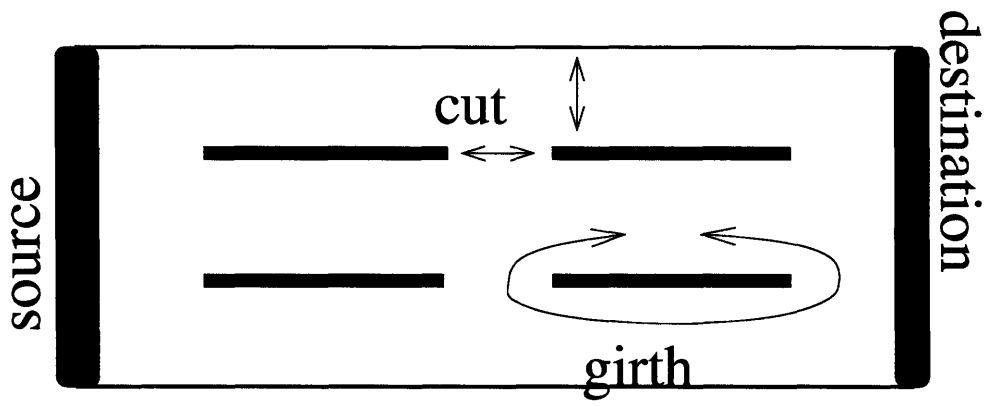

Figure 2-6: The "horizontal slashes" do not reduce the "bandwidth" between source and destination in the absence of faults - but they create holes, of girth equal to twice their length and cut of size equal to their distance, which erode connectivity in the presence of faults.

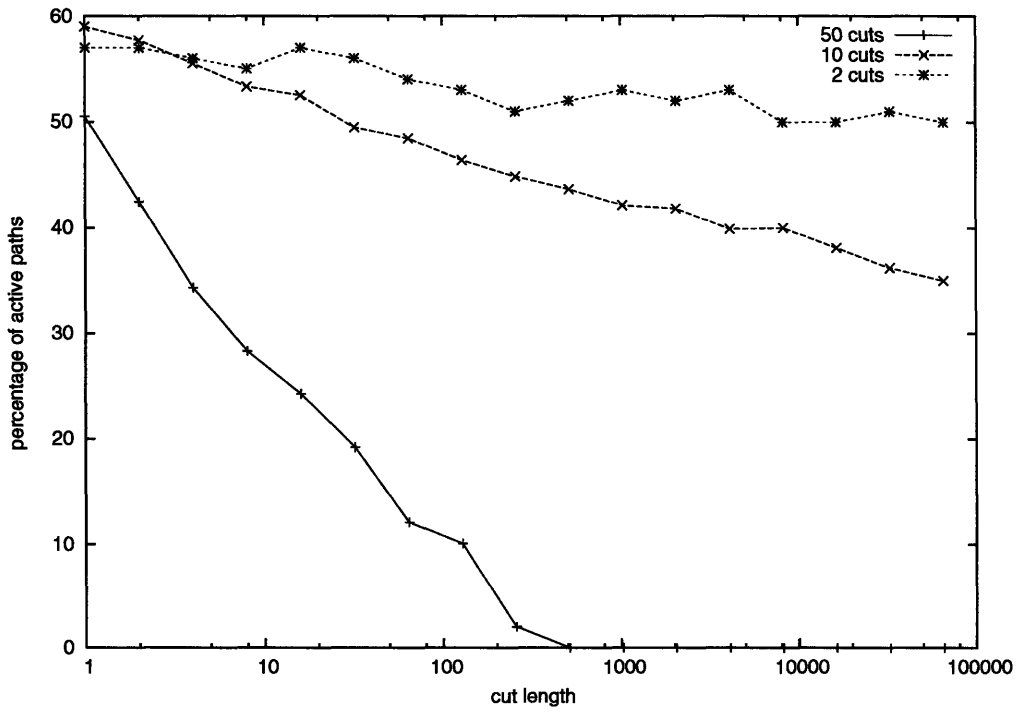

Figure 2-7: The degradation of the efficiency of usage of the strip with the length of the slashes, for different numbers of parallel sequences of slashes.

surrounding region to a distance that is logarithmic with their girth. Figure 2-8 validates our theoretical analysis, showing the minimum number of paths between slashes to guarantee a 50% efficiency, as a function of the slash length (which equals half the corresponding hole's girth). The strip connecting source and destination is 10^5 hops long, 100 paths wide and the link failure rate is 10%.

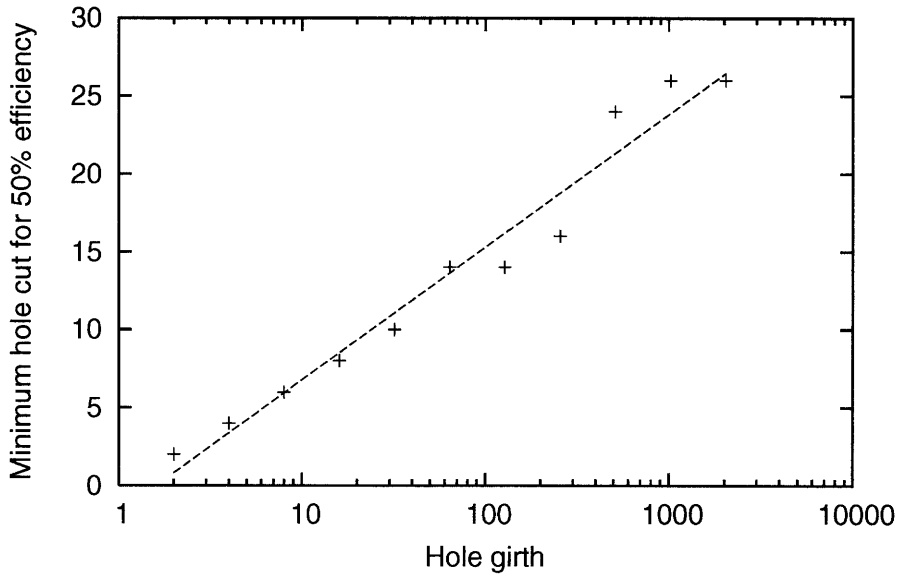

Figure 2-8: The minimum hole cut size (i.e. distance between "slashes") that guarantees 50% efficiency, as a function of the hole's girth (i.e. twice the "slash" size.)

2.6 Conclusions

At the core of this chapter lies a rigorous formalization of the intuitive concept of hole and the notion of robustly linked zones - zones that are connected by a strip (or just a "cone") of width logarithmic in their distance even when one "gives up" a strip (or just a "cone") around every hole of length logarithmic in the girth of the hole itself. Our main result is the proof that robustly linked zones can tolerate with high probability a constant failure rate and random node placement, without losing more than a fraction of the "ideal bandwidth" achievable in the absence of faults and if all nodes where optimally placed. Robust linking between zones therefore characterizes connectivity that is insensitive to the local "noise" caused by faulty or imprecisely positioned nodes or links, and instead depends essentially only on the "big picture" topology of the network.

Our result has a number of important consequences:

- While a truly 1-dimensional support is indeed not sufficient for a network to scalably tolerate failures, it is "almost" sufficient: our result shows that networks can have extremely elongated shapes (e.g. millions of nodes long and only a few dozen nodes wide - the aspect ratio of a typical fishing line!) and still tolerate well a fairly high failure rate.

- Considering connectivity between two zones that, in the absence of failures, are linked by a number of disjoint paths at least logarithmic in the distance between the two zones, rather than between two points that are connected by a single path, has two considerable advantages. First, it allows one to amortize the necessary logarithmic width of the connecting strip over a logarithmic number of fault-free paths. Second, it makes the probability that there are not at least that many fault-free paths between the two zones exponentially small in the width of the strip - whereas clearly a constant failure probability is the best one can achieve when considering simple point to point connectivity.

- The presence of holes, particularly holes with a circumference much larger than

the expected distance between faults, can seriously degrade the ability of a large network to maintain its connectivity in the presence of faults. Two networks that, in the absence of faults, have the same diameter and the same bandwidth, can exhibit radically different connectivity in the presence of faults if one of them contains a large number of large holes with thin "walls" around them. This should be kept in mind when simulating the behavior of large networks in realistic environments, which inevitably have large "holes" (bodies of water, electromagnetically shielded buildings, etc.) - restricting one's analysis to e.g. a solid disc or the unbroken surface of a sphere could lead to excessively optimistic results.

- In order to guarantee high availability of connectivity between two areas of a network, designers often try to ensure that they are connected by a virtual network formed by multiple *completely disjoint* paths. This approach might be valid when the two areas are separated by a number of hops smaller than $1/p$, where p is the probability of failure at any given hop - in other words, when the probability that any single path is entirely fault-free is relatively high. However, it is remarkably ineffective if the distance between the two areas is much larger than $1/p$. In this case the probability that any single path is entirely-fault free becomes exponentially small, and even a fairly large number of disjoint paths stand a very small chance of safeguarding the connectivity between the two areas. In terms of our analysis, a virtual network formed by many completely disjoint paths contains several extremely large and extremely "thin walled" holes. At a distance much larger than $1/p$ the only option to safeguard connectivity (without resorting to an exponentially large number of disjoint paths) is instead to keep the paths connecting two areas tightly linked, in such a way that one can "splice" together non-faulty portions of different, faulty paths to obtain a new, entirely non-faulty path.

- One important consequence of the logarithmic ratio between the width and length of a strip between regions (and around holes) that is sufficient to guar-

antee good connectivity is that, in a geometric radio network or in a nanotechnological ensemble, a larger number of smaller cells is desirable not only because it provides higher bandwidth in the absence of faults. In the presence of faults a larger number of smaller cells guarantees a constant fraction of that larger, "ideal" bandwidth with *higher* probability. This means that cell size reduction can produce by itself an improvement in the overall reliability of the resulting network even when not accompanied by an improvement in the reliability of the individual cells themselves.

Chapter 3

New results for Compact Routing

3.1 Introduction

Is it possible, without increasing the memory capacity of individual nodes (and therefore without increasing the complexity of their routing tables) to keep routing along "almost optimal paths" as a network grows in size and complexity? In particular, is it possible to efficiently route between arbitrary source and destination pairs in networks of n nodes even if each node can store less than the $\log_2 n$ bits required to store a unique identifier?

We show how it is possible to efficiently route between arbitrary pairs of nodes, *using a number of bits per node that is independent of the size of the network.* Our results hold for bounded degree trees, and for the important class of *polynomial growth networks* - any network where the number of nodes within h hops of any one point is at most polynomial in h, and thus any network that is physically implementable assuming bounded link lengths and node volume bounded away from zero. We review the problem (and the current literature) before presenting our results in greater detail.

3.1.1 Compact Routing

The problem of compact routing can be informally described as that of simultaneously minimizing the size of a network's routing tables, the routing information embedded in

the message, and the routing *stretch*, defined as the the highest ratio $\frac{d(u,v)}{d_{opt}(u,v)}$ between the length $d(u,v)$ of the route actually connecting any two nodes u and v and the length $d_{opt}(u,v)$ of the shortest path potentially connecting them. Intuitively, with larger routing tables and/or with more routing information stored in the message, a node is better equipped to choose the best edge(s) through which to forward messages for any given destination.

Different results are achievable depending on the restrictions on the assignment of node addresses and port numbers (informally, edge labels). At one end of the spectrum, addresses and port numbers are preassigned - potentially in the way that least facilitates routing. At the other end, the routing scheme designer has complete freedom to assign any address and port numbers; thus one can potentially "embed" in the address of a node a substantial amount of information on how to reach it. Results can also vary depending on whether a message can change or add to the routing information it is carrying on its way to the destination (e.g. from nodes it is passing through).

The literature on compact routing and its many variants is vast; we briefly review the results that are most relevant to our work, either because we improve upon them or because we use them as starting points for our schemes. For a more comprehensive review of compact routing, see [12].

3.1.2 Lower and upper bounds for generic graphs

Peleg and Upfal [34] show that a fairly severe tradeoff between routing table size and routing stretch is indeed inevitable if all the information required for routing must be stored in the routing tables. They prove that, to achieve stretch s, $n^{\frac{1}{\Theta(s)}}$ bits per node are both sufficient on all (undirected and unweighted) graphs of n nodes and necessary on at least some (undirected and unweighted) graphs of n nodes. This result has been extended to weighted graphs and the constants involved have subsequently been refined by a large number of papers - at the time of this writing Thorup and Zwick provide the tightest known bounds for general graphs in [42].

We remark that the lower bounds above are existential. They are generally ob-

tained on families of graphs with a number of edges equal to $n^{1+\frac{1}{\Theta(s)}}$ and girth (i.e. the size of the smallest cycle) approximately equal to s. Informally, the presence or absence of each edge must be recorded with at least one bit somewhere in the routing tables, since one cannot route through that edge if it is absent (obviously) and one must route at least some messages through that edge if it is present in order to avoid a large stretch. Then, the number of bits stored in a node's routing table must be equal to at least the average number of edges per node. On many graphs that do not sport relatively high density and high girth, one can achieve substantially better memory/stretch tradeoffs.

3.1.3 Interval routing on trees (and other graphs)

On trees of degree g and n nodes, a very simple strategy known as *interval routing* [38] allows stretch 1 (i.e. the message always takes the shortest path between source and destination) with only $O(g \log_2 n)$ bits per node. In the simplest version of interval routing, one needs only assign to nodes sequential addresses from 1 to n in depth first order; then all addresses assigned to nodes in disjoint subtrees belong to disjoint intervals. Every node stores the intervals of addresses contained in the subtree rooted at each of its children, as well as its own address - this obviously takes $(2g+1)\lceil O(\log_2 n)\rceil$ bits per node. It is easy to verify that routing along the shortest path can be achieved by checking, at each node, if the destination address coincides with the node's address; if not, forwarding the message to the child whose interval contains the destination address or, if no such child exists, to the node's parent.

This simple version of interval routing forms the basis of several more sophisticated routing schemes, including the one proposed in this chapter. The most compact routing scheme on trees to this date is that in [42]: assuming freely assigned addresses and port numbers it guarantees stretch 1 with only $\log_2 n + O(1)$ bits per node. Interval based schemes can also provide compact routing solutions on graphs other than trees. Frederickson and Janardan show in [10] how $c-$decomposable graphs admit interval based routing schemes with stretch between 2 and 3 with routing tables

of $O((\log_2(n))^2)$ bits and addresses of $O(\log_2 n)$ bits. Gavoille and Peleg present in [11] an interval based scheme that provides constant stretch with polylogarithmic storage on "almost all" graphs!

3.1.4 Compact routing on "growth restricted" graphs

Compact routing is also simpler than in the general case if the number of vertices that can be reached within a given number of hops does not grow "too rapidly". Talwar in [41] considers graphs that have *doubling dimension* k [13]: graphs where every open ball of diameter $2d$ (i.e. the set of all nodes at distance less than d from a given node) is contained in the union of at most 2^k open balls of diameter d. Talwar obtains a routing scheme with stretch $(1+\epsilon)$ that requires routing tables and message headers that are only polylogarithmic in the aspect ratio of the graph (the ratio between the maximum and the minimum distance between nodes in the graph) i.e. polylogarithmic in the size of the graph if the graph is unweighted. Gupta et al. in [14] improve this result obtaining the same stretch with routing tables of size $O(\log_2(D)\log_2(g))$ where D is the aspect ratio and g is the degree of the graph.

It is tempting to use the assumption of a small, constant doubling dimension to capture the "expansion constraints" imposed on "realistic networks" ([17]), from chips to WANs, by the physical dimensionality of the world: fast connections must be local, and the total number of nodes in a given area or volume is limited. The fact that growth constrained graphs ([21] - informally speaking graphs where doubling a ball's radius increases its volume by at most a constant factor) are a (strict) subset of graphs of constant doubling dimension only reinforces this temptation. Indeed, several peer to peer algorithms have been designed to run on networks of low doubling dimension (e.g. [35],[1],[45]).

Unfortunately, low doubling dimension is a very strong property not satisfied by many real networks that still satisfy many natural definitions of low dimensionality - e.g. embeddability in a low dimensional Euclidean lattice with adjacent nodes occupying sites that are "horizontally, vertically or diagonally" adjacent ([24]). Consider for example the $n-$comb, an unweighted graph formed by a "backbone" line of n

nodes, every node being the first of a "secondary" line of n nodes (see figure 3-1). The $n-$comb can be obviously embedded in the 2 dimensional lattice with adjacent nodes occupying adjacent sites. It can be considered a good approximation of many real world Ethernet based networks; as well as of many real world subway networks. On the other hand, its doubling dimension is $\log_2(n)$ (exactly like an hypercube of the same size!) and can therefore grow arbitrarily large with n.

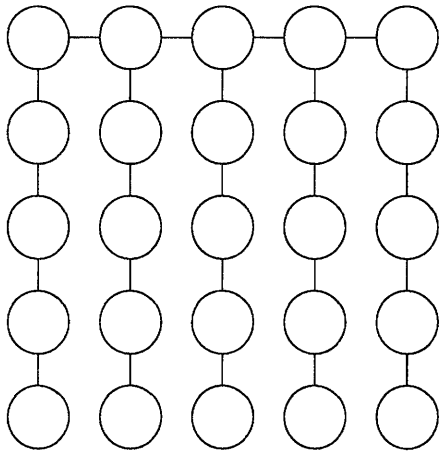

Figure 3-1: The $n-$comb (in the figure, $n = 5$) is formed by a backbone of n nodes (on top), each of them the first of "tooth" also of n nodes. The $n-$comb has a doubling dimension of at least $\log_2 n$, since it can be completely covered by a ball of diameter $3n - 3$, but no ball of diameter $\frac{3n-3}{2}$ can simultaneously cover the tips of two distinct teeth.

3.1.5 Our contribution

We study compact routing under the very restrictive condition of *constant memory* per node (independent of the size of the network). Under these conditions, an individual node's memory may not be sufficient to hold the entire message - in fact, it may not be sufficient to hold a single address! We solve this problem by adopting the well known technique of *wormhole routing* [44]: the destination address and the body of the message are split into a sequence of constant sized blocks that "snake" their way to the destination. Note that any route, even when routing a message between adjacent nodes, will necessarily include at least $\lceil \frac{\log_2 n}{m} \rceil$ nodes since the message needs to collect at least $\log_2 n$ bits of "local information" to ascertain that the destination

has indeed been reached (we call the quantity $\lceil \frac{\log_2 n}{m} \rceil$ the *orientation distance* and denote it with $d_{min}(n, m)$, or simply d_{min} if n and m are clear from the context).

Our schemes assume freedom to assign an $O(\log_2 n)$ bit address to each node (but note that, with memory $m = o(\log n)$, no individual node can *store* its entire address) and that the message header is initially equal to the destination address but can be "annotated" while in transit, although always remaining of size polylogarithmic in the size of the network.

We also show how to work around the "orientation distance limitation" by allowing the initial header to depend on the source (rather than only on the destination alone); for example adopting a scheme similar to that used for ordinary mail delivery, where one can omit the name of the destination country or even city if the mail is being sent from the same country or city.

We study two classes of networks: (unweighted) trees, and polynomial growth (unweighted, undirected) graphs. A graph has (degree k) polynomial growth if at most d^k nodes reside in any ball of diameter d. Graphs with degree k polynomial growth include all graphs that are $k-$dimensional according to the natural definition of [24]: all graphs whose nodes can each be embedded in a unique site of the $k-$dimensional Euclidean lattice Z^k in such a way that nodes adjacent on the graph are mapped into sites at unitary infinite norm distance on the lattice. Therefore, polynomial growth graphs include all graphs modeling networks that can be implemented in physical space assuming a minimum volume for each node and a maximum length for each link. Polynomial growth graphs then form a more "natural" and strictly larger class than graphs of constant doubling dimension (note that the class of polynomial growth graphs is closed under subgraph operation, unlike that of graphs with constant doubling dimension).

We describe a scheme *TreeRoute* for asynchronous wormhole compact routing on (unweighted) trees of size n that reduces the memory requirements of that of Thorup and Zwick from $O(log(n))$ to $O(1)$, at the cost of only a slightly larger stretch. More precisely, *TreeRoute* satisfies the following:

Theorem 3. *TreeRoute makes use of node addresses of* $O(\log_2 n)$ *bits, and routes*

messages formed by a header of $h = O(\log_2 n)$ bits and an arbitrarily long body of b bits. TreeRoute uses m bits of memory per node, where m can be chosen to be as low as $O(1)$, independent of n (although using larger memory will reduce the orientation distance). TreeRoute routes any message of $h + b$ bits between any two nodes at distance d through a route of $d + O(d_{min}) = d + O(\frac{\log_2 n}{m})$ nodes. Every node in the route transmits a total of $O(h + b)$ bits. If we assume that nodes communicate asynchronously by exchanging blocks of B bits, with no exchange taking more than time 1, the total time between the instant the source transmits the first block and the time the destination receives the last block is $O(d + \frac{h+b}{B})$. TreeRoute assumes that ports can be freely named (or, more precisely, that they are numbered in order of non-increasing subtree size), but can be modified to work with preassigned port numbers of at most g bits each by adding g bits to the memory of each node, and increasing the nodes involved by an additive factor d (to $2d + O(d_{min})$).

We also describe a scheme *PolyRoute* for asynchronous wormhole compact routing on (unweighted, undirected) graphs of degree k polynomial growth and n nodes, that extends and improves that of Talwar [41] and Gupta and al. [14], sacrificing a slight increase in stretch in exchange for being applicable on a much wider class of graphs and requiring considerably less memory per node (constant vs. polylogarithmic bits). More precisely, *PolyRoute* satisfies the following:

Theorem 4. *PolyRoute makes use of node addresses of $O(\log_2 n)$ bits, and routes messages formed by a header of $h = O(2^{O(k)}(\log_2 n)^2)$ bits and an arbitrarily long body of b bits. TreeRoute uses m bits of memory per node, where m can be chosen to be as low as $2^{O(k)}$, independent of n (although using larger memory will reduce the orientation distance). PolyRoute routes any message of $h + b$ bits through a route of $O(kd + d_{min}) = O(kd + \frac{\log_2 n}{m})$ nodes. Every node in the route transmits a total of $O(h + b)$ bits. If we assume that nodes communicate asynchronously by exchanging blocks of B bits, with no exchange taking more than time 1, the total time between the instant the source transmits the first block and the time the destination receives the last block is $O(kd + \frac{h+b}{B})$. PolyRoute works with arbitrarily preassigned port numbers*

(at most polynomial in the degree of the graph).

A simple analysis of our schemes would show that, even when no node can store its entire address, it is possible for each node to reconstruct and sequentially output it, involving at most $O(d_{min}) = O(\lceil \frac{\log_2 n}{m} \rceil)$ other nodes, each transmitting $O(d_{min})$ data blocks, in total time $O(d_{min})$.

The rest of the chapter is organized as follows. Section 3.2 introduces some of the basic notions and techniques used throughout the chapter, showing how to perform compact (constant memory) wormhole routing on networks with the simple line (and ring) topology. Section 3.3 describes *TreeRoute*, and proves theorem 3. Section 3.4 describes *PolyRoute* and proves theorem 4. Section 2.6 summarizes our results and discusses some open problems, before concluding with the bibliography.

3.2 Compact Routing on the Line and on the Ring

This section describes a simple compact routing scheme with constant size tables on networks with line or ring topology. The description is informal and supported by visual intuition, but a formal (and very tedious) specification can be easily obtained from it. The techniques developed here are used extensively in the next sections to route on more complex topologies.

3.2.1 Preliminaries

We begin by introducing two simple "primitives" used extensively throughout the chapter, *alignment* and *comparison*. Consider a segment of n nodes $v_1, \ldots, v_n$, with v_i adjacent to v_{i+1}, operating asynchronously, each capable of holding and transmitting to its neighbors one or two datablocks plus an additional constant number (independent of n or the size of the datablocks) of state bits. v_1 is marked with a special flag *First*; otherwise nodes are all initially in the same state.

The first primitive is *align s packets from* v_i *to* v_1. Informally, we can "stream" into the segment, from any one node v_i of it, $s \leq n$ data packets (in order!), in such a

way that in the end every node v_j (for $1 \leq j \leq s$) holds the j^{th} packet (and is notified that it does - note that in general the memory of a single node is not sufficient to hold all the bits to represent j). Figure 3-2 attempts to give a visual intuition of how this is accomplished: each node v_j (including the "input" node v_i) attempts to push every packet it receives to the lowest index "empty" node in the segment. Every node v_j carries a flag marking whether there is empty space at lower indices, and therefore whether it should push packets towards indices lower or higher than j. v_1 raises the sign "no more parking at lower indices" as soon as it receives a packet; a generic node v_j raises it as soon as it sees v_{j-1} raise it and it holds a packet. Note that when a node raises such a sign it necessarily holds its "rightful" packet. It would be easy, but extremely tedious, to formalize the above and prove that:

Lemma 4. *Aligning s packets from v_i to v_1 involves no nodes of indices above $max(i, s)$, each of which has to transmit $O(s)$ packets, and requires time $O(max(i, s))$.*

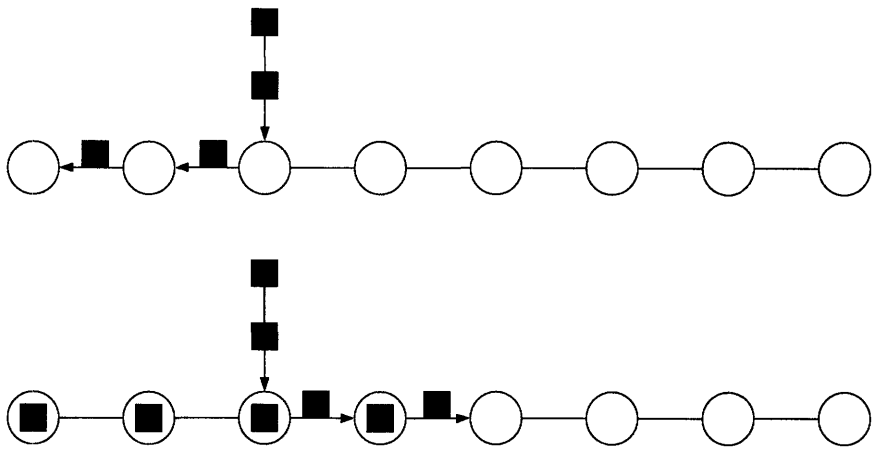

Figure 3-2: Two phases of the alignment to the leftmost node of a stream of datablocks entering from the third node from the left: first datablocks stream to the left, then, as nodes are "filled", to the right.

The second primitive is *compare x to x' over the segment $v_1, \ldots, v_n$*. Informally, given two integers x and x' of sn bits, each "distributed" over the n nodes of the segment (v_1 carrying the most significant bits), we want to compute the difference, also "distributed" over the segment; and then notify each node the sign of the result. This is easily accomplished having each node v_j, starting from v_n, compute the "local"

difference and pass the carry bit to v_{j-1}; v_1 is then able to determine the sign of the result and propagate it to the rest of the segment. Again, it would be easy, but extremely tedious, to formalize the above and prove that:

Lemma 5. *Comparing x and x' over the segment $v_1, \ldots, v_n$ can be accomplished with each node transmitting $O(1)$ bits and requires time $O(n)$.*

3.2.2 A simple scheme for routing on the line and ring

The key idea to efficiently route on the n node line with m bits/node is to divide it into segments of $\Theta(d_{min}) = \Theta(\lceil \frac{\log_2(n)}{m} \rceil)$ nodes each; and then decompose the task of routing to a node into a) routing to the correct segment of the line, and b) routing to a node within the current segment once the address has been aligned into the segment. The address of each node is correspondingly divided in two parts, both of $O(\log_2(n))$ bits: a "global" address containing the information to reach the node's segment and a "local address" containing the information to reach the node from within the segment. The $O(\log_2 n)$ bit header h of each message can then be written as $h =< g, \ell >$, where where g is (the binary representation of) the global address of the destination and ℓ its local address. We can imagine that each bit, or block of bits of the message (whether in the body or in the header) is marked to indicate whether it belongs to the body, to the global address, or to the local address.

The global address of the i^{th} segment is simply i - which can be represented with $O(\log_2 n)$ bits divided into $O(\lceil \frac{\log_2(n)}{m} \rceil)$ blocks of $\Theta(m)$ bits each - and can be "embedded" into the nodes of the segment with the j^{th} node holding the j^{th} block of bits. The message is streamed into a generic node of the line, and the global address is aligned and compared to the current segment's embedded address: if they are not equal, the message is then streamed to one of the two adjacent segments (depending on the sign of the difference), and the process is repeated until the segment of the destination is reached.

At that point the destination's global address is discarded from the header of the message, and the local address is aligned to the first node of the segment. The local

address of the j^{th} node of a segment is simply $0^{j-1}1$; therefore, once it is aligned, one bit per node, to the first node of the segment, the destination can immediately be identified as the only node carrying the single digit of the local address equal to 1, and the message body can be easily wormholed into it.

A very simple modification of the scheme above allows efficient routing on the ring topology with little extra overhead. Remembering that g is be the global address of the destination, let c be the embedded address of the current segment, and s the total number of segments in the ring. Then, if $|g - c| \leq c/2$, the direction of the shortest path to the destination is the same that would be determined with the line scheme above, i.e. routing towards lower indices if $g < c$ and towards higher ones if $g > c$; if $|g - c| > s/2$ the direction is the opposite. Making the above decision requires the involvement of no extra nodes, and no change in the header, but only a (very small) constant factor increase in the size of each node's memory and in the total time and packet transmissions required.

It would be easy to give a more formal description of the process above and prove the following:

Lemma 6. *The routing scheme above for the n node line (and the n node ring) requires headers (and addresses) of $h = O(\log_2(n))$ bits, and constant memory per node. With $O(m)$ bits of memory per node, routing a message of $h + b$ bits between a source and a destination at distance d from each other involves the $d - 1$ intervening nodes, as well as $O(d_{min})$[1] additional nodes at distance $O(d_{min})$ from the source or the destination. Each node involved transmits $O(\lceil \frac{h+b}{m} \rceil$ packets and the total time required is $O(d + d_{min})$.*

3.3 Compact Routing on Bounded Degree Trees.

This section describes and analyzes the algorithm *Treeroute*, and presents the proof of theorem 3.

[1] d_{min}, the orientation distance, is equal to $(\lceil \frac{\log_2(n)}{m} \rceil$ if we require the initial header to be source independent. We show later in this chapter how source dependent headers can reduce the orientation distance to $O(d)$; Lemma 6 continues to hold even in this case.

3.3.1 Overview

TreeRoute is based on classic interval tree routing, modified leveraging the techniques developed in section 3.2. Informally, we aggregate connected sets of nodes to distributedly simulate "virtual nodes" of larger memory that form themselves a virtual tree. The task of routing to a destination is then decomposed into first routing to the destination's virtual node, and then routing to the correct node within the virtual node. Aggregation and address assignment require some subtlety if one wants to avoid paying (poly)logarithmic stretch costs.

We further subdivide the task of routing from source to destination into two phases: *routing upwards* to a common ancestor of source and destination, and *routing downwards* from that common ancestor to the destination itself. We do not necessarily route to the *lowest* common ancestor, but rather to a "sufficiently low" common ancestor. The message header h is correspondingly written as $h =< h_{up}, h_{down} >$: h_{up} is used in the *up* phase and is dropped at the beginning of the *down* phase, where only h_{down} is used. The aggregation process is similar, but not identical, for both phases, producing from a tree T two different virtual trees $T_{up}(T)$ and $T_{down}(T)$.

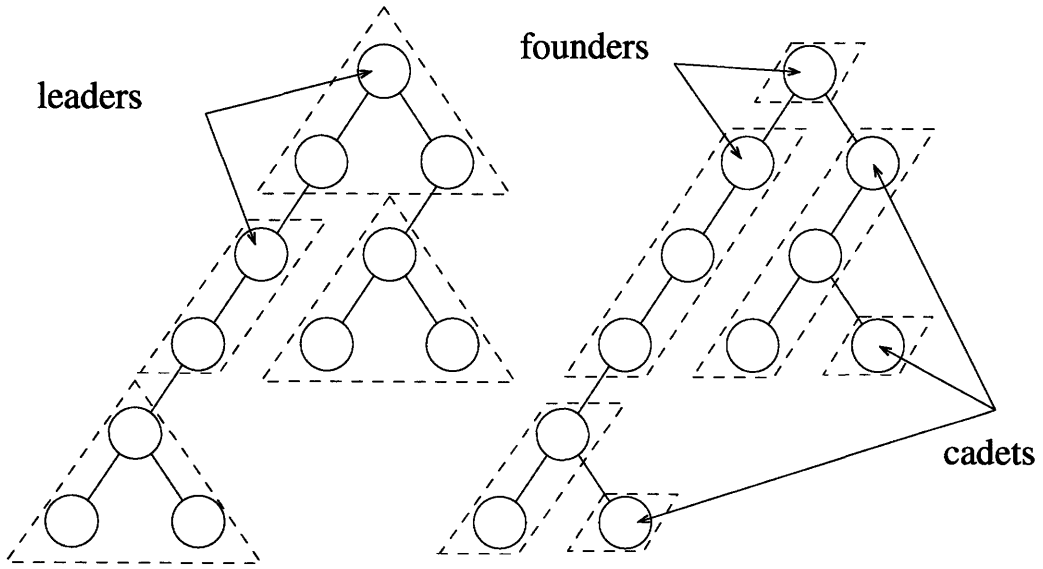

Figure 3-3: The dotted lines show the virtual nodes in the virtual trees $T_{up}(T)$ (left) and $T_{down}(T)$ (right), obtained from the same tree T.

3.3.2 Routing Upwards

The virtual tree $T_{up}(T)$ is formed iteratively as follows. The virtual root contains the root of T and all its descendants within distance $\lceil \frac{\log_2(n)}{m} \rceil$. A virtual node V is a leaf of the virtual tree if none of its nodes have children outside of V; otherwise, for any such child u, V has one virtual child formed by u and all its descendants within distance $\lceil \frac{\log_2(n)}{m} \rceil$ (see figure 3-3).

Assign to each virtual node V a unique address from 1 to $|T_{up}(T)|$ in depth first fashion, as per interval routing. The address portion h_{up} of a node is just the address assigned to its virtual node. Each virtual node V one node that is the ancestor of all other nodes in V; we call that node V's leader. Store in any node in V at distance i form its leader the $(i+1)^{th}$ block of m most significant bits of $max(V)$ and $min(V)$, where $max(V)$ and $min(V)$ are the maximum and minimum address values of the virtual nodes in the subtree rooted at V. It is easy to see that any path from the leader of V to a node in a virtual child of V has length $\lceil \frac{\log_2(n)}{m} \rceil$, and can therefore contain the full description of the $[min(V), max(V)]$.

Upwards routing then proceeds as follows: a message enters from a node in a virtual node V and is streamed upwards until it encounters the leader of that node. Either that leader is the root of T and therefore an ancestor of the destination - in which case the upward phase is completed - or the message can continue upwards until it reaches the leader of $V's$ parent U. At that point, aligning $< h_{up} >$ to $U's$ leader, one can easily establish if the destination address falls in the virtual subtree rooted at U - in which case the upward phase is completed - or whether the message should keep climbing. It would be easy, if tedious, to formalize the above process and formally prove the following:

Lemma 7. *The upward routing phase described above routes a message of $h+b$ bits from the source to a common ancestor u of source and destination that is at less than distance $2\lceil \frac{\log_2(n)}{m} \rceil = 2d_{min}$ from the lowest such ancestor. Only nodes in the direct path between the source and u are involved; each transmits $O(\lceil \frac{\log_2(h+b)}{m} \rceil)$ m bit packets. The total time required is $O(d + d_{min}) = O(d + \lceil \frac{\log_2(n)}{m} \rceil)$.*

3.3.3 Routing downwards

The downward routing phase, and particularly the construction of $T_{down}(T)$ is more complicated. The h_{down} portion of the address/header is actually formed by interleaving three fields, all of $O(\log_2 n)$ length: $h_{interval}$, h_{turn} and h_{port} We explain their role after introducing some notation.

We assume that the ports corresponding to the children of a non-leaf node are in order of non-increasing size of the corresponding subtrees: i.e. port i leads to the i^{th} largest subtree. At the end of this section we show how to remove this assumption at only a moderate additional cost.

Let the i^{th} *child* of a node v be the child at which the i^{th} largest subtree is rooted. Let the first child of v be its *heir*, and let the first child of the i^{th} heir be the $(i+1)^{th}$ heir. Let the $i-$*dynasty* founded by v, for $i \geq 0$, be the set of v and every j^{th} heir of v for $j \leq i$. Let the *cadets* of a dynasty be all the children of nodes in the dynasty that are not heirs, and let the *heir of a dynasty* be the only heir of a node in the dynasty that is not itself in the dynasty. All virtual nodes of $T_{down}(T)$ are dynasties of at most $d_{min} = \lceil \frac{\log_2 n}{m} \rceil$ nodes. The children, if any, of a virtual node are the dynasties whose founders are the cadets and heir of that virtual node. (see figure 3-3). Finally, let the *current tree* of a node v be the subtree of T rooted at its highest ancestor u of which v is a heir (then u is the the root of T, or a cadet).

Every virtual node of d nodes is assigned an address of $2 + \lceil \log_2 n \rceil$ bits, with all but the most significant $dm+2$ being 0s. Thus, one can store the address of a virtual node using only $m + \lceil 2/d \rceil \leq m + 2$ bits/node, starting from the last node of the dynasty (with the trailing 0s being stored "implicitly"). Addresses are assigned in such a way that different virtual nodes can have the same address, but a virtual node always has a lower address than that founded by its heir.

The $h_{interval}$ field is formed by a stack of addresses encoded with the same "implicit trailing 0s" notation. A message routed downwards follows, by default, the first child of every node. Whenever the header of the message is aligned to the last node of a virtual node, the address embedded in the virtual node is compared to the first

address in the $h_{interval}$ stack. If the embedded address is lower, the message passes to the virtual node's heir; otherwise, the message is wormholed into one of the dynasty's cadets using the h_{turn} field to determine which one. The h_{turn} field is a stack of strings in the form 0^i1, whose function is identical to that of the local address used in line routing (see section 3.2); when aligned to the last node of a virtual node, with 1 bit/node, the 1 of the top string marks the exact "turning point" within the dynasty. The h_{port} field is also a stack of strings, each being the binary representation of the port number to take at the next turning point. Upon taking a cadet branch, the message pops the top string from each of the three stacks $h_{interval}$, h_{turn} and h_{port}. We note in passing that the appropriate popping must and can be done as a node begins its downward phase from a node other than the root of T. The necessary information can be easily embedded in the last virtual node of $T_{up}(T)$ visited, without asymptotically increasing the memory requirements.

The key idea in choosing the size and determining the address of a virtual node V founded by a node v is the following. Let α be the address of the virtual node of which v is a heir (let $\alpha = 0$ if v is a cadet or the root of the main tree). V is chosen of the minimum length such that, by adding to α the size μ of the largest cadet subtree of V (or of V itself if larger), and rounding $\alpha + \mu$ up to the lowest number α' that can be embedded in implicit trailing 0s notation in V, the roundup error is not "too much larger" than μ. α' then becomes the address of V.

This guarantees that virtual node addresses never grow too much larger then the size of the *current* tree; and at the same time, that a virtual node holding a b bit address leads to cadet trees of size (approximately) 2^{-b} times the size of the current tree, or smaller. Thus, larger virtual nodes lead to smaller cadet trees, and the larger "backtracking", as well as the larger space taken in the stack, is amortized over a faster reduction in the size of the current tree.

More precisely, let $\{U_1, \ldots, U_p\}$ be the set of virtual nodes that the path from the root of T to any node u crosses without entering their heirs (i.e. the path either ends in U_i or branches to a cadet). It is easy to see that $\sum_{1 \leq i \leq p}(|U_i| - 1) = O(\lceil \frac{\log_2(n)}{m} \rceil) = O(d_{min})$. The left hand term is an upper bound to the number of nodes, visited by a

message to u during the downward phase, that are *not* in the shortest path to u. The inequality also shows that both $h_{interval}(u)$ and $h_{turn}(u)$ have length $O(m \cdot d_{min}) = O(\lceil \log_2(n) \rceil)$. $h_{port}(u)$ is also of length $O(\lceil \log_2(n) \rceil)$ if the i^{th} port always leads to the i^{th} largest subtree, since then the product of all port numbers in h_{port} is at most n.

TreeRoute can be easily modified to work with preassigned port numbers of at most g bits, by a simple lookup table scheme: if a child of a node can be reached through the port with the i^{th} largest label, have it store the port number of its sibling at which the i^{th} largest subtree is rooted. The corresponding entry in h_{port} is i, and it is easy to verify that the overhead is that given in theorem 1.

3.4 Compact routing on polynomial growth graphs

This section describes and analyzes *PolyRoute* and presents a proof of theorem 4. The basic idea of our scheme resembles that of [34]; we improve on it with the techniques of this and the previous sections.

3.4.1 Overview

The routing scheme for polynomial growth graphs is more complex than that for trees. This first subsection provides a high level, informal overview.

The fundamental idea is to construct a minimum spanning tree of the graph. The tree is then recursively decomposed into a Van Emde Boas hierarchy of subtrees called regions. Every region at the i^{th} level of the hierarchy has approximately the same diameter 2^i (within a factor 2), and stores information on how to reach every other "neighboring" region at the same level within distance 2^i, or on how to retrieve such information from the neighboring regions without "going too far". Note that every node belongs to *exactly one* region at each level of the hierarchy (most schemes in the literature use instead overlapping regions).

Then, a message being routed from a source node to a destination node a distance ℓ, attempts to determine, for every i starting from 0, whether the destination lies either in the same level i region of the hierarchy, or in a neighboring one (within

distance 2^i - note that two neighboring regions are not necessarily *adjacent*). The search is bound to succeed by the time the message reaches level $\lceil \log_2 \ell \rceil$. Then, the message is routed to the root of the level i destination region, and makes its way down the tree to the destination node.

There are a number of challenges that our scheme has to overcome. First, note that every node can store only a constant number of bits, but it belongs to a logarithmic number of regions (one for each level); storing a region's information without interfering with the other regions holding the same nodes at different levels requires some finesse.

Second, the same logarithmic levels vs. constant bits issue requires a careful design of the tree routing algorithm. The main difficulty is that the "heavy child" of a node may not be the same at all levels of the hierarchy. This makes generally inapplicable the standard interval tree routing techniques that, when routing "downwards" route by default to the heavy child.

Third, a region may have *polynomially less* nodes than the number of neighboring regions, potentially making it impossible to store in a given region even the identifiers of (not to mention the routing information to) the neighboring regions. This problem can be avoided in graphs with constant doubling dimension - where a careful construction of the regions themselves can lower the number of neighbors a region has to a constant at every level of the hierarchy. In the more general case of polynomial growth graphs, however, the problem is unavoidable, in the sense that there are some graphs for which at least one connected subgraph of a given size will have "too many" neighbors.

Fourth, note that regions that are neighboring according to our definition are simply "close", not necessarily adjacent - they may be separated by a number of regions of the same level that is potentially proportional to the intervening distance, even if the regions have large diameter. One has to find a way to "cross" these intervening regions without traveling, within each, a distance proportional to its diameter.

3.4.2 Multiscale naming and routing on arrays.

This subsection addresses one possible workaround to the requirement of shipping a full $\log_2 n$ address bits to the destination, presenting a technique that will be crucial for our hierarchical routing scheme for polynomial growth graphs; in particular it will allow us to have regions at different levels of the hierarchy coexist in the same (constant memory) nodes.

Intuitively, if we eliminate the requirement that the starting message header be independent of the injection point, we can adopt a scheme very much like that used in ordinary mail communication: when sending departmental email, one can omit the street address and the country of the recipient, and when sending anything but international mail, one can omit the country. If source and destination are known to be at a distance of at most ℓ, an address of $O(d \log_2 \ell)$ bits should be sufficient to identify the destination among one of the $O(\ell^d)$ possible ones.

Let us first show how this can be done on a $1-$dimensional graph. Assume the number of nodes n is a power of 2 (if not, one can simply have some nodes effectively simulate two "virtual nodes" each). Partition the graph into blocks of 2^i nodes, with $0 \leq i \leq \log_2 n$, with every pair of consecutive blocks of size 2^i forming a block of size 2^{i+1}. Every block of size i, carries, in two well specified positions, two bits: an (order i) *address* bit (w.l.o.g. the leftmost of the two) and a(n order i) *free* bit (w.l.o.g. the rightmost of the two). The address bit is the $(i+1)^{th}$ significant bit of the address for all the nodes in the block. The next bit is reserved for use at the $i+1$ block level. If the $i-$order address bit is 0, the $i-$order free bit is the $(i+1)-$address bit; if the $i-$order address bit is 1, the $i-$order free bit is the $(i+1)-$order free bit.

Thus, one can reconstruct the first $i+1$ bits of any node by moving a total of $2^i - 1$ nodes. This is clearly true for $i = 0$, since every node is a level 0 block, and carries its least significant address bit. Inductively, once the node containing $(i-1)-$ order address bit has been reached, if that bit is a 0, the node storing the $(i)-$order address bit is the one storing the $(i-1)-$order free bit in the current block, and can be reached by moving exactly 2^{i-2} nodes to the *right*. If the $(i-1)-$ order address

bit is a 1, then the $i-$ order address bit is stored in next block to the left, in the node storing that block's $(i-1)-$order free bit - which is exactly 2^{i-2} nodes to the *left* of the node storing the current block's $(i-1)-$order address bit.

It is simple, if somewhat tedious, to prove that one can stream an arbitrary message of m bits exactly 2^i nodes to the left or to the right having each of $O(2^i)$ nodes transmit only $O(m)$ bits and in total asynchronous time $O(2^i)$. One can do this by first marking a node 2^{i-1} bits to the left or right, and then having each node between the current and the marked bit produce a counter that moves to the first node without a counter to the left or right of the marked node. If the current node sends a specially tagged counter, the node that ends up carrying that counter marks the $(2^i)^{th}$ nodes to the left or right of the original one.

In fact, using a partial modification of this scheme, one can not only *address* a node, but to create a hierarchy of "superimposed" arrays, sharing the same nodes but formed of progressively larger blocks of size $2, 4, \ldots, 2^i$, where a block of size b can store up to $b^{1-\epsilon}$ bits for any $\epsilon > 0$ even if every node needs to store only $O(1/\epsilon)$ bits. Moreover, it is not difficult to implement such schemes in such a way that the $b^{(1-\epsilon)}$ bits are not "concentrated", but are spread through the block so that, by visiting $\ell > b^\epsilon$ consecutive nodes, one can retrieve $\Omega(\ell/b^\epsilon)$ of these bits.

3.4.3 Efficient tree-hashing in polynomial graphs

This subsection shows how to perform a Van Emde Boas decomposition of a minimum spanning tree of a polynomial growth graph, in such a way that every region of size n at level i can store, and efficiently retrieve, $B = \Omega(n \cdot 2^{-i\epsilon})$ bits, divided into r records indexed by *arbitrary* keys for any $B = O(b)$.

The decomposition process is very simple. Begin with a minimum spanning tree T of the graph of depth Δ. The root of T is the (only) $\lfloor \log_2 \Delta \rfloor-$root and T is the (only) $\lfloor \log_2 \Delta \rfloor-$region. Then, for $i = \lfloor \log_2 \Delta \rfloor - 1, \ldots, 0$, consider the set of all $(i+1)-$trees. In any such tree T, any node v at depth $k \cdot 2^i$ that is the root of a subtree of height at least 2^i becomes an $i-$root, and all its descendants that are not $i-$roots or descendants of an $i-$root descendant of v form the $i-$region of v. Clearly,

every $i-$region is a tree of depth at least 2^i and less than $2 \cdot 2^i$, and therefore its diameter is at least 2^i and less than $4 \cdot 2^i$.

If the whole tree held only one level of regions in its nodes, efficient hashing of records could be carried with a scheme very similar to, and in fact even simpler than, the one presented in the previous section. We first illustrate at a somewhat informal level of detail this simpler scheme before showing how to adapt it to allow multiple levels to share the same nodes.

Assume without loss of generality a single region of n nodes that spans the entire tree, of height h. Consider the *edge* circuit induced by a depth first tour of tree: denote with r the root, and with $s_0, \ldots, s_c$ its $c \leq 2^d$ its children subtrees, if any, ordered by increasing weight (i.e. s_0 is the one with the least number of nodes), and visit $r, s_0, r, \ldots, r, s_c, r$. Write the records in order of increasing key, bit by bit sequentially along this circuit, writing $1/\epsilon$ bits *per edge*, and skipping edges the second time they are visited. The bits of an edge between parent and child are stored in the child. Then, as in the previous section, consider the "backbone" of the tree, formed by the root, its heaviest child, that child's heaviest child etc., and partition it into segments, recording in each segment the range of addresses to be found in the subtrees rooted at the children of that segment that are not themselves part of the segment. Repeat the same process for those subtrees.

To check if a given record is present in the tree, and recover any information it might hold, one need only follow the backbone, checking at each segment if the key sought falls within the corresponding range. If so, perform a binary search to find out which of that segment's subtrees (if any) is responsible for a range containing the key being sought. If no range containing the desired key is found along the backbone, the key is absent from the tree.

The distance traveled from the root to the record sought is equal to the length of the shortest path between the two plus the total extra distance traveled whenever deviations from the backbone are taken. In a tree of n nodes and height h there are at most $\log_2 n = O(d \log_2 h)$ such deviations. If each segment is of length s, the total extra distance spent at each deviation is at most $ds \log_2 s$. Thus, as long as each

segment is $O(h^{\epsilon/2})$ nodes long, the total overhead is at most $O(h^{\epsilon})$.

The fundamental difficulty in extending this process to multiple layers, as mentioned in the previous subsection, lies in the fact that higher level regions can store a vanishingly small amount of information per node (or one would need nodes with at least logarithmic memory) - but in order to follow the backbone, one needs to record at each node which of its children is the heaviest, which may vary depending on the level one is considering. For example, one child v might have more descendants than another child v' within a short distance, but less within a larger distance. v would then be "heavier" at some lower level of the hierarchy, and v' at some higher level. We now show how to solve this problem.

Given a level i region ρ, consider the $m = O(2^{(1-\epsilon)i})$ level $i\epsilon$ subregions $\rho_1, \ldots, \rho_m$ comprising the main region. For each such subregion ρ_j, consider the dynasty of ρ passing through the root of ρ_j; such a dynasty either ends or leaves ρ_j at one of its leaves λ_j. Then record, as a record under a specific, unique key of ρ_j, the identifiers of λ_j and, if any, of the leaves at which the dynasty leaves the next $R(i)$ subregions it enters (we'll determine the exact value of $R(i)$ later). λ_j is identified simply by the key of the appropriate record that is (partially) stored in λ_j, and by the distance from the beginning of the record of λ_j.

Then, traveling to λ_j (in ρ_j) allows one seeking a particular key in ρ to follow the dynasty of ρ from the root, reading along the segments of that dynasty the key ranges held in the subtrees rooted at the cadets of the dynasty. If and when the interval recorded in a segment matches the corresponding key sought, one must abandon the dynasty and follow a cadet branch. The correct branch that must be taken can be found by binary testing, as follows. ρ_j stores the identifier of the "one half leaf", i.e. the leaf leading to the region $\rho_{1/2}$ such that exactly half of the regions rooted at a child of a leaf of ρ_j hold lower key ranges; ρ_j also stores which of the children of that leaf is ρ_h. Every region rooted at a child of a leaf of ρ_j then stores the identifiers of two other such regions and the corresponding leaves of ρ_j, in which one should search for a key if the range held by that region is, respectively, higher or lower than that key; choosing the two leaves and regions in such a way that, at each step, the

number of candidates potentially holding the desired key is halved. Since ρ_j has at most $2^{O(di\epsilon)}$ nodes, the search takes at most $O(di\epsilon)$ such steps - i.e. logarithmically many in the diameter of ρ_j. Once the correct cadet branch is located, search begins again along that dynasty. Note that, since the volume of ρ is $O(2^{di})$ in the path from ρ to the key sought there are at most $O(di)$ deviations from the current dynasty.

Let us now compute the total cost of the search. Consider the (shortest) path from the root of ρ to the key sought; it intersects p level $i\epsilon$ regions $\rho_{j^1}, \ldots, \rho_{j^p}$. The cost $T(\ell)$ of finding the key in ρ of depth ℓ is then equal to the sum of three components:

1. the cost of moving to a specific key in each of $\rho_{j^1}, \ldots, \rho_{j^p}$ (the one leading to the continuation of the current dynasty of ρ), $\sum_{h=1}^{p} T(\ell_h)$, where ℓ_h is the depth of the appropriate leaf in ρ_{j^h}.

2. the cost of finding those "dynasty" keys - by searching in one subregion out of every $R(i)$, starting with the first subregion of the dynasty and starting again with the first subregion of each cadet dynasty followed. Since the path takes at most $O(di)$ cadet branches, and since all regions at the same level have the same depth (within a factor 4), the cost of this step is at most $O(max_h T(\ell_h) \cdot di + \frac{1}{R(i)} \sum_{h=1}^{p} T(\ell_h))$.

3. the cost of performing a binary search for the correct cadet branch whenever one such branch is taken, which is at most $O(di \cdot di\epsilon \cdot max_h T(\rho_h))$.

We can then write (noting that the first of the two terms in the second component, $max_h T(\rho_h) \cdot di$, can be subsumed in the third component, $di \cdot di\epsilon \cdot max_h T(\ell_h)$):

$T(\ell) \leq \epsilon(di)^2 max_h T(\rho_h) + (1 + \frac{di}{R(i)}) \sum_{h=1}^{p} T(\ell_h)$

$= O((di)^2 \epsilon 2^{i\epsilon}) + (1 + \frac{di}{R(i)}) \sum_{h=1}^{p} T(\ell_h)$

with the $T(k) = O(1)$ for k equal to any small constant. It is immediate to verify that, as long as $\frac{di}{R(i)} = O(1/(i^{1+\alpha}))$ for any $\alpha > 0$, the recursion yields $T(\ell) = O(\ell)$. This is clearly achievable since it only requires each subregion to store in one of its records an amount of information that is polylogarithmic in the size of the region itself.

3.4.4 Region aggregation

As mentioned earlier, a region of size n might have $\Omega(n^{d-1})$ neighboring regions, making it impossible to store the routing information about those neighbors entirely within the region itself. Somewhat surprisingly, the polynomial growth condition allows a scheme where a relatively small number of neighboring regions can "gang up" and store together the routing information about all neighbors of every region in the group. More precisely, we first show how to aggregate regions into groups in such a way that each group stores routing information to all neighboring regions *part of groups with fewer total nodes* (breaking ties); and then how to reorganize this information so that each region can efficiently retrieve information about its neighboring regions.

Throughout the rest of this subsection, when we say that a group or region is larger than another, we mean that it either has more nodes, or that it has exactly as many nodes but a larger ID. Initially, every region forms by itself a tentative group. Since each group is formed by "accretion" of regions around this initial region, we can simply assume that this ID is the ID of its initial region.

Definition 7. *A group Γ of level i regions is* $(1-\epsilon)-$stable *(or simply stable when the quantity $(1-\epsilon)$ is clear from the context) if it contains at least $2^{i(1-\epsilon)}$ nodes for each neighboring (level i) region in a smaller group - i.e. for each such region within distance 2^i of a region in Γ.*

Aggregation proceeds in steps, and terminates only when no more unstable groups are left. At each step, consider the largest unstable group Γ, and break up every smaller group into its constituent regions. Γ then absorbs any region within distance 2^i. The aggregation process obviously terminates, since at each step one group strictly grows (if a group Γ) is unstable, it means there is a smaller group close enough for Γ to cannibalize at least one of its region); and to a size larger than any group that is broken up.

In fact, we prove that:

Lemma 8. *The aggregation process of level i regions terminates with every group's diameter being at most $O(2^i \frac{d}{\epsilon} \log_2(\frac{d}{\epsilon}))$ (where d is the dimensionality of the network), and at most $O(2^i \frac{d}{\epsilon})$ for $i = \Omega(\log_2(\frac{d}{\epsilon}))$.*

Proof. We show that no group undergoes more than $O(\frac{d}{\epsilon} \log_2(\frac{d}{\epsilon})$ consecutive growth steps before being broken up - which proves the thesis, since at each step the diameter of the group grows at most by twice the sum of the "neighbor distance", 2^i, and the diameter of the largest possible $i-$level region, less than $4 \cdot 2^i$, for a total diameter that after j steps is less than $10 \cdot j \cdot 2^i$.

Consider a generic group Γ, that evolves through a series of growth steps. Denote with Γ_1 its original region, and with Γ_j the group after $j-1$ additional growth steps. Denote with V_j the volume of Γ_j (i.e. the number of *nodes* in it), and with N_j the number of regions neighboring Γ_j that belong to groups smaller than Γ_j.

If Γ_j is $(1-\epsilon)$ unstable, then $2^{(1-\epsilon)i} \cdot N_j > V_j$. Therefore, remembering that every region at level i holds at least 2^i nodes and that Γ_{j+1} absorbs every region that contributes to N_j, we have that $V_{j+1} \geq V_j + N_j \cdot (2^i) \geq V_j(1 + 2^{i\epsilon})$. Since $V_1 \geq 2^i$, as long as Γ is not broken up, $V_j \geq 2^{i\epsilon j)}$. Since the dimensionality constraints of the graph impose that $V_j \leq (10 \cdot j \cdot 2^i)^d$, it must be that $i\epsilon j = O(di + d \log_2 j)$, which proves the thesis. □

We have shown that every group has $\Omega(2^{(1-\epsilon)i})$ distinct "information storage" nodes for each neighboring region in a smaller group. Then, each group can also reserve, for each larger, neighboring group, $\Omega(2^{(1-\epsilon)i})$ distinct information storage nodes *in some region part of a neighboring group.* If a group has no more than $2^{(1-\epsilon)i/2}$ larger neighboring groups, it can reserve $\Omega(2^{(1-\epsilon)i/2})$ nodes to each, with all these nodes clustered in a single region of a neighboring group. Otherwise, if it has more than $2^{(1-\epsilon)i/2}$ larger neighboring groups, it can reserve $\Omega(2^{(1-\epsilon)i/2})$ distinct nodes to each, with these storage nodes clustered together in a small number of regions, each region containing the information nodes reserved for $\Omega(2^{(1-\epsilon)i/2})$ distinct groups.

We show now how the information within a group can be efficiently retrieved. Note that, although the tree routing scheme presented in the previous subsection

allows efficient navigation within a region, it is not immediately applicable within a group - a collection of neighboring, but not necessarily adjacent regions. We organize the regions of a group as a (complete) tree of regions, with the initial region of the group being the root of the tree, and every region added on step j of the aggregation process being a child, at depth j, of the (depth $j-1$) region closest to it. It is immediate from the construction process that the tree has depth $O(\frac{d}{\epsilon}\log_2(\frac{d}{\epsilon}))$, and that any parent and child are at distance no larger than 2^i from each other.

We allocate information about neighbors in a group following a tree hashing scheme like the one presented in the previous subsection, dividing the address space of such neighbors in intervals. The first interval, containing information about the first $\Omega(2^{(1-\epsilon)i/2})$ neighbors is assigned to the root itself. Each of the remaining intervals contains information pertaining to a number of neighbors proportional to the size of each of the root's subtrees, and is recursively assigned to the corresponding subtree.

Each region also contains enough information to route to its parent; and to all its children if there are at most $\emptyset(2^{(1-\epsilon)i/2})$ of them. Otherwise, we can "explode" each node of the tree into a subtree of degree at most $\emptyset(2^{(1-\epsilon)i/2})$; balanced in such a way that the exploded global tree has depth no larger than $O(\frac{d}{\epsilon}\log_2(\frac{d}{\epsilon}))$. Each region can then simply store the routing information between its children, and the root of its exploded subtree. We then immediately obtain the following:

Lemma 9. *Assuming each node has memory M/ϵ, it is possible to retrieve $\Omega(M \cdot 2^{(1-\epsilon)i/2})$ bits of information for any neighbor of a level i group, starting from any region within the group, by traveling at most $O(\frac{d}{\epsilon}\log_2(\frac{d}{\epsilon}))$ paths between regions at distance 2^i or less from each other, and traveling within each of these region a path long enough to retrieve $\Omega(M \cdot 2^{(1-\epsilon)i/2})$ bits.*

The next subsection shows how to travel efficiently between neighboring regions.

3.4.5 Travel between neighboring regions

This subsection presents and analyzes an efficient, recursive scheme to travel between level i regions that are neighbors, i.e. at distance no larger than 2^i from each other,

after having checked that the destination region is indeed a neighbor of the source and having retrieved the appropriate routing information (which we show is at most polynomial in i - i.e. polylogarithmic in the diameter of the regions and the travel distance) and having routed to the node of the source closest to the destination.

A possible scheme would be to partition the shortest path between source and destination into h segments of length $\ell = O(2^i/h)$, and choose for each segment a level $\log_2(\ell)$ region intersected by that segment. Then, one could route to the destination by recursively routing to each of these lower level regions, and retrieving from each the information necessary to travel to the next such subregion (or to the closest subregion of the destination region). Unfortunately, since each level of the recursion can increase the stretch by a constant multiplicative factor greater or equal to d, this leads to a stretch that is at the very least polylogarithmic in i.

The fundamental idea to overcome this difficulty is to take advantage of the fact that each level i group can provide routing information to each of its neighbors that is polynomial in the diameter of the group, but the scheme above requires only polylogarithmic information; then, we might be able to "holographically" replicate this smaller amount of information "all over" the group, reducing the overhead of retrieving local routing information to a constant (i.e. to a product series whose terms become only vanishingly larger than 1). Another way of interpreting this strategy would be to see every level i group carrying routing information not only for its neighboring level i groups, but also for those groups of level up to $(1+\alpha)i$ for some $\alpha > 0$. Then, routing between source and destination at distance ℓ at level i requires a total travel distance at level $\frac{i}{1+\alpha}$ equal to $\ell(1+2^{-\Theta(i)})$; and thus, recursing, a total distance equal to $\ell \cdot O(1)$.

Unfortunately, while this is always possible in a (carefully partitioned) graph of doubling dimension d - where each point has at most $2^{O(d)}$ regions of level i within distance 2^i *for all* i, it is not necessarily so in the case of degree d polynomial growth graphs. Figure 3-4 illustrates the point. As a function of i (on the x axis) we plot the logarithm of average number of regions of level i within distance 2^i of any given point. In a (properly partitioned) graph of doubling dimension d, the curve never

grows beyond d. In a degree d polynomial growth graph, only the *average* of the curve must remain below level d, since integral of the curve between 0 and i represents corresponds to the logarithm of the volume of a level i region. This means that a degree d polynomial growth graph can look as a graph of dimensionality much higher than d at a given scale, as long as it has "saved dimensionality" at smaller scale. Consider for example the "cube" in figure 3-5. Even though a cube is technically 3−dimensional, our cube is no more than 2−dimensional (i.e. it can be "folded" into a 2 dimensional lattice, with adjacent nodes occupying adjacent sites) because it is built of 1−dimensional "wires"; and yet it still presents all the difficulties of routing on a 3−dimensional object, at least at the larger scale.

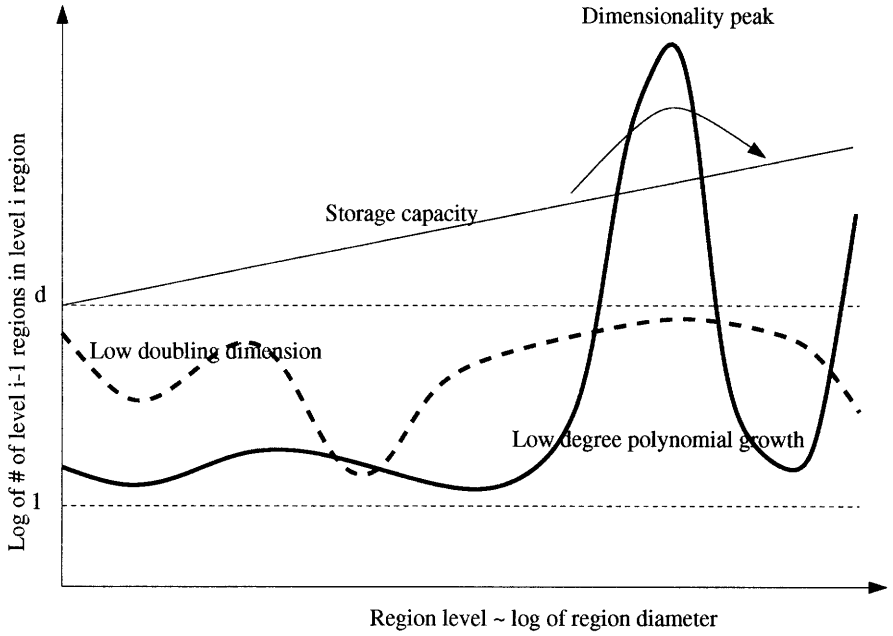

Figure 3-4: The logarithm of the number of level i-1 regions in a level i region, as a function of the region level. The area under the curve is the logarithm of the region volume. The dashed curve corresponds to a graph with doubling dimension at most d - the curve never goes above d. The thick continuous curve corresponds to a graph with degree d polynomial growth - the curve is on average below d, but it can exceed d after being lower for several levels. The thin straight line represents the logarithm of the number of neighbors a region can keep track of as a function of the region level.

Fortunately, the effects of this "dimensionality wobbling" can be kept under control. We prove that, although one might have to resort to searching information for

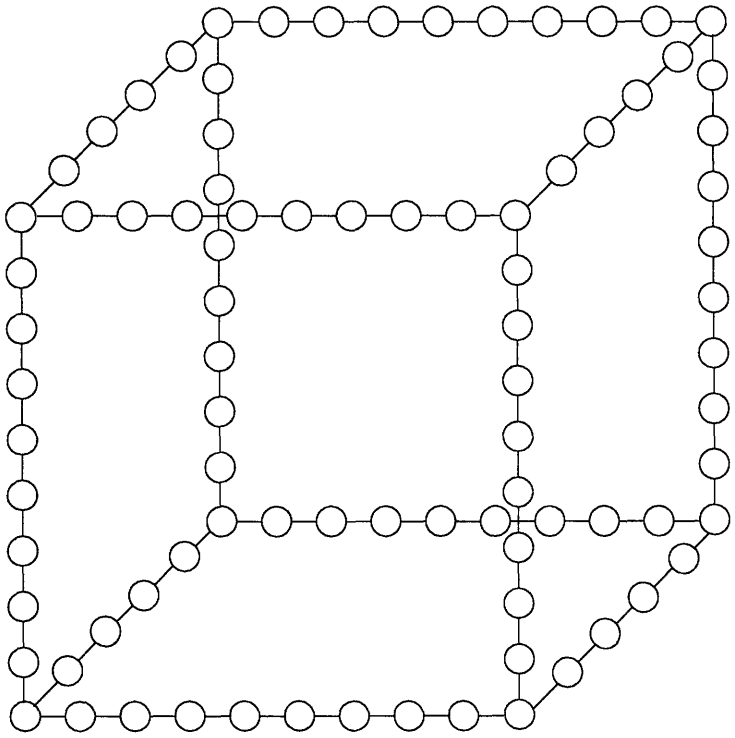

Figure 3-5: A "cube" made of 1-dimensional "wires" has degree-2 polynomial growth (it can be "folded" into two dimensions) but its doubling dimension is 3 and at the largest scale behaves like a 3-dimensional object.

travel at distance 2^i within a group of level i, this will happen only at $O(d)$ levels ; (for some $\alpha > 0$) at all remaining levels routing information can be retrieved from a group of level $(1-\alpha)i$. In any level i region, we store, routing information to all the neighboring regions for the k levels $(1+\alpha)i, \ldots, (1+\alpha)i + k - 1$, as long as there is sufficient storage space (which is certainly the case as long as there are at most $O(\frac{1}{k}2^{-i(1-\epsilon)/2})$ such regions). In case of an "overflow", i.e. if there are more than $O(\frac{1}{k}2^{-i(1-\epsilon)/2})$ regions at one or more of those levels, only routing information to those $O(\frac{1}{k}2^{-i(1-\epsilon)/2})$ with the highest volume is stored. In addition, if all the k levels are overflowing, routing information is stored for all the regions in the next higher non-overflowing level, if any.

Then, when routing at level $i(1+\alpha)$, with an ultimate destination at level $j \geq i(1+\alpha)$:

1. if there is enough information within the local level i subregion to route at level

$i(1+\alpha)+h$ for some $h \leq k$ such that $j \geq i(1+\alpha)+h$, simply route at that level.

2. Otherwise, if there is some level between j and $i(1+\alpha)$ at which overflow does not occur, route at that level, bypassing the intermediate levels.

3. Otherwise, search for routing information in the local level $i(1+\alpha)$ group.

It is immediate from figure 3-4 that any "dimensionality peak" that makes it impossible for local level i regions to store routing information to all level $(1+\alpha)i$ regions cannot have a width of more than $O(d)$ levels (or the volume constraints imposed by polynomial growth restrictions would be violated). Then, by choosing $k = \Theta(d)$, the only situation in which a message is forced to search for information at in the local group, is if the level of the final destination lies within the peak itself - in which case, there are at most $O(d)$ levels at which a message has to resort to "group searching".

3.4.6 Tallying the costs

This subsection tallies all the costs of our construction, in terms of stretch, memory, header length, time and bits transmitted to route between two points at distance ℓ with $i = \lceil \log_2 \ell \rceil$.

In terms of memory, our construction only uses $2^{\Theta(d)}$ bits/node. This quantity depends essentially on the costs of storing local routing information in level 0 regions. The space costs involved in higher levels decrease exponentially with the level.

In terms of header length, splitting a level j path into at most h subpaths means carrying alongside the message the IDs of h groups or regions; if this is done at every level, a message must carry $\Theta(hj)$ IDs of at most $\Theta(dj)$ bits each. This means that the header length is $O(d(\log_2(\ell))^2)$ if we choose h to be a small constant. However, if we choose $h = 2^{\Theta(d)}$, and thus a header length of $O(2^{\Theta(d)}(\log_2(\ell))^2)$ we can make sure that there will be at most one level of group searching, by dropping the scale of the recursion by $\Theta(d)$ levels when at the "final peak".

In terms of stretch, each level of recursion increases the length traveled by a multiplicative factor. The product of these factors for all levels at which the local group is not visited converges to $O(1)$. By choosing a sufficient header length, we can make sure that there is at most 1 level at which the local group is visited, resulting in the maximum possible stretch being multiplied by a factor d. The total distance traveled is then $O(max(\log_2 n, d\ell))$ if the header is initially formed in a source independent fashion by the full destination address, or simply $O(d\ell)$ if the header is initially formed by an appropriate prefix of the destination address suitable for local routing.

Since the underlying techniques for message streaming are those utilized for the line, it is immediate that the number of nodes involved in the transmission is proportional to the distance traveled, that the number of bits transmitted by each node is proportional to the message length (including the header), and that the time taken is proportional to the sum of the message length (including the header) and the distance traveled by the message.

3.5 Conclusions

The previous chapter improves the current lower bounds in compact routing, greatly extending the class of networks for which subpolynomial routing tables are sufficient to obtain constant stretch, while at the same time bringing down the routing table size to a constant number of bits (from at least a logarithmic or polylogarithmic table size). In particular, we showed that constant stretch is achievable with a constant number of bits/node on any *physically implementable* network, i.e. any network in which bounded linked length and node volume bounded away from zero impose a limit on the number of nodes within h hops of any given point that is polynomial in h - as well as on any bounded degree tree.

Several of our techniques are novel, and shed light on the profound difference between bounded doubling dimension [13] and the more natural notion of polynomial growth [24]. Bounded doubling dimension essentially implies bounded dimensionality at all scales; whereas a graph with polynomial growth can behave as a much higher

dimensional object at some scale, if it has "saved dimensionality" at lower scales.

A number of problems remain open. First of all, it would be desirable to refine the constants involved beyond asymptotic notation. Second, it would be interesting to close the gap between compact routing lower and upper bounds, answering the question "what makes compact routing hard"? In particular, is it true that any graph with constant degree only requires a constant number of bits/node to achieve constant stretch routing? The intuition developed in this chapter seems to indicate otherwise, and makes us conjecture that there are indeed constant degree (expander - necessarily) graphs that require more than a constant number of bits/node to achieve constant stretch.

Chapter 4

Conclusions

Can networks scale? I.e. is there a "universal node" with constant resources (memory, reliability etc.) out of which one can build networks of arbitrary complexity - without sacrificing performance (bandwidth, latency etc.) by more than a constant factor?

This thesis addressed two issues within this general problem. First, it tackled the question of long range connectivity in the presence of faults - in some sense whether one can achieve constant bandwidth efficiency without having to increase the reliability of individual nodes as the network size increases. Then, it tackled the question of truly compact routing - in some sense whether one can achieve constant latency without having to increase the memory of individual nodes as the network size increases.

In both cases, the answer is that networks can scale as long their topology satisfies some very mild conditions. Low latency can be achieved with a bounded amount of memory per node (independent of the size of the nework) in any network that is *physically implementable* - i.e. any network in which bounded linked length and node volume bounded away from zero impose a limit on the number of nodes reachable with h hops that is polynomial in h. Efficient bandwidth utilization can be achieved with bounded node reliability (independent of the size of the network) in any network that is not too "thin" and does not sport too many large "holes" (a concept with promising, independent applications): conditions that seem often easy to satisfy in practice.

It would certainly be interesting to combine these results, and extend them (e.g. looking for low latency on expander graphs, or for efficient bandwidth utilization in the presence of other error models). At least as interesting, however, would be a more concrete implementation of the techniques presented here. Achieving the "universal networking node" remains our ultimate goal, one that we believe has promising practical applications in fast growing fields of sensor/actuator networks, nanotechnology, and cellular engineering.

Bibliography

[1] I. Abraham, D. Malkhi, and O. Dobzinski. LAND: Stretch $(1+\varepsilon)$ locality aware networks for DHTs. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA04)*, 2004.

[2] B. Bollobas and F. Chung. The diameter of a cycle plus a random matching. *SIAM Journal on Discrete Mathematics*, 1:328–333, 1988.

[3] L. Booth, J. Bruck, M. Franceschetti, and R. Meester. Continuum percolation and the geometry of wireless networks. *Annals of Applied Probability*, 13(2):722–731, 2003.

[4] Olivier Dousse and Patrick Thiran. Connectivity vs capacity in dense ad hoc networks. In *Proceedings of IEEE INFOCOM*, 2004.

[5] Olivier Dousse, Patrick Thiran, and Martin Hasler. Connectivity in ad-hoc and hybrid networks. In *Proceedings of IEEE INFOCOM*, 2002.

[6] Devdatt Dubhashi, C. Johansson, Olle Haggstrom, Alessandro Panconesi, and Mauro Sozio. Irrigating ad hoc networks in constant time. In *Proceedings of ACM SPAA*, 2005.

[7] Michael B. Elowitz and Stanislas Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–338, 2000.

[8] E.N.Gilbert. Random plane networks. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):533–543, 1961.

[9] M. Franceschetti, J. Bruck, M. Cook, and R. Meester. Continuum percolation with unreliable and spread out connections. *Journal of Statistical Physics*, 2004.

[10] Greg N. Frederickson and Ravi Janardan. Space efficient message routing in c-decomposable networks. *SIAM Journal on Computing*, 19(1):164–181, 1990.

[11] Cyril Gavoille and David Peleg. The compactness of interval routing for almost all graphs. In Shay Kutten, editor, 12^{th} *International Symposium on Distributed Computing (DISC)*, volume 1499 of Lecture Notes in Computer Science, pages 161–174. Springer, 1998.

[12] Cyril Gavoille and David Peleg. Compact and localized distributed data structures. *Distributed Computing*, 2(16):111–120, 2003.

[13] Anupam Gupta, Robert Krauthgamer, and James R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *Proceedings of the symposium on Foundations of Computer Science (FOCS)*, Cambridge, MA, 2003.

[14] Anupam Gupta, Bruce Maggs, and Shuheng Zhou. On hierarchical routing in doubling metrics, 2004.

[15] P. Gupta and P. Kumar. Critical power for asymptotic connectivity in wireless networks. *Stochastic Analysis, Control, Optimization and Applications: A Volume in Honor of W.H. Fleming*, 1998.

[16] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, March 2000.

[17] Kirsten Hildrum, Robert Krauthgamer, and John Kubiatowicz. Object location in realistic networks. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2004.

[18] J. Hill and D. Culler. A wireless embedded sensor architecture for system-level optimization, 2002.

[19] T. Ho, M. Medard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong. Toward a random operation of networks. *Submitted to IEEE Transactions on Information Theory*, 2005.

[20] J. M. Kahn, R. H. Katz, and K. S. J. Pister. Next century challenges: Mobile networking for "smart dust". In *International Conference on Mobile Computing and Networking (MOBICOM)*, pages 271–278, 1999.

[21] David Karger and Matthias Ruhl. Finding nearest neighbors in growth-restricted metrics. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, 2002.

[22] Jon Kleinberg. The Small-World Phenomenon: An Algorithmic Perspective. In *Proceedings of the 32nd ACM Symposium on Theory of Computing*, 2000.

[23] Ralph Kling, Robert Adler, Jonathan Huang, Vincent Hummel, and Lama Nachman. Intel mote: using bluetooth in sensor networks. In *ACM Conference on Embedded networked sensor systems*, page 318, 2004.

[24] Robert Krauthgamer and James R. Lee. The intrinsic dimensionality of graphs. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 438–447, 2003.

[25] Bhaskar Krishnamachari, Stephen Wicker, and Ramon Bejar. Phase transition phenomena in wireless ad-hoc networks. In *Proceedings of the Symposium on Ad-Hoc Wireless Networks (GlobeCom)*, 2001.

[26] E. Lebhar and N. Schabanel. Almost optimal decentralized routing in long-range contact networks. In *Proceedings of the 31st International Colloquium on Automata, Languages and Programming*, pages 179–188, 2004.

[27] T.F. Leighton. *Introduction to parallel algorithms and architectures. Arrays, Trees, Hypercubes.* Morgan Kaufmann, San Mateo, California, 1992.

[28] L. Li, J. Halpern, and Z. Haas. Gossip-based ad hoc routing. In *Proceedings of IEEE INFOCOM*, 2002.

[29] Xiang-Yang Li, Kousha Moaveninejad, and Ophir Frieder. Regional gossip routing for wireless ad hoc networks. *MONET*, 1(1):61–77, 2005.

[30] Chip Martel and Van Nguyen. Analyzing kleinberg's (and other) small-world models. In *Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, 2004.

[31] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *Proceedings of IPTPS02*, March 2002.

[32] R. Meezer and R. Roy. *Continuum Percolation.* Cambridge University Press, 1996.

[33] William J. Ouchark, Jay A. Davis, S. Lakshmivarahan, and Sudarshan K. Dhall. Experiences with the intel hypercube. In *Proceedings of the 1986 SIGAPP Symposium on Applied Computing*, pages 2–7, 1986.

[34] David Peleg and Eli Upfal. A trade-off between space and efficiency for routing tables. *Journal of the ACM*, 36(3):510–530, 1989.

[35] C. Greg Plaxton, Rajmohan Rajaraman, and Andrea W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 311–320, 1997.

[36] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content addressable network. In *Proceedings of the ACM SIGCOMM Conference*, 2001.

[37] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218:329–339, 2001.

[38] N. Santoro and R. Khatib. Labelling and implicit routing in networks. *The Computer Journal*, 28:5–8, 1985.

[39] Yoav Sasson, David Cavin, and André Schiper. Probabilistic broadcast for flooding in wireless mobile ad hoc networks. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, 2003.

[40] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM Conference*, 2001.

[41] Kunal Talwar. Bypassing the embedding: Algorithms for low-dimensional metrics. In *Proceedings of the Symposium on Theory Of Computing (STOC)*, pages 281–290, 2004.

[42] Mikkel Thorup and Uri Zwick. Compact routing schemes. In *ACM Symposium on Parallel Algorithms and Architectures*, pages 1–10, 2001.

[43] D. Watts and S. Strogatz. Collective dynamics of small world networks. *Nature*, pages 393–440, 1998.

[44] Wikipedia. Wormhole routing.

[45] Bernard Wong, Aleksandrs Slivkins, and Emin Gun Sirer. Meridian: A lightweight network location service without virtual coordinates. In *Proceedings of ACM SIGCOMM*, 2005.

[46] Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1), January 2004.