LINE BROADCASTING

by

Jave O. Kane
B.Sc. Simon Fraser University 1989

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
August 1993

National Library of Canada

Acquisitions and Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada

Direction des acquisitions et des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

ISBN 0-315-91204-9

APPROVAL

Name: Jave O. Kane

Degree: Master of Science

Title of thesis: Line Broadcasting

Examining Committee: Dr. Thomas Shermer
Chair

Dr. Joseph G. Peters
Senior Supervisor

Dr. Arthur L. Liestman
Supervisor

Dr. Arthur M. Farley
Computer Science Department
University of Oregon
External Examiner

Date Approved: 10/8/93

PARTIAL COPYRIGHT LICENSE

Title of Thesis/Project/Extended Essay

Line Broadcasting.

Author: ______________________
(signature)

Jave O. Kane
(name)

Aug. 4, 1993
(date)

Abstract

Broadcasting is the process of transmitting information from an originating node (processor) in a network to all other nodes in the network. A *local* broadcasting scheme only allows a node to send information along single communication links to adjacent nodes, while a *line* broadcasting scheme allows nodes to use paths of several communication links to call distant nodes. Local broadcasting is not in general sufficient to allow broadcasting to be completed in $\lceil \log n \rceil$ phases, the minimum time possible for broadcasting in a network of n nodes when no node is involved in more than one communication at any given time; line broadcasting is always sufficent. An optimal line broadcasting scheme is a minimum time scheme that uses the smallest possible total number of communication links. In this thesis, we investigate line broadcasting in cycles and toruses. We give a complete characterization of optimal line broadcasting schemes in cycles, determine the exact cost of line broadcasting in cycles, and develop efficient methods for constructing optimal line broadcasting schemes in cycles and toruses. We conjecture that our torus schemes are optimal. If minimum-time broadcasting using $n - 1$ *local calls* is possible from any originator in a network, then the network is a *minimum broadcast graph.* We define *minimum line broadcast graphs*, a generalization of minimum broadcast graphs in which we can complete minimum-time broadcasting using some fixed total extra length. We find minimum line broadcast graphs for small n, and find several important families of minimum line broadcast graphs.

Acknowledgements

This research was supported by Postgraduate Scholarships awarded by the Natural Sciences and Engineering Research Council of Canada.

I would like to thank Simon Fraser University and the School of Computing Science at Simon Fraser University for the first rank education they have given me, first as an undergraduate and then as a graduate student. Special thanks go to my senior supervisor, Joseph Peters, for his unwavering support and voluminous help. I wish to thank him for his patience and cooperation during the final two years of my M.Sc. program as I worked full time on completing undergraduate Physics in preparation for my Ph.D. Thank you to my supervisor Arthur Liestman for his assistance and encouragement, and for his help while Joe Peters was on sabbatical. I would also like to thank Arthur Farley for having laid the groundwork in the fascinating area of my research, for his early help, and for agreeing to be my external examiner. Thank you to Thomas Shermer and the entire examining committee for their lively and interested examination and for quickly giving me their suggested corrections. Thank you to Kersti Jaager for her masterful handling of the administrative end of the thesis submission and the graduation procedures, and for her daily help as Graduate Secretary in Computing Science.

Finally, I would like to thank my family for their encouragement and for the way they have supported my decisions.

This thesis is dedicated to the memory of Reynard Olav Kane (1940-1988.)

Contents

Abstract iii

Acknowledgements iv

1 Introduction 1

1.1 Overview 1

1.2 Definitions 4

1.3 The Cost Model 5

1.4 Outline of the Thesis 8

2 Line Broadcast Schemes in the Cycle 10

2.1 Overview 10

2.2 Properties and Examples 11

2.3 Nestedness, Flatness, and Fullness 17

2.3.1 Nestedness 17

2.3.2 Flatness 20

2.3.3 Fullness 24

2.3.4 Sufficiency of Nestedness, Flatness and Fullness 26

2.4 Construction and Analysis of Optimal Cycle Schemes 26

2.4.1 Analysis of Optimal 2^k-cycle Schemes 26

2.4.2 Alternate Procedure for Creating Cycle Schemes 29

2.4.3 The Elimination Method for Cycles 34

2.4.4 Embedding Conjecture 34

3 Line Broadcast Schemes in the Torus 36
3.1 Overview 36
3.2 The Product Method 38
3.2.1 Description of the Product Method 38
3.2.2 Analysis of the Product Method 40
3.2.3 Other Uses of the Product Method 44
3.2.4 The Elimination Method for Toruses 47
3.2.5 Other Methods for Producing Torus Schemes 54
4 Minimum Line Broadcast Graphs 57
4.1 Overview 57
4.2 Minimum Line Broadcast Graphs 58
4.3 Variants on MLBGs 65
5 Further Work 67
5.1 Ordering of Call Lengths 67
5.2 Generalizations of the Cycle Properties 69
5.3 Lower Bounds in the Torus 69
Bibliography 70

List of Figures

1.1 Partial ordering of call length sets 7

2.1 Cycle schemes 13

2.2 Layers of the 32-cycle scheme 15

2.3 Nestedness 17

2.4 Crossed calls 18

2.5 Directions of top calls 22

2.6 Contiguous bottom schemes 23

2.7 A recursive construction of optimal cycle schemes 28

2.8 Labelled recursive constructions 30

2.9 Gray code labels and indices for 16-cycle scheme 32

3.1 5×5 and 6×6 local broadcast schemes 37

3.2 The product method for the 4×4 torus 39

3.3 The product method for the 8×8 torus 41

3.4 Line broadcast scheme from product method for the 8×8 torus . . . 42

3.5 Adapting the product method for the 7×6 torus 45

3.6 Adapting the product method for the 4×8 torus 45

3.7 Why the optimal 5×5 scheme must wrap around 48

3.8 A naive plan for the 5×5 torus 50

3.9 Tiling the 10×10 torus 55

4.1 $F(n) = 0$ for $n \leq 3$ 59

4.2 $F(4) = 1$ 59

4.3 $F(6) = 1$ 61

4.4 $F(8) \leq 3$ 62

4.5 $ES(n, L)$, $3 \leq n \leq 8$, $0 \leq L \leq 3$ 64

4.6 Schemes for a member of $\mu(8, 1)$ 64

5.1 Line call ordering 68

Chapter 1

Introduction

1.1 Overview

In broadcasting, information known by one informed processor, the *originator*, is transmitted to all other nodes (processors) in a communication network. In *local* broadcasting, an informed node may use one of its communication links to call an adjacent node during any given time unit, or *phase*. In *line* broadcasting, an informed node may call any other node; the communication path used by the call contains all of the communication links on some simple path between the two nodes, and no link is used in more than one call in a given phase.

When no node is involved in more than one communication at any given phase, the minimum number of phases in which broadcasting can be completed in a network of n nodes is $\lceil \log n \rceil$[1] phases, since there is one originator and the number of informed nodes can at most double at each phase. It is not possible, in general, to inform all nodes in a network in minimum time using local broadcasting, but Farley [9] has shown that there is a *minimum-time line broadcasting scheme* for any originator in any connected network. The question that we address in this thesis is *how much* line broadcasting is needed to complete a minimum-time broadcast from an arbitrary originator in a given graph?

[1] All logarithms in this thesis are base 2.

A broadcasting scheme for a network of n nodes requires $n-1$ calls. Furthermore, $n-1$ calls are sufficient because each node only needs to receive the information once. We can therefore refer to the *broadcast tree* of a broadcasting scheme; the originator is the root of the tree. If minimum-time broadcasting using $n-1$ *local calls* is possible from any originator in a network, then the network is a *broadcast graph.* During the last 15 years, considerable effort has been devoted to the discovery of *minimum broadcast graphs* (broadcast graphs with the fewest possible links) and to the construction of sparse broadcast graphs. (See [1] for a comprehensive study of this subject.) Unfortunately, situations in which a network can be designed to be optimal for a particular communication pattern such as broadcasting are rare. Usually, the topology of the network is determined by other factors and the task is to use the network as 'efficiently' as possible.

One approach to designing broadcast schemes in fixed networks is to use only local calls and then try to minimize time (e.g., the number of phases) or some other measure of cost. If the network uses *store-and-forward* routing, then this is the only possible approach since all communications are local. (See [12] for a recent survey of research in this area.) If the network supports some form of *circuit-switched* routing, then a second possible approach is to insist that one of the parameters, such as the number of phases, is optimized, and then try to minimize some other measure of cost. Usually, this other measure is total time to complete the broadcast taking into account other factors such as switching time at intermediate nodes and transmission rates of links. (A recent example of this approach is [17].)

In this thesis, we will take the somewhat different approach of minimizing the total amount of 'equipment' used to accomplish minimum-time broadcasting. In particular, we will minimize the total number of communication links used (i.e., the sum over all phases of the number of links used in each phase). A simple example where this approach could be useful is the distribution of electronic news on the Internet. At one time, most of the network used telephone lines and most sites were only willing to devote one modem to the net-news. The cost of distributing news depended on the amount of data and on the distance that it was sent. The elapsed time of a phone call to send a particular piece of news is essentially independent of distance travelled, so it makes sense to talk about phases of a broadcast. Assuming that network news readers want their news as quickly as possible, the cost of providing news over a telephone network depends on the total amount of equipment used. In other words, it depends on the long distance telephone charges and these are proportional to the total distance travelled. While the current technology of the Internet involving high-speed trunks and dedicated lines is much more sophisticated, the model still has validity.

We are aware of only three papers on the subject of line broadcasting: the original paper by Farley [9], an unpublished manuscript by Almstrom [5], and [11]. The main result in Farley's paper is a constructive proof that minimum-time line broadcasting is possible in any tree (and hence any connected network). Farley analyzes his construction and shows that an upper bound on minimum *total length* (total number of links used) for broadcasting in minimum time in any network with n nodes is $(n-1)\lceil\log n\rceil$. In this paper, we show that for the cycle with $n = 2^k$ nodes, the optimal total length is asymptotically 1/3 of Farley's upper bound. The bound also holds for $2^{k-1} < n < 2^k$; however we describe exact costs for such intermediate values of n. Farley notes that no tree with $n > 3$ nodes can be a minimum broadcast graph. He discusses finding trees with n nodes which allows the least maximum total length (over all originators,) and discusses *MBTs (minimum broadcast trees)*, trees in which at least two nodes can originate a minimum time local broadcast, to find an upper bound on the least maximum total length. In *path broadcasting*, calls in a given phase must be link and node disjoint. Farley describes *path broadcasting* in linear networks, which are cycles with one link removed. In this thesis we show that optimal line broadcasting in cycles is actually path broadcasting. Almstrom studied a restricted type of line broadcasting on networks that consist of a single path of processors (i.e., a one-dimensional grid). Almstrom's restriction is a constant upper bound on the length of line calls. Farley's results are also mentioned in [11] in a discussion of broadcast, *accumulation*, and *gossip*.

A problem closely related to ours, *embedding*, has received considerable attention. (See [16] for a survey.) Our problem is to find a *constrained* embedding of a broadcast tree into a graph representing the interconnections of a network. The vertices of the broadcast tree are mapped to network nodes one-to-one and edges of the broadcast tree are mapped to paths in the network. The reason our problem is a 'constrained' embedding is because the calls in any phase must use edge-disjoint paths. There is no published literature on this type of embedding problem. Furthermore, broadcast trees are subtrees of binomial trees and we are not aware of any relevant literature on embeddings of binomial trees. An embedding of a tree into a cycle can also be described as a *numbering* of the vertices of the tree. We mention a paper on numbering by Iordanskiǐ [15] in the chapter of this thesis on line broadcasting in cycles.

In this thesis, we investigate line broadcasting in networks using a model in which broadcasting must be completed in $\lceil \log n \rceil$ phases and the optimization measure (or cost) is the total number of links used during the broadcast. We first determine the exact cost of minimum-time line broadcasting in cycles, give a complete characterization of optimal line broadcasting schemes in cycles, and develop efficient methods for constructing optimal line broadcasting schemes. We then use these results to construct what we believe are optimal minimum-time line broadcasting schemes for toruses. Finally, we investigate *minimum line broadcast graphs*, graphs with n nodes in which a minimum-time broadcast can be completed from any originator, and which use the fewest possible total number of communication links for a given *extra length.*

The succeeding sections of this introduction contain definitions and a discussion of our choice of cost model. The discussion of line broadcasting in cycles is in Chapter 2, the discussion of toruses is in Chapter 3, and the discussion of MLBGs is in Chapter 4. Chapter 5 contains various shorter results of relevance to further work in line broadcasting, and suggests possible directions for further work.

1.2 Definitions

In this thesis we will model communication networks as graphs in which nodes represent processors and edges represent communication links. We will use a mixture of standard graph theoretic terminology and network terminology when discussing *broadcast trees, broadcast schemes*, and *networks*. A *broadcast tree* describes the set of calls made in a broadcast. Any broadcast tree that describes a minimum-time broadcast with n nodes, $2^{k-1} < n \leq 2^k$, has as a subgraph the directed binomial tree with 2^{k-1} nodes and is a subgraph of the directed binomial tree with 2^k nodes; the broadcast tree and the two binomial trees have the same root, the *originator.* Each level of a broadcast tree corresponds to a *phase* of a broadcast. The phase of the root of a broadcast tree is 0 and the deepest phase is $\lceil \log n \rceil$. A *broadcast scheme* is an embedding of a broadcast tree with n nodes into a network with n processors. Since the mapping of the nodes of a broadcast tree to the processors of a network by a broadcast scheme is one-to-one, we will often find it convenient to use *node* to refer to processors. Since the correspondence between edges of a broadcast tree and communication links of a network is not one-to-one in all broadcast schemes, we will use *link* when referring to physical links in a network and *edge* or *call* when referring to broadcast trees.

A broadcast scheme always uses a total of at least $n - 1$ links since each of its $n - 1$ calls uses at least one link. In line broadcasting, some of the $n - 1$ calls may be *local calls* which use one link. In this thesis, *scheme* will always mean *line broadcast scheme.* Calls which use $\lambda > 1$ links are *line calls.* A line call contributes *extra length* $\lambda - 1$ to the *total length* of a broadcast scheme. Thus, the total length of a scheme is always *total extra length* plus $n-1$. A *minimum-time broadcasting scheme* is a scheme that has $\lceil \log n \rceil$ phases.

1.3 The Cost Model

We can answer the question "how 'much' line broadcasting is needed?" in several ways. We can give the minimum total length or minimum total extra length needed. We could also find a scheme that uses the fewest number of line calls, or a scheme in which the longest line call is the shortest possible among all schemes. We could also give a set of solutions. For example, for some originator in some graph with n nodes it may be that the following two solutions work:

1. 3 calls of length 2 and $n - 4$ local calls.

2. 2 calls of length 3 and $n - 3$ local calls.

Given the above solutions, we would not consider the following a solution:

3. 3 calls of length 3 and $n-4$ local calls.

Let us imagine an example graph with 9 nodes; 8 calls are required in any broadcast scheme for the graph. Let us say that the *call length set* of a broadcast scheme is the multiset of all call lengths in the broadcast scheme. In the following discussion, We will refer to a broadcast scheme by its call length set; for the moment we are not concerned with the order in which calls are made or with the source and destination of each call. We can write the call length set of a scheme of 8 local calls as follows:

{1,1,1,1,1,1,1,1}

Let us assume that local broadcasting does not suffice, that we need at least some line calls, and that possible solutions are as follows:

1. {3,3,1,1,1,1,1,1}

2. {2,2,2,1,1,1,1,1}

By what criteria is either solution 'better?' Solution 1 uses fewer line calls than does Solution 2. The longest line call in Solution 2 (length 2) is shorter than the longest line call in Solution 1 (length 3). The total length of all line calls is 6 in both cases. The total extra length is different between the two solutions, however, which we quickly see if we split up each call into a local call and an extra length:

1. {1,1,1,1,1,1,1,1}
   {2,2}

2. {1,1,1,1,1,1,1,1}
   {1,1,1}

The total extra length in Solution 1 is 4 while in Solution 1 it is 3.

There are solutions which we would not accept, such as:

3. {3,3,3,1,1,1,1,1}

4. {3,2,2,1,1,1,1,1}

By any reasonable evaluation scheme, Solution 3 is worse than either Solution 1 or Solution 2. Solution 4 is clearly worse than Solution 2, although it is not necessarily worse or better than Solution 1. So, there is at least a partial ordering among solutions; depicting the ordering among Solutions 1-4 above as a digraph on their call length sets, in which an arrow points to a worse solution, gives figure 1.1.

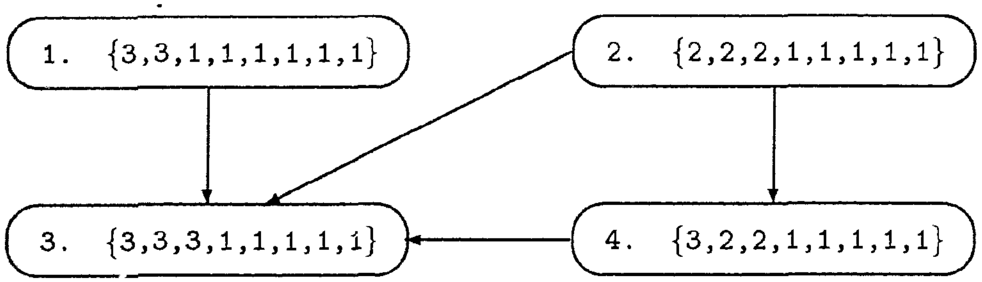


Figure 1.1: Partial ordering of call length sets

The most complete set of 'best' solutions would include every set of calls that has no predecessor in the ordering. The ordering could be described by a partial relation as follows: given two call length sets A and B, place the elements of A into a non-increasing sequence $< a_{n-1}, a_{n-2}, a_{n-3}, \dots a_3, a_2, a_1 >$ and the elements of B into a non-increasing sequence $< b_{n-1}, b_{n-2}, b_{n-3}, \dots b_3, b_2, b_1 >$. Then $A < B$ iff $b_j - a_j \geq 0, j = 1 \dots\ n-1$, and $b_j > a_j$ for some j, $j = 1 \dots\ n-1$.

We may choose a particular cost model for line broadcast schemes because of some practical consideration, such as the cost associated with a line call in an actual application. For example, if we are interested in finding minimum broadcast schemes for networks in which all local calls have more or less the same fixed cost and in which all line calls have more or less the same fixed cost, greater than the cost of a local call, we may prefer schemes which use the fewest line calls. Or, it may be the case in our network of interest that the cost of a line call increases rapidly with the extra length, in which case we may prefer schemes which use as many 'short calls' as possible.

A simple assumption that is perhaps often reasonable in practical terms is that all units of extra length have the same associated cost. Our cost model is then total extra length or total length of the line broadcast scheme; we would prefer line broadcast schemes which minimize the total extra length. As a problem in combinatorial mathematics and graph theory, this cost model is compelling and elegant, and finding lower bound in various networks may require ingenious use of results from graph embeddings, topology, combinatorics and other disciplines. This choice of cost model also allows a complete solution to the line broadcasting problem in the cycle. In our investigations, we will concern ourselves mainly with the metric total extra length.

We will say that an *optimal line broadcasting scheme* for a particular originator is a minimum-time scheme rooted at the originator with minimum total extra length. We will say that a particular graph *requires*, or that a graph and originator *require*, or that an originator in a given graph *requires* a certain total extra length to mean that any minimum broadcast scheme for the graph or for the graph and the originator or for the originator in the given graph must have at least that total extra length. We may also write simply 'extra length' to mean 'total extra length'; our meaning will always be clear from the context.

1.4 Outline of the Thesis

In Chapter 2 we discuss line broadcasting in cycles. We give a complete characterization of optimal line broadcasting schemes in cycles, determine the exact cost of line broadcasting in cycles, and develop efficient methods for constructing optimal line broadcasting schemes in cycles. The results in this chapter are the most complete of any in the thesis and are the most significant contributions made in this thesis. The proof of the optimality of our cycle schemes is very interesting because despite the fact that the cycle schemes in Figure 2.1 *look* as though they should be optimal, the proof is subtle and the order of its arguments is critical - first *nestedness*, then *flatness* and the *unused link*, and only then *contiguity* and *fullness*. The discovery of the *nestedness* property was initially unexpected and whether the properties of optimal cycle schemes generalize in a useful way to other graphs is an interesting open problem. The main open problem after the present research on cycles is whether and how general results on embeddings into linear networks simplify or relate to the present results.

Chapter 3 contains the discussion of line broadcasting in toruses. We develop efficient methods for constructing optimal line broadcasting schemes in toruses. One method is the *product method*; it produces schemes for toruses whose numbers of rows and columns are powers of 2. We conjecture that torus schemes produced by the product method are optimal. Another method is the *elimination method for toruses*; it produces schemes for certain toruses whose numbers of rows and columns are not powers of 2. A third method produces schemes for certain other toruses; this third method *tiles* a torus with another, possibly *ad hoc*, scheme. Finding a tight lower bound on total extra length required in the torus is the main open problem in toruses after the present research. It is apparently a tough problem and all the more interesting because the product method schemes seem so obviously optimal. We have suggested various ways of tackling this problem; our suggestions are in Chapter 3 and in Chapter 5.

In Chapter 4 we discuss *minimum line broadcast graphs* (MLBGs.) If minimum-time broadcasting using $n - 1$ *local calls* is possible from any originator in a network, then the network is a *minimum broadcast graph.* An MLBG is a generalization of a minimum broadcast graph in which we can complete minimum-time broadcasting using some fixed total extra length. We find MLBGs for small n, and find several important families of MLBGs. This chapter outlines problems to be solved and presents some preliminary results. One problem to be solved is to find and characterize other families of MLBGs. Another problem is to find the minimum total extra length L, for each number n of nodes in the graph, at which *any* graph with n nodes is an MLBG, given L. There are probably many new families that can be found without much work, and the search for still others may challenge the researcher to use a variety of mathematical methods. Still another problem is posed in our conjecture that a *minimum broadcast tree* (MBT) is a 'best' tree to use to minimize total extra length needed by any originator.

Chapter 5 contains various shorter results of relevance to further work in line broadcasting, and suggests possible directions for further work.

Chapter 2

Line Broadcast Schemes in the Cycle

2.1 Overview

In this chapter, we investigate line broadcasting in cycles. We determine the exact cost of minimum-time line broadcasting in cycles, give a complete characterization of optimal line broadcasting schemes in cycles, and develop efficient methods for constructing optimal line broadcasting schemes. The basis of our results is a set of three properties of broadcast schemes - *nestedness, flatness*, and *fullness*. We prove that these three properties are both necessary and sufficient for optimality.

Sufficiency is established by showing that all flat, nested, full schemes with n nodes have the same cost. The cost is found by analyzing one particular method for creating flat, nested, and full schemes for cycles with 2^k nodes, $k \geq 0$, and then using the fullness property to adapt these schemes to other cycles. The proofs of necessity and sufficiency appear in Section 3 of this chapter. The cost analysis and methods for constructing optimal schemes are in Section 4. In Section 2, we discuss several examples of optimal schemes to introduce terminology and to give an informal and intuitive overview of the proofs in Sections 3 and 4.

We mentioned in the introduction that an embedding of a tree into a graph can be described as a numbering of the vertices of the tree. In particular, an embedding of a tree into a linear network or a cycle can be viewed as one-to-one and onto mapping of the integers in $[1, n]$ into the n vertices of the tree. The total length of the embedding is the sum over each edge in the tree of the difference of the labels of the endpoints of the edge. An optimal embedding corresponds to a *minimal* or *min-sum* numbering. Iordanskiǐ [15] investigates the maximum min-sum among all trees with a fixed degree bound. He appears to use a concept similar to one which we develop in this chapter, the concept of *layers*. He also seems to have a concept similar to our concept of *contiguity* and concepts similar to consequences of our concept of *nestedness*. His results are less restrictive than ours, because they do not involve our constraint that the embeddings of calls in a given level of the tree are edge-disjoint. His results appear to be correct, but [15] contains little in the way of proofs or detailed discussion, and we have not been able to find proofs or detailed discussion of his results elsewhere.

2.2 Properties and Examples

We will refer to a scheme as a triple $S(G, T, \Phi)$. $G(V, E)$ is the target graph of the embedding. $T(V, C, u)$ is the broadcast tree, where C is the set of calls (edges) of T, and u is the root of T. Φ is the embedding. It maps each call in C to a simple path in G. It is understood that Φ maps the endpoints of calls consistently with the structure of T. When we write about a *subscheme* $S'(G', T', \Phi')$ of $S(G, T, \Phi)$, we will mean that G' is a subgraph of G, T' is a subtree of T, and Φ' is a subset of Φ. When we use the term *subtree* without modification, we will mean that paths in the subtree extend out to include leaves of T. We will sometimes refer to phases, nodes, calls, paths or subtrees of S; we will always actually be referring to T.

When P is a path in T, we will write P^{Φ} to mean the connected path induced in G by the embedding of P. P^{Φ} is not necessarily simple; it may fold on itself, and it may 'wrap around' the cycle and overlap itself, although in an optimal scheme no such overlap can occur, as we will see later on. There is a connected, undirected, path σ in the cycle which contains all the nodes on P^{Φ} and exactly all the links on P^{Φ}. σ is exactly the cycle if P^{Φ} wraps around the cycle. In an optimal scheme, σ will be simple. We will refer to σ as the *segment* of P or of P^{Φ}. Since Φ preserves the connectedness of T, any subtree T' of T must also have a corresponding segment (extending the term in the obvious manner.) When S' is a subscheme of S, we can write about the segment of S' with obvious meaning. We will say that S' is *contiguous* if its segment contains only nodes of S'.

Figure 2.1 shows several schemes on cycles. Parts (a), (b), (c), (d), and (f) of the figure show schemes for $4, 8, 16, 32$- and 64-cycles, respectively. Parts (e) and (g) show schemes for 22- and 55-cycles. Nodes are shown as black dots, and calls as arrows or short lines. Links of the cycle are not shown. In particular, the link connecting the leftmost and rightmost node in each part is not shown; that link is not used by any call in any of the schemes shown (although nothing in the definition of line schemes prohibits the use of that link.) The number under a node is the phase at which the node is informed; the originator is informed at phase 0. An arrowhead on a call, if present, shows the direction of the call. This is not really necessary since the receiver of the call is always informed later than the sender. The 4-cycle scheme in part (a) appears repeatedly in the other schemes as a subscheme, and when it does, the arrowheads are omitted to reduce clutter.

Each scheme is shown in two ways; the first shows the nodes all on one line, and the second shows one path in the scheme laid out flat and the rest of the scheme hanging below that path. The phases and positions of nodes on the cycle are the same in the two representations. The total extra length of any call shown is exactly the number of nodes which is under the call (this is best seen in the first representations.)

The schemes in Figure 2.1 are all minimum-time schemes, which can be verified by examining the phases at which nodes are informed. They are also all optimal, as we will prove in later sections. We will introduce the proof informally here by pointing out some features of the schemes which optimal schemes would intuitively seem to require. The most basic of these features are three independent properties which we call *nestedness, flatness*, and *fullness.* Nestedness and flatness are properties of the embedding; fullness is a property of the broadcast tree. The proof of optimality is based on a demonstration that the three properties are necessary and sufficient for optimality.

The first representation suggests one accounting method for total cost. Each link

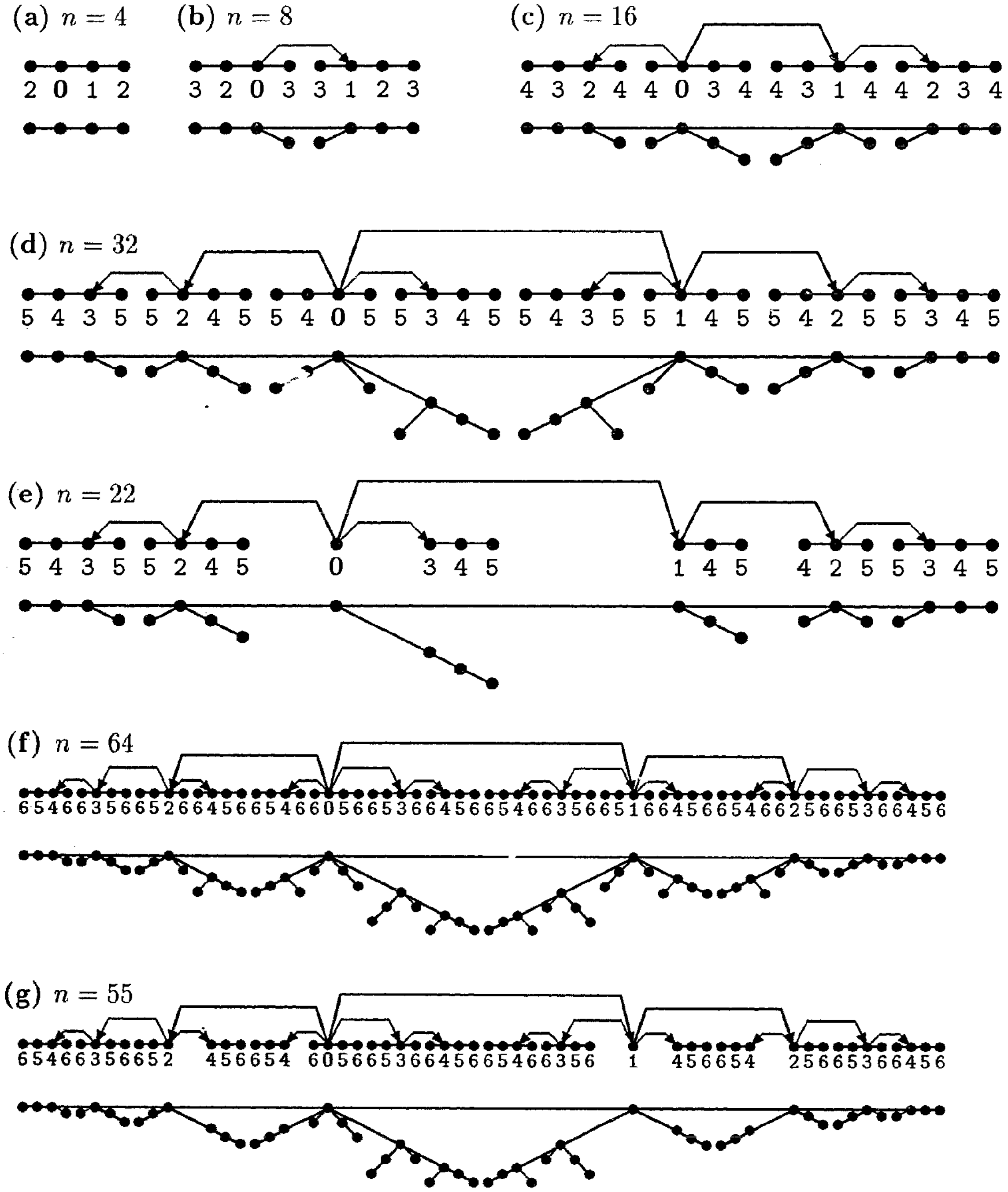
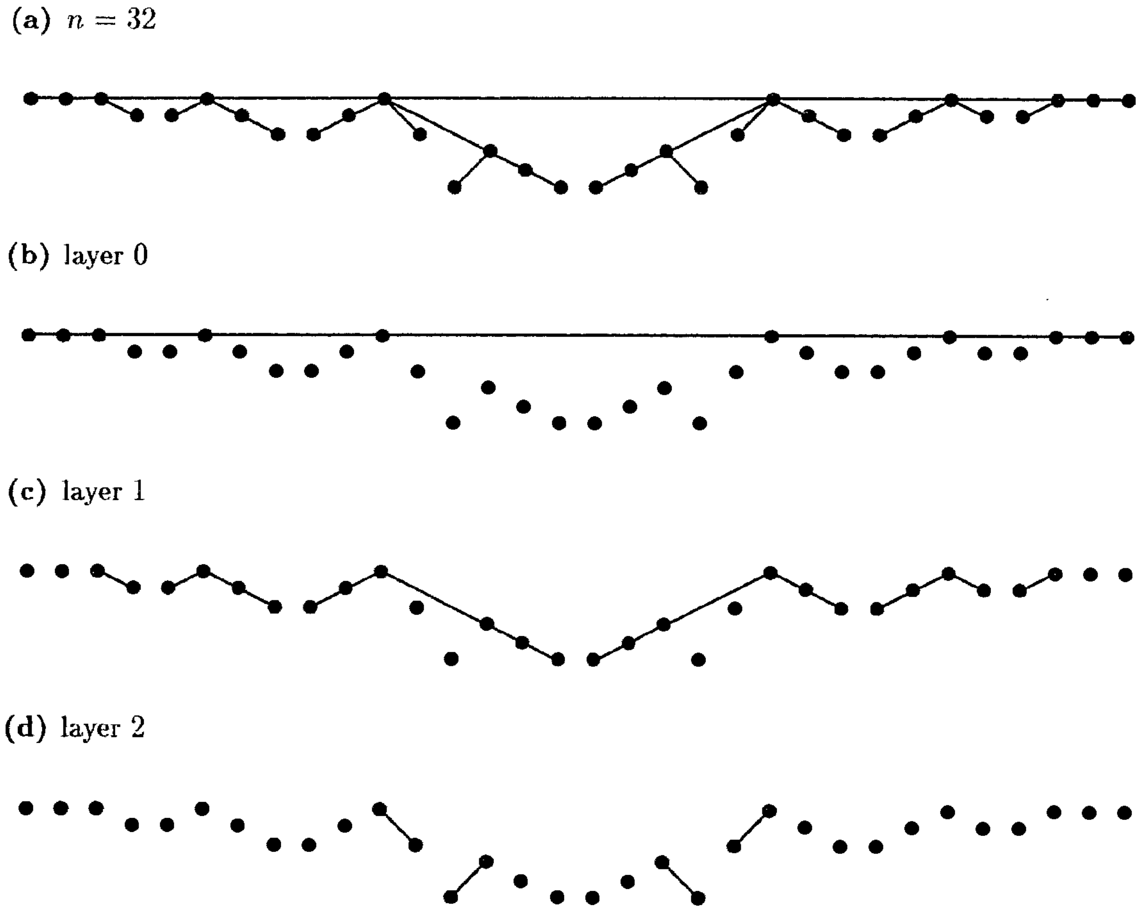


Figure 2.1: Cycle schemes

ℓ of the cycle is used by some number $\gamma(\ell)$ of calls in the scheme, where $\gamma(\ell)$ is the *congestion* of ℓ. The sum of the congestions of all links is the total length of the scheme. We will develop and use a less obvious accounting method, one which allows us to prove the necessity of fullness. From the same schemes, we note that calls are *nested*; later calls are shorter and stay *under* earlier calls, and calls never *cross*. Thus, for any pair of calls, either one of the calls is completely *under* the other, or the calls don't share any links. (We will also talk of nodes and links being *under* a call with obvious meaning.) It seems that a scheme that is not nested would have unnecessary congestion. One consequence of *nestedness* is that there must be one link of the cycle that is not used by any call. Intuitively, this *unused link* makes sense; if there is no unused link, then the broadcast tree must be embedded in such a way that it wraps around the cycle and overlaps itself. We will prove that a shorter scheme without wrap-around is always possible.

In the second representations, we see that a *top path* of calls has been laid out *flat*. It can be seen that every link of the cycle but the unused link is on the top path, and that the rest of the scheme is completely under the top path. Further examination reveals that removal of the top path leaves a set of subschemes, which we call *bottom schemes*, and that each bottom scheme has a flat top path. Removal of these top paths from bottom schemes gives sub-subschemes, and so on. (Examine the subscheme structure of the originator in parts (f) and (g) for example.) Thus, it appears that we can decompose a scheme into *layers* by repeatedly removing the flat top paths of subschemes. The top path of the entire scheme forms *layer* 0 of the scheme, the set of top paths of the subschemes that remain after layer 0 is removed constitute layer 1, and so on. Figure 2.2 shows the layers of the 32-cycle scheme. We give the name *flatness* to the property of a scheme that its layers are embedded flat into the cycle. This property is reasonable, because the less a scheme is 'folded up' on itself, the less congested it is. The layer structure turns out to be a precise property of the broadcast tree. We also notice that at each layer, the bottom schemes are *contiguous* in the cycle; paths in different bottom schemes do not cross. This property is a consequence of nestedness and allows us to show that there is a *recursive* optimality to any optimal cycle scheme.



(a) $n = 32$

(b) layer 0

(c) layer 1

(d) layer 2

Figure 2.2: Layers of the 32-cycle scheme

It certainly appears that we want the shallower layers of a scheme to be as *full* as possible (i.e., there are as many calls as possible at layer 0, then layer 1 is filled, and so on). If the shallow layers are not full, then there may be another scheme in which shallower layers *are* full, so that in place of a call in the former scheme which was under, and so, 'stretched out', p other calls, there is in the new scheme a call which is under fewer than p calls. We use the term *fullness* to refer to this third desirable situation in which shallower layers contain as many calls as possible.

Further examination of the first representation in parts (a), (b), (c), (d) and (f) reveals that an optimal scheme for a 2^{k+1}-cycle can be produced from a 2^k-cycle scheme by placing two mirror image 2^k-cycle schemes beside each other and joining their originators with a line call. Studying these parts of the figure again, we notice a second recursive method for creating the schemes; a scheme for a 2^k-cycle can be described as a modified scheme for a 2^{k-1} cycle, in which two new nodes have been added to the center of the top path, and each of those two new nodes made the root of a bottom scheme which looks exactly like an optimal 2^{k-2} cycle scheme. We use this second recursive method to find the total extra length of all optimal schemes with 2^k nodes, and also to determine the maximum possible number of calls in each layer of a scheme with n nodes, $2^{k-1} < n \leq 2^k$.

The 22-cycle scheme shown in part (e) is an adaptation of the 32-cycle scheme, with the deepest layer entirely removed and some of the calls in the next layer removed. The nodes which those calls informed in the 32-cycle are also removed from the cycle, thus shortening some calls (examine, in the first representations, how the number of nodes under some calls changes from the 32-cycle scheme to the 22-cycle scheme. These calls are deliberately drawn the same way in the two schemes to emphasize the obvious correspondence between a node, call or phase in the 22-cycle scheme with the node, call or phase directly above it in the 32-cycle scheme. Similarly, the 55-cycle scheme shown in part (g) is an adaptation of the 64-cycle scheme with some of the deepest layer calls removed. Our proof that this *elimination method* produces optimal schemes when the number of nodes is not a power of 2 is based on the *fullness* property of optimal schemes, and the analysis of the method is based on our knowledge of the maximum possible number of calls in each layer. We note that to produce, for example, an optimal 15-cycle scheme, we couldn't just continue eliminating deeper layer calls from the 32-cycle scheme, because the top path would then be too long. A minimum-time scheme for a 15-cycle uses 4 phases whereas the top path of the 32-cycle scheme in part (d) requires 5 phases.

2.3 Nestedness, Flatness, and Fullness

In the first three subsections of this section, we prove that nestedness, flatness, and fullness are necessary properties of optimal line broadcasting schemes on cycles. In the fourth subsection, we show that these three properties are sufficient for optimality.

2.3.1 Nestedness

Definition 1 (Nested) *A broadcast scheme S is nested if no call of S passes through an informed node.*

Lemma 1 *Every optimal cycle scheme is nested.*

Proof: Assume S is an optimal scheme that is not nested. Then some call c in S goes through an informed node w, as shown in Fig. 2.3, part (a). (In the figure, dashed lines indicate paths of one or more links.) Since every link between u and v is used by c, w cannot originate a call while c is being made. A cheaper scheme is possible by letting w, instead of u, inform v, as shown in Fig. 2.3, part (b). Thus, a non-nested scheme cannot be an optimal scheme. □

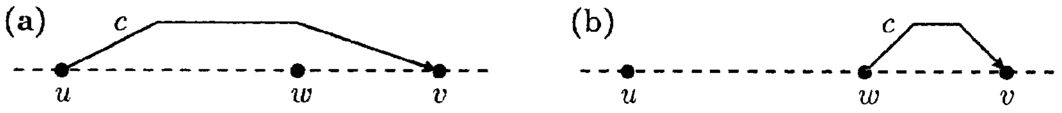


Figure 2.3: Nestedness

We note that nestedness is a property of the embedding.

Definition 2 (Under) *Let c and d be two calls in a scheme for a cycle with $c \neq d$. Then d is under c if every link which is used by d is also used by c. Also, a node v is under c if v is not the sender or receiver of c but c goes through v, and a link ℓ is under c if c uses ℓ.*

It follows immediately that a node is under a call only if the call has total length greater than 1 and, in a nested scheme, only if the node is informed after the sender and receiver of the call. In a nested scheme, the calls c and d in Definition 2 can never *cross*; either c and d use disjoint sets of links (but may share one node if it is the sender of both calls) or one call is under the other. The only other possibility is that each call uses a link which the other does not, as shown in Fig. 2.4. Since c and d share an edge, they cannot occur in the same phase. Furthermore, an endpoint of d, x in the figure, is under c, and an endpoint of c, v in the figure, is under d. If d occurs first, then x is informed before c occurs and nestedness prohibits c. Similarly, if c occurs first, then nestedness prohibits d. One consequence of this result is that optimal line broadcasting in the cycle is actually path broadcasting; calls in the same phase cannot share any links and cannot share either sender or receiver. Thus, they must be link *and* node disjoint.

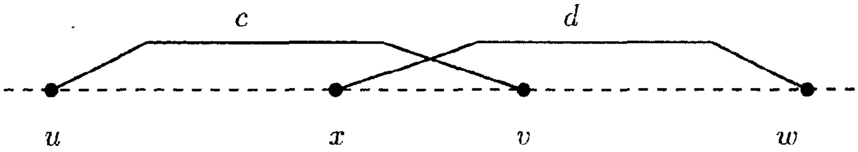


Figure 2.4: Crossed calls

Nestedness implies some other useful properties. Assume S is a nested cycle scheme. Let P_{uv} be the directed path in S from node u to node v, and let σ_{uv} be the segment of P_{uv}. Note that u is the first informed node on P_{uv}, so by nestedness, no call on P_{uv} either informs u or goes through u. It follows that one of the links incident on u is not on σ_{uv}; this establishes the following property:

Property 1 *The segment of a directed path in a nested cycle scheme is simple.*

Furthermore, each node on σ_{uv} is either u or a descendant of u in S. To see this, suppose that node x is neither u nor a descendant of u in S. Then, u cannot be the originator θ of S. Also, θ is under no call of S, by nestedness. Thus if x is on σ_{uv}, then the path from θ to x, which contains no node on P_{uv}, must contain a call c whose sender is on neither σ_{uv} nor P_{uv}, which goes *through* a node on P_{uv}, and whose receiver is on σ_{uv} but is not on P_{uv}. The receiver must be under a call d on P_{uv}; ie., c and d cross.

Assume that x has an ancestor w in S. Let P_{wx} be the path in S from w to x, and let σ_{wx} be the segment of P_{wx}. Assume that w is not a descendant of u (but allow that w may be u.) Then no node on P_{wx} is a descendant of u; It follows that no link on σ_{wx} is on σ_{uv}; σ_{wx} and σ_{uv} have in common at most their mutual endpoint u (in case x is u.)

Now arbitrarily assign the directions 'left' and 'right' in the cycle. If P is a simple path in S that begins at θ, and σ is the segment of P, then by the above discussion, θ is an endpoint of σ. Thus we can unambiguously say that σ is to the left or to the right of θ. Let Σ_L (Σ_R) be the set of all such segments which are to the left (right) of θ. Let σ_L (σ_R) be the longest (ie., in links) path in Σ_L (Σ_R). Then every link used by S is under either σ_L or σ_R. By the above discussion, θ is the one and only node shared by σ_L and σ_R, and σ_L and σ_R share no links. It follows that some link ℓ on the cycle is under neither path, so there is a link in the cycle which S does not use. However, ℓ is the only such link; otherwise, since σ_L and σ_R together form a connected path in the cycle, there would be some node which S does not inform. Thus there is *exactly* one unused link. Also, the term *between* is unambiguous in S; its reference frame is the segment of S.

Property 2 *Let S be a nested scheme on a cycle. Assume u roots a subscheme S_u of S, w roots a subscheme S_w of S, u is not in S_w, and w is not in S_u. Then the segments of S_u and S_w share no links in the cycle.*

To see this, let v be a node in S_u and x be a node in S_w, and assume that both v and x are between u and w. Let σ_{uv} (σ_{wx}) be the segment of P_{uv} (P_{wx}). By the above discussion, σ_{uv} and σ_{wx} share no links, and since each is connected, v is between u

and x, which proves that P_{uv} and P_{wx} do not cross. Note that w may be a descendant of u in S or vice-versa.

2.3.2 Flatness

Let $T(V, C, u)$ be a broadcast tree with n nodes. We will say that $c \in C$ is a *top call* or *layer 0* call of T if it is the first or second call made by u or if it is the first call made by a node which was itself informed by a top call. The node u, and any node informed by a top call of T, are *top nodes* of T. The *top path* or *layer 0* path of T is the simple path in T whose edges are the top calls of T. The set of top calls of T is *layer 0* of T. The node u is the *root* of the top path of T. If $c \in C$ is not a top call of T, then c is a *bottom call* of T. Let B be the set of subtrees we obtain by removing layer 0 of T (no nodes, just calls, are removed.) B is the set of *layer 1* or *bottom* trees of T. All the above definitions apply to any layer 1 tree T' of T, since T' is also a broadcast tree. The top path of T' is a *layer 1* path of T. Any call on T' is a *layer 1 call* of T. The set of all calls on all layer 1 paths of T is *layer 1* of T. When we remove layers 1 and 2 of T, we obtain another set of subtrees of T, the *layer 2* trees of T. Continuing, we see that each call in C belongs to exactly one layer, say p, of T, and is a layer p call. The definitions of *layer p tree*, *layer p path*, *layer p call*, and *layer p* all follow by the obvious extensions.

We will agree to say that layer p is in T only if there is at least one layer p call in T. If layer p is not in T, then there are no layer $p+1$ trees and so, no deeper layers. Thus, the layers of T are exactly all layers from 0 to $q-1$ for some q, the number of layers of T. We will say that P is a *layer path* of T if it is a layer p path of T for some p, $0 \leq p \leq q-1$. We also note that an end node of a layer path is either the root of the path or a leaf of T. So, if there are calls on a layer p path, then there is at least one leaf node of T on the path; we call the leaf a *layer p leaf*. Thus, there are layer p calls in a scheme only if there are layer p leaf nodes in the scheme.

A layer structure is a property of a broadcast tree. When we write about the layers of a scheme, we will actually be referring to the layers of its broadcast tree. We will use the term *bottom scheme* to refer to a subscheme whose broadcast tree is exactly some bottom tree of the broadcast tree.

The next property, flatness, is a property of an embedding.

Definition 3 (Flat) *Let S be a cycle scheme. Let P be a simple path in S. P is embedded flat when*

- *if a node on P makes two calls on P, then the two calls are embedded in opposite directions into the cycle.*
- *if a node on P makes a call on P and receives a call on P, then the two calls are embedded in the same direction into the cycle.*

S is flat if each layer path of S is embedded flat.

Lemma 2 *The top path of an optimal cycle scheme is embedded flat.*

Proof: Let S be the scheme. First, let u be the root of the top path; u is the originator of the S. Referring to Fig. 2.5, part (a), assume that u calls v at phase 1 and that the next call by u is at phase $t > 1$ to w, such that w is under the call from u to v. As illustrated in part (b), we can obtain a cheaper scheme by making w the originator and having w call v at phase 1 and then u at phase t (note that nestedness prohibits the sort of situation shown in part (c).)

Next, assume that u is *not* the root. Referring to Fig. 2.5, part (d), assume that v calls u at phase t, that u's first call c, at phase s, say, is to w, and that w is under the call from v to u (note that by nestedness, u cannot call *through* v.) As shown in part (e), we can obtain a cheaper scheme by having v call w at phase t and having w call u at phase s.

In each of the cheaper schemes, each of u, v, w can make the same set of calls, other than the ones shown, that it did in S; it is informed no later, and is sending or receiving calls at no other phases, than it was in S. □

Lemma 3 *If S is a nested scheme on a cycle, and S' is a bottom scheme of S, then S' is contiguous.*

Before proving the lemma we first establish a property of all schemes.

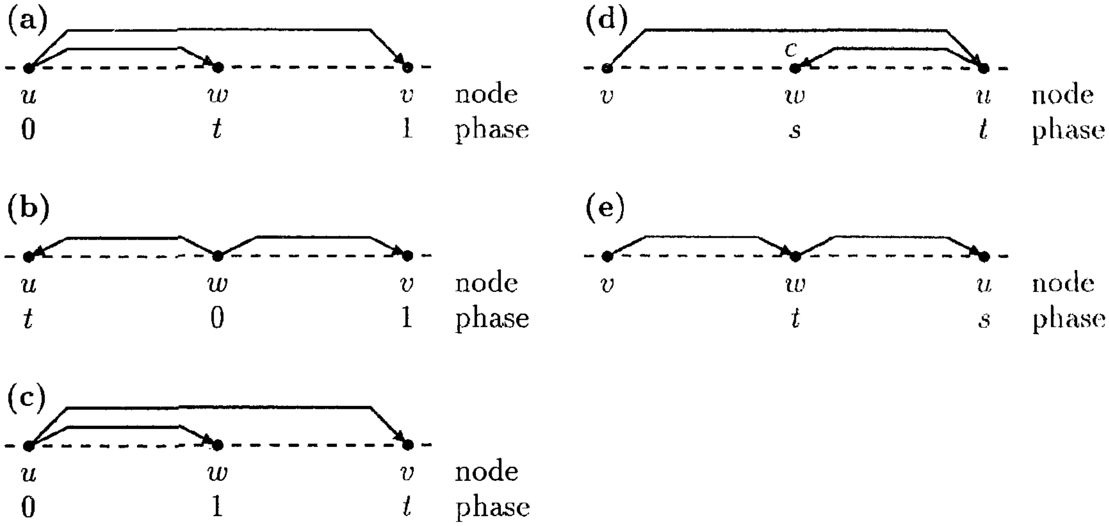


Figure 2.5: Directions of top calls

Property 3 *If S is a scheme, u is a top node of S,and S' is the bottom scheme of u in S, then u has two neighbors on the top path of S if there are calls in S'.*

To see this, first assume that u is the root of the top path. Then only u's third and later calls are bottom calls. Otherwise, u is informed by a top call and only u's second and higher calls are bottom calls. If u makes no bottom calls, then u is the only node in S' and S' contains no calls.

Proof of Lemma: By Property 3, if u is the root of $S' = S_u(G_u, T_u(V_u, C_u, u), \Phi_u)$, then u has a neighbor v and a neighbor w on the top path of S. Either u calls v and w, or v (say) calls u and u calls w. Consider the first case, in which u calls both v and w; u is the originator of S. Referring to Fig. 2.6, part (a), let $S_v(G_v, T_v(V_v, C_v, v), \Phi_v)$ be the *total scheme* of v (the largest subscheme of S which v roots,) and $S_w(G_w, T_w(V_w, C_w, w), \Phi_w)$ be the total scheme of w. Then $u \notin V_v, V_w$, and $v, w \notin V_u$. By Property 2, no node or call in S_w or S_v is closer to u than any node or call in S_u. Since the three subschemes account for all nodes and calls in S, S_u must be contiguous. The other case is similar. We define S_u and S_w as before, but now v is either the originator of S or is between the originator and u on the top path of S. In either subcase, S_v is S without the call from v to u and without the total

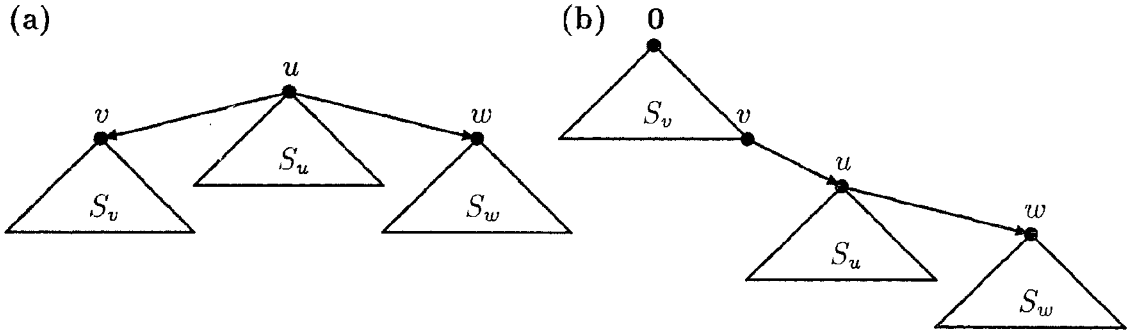


Figure 2.6: Contiguous bottom schemes

scheme of u. Fig. 2.6, part (b) shows the subcase where v is not the originator. In either subcase $u \notin V_v, V_w$, and $v, w \notin V_u$, and we can apply Property 2 as before. ▣

The previous lemma leads to two more results. Let $S(G, T, \Phi)$ be a nested scheme whose top path P is embedded flat. By Lemma 2 and Property 1, the induced path P^Φ is simple. Let u be any top node of S and let S_u be the bottom scheme of u in S. By Property 3, there are calls in S_u only if u has two top node neighbors. These neighbors, and u, are informed before any descendant of u in S_u, so by nestedness, no call in S_u goes through u or those neighbors. This discusssion establishes the following property:

Property 4 *If S is a nested scheme and the top path of S is embedded flat, then no top call of S is under any other call of S and each bottom call of S is under exactly one top call of S.*

We can now see that if $S = (G, T(V, C, u), \Phi)$ is an optimal scheme, then every bottom scheme $S'(G', T'(V', C', u'), \Phi')$ of S is itself usable as an optimal scheme on a $|V'|$-cycle. By Property 4 each bottom call of S is under exactly one top call of S. Since S uses every link of the cycle except for the unused link, its segment is $|V| - 1$ links long, regardless of any other detail of the scheme, including any details of the bottom schemes. But the total cost of S *includes* the total cost of each bottom scheme. Now, each bottom scheme is contiguous, so no detail of one bottom scheme affects the cost of any other bottom scheme. Therefore, each bottom scheme must 'look like' an optimal cycle scheme; in particular, its top path must be embedded flat. Repeating the argument recursively, we see that S will have to satisfy Definition 3. So, we have proved the following lemma.

Lemma 4 *Every optimal cycle scheme is flat.*

Flatness and nestedness are completely independent properties, despite the fact that proof of Lemma 2 involves the nestedness of an optimal scheme. A top path can fold on itself (ie., not be embedded flat) and yet still obey nesting. On the other hand, two bottom schemes in a flat scheme can cross and thus violate nesting.

We can use any broadcast tree T to create a flat, nested cycle scheme $S(G, T, \Phi)$. First, we lay out the top path P of T flat and without overlap into a cycle which has as many nodes as there are top nodes. Next, we replace each node by a contiguous embedding of the bottom tree rooted by that node; we thus creates the bottom schemes of S. In the process, we stretch out the initial induced path P^Φ by inserting nodes in the cycle to the left and right of nodes on P^Φ. The details of the bottom schemes do not affect the total number of links on the segment of the stretched P^Φ. To determine the details of the bottom schemes, we simply repeat the entire procedure recursively for each bottom tree, stopping the recursion when we reach subtrees which contain no calls.

2.3.3 Fullness

Let $S(G, T, \Phi)$ be a flat, nested cycle scheme. By Property 4, each bottom call of S is under exactly one top call of S. Now, remove the top path of S. The bottom (layer 1) schemes are contiguous, by Lemma 3, so no call in one bottom scheme is under any call in any other bottom scheme. Also, no layer 1 call is under any other call in the bottom scheme of which it is a top call. Each bottom scheme is flat and nested. If follows that each bottom call of each layer 2 scheme is under exactly one layer 0 call, one layer 1 call, and no other calls. Continuing, we can see that a layer p call is under exactly one layer r call, $0 \leq r \leq p-1$ and under no other calls. Let $q-1$ be the deepest layer of S. A layer $p-1$ call has no calls under it since such calls would be in a deeper layer than $q-1$. It follows that layer $q-1$ calls are all local calls.

These observations lead to an accounting method for total extra length which immediately allows us to prove that an optimal cycle scheme has as many calls as possible at lower layers. Consider *removing* a layer $q-1$ leaf u. More precisely, remove from T the call c which informs u, and then remove u from the cycle by merging the links incident on u. We know that c was a local call; removing it from T saves no extra length. We know that u was under $q-1$ calls in T, all of which went through u and which had extra length ≥ 1. When we remove u from the cycle, we shorten each of those $q-1$ calls by one link. Thus, we have a new cycle scheme with $n-1$ nodes and with exactly $q-1$ less extra length. If the original scheme was flat and nested, then so too is the new scheme; no call that obeyed nesting in the original scheme now goes through any new, let alone informed, node, and the only change we have made, if any, to a layer path which was embedded flat in the original scheme is to shorten it by one call.

Since the new scheme is flat and nested, we can repeat the procedure, performing it once for each call in the scheme. As we do so, we charge each unit of extra length exactly once, to one call in the scheme. Thus, we have a method for finding the total extra length of any flat, nested cycle scheme.

As we have seen, any broadcast tree can produce a flat, nested scheme. Let the *capacity* $\mathcal{M}(n,p)$ be the maximum size of layer p in some set of broadcast trees on n nodes. It follows from the preceding discussion that the least expensive schemes in the set are the ones in which layers are filled in order, with a layer partly full only if there are no higher layer calls.

Definition 4 (Full) *Assume that we have a capacity $\mathcal{M}(n,p)$ which is the maximum size of each layer p in a set of broadcast trees with n nodes and q layers, $0,\ldots,q-1$. A scheme in the set is full with respect to M if layer r of the scheme, $0 \leq r < q-1$ has $\mathcal{M}(n,r)$ calls and layer $q-1$ has $\leq \mathcal{M}(n,q-1)$ calls.*

We have proved the following lemma.

Lemma 5 *Every optimal cycle scheme is full.*

Note that the *trivial* capacity allows $n-1$ calls in layer 0. Any local broadcast scheme for the cycle is full with respect to the trivial capacity; the scheme has just a top path and no bottom calls. So, optimality requires extra length only if there is a nontrivial capacity.

2.3.4 Sufficiency of Nestedness, Flatness and Fullness

Assume that the n-cycle scheme S has q layers numbered $0,\ldots,q-1$, and $\omega(n)$ calls in layer $q-1$. If S is flat, nested and full, then it follows that the total extra length of S is just

$$\epsilon(n) \quad = \quad \sum_{p=0}^{q-2} \mathcal{M}(n,p) \cdot p + \omega(n) \cdot (q-1) \tag{2.1}$$

Since *all* nested, flat and full n-cycle schemes have this same cost, they are all optimal; we have proved that nestedness, flatness and fullness are sufficient conditions for optimality. Therefore we have proved the following theorem.

Theorem 1 *A cycle scheme is optimal iff it is flat, nested and full.*

2.4 Construction and Analysis of Optimal Cycle Schemes

In this section, we completely analyze optimal cycle schemes with $n = 2^k$ nodes, $k = 0, 1, 2, \ldots$, in the process obtaining enough information to describe the cost of optimal cycle schemes for all other values of n. We discuss two recursive procedures for constructing optimal 2^k-cycle schemes, and how to use any optimal scheme for a 2^k-cycle to create an optimal scheme for an n-cycle, $2^k - 1 < n < 2^k$.

2.4.1 Analysis of Optimal 2^k-cycle Schemes

Because there is only one minimum time broadcast tree with 2^k nodes (the directed binomial tree with 2^k nodes,) to find the total extra length of an optimal cycle scheme with 2^k nodes, we could simply determine the number of layers $\Lambda(k)$, and the size $M(k,p)$, of each layer p, $p = 0, \ldots, \Lambda(k) - 1$, in the broadcast tree with 2^k nodes. Since, as we saw in Section 2.3.3, we can charge extra length p to a layer p call in a flat nested scheme, the total extra length is just $\sum_{p=0}^{\Lambda(k)-1} M(k,p) \cdot p$.

As our notation in the previous paragraph implied, $M(k,p)$ is in fact the capacity of the set of minimum time broadcast trees with k phases. This follows from the fact that a broadcast tree with k phases is simply the broadcast tree with 2^k nodes but with some subtrees removed. If follows that an optimal cycle scheme on n nodes, $2^{k-1} < n \leq 2^k$, is full with respect to the capacity $M(k,p)$ described in the previous paragraph. We can write $M(k,p)$ instead of $\mathcal{M}(n,p)$ since the two are the same if $2^{k-1} < n \leq 2^k$. We can determine $M(k,p)$ by a recursive description of broadcast trees on 2^k nodes; the description leads to a recurrence relation which we will solve. We will not attempt a closed form formula for $\epsilon(n)$, the total extra length of an optimal cycle on n nodes, $n > 0$. However, once we have $M(k,p)$, $\epsilon(n)$ is easily found for any n via Equation (2.1), using $M(k,p)$ in place of $\mathcal{M}(n,p)$. A complete analysis of extra length is also easier for cycles with 2^k nodes, using the same recursive description just mentioned.

We now determine $\Lambda(k)$, $\epsilon(2^k)$ and $M(k,p)$ for $k \geq 0$. The trivial cases are $k = 0$ and $k = 1$, since optimal schemes in these cases are just local broadcast schemes with 1 layer. We determine $\Lambda(k)$ in general as follows: the originator makes its first two calls on layer 0, its next two on layer 1, and so on. Since the originator makes a call in each phase, there are at least $\lceil k/2 \rceil$ layers. Every call in the broadcast tree has a layer found by removing layers; removing each layer we descend 2 phases further into the tree, to a maximum of k phases. Thus, there are no more than $\lceil k/2 \rceil$ layers, ie. there are exactly $\lceil k/2 \rceil$ layers.

At this point, we take a shortcut to determine $\epsilon(2^k)$; instead of first finding $M(k,p)$, we directly use the same recursive construction which we will use to find $M(k,p)$. For brevity, call an optimal scheme on an n-cycle an *n-scheme*. It is easy to describe the bottom schemes of a 2^k-scheme. We will say that the originator of a cycle scheme is a *1-node*. We will say that any other top node of a cycle scheme is a *j-node* if it is informed at phase j. The originator calls the other 1-node. Each 1-node then calls a 2-node, and next, begins a 2^{k-2}-scheme. The k-nodes make no calls (they are the ends of the top path.)

In an optimal scheme, each bottom scheme is flat, nested and full. It follows that, given an optimal 2^{k-1}-scheme, to construct an optimal 2^k-scheme we simply lengthen the top path of the 2^{k-1}-scheme by adding two new nodes to the center of the top path, and then make each of the two new nodes the root of a 2^{k-2}-scheme, such that each of those 2^{k-2}-schemes is contiguous. We demonstrate this procedure for $k = 4$ in Figure 2.7. Following the labelling of the dashed boxes, we see that boxes 1 and 2 together contribute extra length $\epsilon(2^{k-1})$, while boxes 3 and 4 each contribute extra length $\epsilon(2^{k-2}) + 2^{k-2} - 1$. The addend $2^{k-2} - 1$ is due to the number of new calls under the top path. Therefore,

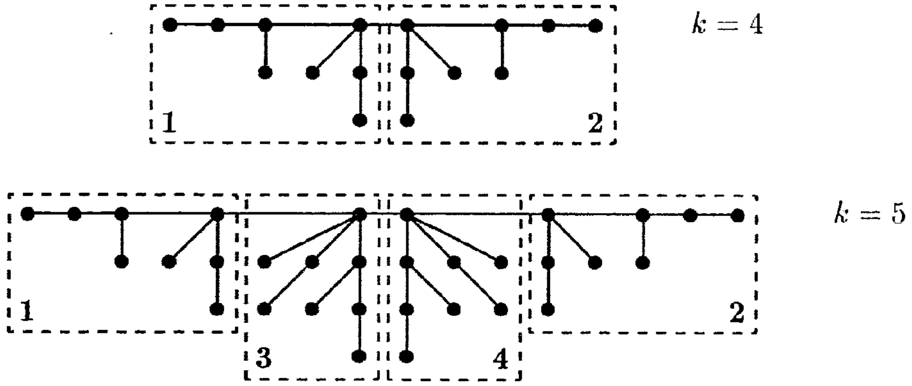


Figure 2.7: A recursive construction of optimal cycle schemes

$$\begin{aligned}\epsilon(2^k) &= \epsilon(2^{k-1}) + 2 \cdot \epsilon(2^{k-2}) + 2^{k-1} - 2, k > 1\\ \epsilon(2^0) &= 0\\ \epsilon(2^1) &= 0\end{aligned}$$

The solution to the recurrence relation is

$$\epsilon(2^k) = \frac{1}{9}[2^k(3k-8) - (-1)^k] + 1, \tag{2.2}$$

which is easily verified by direct substitution. The total length of an optimal scheme on 2^k nodes is then $\epsilon(k) + (2^k - 1)$, which is $\frac{1}{9}[2^k(3k+1) - (-1)^k]$, or just $\frac{1}{9}[n(3\lceil\log n\rceil +$

$1) - (-1)^{\lceil \log n \rceil}]$. Farley's upper bound for total length in any network on n nodes was $(n-1)\lceil \log n \rceil$; asymptotically in n, our result is 1/3 of Farley's upper bound.

Finally, we determine $M(k,p)$. Referring again to Figure 2.7, and arguing much the same as we just did for $\epsilon(2^k)$, we can easily derive the following recurrence relation:

$$\begin{aligned}
M(0,p) &= 0, p \geq 0 \\
M(1,0) &= 1 \\
M(1,p) &= 0, p > 0 \\
M(k,0) &= 2k-1, k > 0 \\
M(k,p) &= M(k-1,p) + 2 \cdot M(k-2,p-1), p > 0, k > 0
\end{aligned}$$

The solution to this recurrence relation is

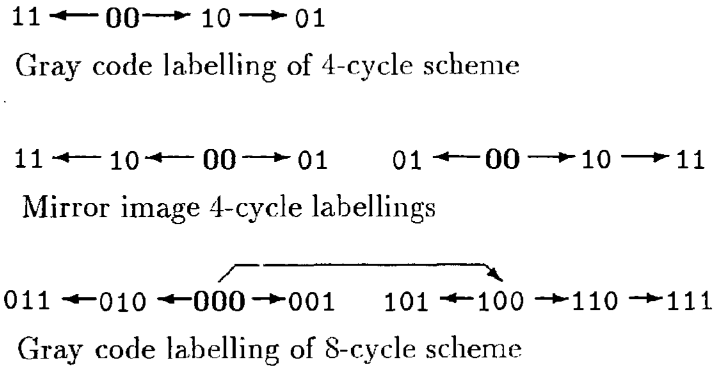
$$\begin{aligned}
M(k,p) &= 2^p \cdot \left[2 \cdot \binom{k-p-1}{p+1} + \binom{k-p-1}{p}\right] && (2.3) \\
&= 2^p \cdot \left[2 \cdot \left(\frac{k-p}{p+1}\right) - 1\right] \cdot \binom{k-p-1}{p}, && (2.4)
\end{aligned}$$

which can be confirmed by substitution.

2.4.2 Alternate Procedure for Creating Cycle Schemes

Next, we give an alternate procedure for creating optimal schemes for the 2^k-cycle, analyze the scheme using a binary reflected Gray code labeling naturally induced by the procedure, and find that the result of the analysis agrees with equation (2.2). Our motivation in developing this analysis was the search for a tight lower bound on total extra length required by the torus. Our hope was to extend the Gray code labelling which we use here to a 2-part or concatenated labelling of nodes in the cycle, where the concatenated label is formed from the labels which the node has in the two factor cycles of the torus (see Chapter 3.) We note that a different and more transparent determination of the cost of the first line call in a scheme produced by this alternate procedure appears in Chapter 3 in the analysis of the product method for toruses.

We begin the procedure by embedding the minimum time broadcast tree with 2^k nodes into the cycle. We find such an embedding in a recursive manner; the embedding for the 2^k-cycle is found directly from the embedding for the 2^{k-1}-cycle. Figure 2.8 shows the embedding for the 4-cycle and how we use it to produce an embedding for the 8-cycle. In the figure, we omit the links in the cycle; we assume that there is an link in the cycle between nodes which are adjacent in each picture, and between the leftmost and rightmost nodes.



Gray code labelling of 4-cycle scheme

Mirror image 4-cycle labellings

Gray code labelling of 8-cycle scheme

Figure 2.8: Labelled recursive constructions

For the 4-cycle, we first label the nodes as shown, where the node labelled (00) is the originator. For this discussion, we will use the shorthand "(00) calls (10)" to mean, for example, that the node labelled (00) calls the node labelled (10). In the scheme for the 4-cycle, (00) calls (10) at time 1, and then at time 2, (10) calls (11) while (00) calls (01). Given a scheme for the 2^{k-1}-cycle, we find a scheme for the 2^k-cycle as follows: we first place two 2^{k-1}-cycles side by side, with the labellings of the cycle on the right being a mirror image of the labellings of the cycle on the left, and with the nodes labelled $(0\ldots0)$ as close together as possible. Our description is somewhat informal here, but Fig. 2.8 makes clear by example what we have in mind. Note that in the third picture in the figure, if we transposed the two 4-cycles, the two nodes labelled (00) would be further apart. We make a 2^k-cycle from the two 2^{k-1}-cycles, in the obvious manner, assuming that any pair of adjacent nodes share a link and that the rightmost and leftmost nodes share a link. We then relabel all nodes, prefixing the labellings of nodes which we took from one of the 2^{k-1}-cycles with 1 and the labellings of the remaining nodes with 0. We produce a line broadcast scheme for the 2^k-cycle by having $(0\ldots0)$ call $(10\ldots0)$ in the first time unit of the scheme, and then having each of those two nodes begin the scheme for the 2^{k-1}-cycle.

The procedure described above produces a scheme on k phases for the 2^k cycle, since the first phase of the scheme uses one call to begin two subschemes on $k-1$ phases. If we show that the procedure described above produces a flat, nested, full scheme, then by Theorem 1 we will show that the scheme is optimal, and, since it is on k phases, that it is an optimal minimum-time scheme. However, we will analyze the extra cost of the scheme in a different way, arrive at exactly the extra cost given by equation (2.2), and thus prove that the scheme is optimal. The labelling of the nodes in the cycle is exactly a binary reflected Gray code as described in [7], p. 173. We can explain a node's label $(g_k\ldots g_1)$ as follows: for each node, there is a 'chain' of calls leading from the originator (labelled $(0\ldots0)$) of the scheme to the node. The bit $g_t, 1 \leq t \leq k$, is a 1 if and only if some call in that chain was made at phase $k-t+1$ of the scheme. For example, the originator is labelled $(0\ldots0)$ because its chain is empty, while the most far-flung node is labelled $(1\ldots1)$ because its chain contains a call made in every phase of the scheme. We will use this observation later.

Obviously, with the Gray code labelling, the length of a call from the node labelled $G(r)$ to the node labelled $G(s)$ is $|r-s|$. From [7], p. 176, we have that if $G(b)$ is $(g_k\ldots g_1)$, and b is $(b_{k-1}\ldots b_0)_2$, then

$$b_t = \sum_{m=t+1}^{k} \overline{g_m} \pmod 2, 0 \leq t < k \tag{2.5}$$

Now, sender and receiver are always Hamming distance 1 apart. In fact, it follows by a simple inductive argument that if a node is first informed at phase $t, 0 \leq t \leq k$, then bits g_1 to g_{k-t} in the label $(g_k\ldots g_1)$ of the node are all 0's, and that 'the node complements' each of those 0 bits, one at a time, in order from left to right, to 'get the label' of the node to call next. For example, if $n=4$, the node labelled (1000) is informed at phase 1, and it then calls (1100) at phase 2, (1010) at phase 3 and (1001) at phase 4. So, we have

$$\begin{aligned} G(s) &= (g_k \ldots g_{i+1} 0 g_{i-1} \ldots g_1) \\ G(r) &= (g_k \ldots g_{i+1} 1 g_{i-1} \ldots g_1) \end{aligned}$$

Now, referring to Equation (2.5), we have b_t being the same for $G(r)$ and $G(s), t \geq i$. Since $G(r)$ and $G(s)$ differ in only in one bit, if follows from (2.5) that the b_t's are complements, for $t < i$. To illustrate this, Fig. 2.9 shows a detailed example for 16 nodes ($k = 4$,) which shows the portions of the binary representations of s and r which are complements of each other (by enclosing those portions in [ ].)

phase	G(s) →G(r)	s →r	s →r
		(dec.)	(binary)
1)	0000 →1000	10 →05	[1010] →[0101]
2)	0000 →0100	10 →13	1[010] →1[101]
	1000 →1100	05 →02	0[101] →0[010]
3)	0000 →0010	10 →09	10[10] →10[01]
	1000 →1010	05 →06	01[01] →01[10]
	0100 →0110	13 →14	11[01] →11[10]
	1100 →1110	02 →01	00[10] →00[01]
4)	0000 →0001	10 →11	101[0] →101[1]
	1000 →1001	05 →04	010[1] →010[0]
	0100 →0101	13 →12	101[1] →101[0]
	1100 →1101	02 →03	001[0] →001[1]
	0010 →0011	09 →08	100[1] →100[0]
	1010 →1011	06 →07	011[0] →011[1]
	0110 →0111	14 →15	111[0] →111[1]
	1110 →1111	01 →00	000[1] →000[0]

Figure 2.9: Gray code labels and indices for 16-cycle scheme

If we let $\beta(t)$ be the length of the calls made in phase $k - t$, or equivalently, the length of the longest call in the scheme for the 2^t-cycle, then we have $\beta(t) = |r - s|$, if the call is between nodes with labels $G(r)$ and $G(s)$. Clearly, $\beta(0) = 0$, and examining the example of Fig. 2.9, we see that

$$\begin{aligned}\beta(1) &= (1)_2 - (0)_2 &&= (1)_2\\ \beta(2) &= (10)_2 - (01)_2 &&= (01)_2\\ \beta(3) &= (101)_2 - (010)_2 &&= (011)_2\\ \beta(4) &= (1010)_2 - (0101)_2 &&= (0101)_2\end{aligned}$$

Consider the label of the first sender, $G(s) = (g_k \dots g_1) = (0 \dots 0)$. From (2.5),

$$\begin{aligned}b_k &= 0\\ b_{k-1} &= \sum_{m=k}^{k} \overline{g_m} \pmod 2 = 1\\ b_{k-2} &= \sum_{m=k-1}^{k} \overline{g_m} \pmod 2 = 0\end{aligned}$$

and by induction,

$$b_{k-t} = \begin{cases} 0, & t \text{ even} \\ 1, & t \text{ odd} \end{cases} \tag{2.6}$$

Thus, s is of the form (1010...101) if k is even and (1010...10) if k is odd; r is of the form (0101...0) if k is even and (0101...1) if k is odd. So the length of the first call is given by the binary number

$$\beta(k) = \begin{cases} (01)^{k/2}, & k \text{ even} \\ (01)^{(k-1)/2}1, & k \text{ odd,} \end{cases} \tag{2.7}$$

the exponents meaning string concatenation. In general, $\beta(0) = 0$ and

$$\beta(k) = \beta(k-1) + (-1)^k \tag{2.8}$$

The solution to this recurrence relation is

$$\beta(k) = [2^k - (-1)^k]/3. \tag{2.9}$$

If we let $\alpha(k)$ be the extra length of the first call made in the scheme for the 2^k-cycle, then $\alpha(0) = 0$, and

$$\alpha(k) = [2^k - (-1)^k]/3 - 1, k > 0 \tag{2.10}$$

Then, the total extra length $\epsilon'(k)$ for the scheme for the 2^k-cycle is given recursively by

$$\epsilon'(k) = \begin{cases} 0, & k = 0 \\ 2 \cdot \epsilon'(k-1) + \alpha(k), & k > 0. \end{cases} \tag{2.11}$$

The solution to this recurrence relation is equation (2.2), $\epsilon(2^k)$. So, the cycle schemes produced by this alternate procedure are optimal.

2.4.3 The Elimination Method for Cycles

In Section 2.3.2, we described creating a cycle scheme on n nodes by creating a top path and then creating the bottom schemes. We can use this method to create optimal minimum time schemes on n nodes, $2^{k-1} < n < 2^k$, making sure that the broadcast tree is the rooted binomial tree on 2^k nodes, with some subtrees removed. An alternate method is to simply create an optimal scheme on 2^k nodes, using, for example, one of the recursive procedures described in the previous section, and then eliminate the most expensive calls in the scheme until we have $n-1$ calls left; we call this procedure the *elimination method.* We begin with a flat, nested, full scheme with 2^k nodes and k phases and remove, in the manner discussed in Section 2.3.3, a deepest layer leaf node. We continue removing leaf nodes until we have removed a total of $2^k - n$ nodes. As argued in Section 2.3.3, removing the leaf nodes preserves nestedness and flatness, and the resulting scheme is automatically full, so by Theorem 1, the scheme is an optimal minimum time scheme. The cost of the scheme is easily obtained by using Equations (2.1) and (2.3) and noting that there are $\lceil k/2 \rceil$ layers in the flat, nested and full scheme with 2^k nodes.

2.4.4 Embedding Conjecture

We feel, partly on the basis of the apparent similarity of some of Iordanskiǐ's results to ours, that the conjecture we present in this section is correct. We first discuss the difference between *unconstrained* embeddings of tree into linear networks and the embeddings used in our cycle schemes (a linear network is a cycle or a path.) In an unconstrained embedding, an undirected, unordered tree is embedded. In creating our cycle schemes, we embed a broadcast tree into the cycle. The tree is directed and has a *level-ordering*; each node in the tree has a level which is the phase, of the scheme, at which the node is informed. Each edge in the tree, which represents a call in the scheme, can also be assigned a level, which is the level of the node which the call informs. Given this assignment, we can describe a constraint on our embeddings which is imposed by the edge-disjointedness feature of the line broadcasting model; no pair of calls at the same level can be embedded into the cycle so that their embeddings share a link.

Conjecture 1 *An optimal cycle scheme $S(G, T, \Phi)$ with n nodes has the same total length as the optimal unconstrained embedding of the undirected version of the broadcast tree T into the cycle with n nodes or into a linear network with at least n nodes, where the unconstrained embedding maps nodes of the tree one-to-one.*

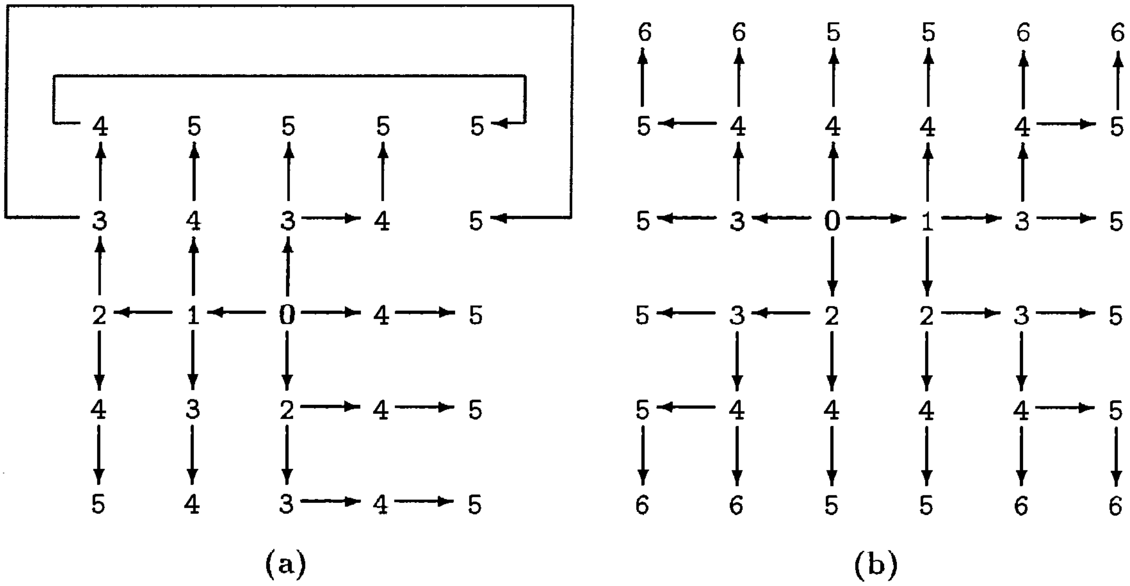
Chapter 3

Line Broadcast Schemes in the Torus

3.1 Overview

We now look at a constructive approach to finding upper bounds on total extra length required by the torus. We do not have a good lower bound. A torus is like a grid graph, in which each node is connected to its neighbors to the north, south, east, and west, except that the torus 'wraps around' north-south and east-west. Note that the torus is vertex-transitive. We will refer to a torus as an $m \times n$ torus to mean that it has m rows and n columns. An $m \times n$ torus can also be described a a *product* of an m-cycle and an n-cycle. We will sometimes refer to a torus scheme as a product of two cycle schemes, with an obvious meaning.

It is straightforward to complete local broadcasting in minimum time in small toruses. Figure 3.1 shows minimum local broadcast schemes for the 5×5 torus, in part (a), and the 6×6 torus, in part (b). When showing an $m \times n$ torus scheme in a figure, we will show a node by a number indicating the phase at which the node is informed, or sometimes, by a 2-part Gray code labelling. Arrows will point from source nodes to destination nodes (the arrowheads are redundant since the sender is alway informed before the receiver.) We will generally omit the edges between nodes to keep figures simpler, with the understanding that each node is connected to its neighbors to the north, south, east, and west, and that the torus wraps around north-south and east-west. Note that both the 5 × 5 and the 6 × 6 torus schemes are local broadcast schemes. Also note that the 5 × 5 scheme wraps around one dimension of the torus; by this we mean that the scheme could not be drawn within a 5 × 5 grid. We prove in Section 3.2.4, there is no local scheme for the 5 × 5 torus which does not wrap around the torus.



(a) (b)

Figure 3.1: 5 × 5 and 6 × 6 local broadcast schemes

We now show a weak lower bound and an upper bound for total extra length required by an $m \times n$ torus. To set a lower bound, first observe that the diameter of an $m \times n$ torus is $\lfloor m/2 \rfloor + \lfloor n/2 \rfloor$. In both a 4 × 4 and a 5 × 5 torus, the diameter is 4. So, the shortest path from the originator to the most distant point in the torus is $\lfloor m/2 \rfloor + \lfloor n/2 \rfloor$ edges long. Since we have $\lceil \log(mn) \rceil$ phases in which to complete broadcasting, we require at least $\lfloor m/2 \rfloor + \lfloor n/2 \rfloor - \lceil \log(mn) \rceil$ extra edges to reach that most distant node, when $\lfloor m/2 \rfloor + \lfloor n/2 \rfloor > \lceil \log(mn) \rceil$. In a 9 × 9 torus, $\lfloor m/2 \rfloor + \lfloor n/2 \rfloor = 8$. $\lceil \log(mn) \rceil = \lceil \log(9 \cdot 9) \rceil = 7$. So, the 9 × 9 torus requires extra length of at least 1.

To show an upper bound, we present the *product method*, which produces schemes for the $2^k \times 2^k$ torus from the scheme for the 2^k-cycle.

3.2 The Product Method

3.2.1 Description of the Product Method

Figure 3.2 shows how we use the product method to produce a line broadcast scheme for the 4×4 torus. Part (a) shows a 4-cycle scheme labelled with the Gray code labelling described in Section 2.4.2. We begin with the labelling because we feel that it may suggest a lower bound analysis in future work. Part (b) shows the 4×4 torus as the product of two labelled 4-cycles. In the figure, we show each cycle with its line broadcast scheme. The labelling of each node is the concatenation of the node's label in one cycle of the product with its label in the other cycle of the product (we show the two components of each label offset.) Second, we choose a subset of the calls in the schemes. The originator is the node whose label is all 0s. We begin by choosing the first call from the scheme for one of the component cycles of the product; in the figure, we choose the call from 0000 to 0001. At each subsequent step, we choose, for each informed node, the next call from the scheme for the alternate cycle. For example, the second choice is the two calls 0000 to 0100 and 0001 to 0101. Part (c) shows the resulting line broadcast scheme for the 4×4 torus, with phases shown instead of labellings.

In the generalization of the method to the $2^k \times 2^k$ torus, we produce the product of two 2^k-cycles and select calls from the line broadcast schemes for the two component cycles. The originator is the node whose label is all 0s. The first call we choose is the first call from the scheme for one of the component cycles. At step $p = 0, 2, 4 \ldots$, we make, for each informed node, the call which that node would make at step $p/2$ in the line broadcast scheme for the one component cycle, while at step $p = 1, 3, 5, \ldots$, we make for each informed node the call which that node would make at step $(p-1)/2$ in the line broadcast scheme for the other component cycle. In the $2^k \times 2^k$ torus, the product method amounts to alternating at each step between the two dimensions of the torus (with the choice of call determined by the line broadcast scheme for the

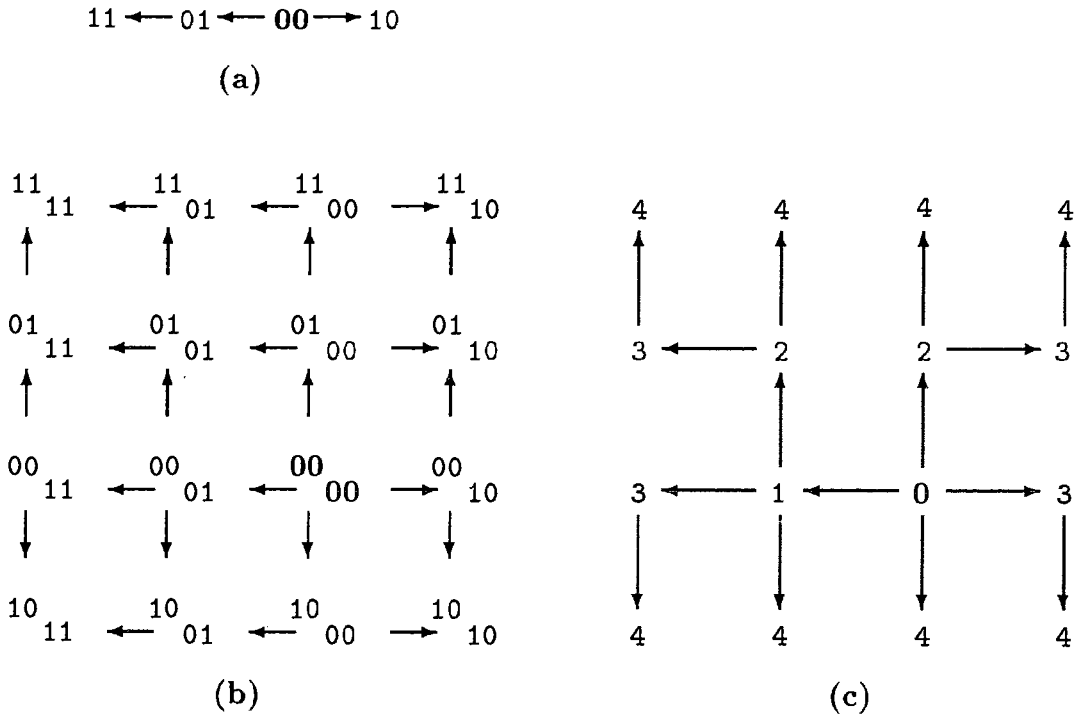


Figure 3.2: The product method for the 4 × 4 torus

2^k-cycle.)

Figures 3.3 and 3.4 illustrates the production of an 8 × 8 torus scheme by the product method. Figure 3.3 part (a) shows the labelled 8-cycle scheme, and part (b) shows the 8 × 8 torus as a product of two 8-cycles, along with the line broadcast schemes for each cycle. Figure 3.4 shows the line broadcast scheme for the 8 × 8 torus which results from choosing calls as described above.

3.2.2 Analysis of the Product Method

The extra length used by the product method is given as follows:

Claim 1 *For the $2^k \times 2^k$ torus, $k \geq 2$, the product method uses total extra length*

$$\mathcal{E}(2^k) = \frac{1}{5}[4^k + (-1)^{k-1}] - 2^k + 1. \tag{3.1}$$

Proof: For the 4 × 4 torus, $k = 2$, and the total extra length is 0 (see Fig. 3.2); setting k to 2 in Equation (3.1), we obtain

$$\begin{aligned} \frac{1}{5}[4^k + (-1)^{k-1}] - 2^k + 1 &= \frac{1}{5}[4^2 + (-1)^{2-1}] - 2^2 + 1 \\ &= \frac{1}{5}[16 - 1] - 4 + 1 \\ &= 3 - 4 + 1 \\ &= 0 \end{aligned}$$

So, the formula is correct for $k = 2$. Assume that the formula is correct for k, and define $D(j)$ as the total extra length of the first three calls in the scheme for the $2^j \times 2^j$ torus. Then, $\mathcal{E}(2^{k+1}) = 4 \cdot \mathcal{E}(2^k) + D(k+1)$. The first call in the scheme for the $2^k \times 2^k$ torus is the first call in the scheme for the 2^k-cycle; the second and third calls are also the first call in the scheme for the 2^k-cycle, though in the torus they are made in the other dimension than the one in which the first call was made. The formula for $D(k)$ is $2^k - 3 + (-1)^{k-1}$, which we show as follows: the extra length $p(k)$ of each of the first three calls for the $2^k \times 2^k$ torus is given by $p(k) = (2^k - 3 + (-1)^{k-1})/3$, as we show in a moment. Then, 3 times $p(k)$ is $D(k)$ as given. To show that the formula

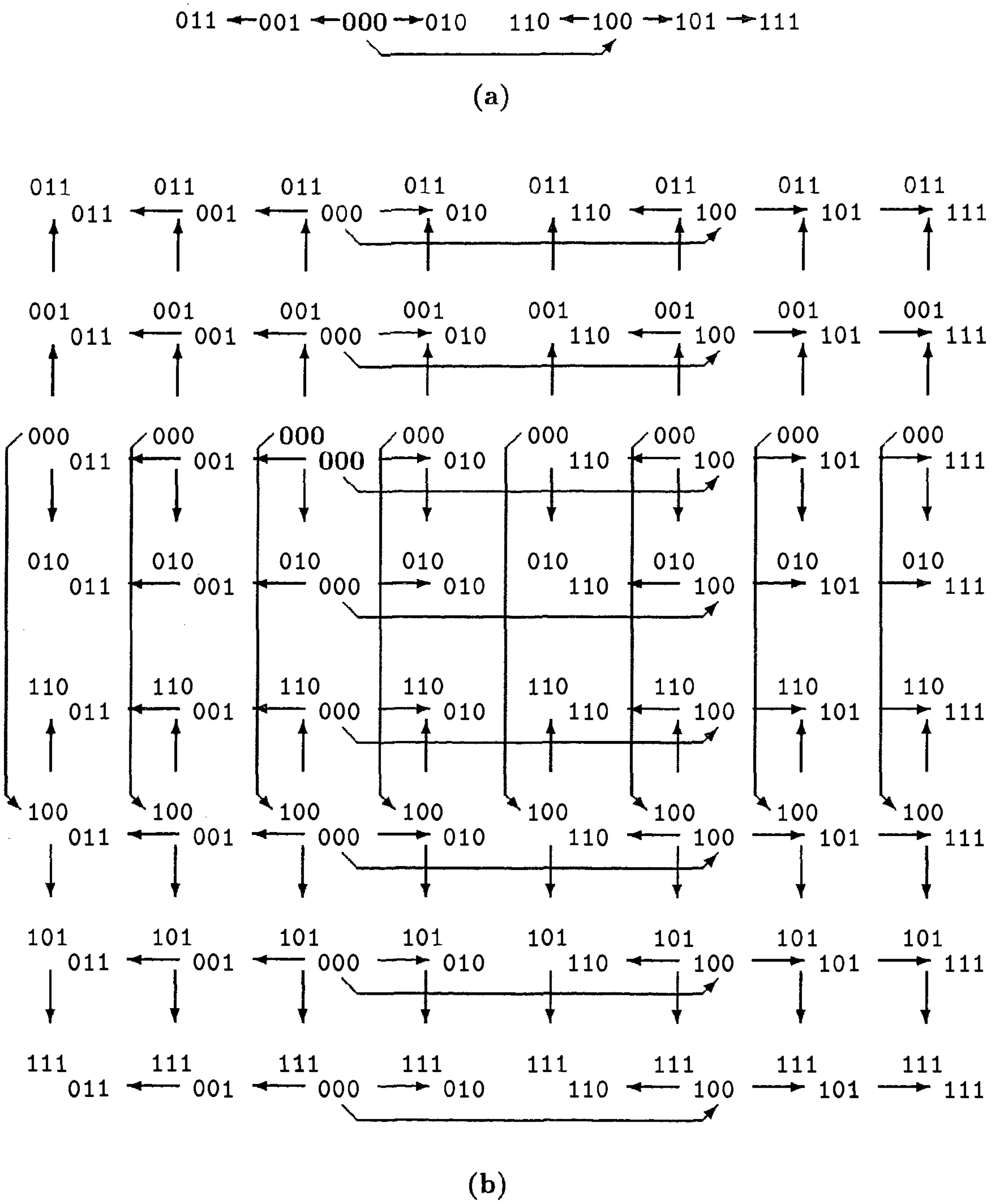


Figure 3.3: The product method for the 8 × 8 torus

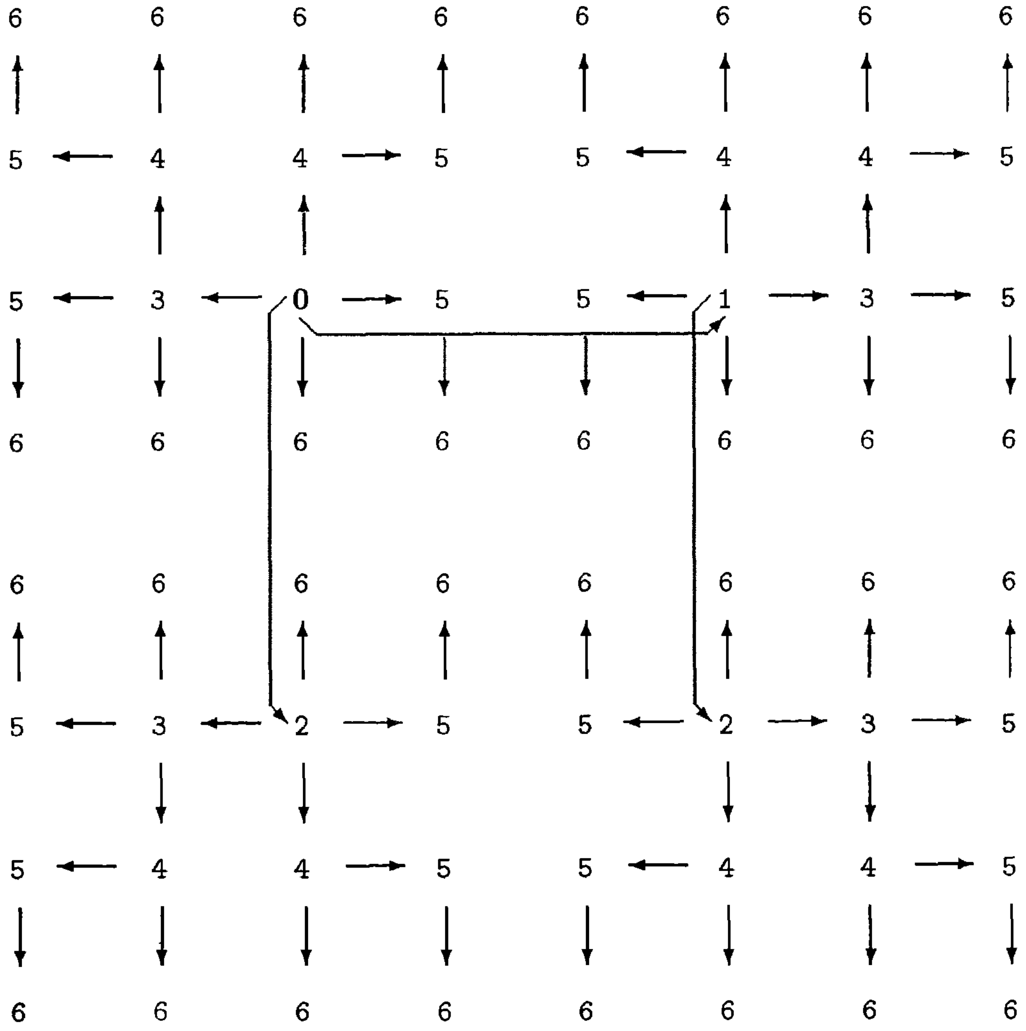


Figure 3.4: Line broadcast scheme from product method for the 8 × 8 torus

for $p(k)$ is correct, we define $x(k)$ as the number of cycle edges between the originator and nearest end node of the 2^k-cycle scheme. It is easy to see that

$$\begin{aligned}
x(0) &= 0 \\
x(1) &= 0 \\
x(2) &= 1 \\
x(3) &= 2^{3-1} - x(3-1) - 1 \\
&= 4 - 1 - 1 \\
&= 2 \\
x(4) &= 2^{4-1} - x(4-1) - 1 \\
&= 2^3 - x(3) - 1 \\
&= 8 - 2 - 1 \\
&= 5,
\end{aligned}$$

and that in general,

$$x(k) = 2^{k-1} - x(k-1) - 1.$$

The solution to this recurrence relation is $x(k) = (2^{k+1} - 3 + (-1)^k)/6$.

It is also easy to see that

$$\begin{aligned}
p(0) &= 0 \\
p(1) &= 0 \\
p(2) &= 0 \\
p(3) &= 2 \cdot x(3-1) \\
&= 2 \cdot x(2) \\
&= 2 \cdot 1 \\
&= 2 \\
p(4) &= 2 \cdot x(4-1) \\
&= 2 \cdot x(3)
\end{aligned}$$

$$\begin{aligned} &= 2\cdot 2 \\ &= 4, \end{aligned}$$

and that in general,

$$\begin{aligned} p(k) &= 2\cdot x(k-1) \\ &= (2^k-3+(-1)^{k-1})/3 \end{aligned}$$

As expected, this result agrees with the Gray code analysis of the length of the first call in a 2^k-cycle, as given in Section 2.4.2. Thus,

$$\begin{aligned} \mathcal{E}(2^{k+1}) &= 4\cdot\mathcal{E}(2^k)+D(k+1) \\ &= 4\cdot(\frac{1}{5}[4^k+(-1)^{k-1}]-2^k+1)+2^{k+1}-3+(-1)^{(k+1)-1} \\ &= \frac{1}{5}[4^{k+1}-4\cdot(-1)^k]-2^{k+2}+4+2^{k+1}-3+(-1)^k \\ &= \frac{1}{5}[4^{k+1}+(5-4)\cdot(-1)^k]-2^{k+1}+1 \\ &= \frac{1}{5}[4^{(k+1)}+(-1)^{(k+1)-1}]-2^{(k+1)}+1, \end{aligned}$$

and the result is proved by induction. □

For the $2^k\times 2^k$ torus, then, the total length is $\mathcal{E}(2^k)+(2^k-1)$, which is $\frac{1}{5}[4^k+(-1)^{k-1}]$, or simply $\frac{1}{5}[n+(-1)^{\lceil\log n\rceil-1}]$, since the number of nodes n is $2^k\cdot 2^k=4^k$. Farley's upper bound for total length is $(n-1)\lceil\log n\rceil$, so our upper bound is asymptotically $5\lceil\log n\rceil$ times smaller than Farley's upper bound.

3.2.3 Other Uses of the Product Method

We can use the schemes produced by the product method to generate schemes for 'odd-sized' toruses, by rounding up the dimensions of the torus to a power of 2 and using the result for that power of 2. For example, suppose that we have a 7×6 torus. We have an upper bound for an 8×8 torus from the product method, and since a 7×6 torus 'fits' into an 8×8 torus for the purposes of the product method, we can simply

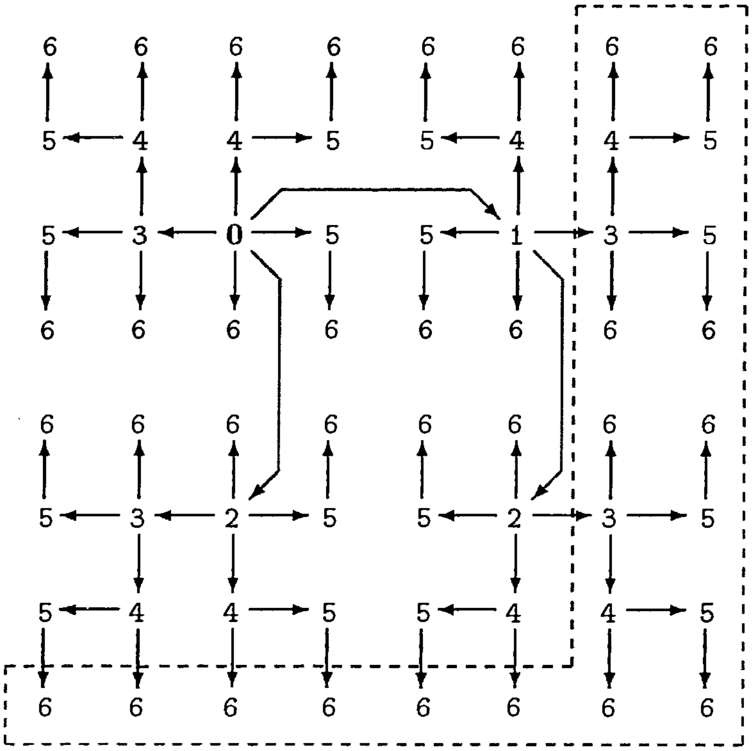


Figure 3.5: Adapting the product method for the 7 × 6 torus

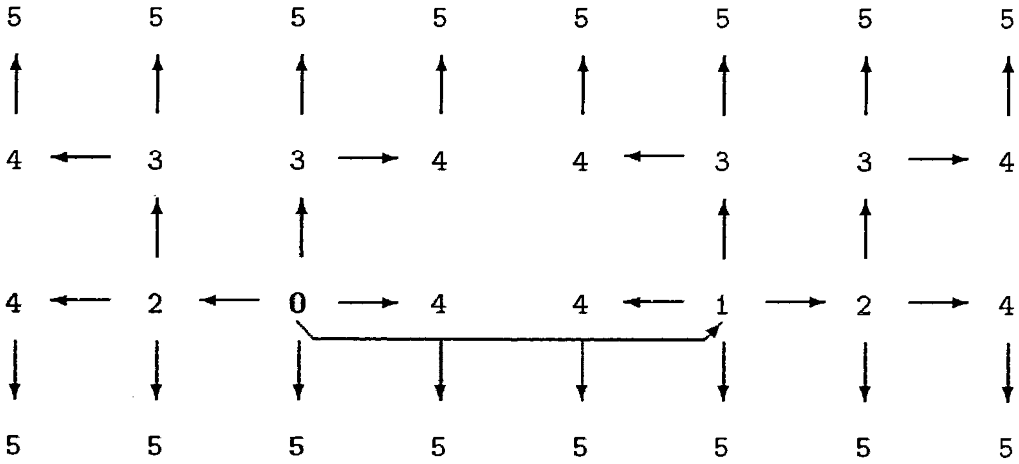


Figure 3.6: Adapting the product method for the 4 × 8 torus

use the scheme used for the 8×8 torus but without making calls that the 7×6 torus doesn't 'require'; see figure 3.5. Figure 3.6 shows a scheme for 4×8 torus. Results for $m \times 2m$ toruses can be used as upper bounds for $m \times n$ toruses where $n < 2m$.

We can let $k = \lceil \log \max(m, n) \rceil$ and use the product method. So, we can establish an upper bound for any m and n. We can easily improve this bound in several ways. First of all, we can discount some extra edges when m is not a power of 2. If we refer to figure 3.5, and imagine that the torus were actually of size $7 \cdot 4^k \times 6 \cdot 4^k, k > 0$, then the outlined area in the figure would actually contain line calls which the 7×6 scheme would not include and which we could discount. There are, however, better ways to choose the calls to 'ignore', as discussed in the next section.

We can also use the product method to directly generate schemes for $2^k \times 2^j$ toruses; we find the schemes for the 2^k-cycle and 2^j-cycle and in selecting calls, we first broadcast only in one dimension until we have a set of 'square' toruses in which to complete broadcasting. Assuming $j > k$, we first complete a 2^{j-k}-cycle scheme and then do $(j - k)$ copies of the $2^k \times 2^k$ torus scheme in parallel.

We can also consider other graphs which can be described as products; in producing schemes for such graphs, we begin by finding schemes for each component of the product. We are not restricted to binary products; if a graph is a product of δ factors, we begin by finding a scheme for each of the δ factors. For example, we can easily find schemes for all δ-dimensional cycles (where a torus is a 2-dimensional cycle.)

Our experience with the product method leads us to make the following conjecture.

Conjecture 2 *The product method (with its extensions as just described) produces optimal schemes for all δ-dimensional cycles, $\delta \geq 1$, where the number of nodes in each factor cycle is a power of 2.*

The hunch behind the conjecture might be described as an intuition about the 'orthogonality' of a product. The conjecture is true for $\delta = 1$; in that case we have cycles, and will reproduce our optimal cycle schemes of Chapter 2. It might be instructive to try to find optimal schemes for $2^k \times m$ toruses, for $1 < m \leq 6$. The product method schemes in these cases use exactly the extra length used by an optimal 2^k-cycle scheme, because a local broadcast scheme suffices for the second ('m') dimension when $1 < m \leq 6$. It seems compellingly obvious that the addition of the second dimension in these cases does not allow us to use less extra length than we used in the 2^k-cycle; perhaps the proof of this restricted result could be generalized to allow m to also take on the values $8, 16, 32, \ldots$, as in the conjecture. If the conjecture is correct, then it probably generalizes to $m \times n$ toruses for less restricted values of m and n. However, it cannot be true for *unrestricted* m and n, as we discuss in the next section.

3.2.4 The Elimination Method for Toruses

In Section 2.4.3 we described an elimination method for cycles. The method involved removing the costliest calls from an optimal 2^k-cycle scheme to produce an optimal scheme for any other value of n, $2^{k-1} < n < 2^k$. In this section, we describe an elimination method for toruses. This method involves removing the costliest *rows* and *columns* of calls to leaf nodes from a $2^k \times 2^k$ torus to produce a scheme for an $m \times n$ torus, where $m, n \leq 2^k$. The method can be generalized to $2^k \times 2^j$ toruses, but for simplicity we will deal only with $2^k \times 2^k$ toruses. The present aim is not to produce *optimal* schemes for general $m \times n$ toruses, as we produced optimal schemes for general n-cycles. We should not expect to do so, since we have not proved the optimality of the $2^k \times 2^k$ torus schemes. We will instead systematically describe the construction of $m \times n$ torus schemes for some values of m and n, discuss for what values of m and n the constructions are valid, and present some results which could lead to an analysis of the cost of schemes produced by the method.

An example of this method is producing the 6×6 scheme from the 8×8 scheme. Comparing Figs. 3.3 and 3.1, we see that if we remove the middle two columns and middle two rows of the 8×8 scheme, we remove all extra length from the scheme and produce exactly the 6×6 scheme. However, we now argue that there is no way to produce an optimal 5×5 scheme by eliminating rows and columns from a larger product method scheme. Every optimal 5×5 scheme wraps around the torus in one dimension. But no 'wrap-around' can involve all leaf nodes in one row or column, or else the wrap-around is not actually a wrap-around, just a 'shift' of the entire scheme by one column. A shift does not change the scheme, since the torus is vertex-transitive. But no product method scheme wraps around, so producing from such a scheme another scheme which wraps around would require removing only *part* of a row or column and leaving the rest; the elimination method does not do this.

To show that every optimal 5×5 scheme wraps around, we try to construct a local broadcast scheme which does not wrap around. Finding such a scheme is equivalent to finding a suitable originator in a 5×5 grid graph and then finding a local broadcast scheme for the graph and for that originator. There is no such originator, however. If we examine the grid graph in Fig. 3.7, part (a), we note that only the 5 candidate originators in the cross-shaped are within 5 links of each of the four circled nodes in the corners of the grid. Thus, only those 5 originators could possibly reach each of the corner nodes in the 5 available phases using only local calls. Up to isomorphism,

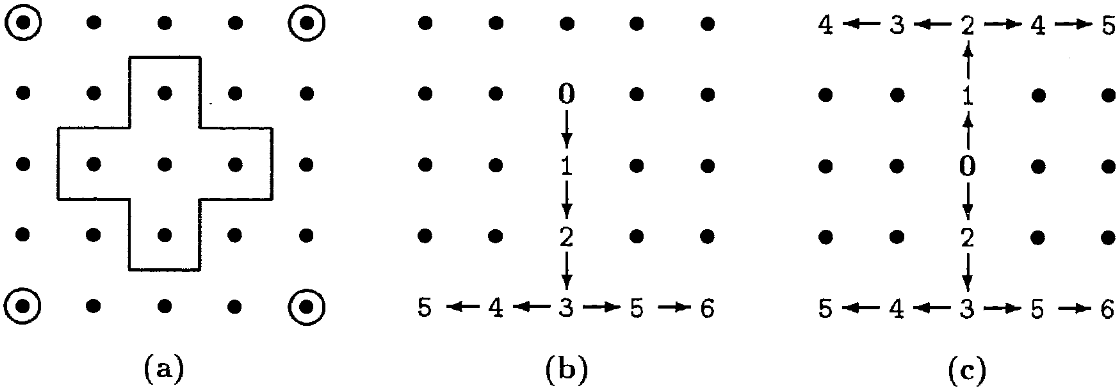


Figure 3.7: Why the optimal 5×5 scheme must wrap around

there are only two distinct originators in the cross-shaped area; we examine these two originators in parts (b) and (c). A local scheme beginning at the originator shown in part (b) cannot inform both of the labelled corner nodes in 5 phases. Each corner node is 5 links from the originator; the path of local calls which informs either node must be 5 links long, and no node on the path can 'delay' before making its call on the path. However, a single such path cannot inform both nodes; there must be two such paths (although the two paths may share some links.) If we select one of the two nodes to inform by such a path, then at some node on that path, a 'delay' of one phase must occur before the second path is started; the delay may occur as 'early' as at the originator, in which case the two paths share no links. Thus the node which we did not select cannot be informed by phase 5. A 'failed' example subscheme is shown. A local scheme beginning at the originator in part (c) cannot inform all four of the corner nodes in 5 phases. Each corner node is 4 links from the originator and cannot be informed before phase 4. The first call by the originator informs a node that is closer than the originator to only two of the corner nodes, in adjacent corners of the graph. Thus the other two corner nodes are 'delayed' by one phase; neither of those latter two nodes can be informed before phase 5. At some phase after a call is made that is closer than the originator to one of those two latter nodes, a further call must be made that again delays one of those two latter nodes; that twice-delayed node cannot be informed by phase 5. Again, a 'failed' example subscheme is shown.

There is another limitation on the use of the product method, which we illustrate with the 8×8, 6×6 and 5×5 schemes. The number of nodes in the 8×8 torus is 64, which is 2^6. Therefore, the broadcast tree of a (minimum time) scheme for the 8×8 torus has 6 phases. The 6×6 torus has 36 nodes, while the 5×5 torus has 25 nodes. Now, $2^5 < 36 < 2^6$, so the 6×6 broadcast tree has 6 phases. However $25 < 2^5$, so the 5×5 tree has only 5 phases. This fact does away with a naive plan, illustrated in Fig. 3.8, to produce an optimal scheme for the 5×5 torus. In part (a), we create an optimal 8-cycle scheme, in part (b) we use the elimination method for cycles to produce an optimal 5-cycle scheme, and then in part (c) we produce a 5×5 torus scheme as the product of these two 5-cycle schemes. The resulting scheme has 6 phases, which is too many. At any rate, the scheme does not wrap around, and as we have seen, no such 5×5 scheme is optimal.

In general, the plan takes an optimal m-cycle scheme and an optimal n-cycle scheme and produces a scheme for the $m \times n$ torus as the product of the two cycle schemes. The plan works in many cases. For example, if we examine the 8-cycle scheme in Fig. 3.8 again, we can see that by eliminating the leaf nodes under the long call, we obtain an optimal 6-cycle scheme. If we then examine the 6×6 scheme in Fig. 3.1 again, we can see that it actually *is* the product of two 6-cycle schemes. The plan will *not* work when $\lceil \log(m \cdot n) \rceil$ is less than $\lceil \log m \rceil + \lceil \log n \rceil$, as is the case

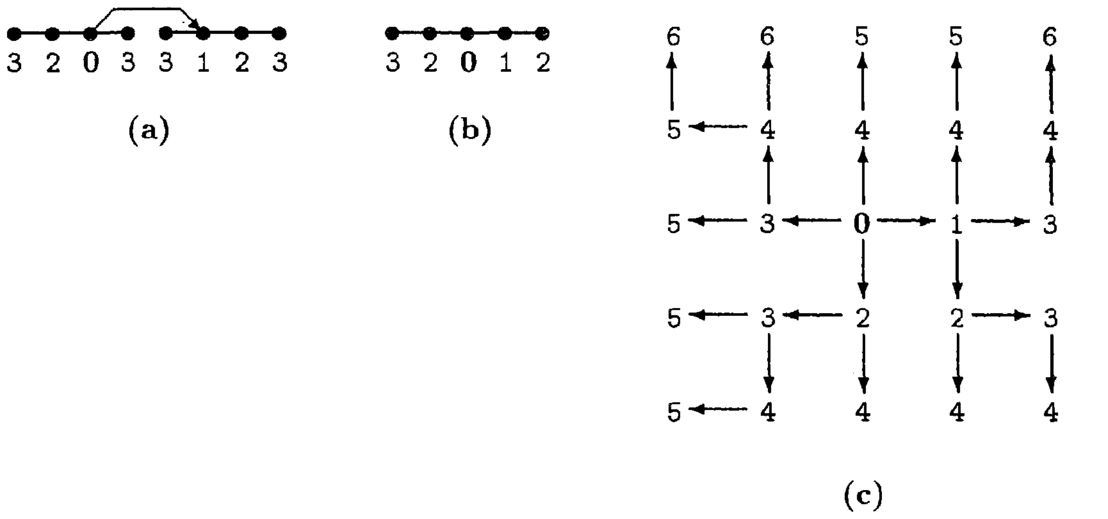


Figure 3.8: A naive plan for the 5 × 5 torus

when m and n are both 5. The problem in such cases is that all $\lceil \log m \rceil$ phases are retained in one dimension and all $\lceil \log n \rceil$ phases in the other dimension, whereas one phase must be lost in the torus because the total number of nodes has dropped below $\frac{1}{2}m \cdot n$.

So, we will restrict ourselves to values of m and n where

$$\lceil \log(m \cdot n) \rceil = \lceil \log m \rceil + \lceil \log n \rceil \tag{3.2}$$

We note, however, that we *could* eliminate two more rows and two more columns from the 6 × 6 scheme and arrive at the 4 × 4 scheme; we would first eliminate the two outer rows of leaf nodes and then the two outer columns of leaf nodes left after eliminating the rows. However, this exercise is not particularly revealing; to satisfy (3.2), we have had to eliminate *all* leaf nodes from the scheme with which we began, because all of those leaf nodes were informed in the last (and now 'illegal') phase.

Our results on the cost of schemes produced by the elimination method are preliminary. We restrict ourselves to the case where we only eliminate some *rows of leaf nodes.* We draw a distinction between rows and columns of a scheme. If we examine the 4 × 4 scheme in Fig. 3.2 and the 8 × 8 scheme in Fig. 3.4, we note that the first call in the scheme is in one dimension of the torus, and that the next two are in the other dimension. With the way we have chosen to select calls from the factor cycle schemes, we can see that this distinction between the two dimensions continues recursively; the four 4×4 subschemes in the 8×8 scheme all have the same feature, and the dimension in which the *single* call occurs is the same at all levels. We will define the *rows* of the scheme as the factor cycles which cross the *double* calls. That way, when we eliminate a row, we will shorten more calls than if we were to eliminate a *column*, which always only goes through the single calls.

The reason we have chosen to restrict ourselves to row eliminations is that eliminating both rows and columns complicates the analysis. For example, assume that we first eliminate some rows. What this does is to *shorten* some columns. This means that we cannot independently analyze the cost of eliminating the columns as though they had been removed first.

The elimination method for toruses is similar to the elimination method for cycles in one respect. We described the elimination method for cycles as repeated removal of leaf nodes from a cycle scheme. We will describe the elimination method for the torus as removal of rows of leaf nodes from a torus scheme. For example, consider producing a 6×8 scheme from the 8×8 scheme shown in Figure 3.3; we eliminate the middle two rows of leaf nodes. We *could* eliminate the outer two rows of leaf nodes, but we would not save any extra length that way, so the resulting scheme could not be optimal. We also note that in the 4×4 scheme in Fig. 3.2, 2 out of the 4 rows are rows of leaf nodes. Since the 4×4 scheme is the 'building block' of all $2^k \times 2^j$ schemes, it follows that exactly half of all the rows in any such scheme are rows of leaf nodes. A similar feature holds for cycle schemes, and can be seen to account for the feature in the torus; exactly half of all leaf nodes in a 2^k-cycle scheme are leaf nodes, since the number of informed nodes doubles in the last phase and all nodes informed in the last phase must be leaves of the broadcast tree. The reason this feature of the cycle accounts for the feature in the torus is that the rows of leaf nodes in the torus correspond to the leaf nodes of the 'second' dimension of the cycle (the direction chosen second as calls are selected from the factor schemes.)

Our investigation is an attempt to determine the cost distribution of the rows of leaf nodes. When we eliminate a leaf node, we shorten all the calls which that leaf is under. As in Section 2.3.3 on cycles, this suggests an accounting method for the cost of calls in a torus scheme. In particular, we are trying to find an accounting method for the cost of local calls which are parallel to *columns* of the scheme. We note that not all calls informing the leaf nodes of a row have the same cost. For example, consider Fig. 3.4. Only two of the calls informing leaf nodes in a middle row are under the long calls of the scheme; it is the elimination of *these* two calls that saves extra length. The situation becomes more complicated for larger toruses. Two calls will be under the 'topmost' long calls; *four* will under the long calls at the next step of recursion, and so on, where a step of recursion is defined as the first three calls, ie. the splitting of a grid of the torus into four subgrids. Out of those latter four calls, only *two* will be under the long calls at both of the first two steps of recursion.

There should be a recursive description of the situation, analogous to the recursive description used to find $\epsilon(2^k)$ and $M(k,p)$ for cycles in Section 2.4.1. We have not developed this description yet. However, it should provide a ***distribution***, a set of total costs of rows and the frequency of occurence of each cost. In principal, this distribution tells us the cost of the cheapest scheme we could produce by the elimination of rows of leaf nodes; we simply eliminate as many of the costliest rows as we can, then as many of the next costliest, etc. It is also possible that a description of this sequence could lead to insights regarding a lower bound on cost for the $2^k \times 2^k$ torus.

What we have done is to find by 'brute force programming' the total extra length of each row of leaves in the $2^k \times 2^k$ torus and looked for patterns. The patterns appear to be generalized Fibonnaci sequences. The number of cost values is $\mathcal{F}(k)$, the kth Fibonnaci number. For $k = 4$, the rows of leaves come in three possible total extra lengths: $0, 2,$ and 4 ($\mathcal{F}(4) = 3$.) For $k = 5$, the total extra lengths are $0, 2, 4, 8, 10$ ($\mathcal{F}(5) = 5$.)

For any value of k, the possible extra lengths are simply each of the first $\mathcal{F}(k)$ values of the folowing sequence, multiplied by 2:

```
0,1,2,4,5,8,9,10,16,17,18,19,20,21,32,33,34,36,37,40,
41,42,64,65,66,68,69,72,73,74,80,81,82,84,85,128,...
```

This series is actually composed of successively longer pieces, where piece 0 has length 1, and piece r has length $\mathcal{F}(r)$, $r > 0$. Piece 0 is (0); piece 1 is (1), piece 2 is (2); to get piece r, $r > 2$,

1. take piece $r - 1$ and add 2^{r-2} to each member.

2. Then, take piece $r - 2$ and add 2^{r-1} to each member.

Appending the second piece to the first gives piece r. Note that piece $r - 1$ has length $\mathcal{F}(r-1)$ and piece $r - 2$ has length $\mathcal{F}(r-2)$, so piece r has length $\mathcal{F}(r-1) + \mathcal{F}(r-2)$, which is $\mathcal{F}(r)$ as desired. Here is the sequence again, with the pieces bracketted:

```
              piece 3
                 |
(0),(1),(2),(4,5),(8,9,10),(16,17,18,20,21),
(32,33,34,36,37,40,41,42),
(64,65,66,68,69,72,73,74,80,81,82,84,85),(128,...
```

Now, each cost occurs a certain number of times. eg., for $k = 4$, the possible costs are $0, 2, 4$; 0 occurs 2 times, 2 occurs 2 times, and 4 occurs 4 times (total of all occurrences is 8, which makes sense since there are 8 rows of leaf nodes in the $2^4 \times 2^4$ torus.) The occurences are described by another series, which looks like this (already bracketted:)

```
(2),(2),(4),(4,4),(4,4,8),(4,4,8,8,8),
(4,4,8,8,8,8,8,16),...
```

The length pattern is the same as in the previous sequence; piece r has length $\mathcal{F}(r)$. To obtain piece $r, r > 2$, we take piece $r - 1$ and append to it piece $r - 2$ with all elements of piece $r - 2$ multiplied by 2. There is no other factor dependent on k for this series; we just take the first $\mathcal{F}(k)$ elements for the $2^k \times 2^k$ torus. Element r of this sequence is the number of occurrences of the rth cost in the cost sequence.

We have not developed a formula which generates the rth element of either sequence. We have made a further observation, however. We described a cost sequence for the $2^k \times 2^k$ torus as 2^{k-1} added to the first $\mathcal{F}(k)$ elements of the 'base' sequence

```
0,1,2,4,5,8,9,10,...,          (*)
```

which is composed of the pieces `(0),(1),(2),(4,5),(8,9,10),....` We note that the rth piece of the base sequence, $r > 0$, is 2^{r-1} added to the first $\mathcal{F}(r)$ elements of the base sequence, the zeroth piece being `(0)`.

3.2.5 Other Methods for Producing Torus Schemes

The product and elimination methods cannot produce optimal schemes for all toruses. The first example we noted was the 5×5 torus. Our optimal 5×5 scheme can itself be used to produce larger schemes. We can simply use it to *tile* a larger torus, and then connect the originators of the tiles by line calls. For example, we can produce a scheme for the 10 × 10 torus. A tiling of that torus is shown in Fig. 3.9. The wrap-around is now flattened out in each tile; it is easy to see that the left and right sides of the entire tiling will fit properly. It is also easy to see that it will take 3 calls of extra length 4 to join the 4 originators of the tiles. The tiles contribute no extra length, so the total extra length of the resulting 10 × 10 scheme will be 12. We note that $\lceil \log(10 \cdot 10) \rceil < \lceil \log 10 \rceil + \lceil \log 10 \rceil$, so we would not attempt to use the elimination method on the 16 × 16 torus to produce a scheme for the 10 × 10 torus.

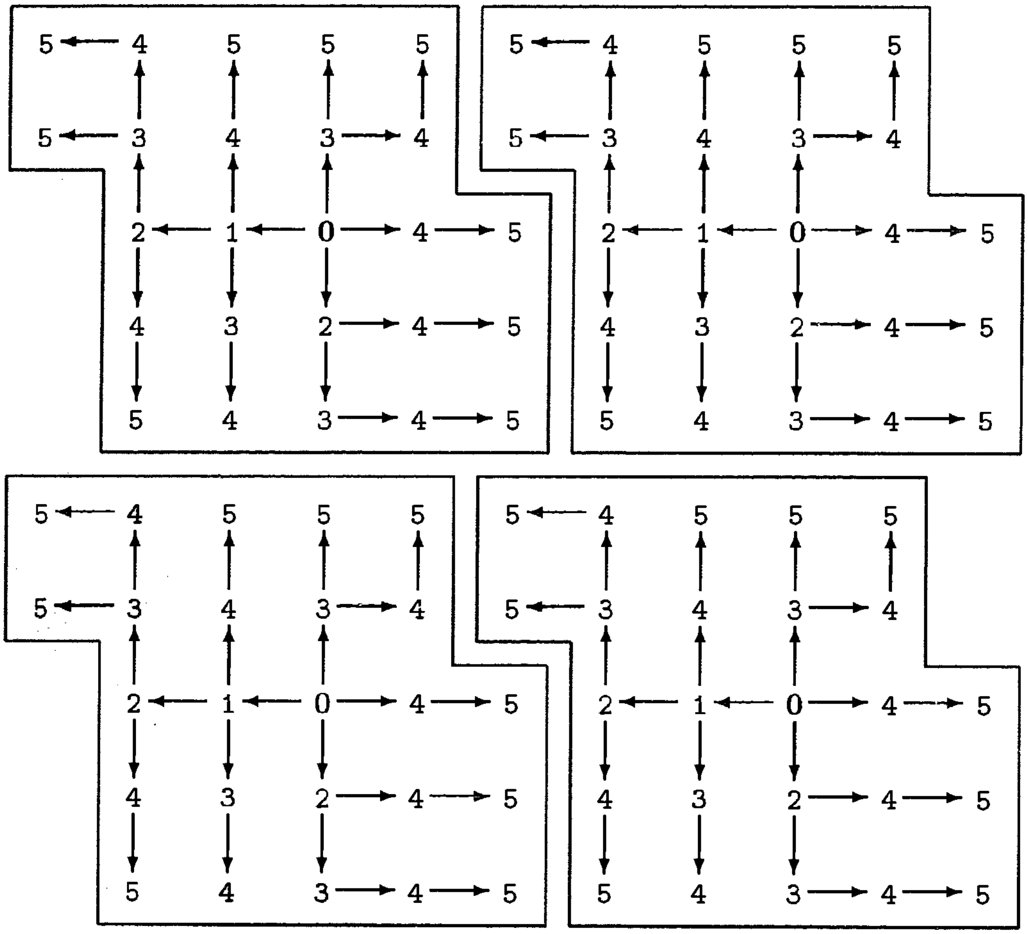


Figure 3.9: Tiling the 10 × 10 torus

We could use any suitable scheme to tile a torus; we could use the 10 × 10 scheme itself as a tile. We might also use an optimal scheme for say, the 7 × 7 or 7 × 11 torus, which could perhaps not be produced by the product and elimination methods, to tile another torus. We note that we should not 'have to' use the 6×6 scheme to tile the 12×12 scheme, since $\lceil \log(12 \cdot 12) \rceil = \lceil \log 12 \rceil + \lceil \log 12 \rceil$; we *can* use the elimination method on the 16×16 torus to produce a scheme for the 12×12 torus. Similarly, $\lceil \log(24 \cdot 24) \rceil = \lceil \log 24 \rceil + \lceil \log 24 \rceil$, so elimination on the 32×32 torus produces a scheme for the 24×24 torus. However, $\lceil \log(20 \cdot 20) \rceil < \lceil \log 32 \rceil + \lceil \log 32 \rceil$ so the 10×10 scheme could tile the 20×20 torus while the elimination method would not work.

Finally, we note that the generalization of the restriction in Equation (3.2) becomes more severe for δ-dimensional cycles as δ increases. That is so because δ-dimensional 'volume' changes quickly as a function of factor 'length'. That is, eliminating only a small fraction of the 'rows' in each dimension can reduce the total number of nodes by more than $\frac{1}{2}$.

Chapter 4

Minimum Line Broadcast Graphs

4.1 Overview

In this section we discuss an approach, inspired by the idea of *minimum broadcast graphs* (MBGs) (see [2], [20],) to investigating what we can do with a given extra length. A minimum broadcast graph is a graph with n vertices in which we can complete local broadcast from any originator in $\lceil \log n \rceil$ time units, and which has the minimum possible number of edges. In general, it seems to be extremely difficult to find an MBG for an arbitrary n, while for some values of n, the result is straightforward; for example, any k-cube is an MBG with 2^k vertices. If we allow line calls, we may be able to complete broadcast in minimum time in a graph with n nodes which has fewer edges than an MBG with n nodes. Of course, if the MBG with n nodes is a tree, then *no* graph with n nodes in which we can complete any broadcast has fewer edges than the MBG. Morever, we already know from Farley's result, mentioned in section 1, that given enough extra length, we can always complete line broadcasting in minimum time in a tree with n nodes. What we may ask is: how few edges we can have in a graph with n nodes, and still be able to complete line broadcasting in minimum time using only some given extra length?

4.2 Minimum Line Broadcast Graphs

Definition 5 (Minimum line broadcast graph) *A minimum line broadcast graph (MLBG) with n nodes and total extra length L is a graph $G = (V, E)$ with $|V| = n$, in which we can complete line broadcasting in G from any $v \in V$ in $\lceil \log n \rceil$ time units using $\leq L$ total extra length, and such that there is no graph $A = (V, E_2)$ with $|E_2| < |E|$ in which we can complete line broadcasting from any $v \in V$ in $\lceil \log n \rceil$ time units using $\leq L$ total extra length.*

For a given n and total extra length L, there may be a set of MLBGs; let us refer here to the set as $\mu(n, L)$. We already have a member $\mu(n, 0)$ for some values of n; that member is the known MBG with n nodes. For any n and for large enough L, Farley's result gives us a member of $\mu(n, L)$; that member can be P_n, the path with n nodes, or any tree with n nodes. In fact, for any n, there are only a finite number of values of L which are of interest, starting at 0 and going up to a value large enough for Farley's result to apply. If we refer to the value of L at which Farley's result applies for a graph with n nodes as $F(n)$, then $\mu(n, L)$, $L \geq F(n)$, contains only trees with n nodes. It seems reasonably obvious that if extra length L suffices for an end (leaf) node of P_n, then L will suffice for any tree with n nodes. The proof should involve a simple exchange argument. We state with confidence then that $F(n)$ is simply the total extra length of an optimal scheme for an end node of P_n.

$F(n) = 0$ for $n \leq 3$, as we can see in Figure 4.1, which shows MLBGs for $n \leq 3$ and minimum time local broadcast schemes for all distinct (up to isomorphism) originators. It is known already that any MBG with 4 nodes contains 4 edges; the 2-cube is an MBG for $n = 4$, as shown if Figure 4.2, part (a). So $F(4) > 0$, and $\mu(4, 0)$ contains the 4-cube. $F(4)$ is, in fact, 1; figure 4.2 part (b) shows that $P_4 \in \mu(4, 1)$ by showing schemes for all originators (up to isomorphism.) Farley has shown in [9] that for $n > 3$, no tree with n nodes is a minimum time local broadcast graph, so $F(n) > 0$ for all $n > 3$.

It is easy to show that $F(5) = 1$ by drawing a scheme for the end node of P_5 that uses only $L = 1$, and noting that $L = 0$ will not suffice, since P_5 is a tree with more than 3 nodes. For $n = 2^k$, we can easily suggest what is probably an optimal scheme

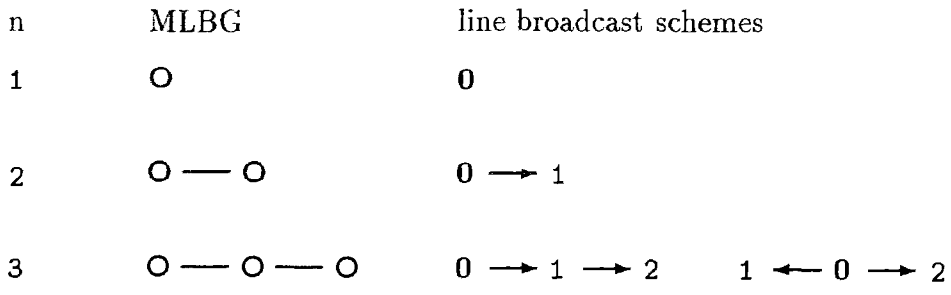
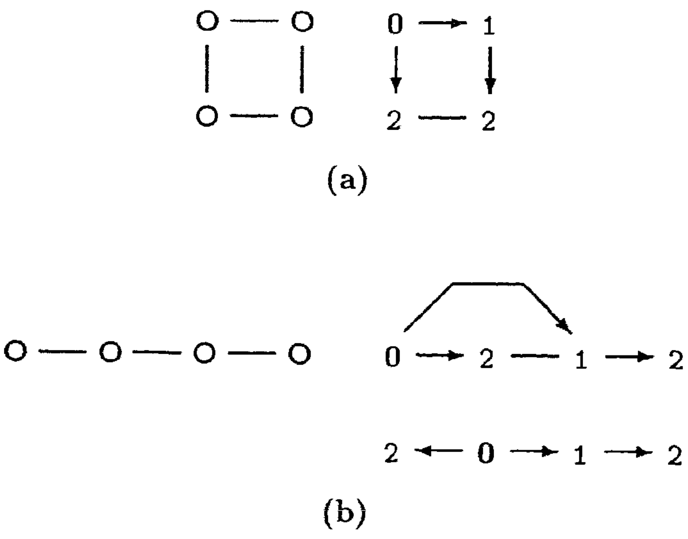


Figure 4.1: $F(n) = 0$ for $n \leq 3$



(a)

(b)

Figure 4.2: $F(4) = 1$

for the end node of P_n. The total extra length of the scheme clearly serves as an upper bound on $F(n)$ for $2^{k-1} < n < 2^k$. Since we will not *prove* that the scheme is optimal, we will call the total extra length of the scheme $F_u(k)$ to indicate that it is technically only an upper bound on $F(n)$, and that it is a function of k, not n directly. When $k = 1$ ($n = 2$), we require no extra length. For $k = 2$, we can use extra length 1; the end node calls the node 2 edges away, and effectively splits the path into two paths of length 2. We generalize this scheme for $k > 2$ in the obvious fashion. In general, if our upper bound for k is $F_u(k)$, then

$$F_u(k) = \begin{cases} 0, & k = 1 \\ 2^k/2 - 1 + 2 \cdot F_u(k-1), & k > 1 \end{cases}$$

The solution to this recurrence relation is

$$F_u(k) = 2^{k-1} \cdot (k-2) + 1. \tag{4.1}$$

We may also want to know what value of L is just large enough that *some* tree with n nodes is in $\mu(n, L)$; let us call that value $f(n)$. We know that $f(6) > 0$ since $6 > 3$. If follows from Figure 4.3, then, that $f(6) = 1$; that figure shows that a *minimum broadcast tree* (MBT) with 6 nodes is in $\mu(6, 1)$. A minimum broadcast tree is a broadcast tree in which one or two nodes in the tree can originate a minimum time local broadcast; those one or two nodes form the *(local) broadcast center* of the tree. If there are two nodes in the broadcast center then they are neighbors in the tree. Binomial broadcast trees are MBTs; a binomial broadcast tree with 2^{k+1} nodes can be split into two identical MBTs (also binomial broadcast trees) with 2^k nodes by removing the edge joining the two nodes in the broadcast center. Similarly, we can show that $f(7) = 1$. We can also conclude that $f(8) \leq 3$; Figure 4.4 shows that the MBT with 8 nodes is in $\mu(8, 3)$.

In [9], Farley proposed a scheme for broadcasting from any node in an MBT. We illustrate the procedure by describing a scheme for the worst-situated leaf of an MBT with 2^k nodes, which is simply the familiar binomial tree with 2^k nodes. In an MBT there are exactly two nodes capable of originating a minimum-time local broadcast; they are neighbors in the tree and by removing the edge between them we obtain two

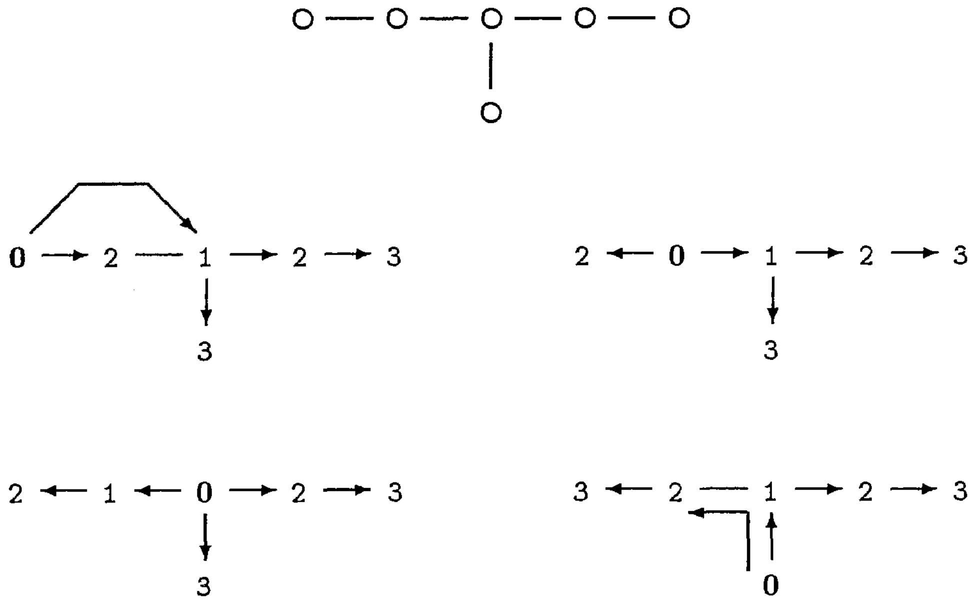


Figure 4.3: $F(6) = 1$

MBTs with 2^{k-1} nodes. The leaf simply calls the more distant of those two nodes; that more distant node then originates a minimum-time local broadcast in its subtree with 2^{k-1} nodes, while the leaf repeats the entire procedure recursively in the other subtree; it is also the worst-situated leaf in that subtree. It is easy to show that the total extra length of this scheme is an upper bound on the total extra length for any node in any MBT with n nodes, $2^{k-1} < n \leq 2^k$. Farley gives the upper bound as

$$m(k) = \frac{1}{2}k(k-1). \tag{4.2}$$

Comparing this result to (4.1), we can see that $m(k)$ is asymptotically much smaller than $F_u(k)$.

We may wonder if there is a better choice of tree than the MBT, perhaps one of much lower diameter. The lowest diameter tree is STAR$_n$, the star with n nodes; it has one central node and $n-1$ isomorphic leaves and is of diameter 2. We consider the scheme for any leaf of STAR(k), which we define as the star with 2^k nodes. It is fairly obvious that the leaf should call the central node so that the central node may make $k-1$ local calls. All calls are either of length 1 or length 2, so using the

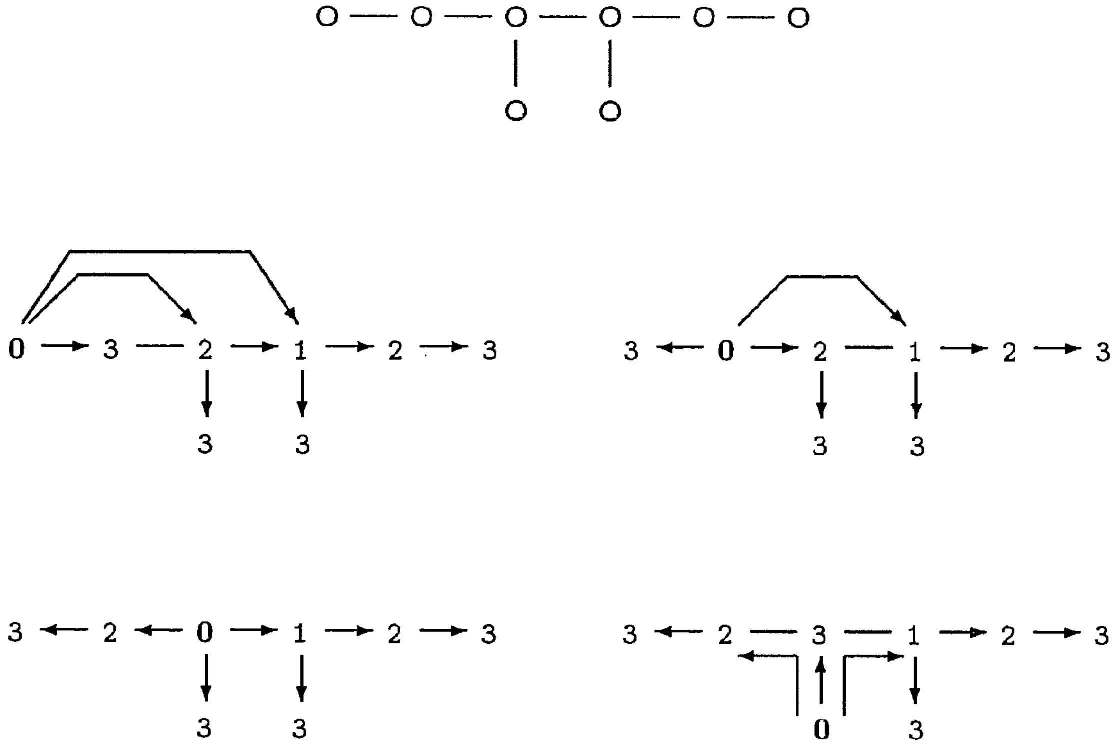


Figure 4.4: $F(8) \leq 3$

maximum number of local calls decreases the total extra length as much as possible. All calls of length 2 will be between leaf nodes and go through the central node; note that it is actually impossible to specify a scheme which violates edge-disjointedness since an edge could only be used twice if one node were involved in 2 calls at once. The analysis is easy, then. There are $k-1$ calls of length 1 and $(2^k-1)-(k-1)$ calls of length 2. So the extra length required by STAR(k) is

$$s(k) = 2^k - k \tag{4.3}$$

Comparing this result to (4.2) we see that $s(k)$ is asymptotically much worse than $m(k)$. On the basis of what we have described, and after further investigation, we are led to the following conjecture:

Conjecture 3 *Let $f(n)$ be that value of extra length L which is just large enough that there exists a tree with n nodes in $\mu(n, L)$. For $n = 2^k$, $f(n) = \frac{1}{2}k(k-1)$ and the tree is the MBT with 2^k nodes.*

As we have said, we have a member of $\mu(n, 0)$ for many values of n (the MBG with n nodes.) We have also a member of $\mu(n, F(n))$ (ie., P_n) and we know that $F(n) > 1$ for $n > 3$. It may also be of interest to find members of $\mu(n, L)$, for $0 < L < F(n)$, because, given a graph $G = (V, E)$ and graph $G_m = (V, E_m) \in \mu(|V|, L)$, we know that if $|E| < |E_m|$ then there is no line broadcast scheme for G which uses $\leq L$ extra length. Let $n = |V|$. If we define $ES(n, L)$ as $|E_m|$ for $G_m = (V, E_m) \in \mu(n, L)$ then we could think of this investigation as a matter of filling in a table of values of $ES(n, L)$ in which the rows are labelled with values of n and the columns are labelled with values of L; perhaps we might represent an entry for some $ES(n, L)$ by an example member of $\mu(n, L)$. Figure 4.5 shows part of the table, including some of the entries we have described so far. A blank entry in row n, column L means that the $L > F(n)$, and T$_n$ means any tree with n nodes, P_n for example. MBT$_n$ means an MBT with n nodes.

Figure 4.6 shows the schemes for both originators (up to isomorphism) for the representative of $\mu(8, 1)$ shown in the table.

We appear to have systematic descriptions of three parts of the table already:

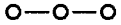
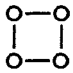
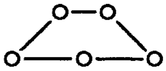
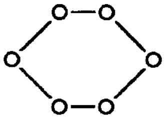
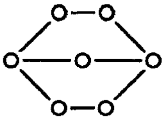
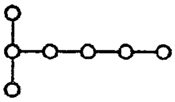
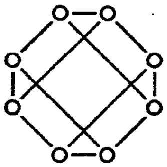
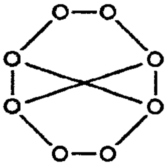
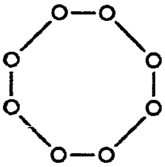
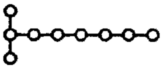
n \ L	0	1	2	3	4	5
3						
4		T_4				
5		T_5				
6		MBT_6	T_6			
7		MBT_7		T_7		
8				MBT_8		T_8

Figure 4.5: $ES(n, L)$, $3 \le n \le 8$, $0 \le L \le 3$

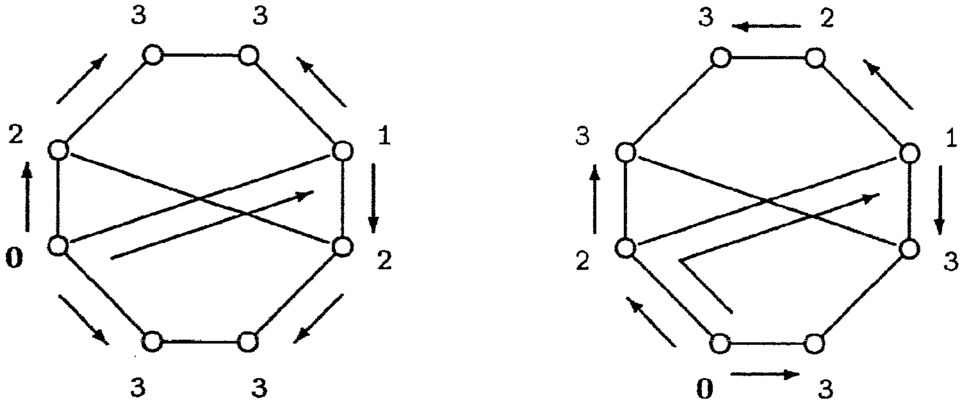


Figure 4.6: Schemes for a member of $\mu(8, 1)$

1. The first column.
2. the 'path' in the table formed by the last entry of interest in each row.
3. The path formed by the MBT entries.

We can also add another path of interest, the path formed by the cycle entries. We see that $\mu(4,0)$, $\mu(5,0)$, $\mu(6,0)$ and $\mu(8,2)$ contain cycles. Because we know the optimal total extra length $\epsilon(n)$ (from Section 2.4) of an n-cycle scheme, we know that the column for $L = \epsilon(n)$ is the first column (starting from $L = 0$) which contains a cycle.

We are likely to observe a property of $ES(n, L)$ which is similar to one we observe of $B(n)$, the analogous function for local broadcasting, described in [2] and [20]. The property might be called the 'slack' property of the function, and it results from the '$\lceil\ \rceil$' in '$\lceil \log n \rceil$', the formula describing the minimum number of time units required to complete broadcast in a graph with n nodes. When n is slightly larger than some 2^k, we make only a few calls, relative to the number of informed nodes, in the last time unit; we would expect it to be easier to find graphs in which we can complete the broadcast in minimum time, now that we have an entire extra time unit and only a few more nodes to inform in that time unit. We might, then, expect $F(2^k)$ to be more than $F(n)$ for n slightly more than 2^k, and we might expect $F(n)$ to rise as n increases from 2^k to 2^{k+1}. The portion of the table shown in Figure 4.5 is too small to illustrate this expected property.

4.3 Variants on MLBGs

We may wish to define restricted variants of MLBGs. In [19], it is pointed out that a practical restriction on broadcast graphs is that of bounding the degree of vertices in the graph; the paper discusses finding 'sparse graphs' of bounded degree in which local broadcast may be completed 'quickly', if not in minimum time. We may also wish to examine this restriction in the context of line broadcasting; we may look for degree-bounded graphs with n nodes which have the fewest possible edges and in which we may complete line broadcasting from any originator in minimum time using less than

L extra length. Restricting our search to graphs with n nodes with a given degree bound may make it impossible to complete local broadcast in the graph in minimum time. When we may make line calls, we can still always complete the broadcast in minimum time, but we may need to use more extra length than we would have had to without the degree bound.

An examination of Figures 4.3 and 4.4 suggests a further restriction we may wish to make in the case of line broadcasting, one which is not applicable to local broadcasting. We note in those figures that only some of the possible originators require the total extra length L available. (In fact, in these examples, only *some* of the originators require any extra length at all; that is to be expected for at least one originator since the MLBGs shown are MBTs.) We may wish to find those MLBGs for which the *average* extra length required is minimimized. By 'average' we might mean the sum over all originators, ignoring isomorphism, of the extra length required, divided by n, the number of possible originators; such a definition would lead to an expected total extra length of completing a minimum time line broadcast if all nodes are equally likely to be the originator. For the MLBG with 6 nodes shown in Figure 4.3, the average required extra length is

$$\begin{aligned}\frac{2(1)+2(0)+1(0)+1(1)}{6} &= \frac{2+0+0+1}{6}\\ &= \frac{1}{2},\end{aligned}$$

compared to L, which was 1. Similarly, for the MLBG with 8 nodes shown in Figure 4.4, the average required extra length is $\frac{12}{8}=\frac{3}{2}$, compared to L, which was 3.

Chapter 5

Further Work

5.1 Ordering of Call Lengths

For this section, we recall the definition of *call length set* given in Section 1.3. Finding optimal line solutions in a graph of interest would perhaps be simpler if we knew the following property to hold for the graph:

Definition 6 *Given a graph $G = (V, E)$ and an originator $v \in V$, we say that G and v have the line call ordering property if for every minimum broadcast scheme A on G and v in which some call made in some time unit is longer than some call made in some earlier time unit, there is another broadcast scheme which has the same call length set as A and in which no call made in any time unit is longer than any call made in any earlier time unit. We also say that G has the line call ordering property if the property holds for G and any $v \in V$.*

If this property holds for the graph of interest, then in doing the analysis, we would have the possible advantage of considering only those schemes in which calls are made in order of length, with each longer call being made either in an earlier time unit than or in the same time unit as any shorter call. In fact, it is easy to find graphs which do not have the line call ordering property. Consider the graph in Figure 5.1. In the left picture in the figure, the originator (labelled '0') *must* begin by making a local call, and then the originator *must* make a line call of length 2 if the broadcast is to be completed in minimum time. In the right picture, we see a scheme which uses 1 line call of length 2, and 2 local calls; its call length set is {1,1,2}. The scheme begins with a local call and the call of length 2 is made in the next phase. No minimum time schemes featuring the given originator in the right picture and having that call length set can avoid beginning with a local call. In fact, no scheme for *either* originator which has the call length set {1,1,2} can avoid beginning with a local call. If either originator begins a scheme which has the call length set {1,2,2}, then the scheme could begin with a line call, but we would presumably not want to consider such schemes, given the partial relation on call length sets described in Section 1.3. The graph shown in the figure is an example of a *star* graph, and in a star graph, the 'central' node can only make local calls, while the outer nodes can only make calls of length 2, and may have to do so if the broadcast is to be completed in minimum time.

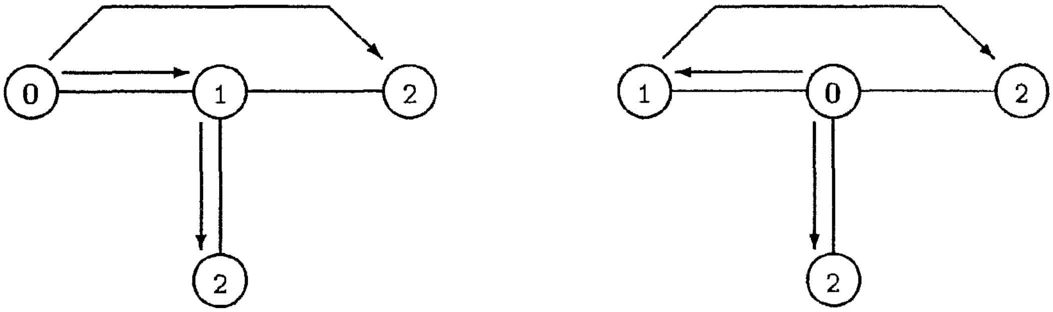


Figure 5.1: Line call ordering

We may speculate on what properties a graph must have to make the line call ordering property hold. One possibility is that the connectedness of the graph determines whether or not the line call ordering property holds. Also, we note that our example graphs for which the property does not hold are not vertex-transitive. Our experience in working with cycles and toruses, both of which are vertex-transitive, leads us to conjecture the following:

Conjecture 4 *Any vertex-transitive graph has the line call ordering property.*

If this conjecture is correct, it could simplify the search for a lower bound on total length required in the torus.

5.2 Generalizations of the Cycle Properties

In general graphs, there are no simple analogs of nestedness, flatness and fullness, although weak generalizations of these properties exist and can be shown to be necessary for optimality. In the cycle, optimal line broadcasting turned out to be path broadcasting; in higher-degree graphs the fact that one node can switch-through multiple calls deprives us of this simplification. A generalization of nestedness is that an informed node should always be making a call if calls are going through it; otherwise, it should be the sender of one of those calls, thus making that call shorter. One interesting line of further work might be line broadcasting in degree-3-regular or degree-3-bounded graphs. At a degree 3 node, at most one call can be going through the node, but the node may be originating a call at that time.

5.3 Lower Bounds in the Torus

We have had little success in finding a tight lower bound for the torus, despite Conjecture (2). We have searched the literature for generalizations of Iordanskiǐ's work. A *pairwise* numbering of the vertices in a tree would correspond to an embedding of the tree into an infinite grid graph. The total length of the embedding is the sum over each edge in the tree of the 'cost' of the edge. The cost of an edge is the difference in the first components of the labels of the endpoints of the edge, added to the difference of the second components. Obvious generalizations exist for general δ-dimensional cycles. A geometric approach is another possibility; we could simply try to find the optimal connection into a broadcast tree of n nodes having integer x and y coordinates in the plane, using straight lines. The closest reference we could find to such work is [8], although the vast literature on topological embeddings might provide better leads.

Another approach might simply involve trying to find a proof similar in nature to the cycle proof. We would try to find a set of important properties of optimal schemes, or, of optimal embeddings. We might find exchange arguments which allow us to narrow our search to a more easily studied subset of all possible schemes. For example, it might be possible to prove that for every scheme which uses 'bent' (with the obvious meaning) calls, there is another scheme which is no more expensive and which uses only 'straight' calls. In conjunction with Conjecture (4), we obtain a subset of all possible schemes of which subset the product method schemes seem to be significant members.

Finally, a brute force approach is simply to try to minimize the total length of the embedding of a binomial tree into an infinite grid graph. We have no insight into how difficult this task might be, although we suspect that it is very hard.

Bibliography

[1] J.-C. Bermond, P. Fraigniaud, and J.G. Peters. Antepenultimate Broadcasting. Technical Report TR 92-03, School of Computing Science, Simon Fraser Univ., 1992.

[2] J-C. Bermond, P. Hell, A. L. Liestman and J. G. Peters. Sparse Broadcast Graphs, Disc. Appl. Math. 36 (1992), pp. 97-130.

[3] J-C. Bermond and C. Peyrat. The de Bruijn and Kautz networks: a competition for the hypercube, *in Hypercube and Distributed Computers* , F. Andre and J. Verjus, eds., North-Holland, pp. 279-294, 1989.

[4] J-C. Bermond and C. Peyrat. Broadcasting in de Bruijn Networks. *Proc. 19th S-E Conference on Combinatorics* , Congressus Numerantium 66, pp. 283 - 292, 1988.

[5] C. Almstrom. Limited Line Broadcasting in the Infinite 1-dimensional Grid Graph. Manuscript, 1988.

[6] E.J. Cockayne and S.T. Hedetniemi. A Conjecture Concerning Broadcasting in M-Dimensional Grid Graphs. Technical Report CS-TR-78-14, University of Oregon, 1978.

[7] N. Deo, J. Nievergelt and E.M. Reingold. *Combinatorial Algorithms: Theory and Practice*, Prentice-Hall, Englewood Cliffs, New Jersey, 1977, pp. 173-176.

[8] E. N. Gilbert. Random Minimal Trees. *J. Soc. Indust. Appl. Math.* Vol. 13, No. 2, pp. 376-387, June 1965.

[9] A. M. Farley. Minimum-Time Line Broadcast Networks. *Networks* 10, pp. 59-70, 1980.

[10] A.M. Farley and S.T. Hedetniemi. Broadcasting in Grid Graphs. *Proc. 9th S-E Conf. Combinatorics, Graph Theory, and Computing*, Utilitas Mathematica, pp. 275-288, 1978.

[11] R. Feldmann, J. Hromkovic, S. Madhavapeddy, B. Monien, and P. Mysliwietz. Optimal Algorithms for Dissemination of Information in Generalized Communication Modes. Report Nr. 115, Dept. of Mathematics and Computer Science, Univ. of Paderborn, 1993.

[12] P. Fraigniaud and E. Lazard. Methods and Problems of Communication in Usual Networks. *Discr. Appl. Math.*, to appear.

[13] M. R. Garey and D. S. Johnson. *Computers and Intractability - a Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, New York, 1979.

[14] M.C. Heydemann, J. Opatrny and D. Sotteau. Broadcasting and spanning trees in de Bruijn and Kautz networks. Disc. Appl. Math. 37/38 (1992), pp. 297-317.

[15] M.A. Iordanskiǐ. Minimal Numberings of the Vertices of Trees. *Soviet Math. Dokl.* 15(1974), pp. 1311–1315.

[16] B. Monien and H. Sudborough. Embedding one Interconnection Network into Another. *Computing Suppl.* 7(1990), pp. 257-282.

[17] J.G. Peters and M. Syska. Circuit-Switched Broadcasting in Torus Networks. *Technical Report TR 93-04*, School of Computing Science, Simon Fraser Univ., 1993.

[18] S. T. Hedetniemi, S. M. Hedetniemi, and A. L. Liestman. A Survey of Broadcasting and Gossiping in Communication Networks, *Networks* 18, pp. 319-349, 1988.

[19] A. L. Liestman and J. G. Peters. Broadcast Networks of Bounded Degree. *SIAM J. Disc. Math.* Vol. 1, No. 4, pp. 531-540, November 1988.

[20] A. L. Liestman and J. G. Peters. Minimum Broadcast Digraphs. Disc. Appl. Math. 37/38 (1992), pp. 401-419.