2008

Broadcasting in cycles with chords

Lisa L. Kovalchick
West Virginia University

Broadcasting in Cycles with Chords

Lisa L. Kovalchick

Dissertation submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy
in
Computer and Information Science

Frances L. Van Scoy, Ph.D., Chair
John Atkins, Ph.D.
Y. V. Ramana Reddy, Ph.D.
George Trapp, Ph.D.
Laura Pyzdrowski, Ed.D.

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2008

Keywords: Broadcasting, Multiple Message Broadcasting, Line Broadcasting, Cycles with Chords

ABSTRACT

Broadcasting in Cycles with Chords

Lisa L. Kovalchick

Broadcasting is the process of information dissemination in which one node, the originator, knows a single piece of information and using a series of calls must inform every other node in the network of this information. We assume that at any given time, a node can communicate the message to another node, with which it shares an edge, by acting as either a sender or receiver, but not both. Multiple message broadcasting considers the case when the originator has m messages, where $m > 1$, to disseminate. Whereas broadcasting limits the communication of a message from one node to another node via a single edge, line broadcasting allows one node to send a message to any other node in the network as long as a simple path exists between the sending node and the receiving node and every edge along the path is not in use.

In this dissertation, we consider the problem of broadcasting in a cycle with chords and we develop broadcast schemes for this type of network.

We begin by investigating the problem of broadcasting in a cycle with one and two chords, respectively. Then, we consider the problem of multiple message broadcasting in cycles with one and two chords. Finally, we consider the problem of line broadcasting in cycles with chords.

Through our investigations, we develop two algorithms for the problem of broadcasting in a cycle with one and two chords, respectively and we analyze the correctness and complexity of these algorithms. Then, we discuss problems associated with multiple message broadcasting in cycles with one and two chords. Finally, we use techniques developed for line broadcasting in cycles to create minimum time broadcast schemes for cycles through the addition of chords.

Using techniques developed in this dissertation, we are able to broadcast in minimum time in cycles with chords. In cycles whose size is a power of 2, we have proved that the number of chords that we add to the cycle is the minimum number of chords required to broadcast in minimum time in such a cycle.

ACKNOWLEDGEMENTS

I wish to thank my graduate advisor, Dr. Frances Van Scoy, who gave me the opportunity to work with her and guided me through my schooling. Dr. John Atkins, Dr. Ramana Reddy, Dr. George Trapp, and Dr. Laura Pyzdrowski are greatly appreciated for serving on my Graduate Committee. Finally, I wish to thank my parents, James and Dorothy Kovalchick and my sister, Amanda Kovalchick for their support and encouragement.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Preliminaries

Broadcasting is the process of information dissemination in which one node, the originator, knows one or more pieces of information and using a series of calls must inform every other node in the network of this information. A call is defined as the movement of information during a single time interval between two adjacent nodes (neighbors). We assume that time is discrete, meaning that any calls made within the same time unit occur simultaneously. Many different broadcast models are based on the above definition; [HHL88] provides a comprehensive investigation of these models. The model of broadcasting that we will be using adds two additional constraints.

1. A node can be involved in at most one call per time unit.

2. The originator will have only a single message to disseminate.

We use a connected undirected graph, $G = (V, E)$, to model the communication network, where V represents the set of n nodes and E represents the set of edges (i.e., communication links) between pairs of nodes. A broadcast algorithm is defined as a sequence of calls with the following constraints.

1. A message must arrive at a node before that node can pass the message on to another node.

2. At any given time unit, any node may act as either a sender or receiver of a message, but not both.

3. At the end of a broadcast algorithm, every node in the network has received the message.

Given a connected undirected graph $G = (V, E)$ and an originator, node u, we define the broadcast time of node u, $b(u)$, to be the minimum number of time units required to complete broadcasting from node u. We also define the broadcast time of the graph G, $b(G)$, to be the maximum broadcast time of any node u in G.

1.2 Formal Definition

Let G be a graph that has C_n as a subgraph, where C_n contains n nodes and n edges such that $n > 2$ and, let there be one or more chords (edges not in C_n whose endpoints lie in C_n) connecting pairs of nodes in the C_n, can we develop a broadcast scheme for such a graph?

Figure (1.1) depicts a cycle containing 6 nodes with a single chord, c, dividing the cycle in half.

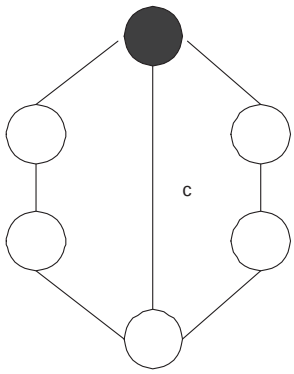

Figure 1.1: A cycle containing 6 nodes with a single chord, c.

Since we are working with an undirected graph, the number of unique chords of a cycle with n nodes must range between 0 and $\frac{n\cdot(n-3)}{2}$, inclusive for $n \geq 3$. A cycle with $\frac{n\cdot(n-3)}{2}$ chords is merely the complete graph on n nodes, K_n. [HHL88] reports that the time to broadcast in K_n is equal to $\lceil \log_2 n \rceil$; an algorithm already exists that meets this time bound. However, the problem of broadcasting in a cycle with fewer than $\frac{n\cdot(n-3)}{2}$ chords remained an open problem, until now.

1.3 Motivation

The current push to obtain faster algorithms for many scientific problems often results in the use of either parallel or distributed computing. The running time of algorithms which employ parallel or distributed computing is often determined by the communication time between processors. If it is possible to lower the amount of time taken for processor communication, it is often possible to decrease the running time of such algorithms. It is for this reason that broadcasting algorithms are an important part of current computing. While the problem of broadcasting a single message throughout a network is well understood for several classes of graphs including grids [FH78], cycles [S99, W99], trees [SCH81], etc., until now, no general solution existed for cycles with

chords. In this dissertation, we present algorithms for broadcasting in cycles with one and two chords, respectively, and analyze the correctness and complexity of these algorithms. Then, we discuss the problems associated with multiple message broadcasting in cycles containing one and two chords. Finally, we use techniques developed for line broadcasting in cycles to create minimum time broadcast schemes for cycles through the addition of chords.

Chapter 2

History

2.1 Previous Results

The problem of broadcasting was first formulated in 1977, by Slater, Cockayne and Hedetniemi who studied the amount of time necessary for a single person to share a single piece of information with everyone else in a network. As a result of this work on broadcasting, a whole new area of research developed.

As mentioned in Chapter 1, the broadcast time of node u, $b(u)$, is defined as the minimum number of time units required to complete broadcasting from a given node u. When broadcasting begins, only a single node, u, has the message and, according to the constraints, the number of informed nodes can at most double at each step (i.e., each informed member communicates with an uninformed member at each time step), thus $b(u) \geq \lceil \log_2 n \rceil$. Obviously, broadcasting can be completed in minimum time in a complete graph (i.e., $b(u) = \lceil \log_2 n \rceil$ for K_n); although, when the number of nodes, n, is large the complete graph requires too many communication links (edges) in order to be practical. This is one of the reasons that researchers have studied many other types of

graphs in order to find a compromise between broadcast time and the number of communication links required.

One of the first broadcast problems to be studied involved finding the minimum number of communication links required in order to complete broadcasting in minimum time. In other words, if we start with the complete graph on n nodes, K_n, we are interested in the maximum number of communication links that can be removed while still being able to broadcast from any processor in minimum time. In [JG79], Johnson and Garey showed that the problem of determining whether $b(v) \geq k$ for a node v in an arbitrary graph for fixed $k \geq 4$ is NP-Complete. As a result, several authors studied methods for constructing graphs with small numbers of edges in which broadcasting can be completed from any node in minimum time. For example, [SW84] developed a dynamic programming formulation for optimal broadcasting in general networks, they also give an exact algorithm based on its development. [F79] and [FHMP78] are among the authors that have discovered graphs with the minimum number of edges possible to complete broadcasting in minimum time.

It is not always necessary to complete broadcasting in minimum time. Obviously, the performance of a parallel computer relies upon the amount of time necessary to complete broadcasting; although, there may be cases when the cost of constructing such a computer is slightly more important than achieving the very best performance. Redundant paths, extendability (i.e., the difficulty and expense of adding nodes to the communication network) and the diameter of the communication network are all issues that involve improving the performance of a computer. On the other hand, fixed degree, simple construction and a small number of wires are issues that help to lower the cost of a computer. For example, in the real world, having a bounded maximum degree (i.e., a maximum number of communication links connecting a node to its neighbors) for each node may

be more important than getting the very best performance for specific situations. Thus, broadcasting has been well studied for many classes of graphs, including trees, grids and cycles.

Obviously, a graph must be connected in order to complete broadcasting; otherwise, it would be impossible to inform every node in the graph. A tree is a graph that contains the minimum number of edges possible in order for the graph to be connected (i.e., a tree with n nodes has $n-1$ communication links). Thus, a tree is the cheapest communication network based on the number of communication links required. [SCH81] studied broadcasting in trees. Specifically, they present an algorithm which determines the broadcast time in a tree. This algorithm runs in linear time and actually finds the broadcast center of the tree (i.e., the set of all nodes from which broadcasting can be completed in the least amount of time). [P81] studied minimum broadcast trees (mbts), which are defined as a special class of rooted trees that allow broadcasting from the root to all other nodes of the tree in minimum time (over all rooted trees with n nodes). In addition, they give an algorithm which decides membership in the class, another algorithm to construct all mbts with a given number of nodes and a recursive formula to count these trees.

A grid is defined as an n x m undirected graph consisting of n rows and m columns of nodes. Each node in a grid has communication links to four other nodes (its neighbors), except nodes which form the border of the grid, which have either two or three neighbors (see Figure (2.1)). Grids are one of several common topologies used when constructing parallel computers. Reasons to choose a grid topology emphasize the performance of a computer. We define the diameter of a graph as the number of edges in the maximum shortest path over all pairs of nodes. Grids have a relatively small diameter, which helps improve communication time. For example, the diameter of a 2-dimensional grid, $G_{m,n}$, containing m rows and n columns is $m+n-2$. Grids are also extendable (i.e., it is relatively easy to add processors to the network at a later time) and

they provide redundant paths, which may be useful to improve communication times. Thus, it is important to study broadcasting in such a graph.

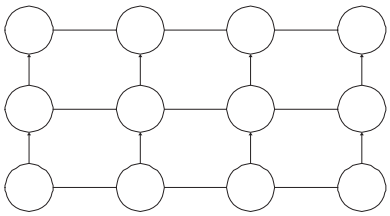

Figure 2.1: A 3 x 4 grid.

Broadcasting in grid graphs was first presented in [FH78]. They presented broadcast times for an n x m grid graph which relied on the location of the originator of the message (i.e., a corner node, a side node, or an interior node). They also looked at broadcast times required for wrap-around grids and ILLIAC-grids. Finally, they looked at broadcasting in an infinite 2-dimensional grid, which gave rise to several other papers including [K79] and [P80].

A cycle is another popular model used when building parallel computers. A cycle is an undirected graph on n nodes containing n edges; nodes in a cycle have the special property that every node has degree 2 (i.e., is connected to exactly 2 other nodes). When building a parallel computer using a cycle topology, few communication links are needed, which reduces the number of wires necessary. A cycle also cuts cost by providing easy construction, since it simply involves connecting processors in a ring-like fashion. In such a graph, the only realistic method for broadcasting is for the originator to send the message to one of its neighbors and for each informed node to send the message to an uninformed neighbor (if such a neighbor exists) at each successive step. See Figure (2.2), for an example of broadcasting in a cycle containing 6 nodes; in this example, the node colored black is the originator.

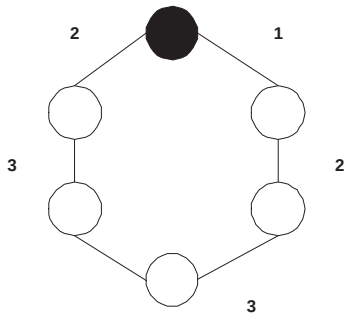

Figure 2.2: Broadcasting in a cycle containing 6 nodes.

Using this scheme it is obvious that the time to broadcast from any given node in a cycle is:

$$\begin{aligned} \lfloor \frac{n}{2} \rfloor &= D \qquad \text{if } n \text{ is even,} \\ &= D+1 \qquad \text{otherwise,} \end{aligned}$$

where D represents the diameter of the graph. Although using a cycle topology reduces the cost of the computer, it usually does not provide the performance necessary. If we are able to develop efficient schemes for broadcasting in a cycle with chords, we may be able to reach a compromise between cost and performance which can be useful to industry.

2.2 Related Work

Along with single message broadcasting, as defined in Chapter 1, many other broadcast models have been studied. Several surveys have been done in this area including [HHL88, FL94] and [HKMP96]. Some of the more common models include reliable broadcasting, broadcasting using the postal model, multiple message broadcasting and line broadcasting.

Reliable broadcasting is concerned with the idea of broadcasting in the presence of faults. A fault is defined as the failure of one or more communication lines (edges) or the failure of one

or more communication sites (nodes). The study of reliable broadcasting is important for several reasons including allowing for network maintenance to be done in a way that would reduce the frequency of non-isolated network failures and making it easier to determine the effects of making a site or line inoperable in order for maintenance to occur. Some networks in which reliable broadcasting schemes have been studied include stars [MBHI96, GRV98], meshes [JW01] and hypercubes [GLPR94, WGLM00].

As an example of the postal model, consider people living in the same town who communicate by writing letters to one another and dropping these letters in the mailbox. Once a letter is placed in the mailbox, the sender forgets about the letter assuming that it will be delivered to the recipient in a reasonable amount of time. The sender is not concerned with the route that the letter travels and may send several letters, before receiving delivery notification of the first letter. Broadcasting using the postal model allows one to ignore the actual underlying connection network and to assume that the underlying network is a complete graph with message latencies which represent the amount of time that passes between the time a message is sent and the time that the message is received. This model was introduced in order to study parallel and distributed systems which use packet switching networks as opposed to circuit switching. [BK92, M92] and [GS99] are just a few of the researchers who have studied the postal model.

When working with parallel or distributed architectures, sometimes, it may be more beneficial to break a message into pieces and to send each piece separately. For example, this could occur when the cost of sending a message is proportional to the size of the message. Often, broadcasting schemes that work well for a single message are inefficient when used for broadcasting multiple messages. Thus, multiple message broadcasting has been studied. Some examples in which multiple message broadcast schemes have been investigated include grids [W99, WV96, VB94], paths

[S99], hypercubes [QA97, S99] and complete graphs [KC95].

Circuit-switched networks are networks in which a dedicated path from sender to receiver is established for the duration of a call. Early telephone systems were an example of circuit-switched networks. In this type of telephone system, whenever a call is made between a sender and a receiver, the physical line that connects the two parties is dedicated to those parties only and cannot be used by anyone else for the duration of the call. This type of communication system can be studied using the line broadcasting model. Line broadcasting allows for "long distance" calls to be made between nodes. In line broadcasting, a node can call any other node as long as there is an available path between the two nodes. An available path is defined as any simple path in which none of the edges are involved in another call. Some examples in which line broadcasting schemes have been investigated include cycles [KP98], grids [W99] and complete binary trees [AGR01].

Chapter 3

Broadcasting in Cycles with a Single Chord

3.1 Background

In order for a cycle, C_n, to exist, the graph must contain at least 3 nodes. The most obvious algorithm for broadcasting in a cycle is for every informed node to pass the message along to one of its uninformed neighbors at each time step, if such a neighbor exists; we refer to this algorithm as the chordless cycle algorithm. Using the chordless cycle algorithm, cycles containing between 3 and 6 nodes (inclusive) can broadcast in minimum time (i.e., $b(C_n) = \lceil \log_2 n \rceil$, for $3 \leq n \leq 6$) as shown in Figure (3.1).

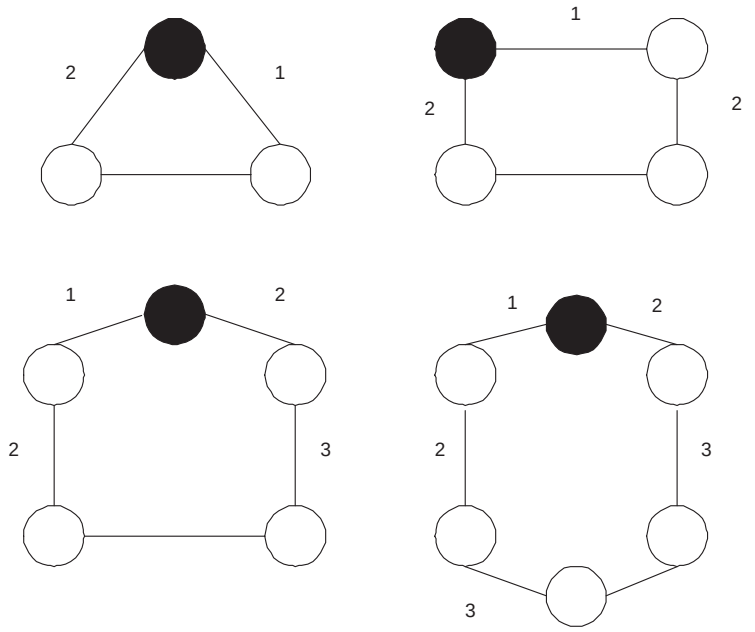

Figure 3.1: Broadcasting in minimum time in C_3, C_4, C_5 and C_6 (the originator nodes are colored black).

The time to broadcast in a cycle with an even number of nodes is equal to the diameter of the graph (i.e., $\lfloor \frac{n}{2} \rfloor$) and the time to broadcast in a cycle with an odd number of nodes is equal to the diameter of the graph plus one (i.e., $\lfloor \frac{n}{2} \rfloor + 1$). Both the function representing the time to broadcast in a cycle and the function representing the minimum broadcast time in a graph (i.e., $\lceil \log_2 n \rceil$) are monotonically increasing and for cycles containing more than 6 nodes (i.e., $n > 6$), the latter function grows slower. Thus, minimum time broadcasting cannot be obtained for cycles containing more than 6 nodes (see Table 3.1).

Table 3.1: Comparing the time to broadcast in a chordless cycle to the minimum time to broadcast in a graph.

Number of nodes	Time to broadcast in a chordless cycle	Minimum time to broadcast in a graph
3	2	2
4	2	2
5	3	3
6	3	3
7	4	3
8	4	3
9	5	4

During broadcasting, at each time step, each informed node may pass the message along to one of its uninformed neighbors; thus, the number of nodes informed at each successive time step can at most double. For example, at time step 0 a single node (the originator) is informed of the message; this node can inform one of its neighbors at time step 1. After this time step, two nodes are informed (i.e., the originator and the node informed at time 1). Then at time 2, both the originator and the node informed at time 1 can inform one of their uninformed neighbors. After this time step, at most 4 nodes are informed (i.e., the originator, the node informed at time 1 and at most 2 nodes informed at time 2). Continuing this process for consecutive time steps, we see that the number of informed nodes can at most double from one time step to the next. In order for the number of nodes to double at time n, each informed node at time $n-1$ must have an uninformed neighbor that it can inform at time n. Table 3.2 shows the maximum total number of nodes informed at time steps 0 through 5 and the maximum number of new nodes informed at each time step.

Table 3.2: Maximum number of nodes informed at time steps 0 through 5.

Time	Total Number of Informed Nodes	Number of Newly Informed Nodes
0	1	
1	2	1
2	4	2
3	8	4
4	16	8
5	32	16

The slow broadcast time of cycles is due to the fact that, after time 2, at most 2 new nodes can be informed at each time step. This occurs because each node of a cycle has exactly 2 neighbors and all nodes, except the originator, must be informed by one of their neighbors, leaving only a single neighbor for each node to inform (see Figure (3.2)).

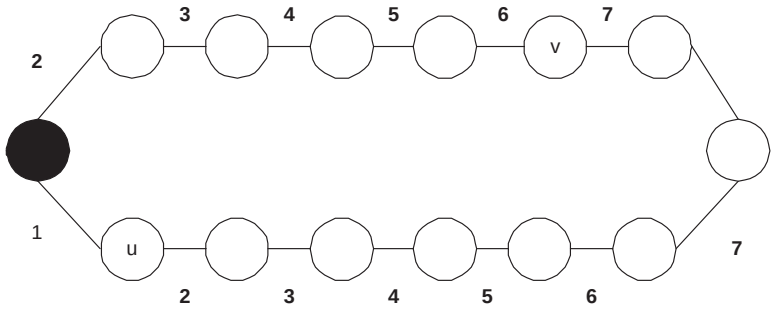

Figure 3.2: Broadcasting in C_{14} (the originator node is colored black).

To improve the broadcast time of such cycles, chords must be added. In this chapter, we discuss the placement of a single chord in the cycle. Then, we develop two algorithms for broadcasting in cycles containing a single chord. The first algorithm considers the case when the chord divides the cycle in half. The second algorithm considers the case when the chord divides the cycle into two cycles which differ in length by at least 2 nodes. Finally, we discuss the location of the originator in the cycle.

3.2 Placement of the Chord

According to the definition of a cycle, each node of the cycle has exactly 2 neighbors (i.e., is connected to exactly 2 other nodes). The addition of a chord connecting 2 nodes of a cycle will increase the number of neighbors of both of the nodes by one neighbor each. This allows for 2 nodes of the cycle to each inform an additional node (as long as that node was not already informed) during the broadcasting process, which can decrease the time to broadcast compared to the same size cycle without chords.

As an example, comparing Figure (3.2) to Figure (3.3) shows that both node u and node v were able to inform an additional node, by making use of the chord.

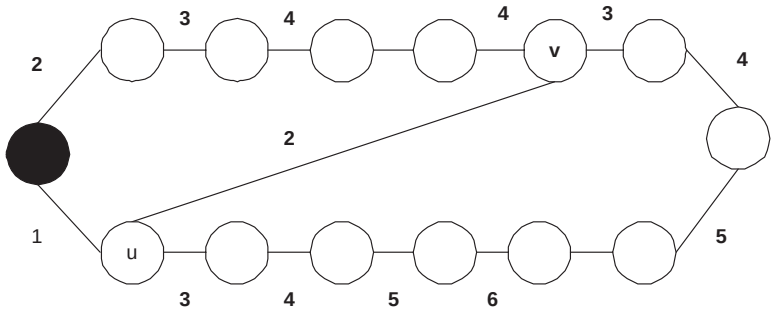

Figure 3.3: Broadcasting in C_{14} with a single chord (the originator node is colored black).

Table 3.3 compares the number of nodes informed at each time step in a cycle without chords (i.e., a chordless cycle) to the maximum number of nodes that can be informed at each time step. Time 3 is the first place that the number of nodes informed by the chordless cycle algorithm differs from the maximum number of nodes that can be informed at that time step. In a cycle, we want to increase the number of nodes informed as early as possible because this increase can increase the number of nodes informed at each additional time step. The addition of a single chord to a cycle, allows us to increase the number of nodes informed at time 3 from 2 to a maximum of 4 nodes

(see Figure (3.4)). The 2 additional nodes informed at time 3 then have the ability to each inform an uninformed neighbor at time 4, which can increase the number of nodes informed at time 4 by at most 2. At all times after time 4, at most 4 new nodes can be informed, since after time 3, no remaining nodes have more than 2 neighbors.

Table 3.3: Comparison of nodes informed at time steps 0 through 5.

Time	Number of Newly Informed Nodes in a Chordless Cycle	Maximum Number of Newly Informed Nodes
0		
1	1	1
2	2	2
3	2	4
4	2	8
5	2	16

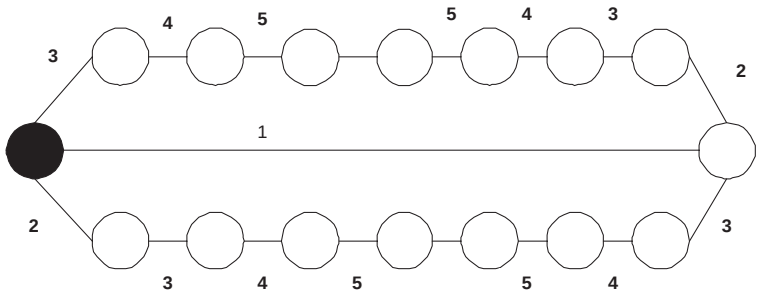

Figure 3.4: Sending the message across the chord as soon as possible.

In order to maximize the number of nodes informed at time 3, thus, increasing the total number of nodes informed at each time step after time 2, we must make the originator one of the endpoints of the chord (compare Figure (3.4) with Figure (3.5)).

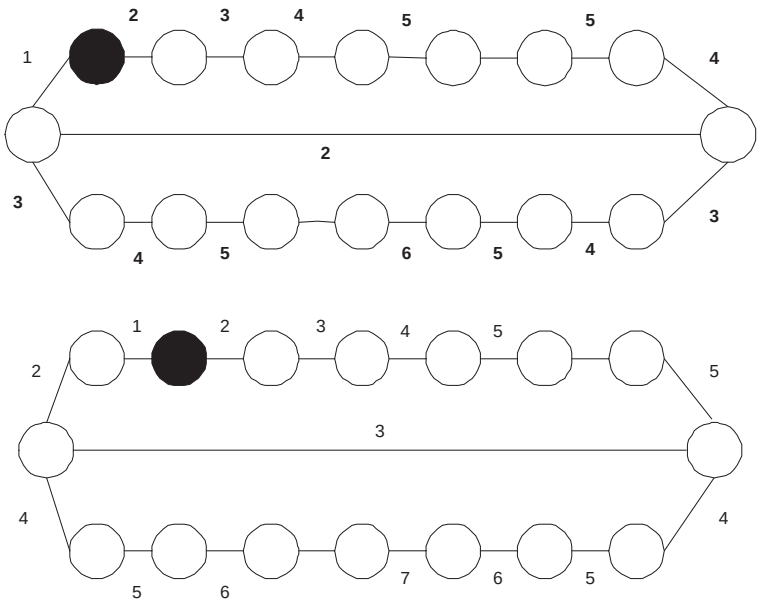

Figure 3.5: Delays in sending the message across the chord.

The next two sections present algorithms for broadcasting in a cycle containing a single chord with the originator as an endpoint of the chord.

3.3 The Equal Split Chord Originator Algorithm

Our first algorithm considers the case in which the chord divides the cycle in half or, in the case of an odd number of nodes, into two cycles which differ in size by a single node. We assume that the originator is a node that forms one of the endpoints of the chord. We call this the Equal Split Chord Originator (ESCO) algorithm. When the chord is added to the cycle, it creates two cycles which share a single edge (call these cycles A and B). We will label the cycles such that cycle A will always be either the same size as cycle B or, if the original cycle contains an odd number of nodes, cycle A will contain one more node than cycle B. ESCO proceeds by first sending the message across the chord and then one informed node sends the message to cycle A while the other

informed node sends to cycle B. Algorithm (3.3.1) is a formal description of our technique. See Figure (3.6) for an example of broadcasting using Algorithm (3.3.1).

Function $\mathrm{ESCO}(\mathbf{G}, n)$

1: The originator sends across the chord.
2: The originator sends to its neighbor in cycle A while the node informed in step 1 sends to its neighbor in cycle B.
3: The originator sends to its neighbor in cycle B while the node informed in step 1 sends to its neighbor in cycle A. The nodes informed in step 2 send to their uninformed neighbor.
4: **while** Not all nodes are informed. **do**
5: Nodes informed in the previous step send to their uninformed neighbor.
6: **end while**

Algorithm 3.3.1: Equal Split Chord Originator Algorithm

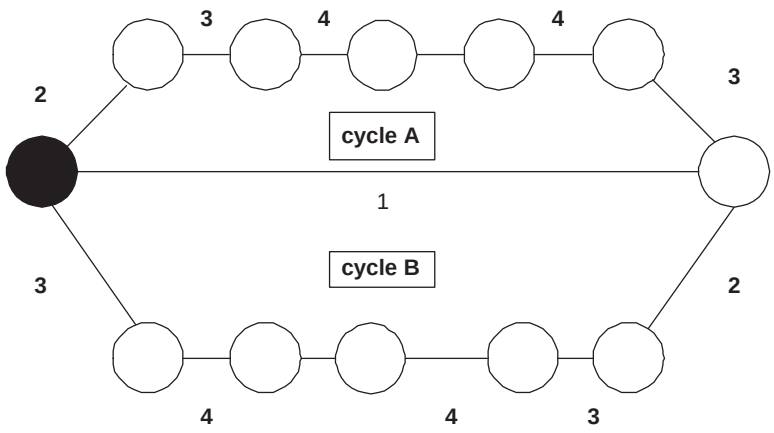

Figure 3.6: Broadcasting using Algorithm (3.3.1).

3.4 Correctness and Analysis of ESCO

The correctness of Algorithm (3.3.1) follows from the observations below.

1. The message arrives at a node before that node can pass the message on to another node. This follows clearly from the way the algorithm is defined.

2. At any given time, any node may act as either a sender or receiver of a message, but not both. This occurs since each node is used at most once in each step.

3. At the end of the broadcast algorithm, every node in the network has received the message. This is obvious from the condition of the while loop in the algorithm.

Lemma 3.4.1 *The time required to broadcast using Algorithm (3.3.1) is equal to* $\lfloor \frac{n}{2} \rfloor + 1$*, where* n *represents the number of nodes in the larger cycle formed by the chord (cycle* A*).*

Proof: Cycle A and cycle B differ in length by at most one node and according to the broadcast algorithm, after steps $1 - 3$, both cycles contain the same number of informed nodes. Thus, the larger cycle (cycle A) must finish broadcasting at the same time as, or after, the smaller cycle (cycle B); therefore, we only need to consider cycle A, when analyzing the running time of the algorithm. Further analysis of cycle A requires us to consider the case when cycle A contains an even number of nodes and the case when cycle A contains an odd number of nodes.

Case 1**:** Cycle A contains an even number of nodes.

Farley's lower bound is realized by such cycles; this bound states that the minimum broadcast time is equal to $2 \cdot (M - 1) + D$, where M represents the number of messages being broadcast and D represents the diameter of the graph [F80]. Since we are dealing with a cycle, $D = \lfloor \frac{n}{2} \rfloor$ and since we are only interested in broadcasting one message, $M = 1$. Using this information, we calculate the minimum broadcast time for the cycle to be $\lfloor \frac{n}{2} \rfloor$. Figure (3.7) shows a broadcast algorithm on C_n where n is even, which obtains this bound. Figure (3.8) shows Algorithm (3.3.1) on $C_{n+(n-2)}$ with a chord dividing the cycle in half (i.e., cycle A contains n nodes and cycle B contains n nodes). Comparing Figure (3.7) with cycle A of Figure (3.8), the last two nodes are informed at time $\lfloor \frac{n}{2} \rfloor$ in Figure (3.7); however, at that time, one node, x, remains to be informed in cycle A

of Figure (3.8). The uninformed node can be informed by either of its neighbors at time $\lfloor \frac{n}{2} \rfloor + 1$. Thus, the time required to broadcast when cycle A contains an even number of nodes is $\lfloor \frac{n}{2} \rfloor + 1$.

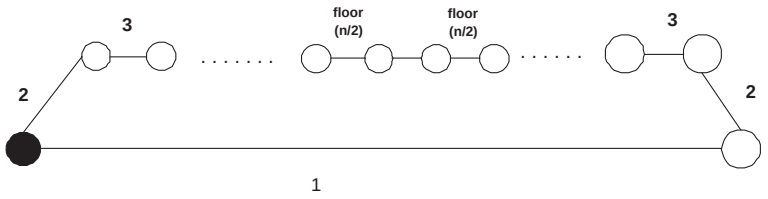

Figure 3.7: Case 1: Broadcasting in C_n, where n is even, using the chordless cycle algorithm.

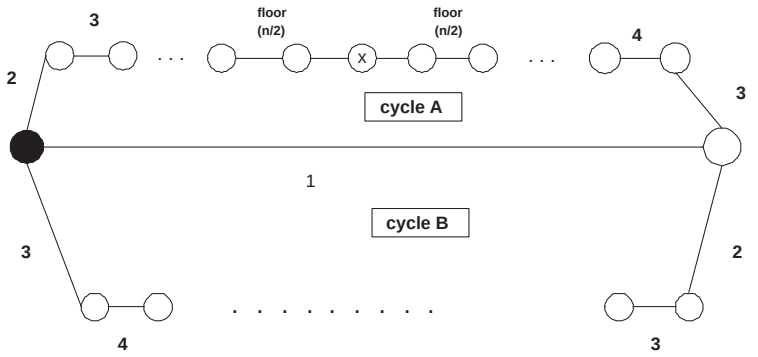

Figure 3.8: Case 1: Broadcasting in $C_{n+(n-2)}$, where n is even, using Algorithm (3.3.1).

Case 2**:** Cycle A contains an odd number of nodes.

Farley's lower bound cannot be achieved in such cycles, since there are two nodes which are both the maximum distance from the originator. The time required to broadcast is $2 \cdot (M-1) + D + 1$. Once again, $D = \lfloor \frac{n}{2} \rfloor$ and $M = 1$; thus, the time required to broadcast becomes $\lfloor \frac{n}{2} \rfloor + 1$. Figure (3.9) shows a broadcast algorithm on C_n where n is odd, which achieves this bound. Figure (3.10) shows Algorithm (3.3.1) on $C_{n+(n-2)}$ with a chord dividing the cycle in half (i.e., cycle A contains n nodes and cycle B contains n nodes). During the next to last time step, in Figure (3.9), two nodes are informed; however, only one of these two nodes is needed to pass the message on to the last node to be informed. Referring to Figure (3.9), we could increase the time at which the

neighbor of node x is informed by 1 without affecting the overall time needed to broadcast. This is precisely what happens in cycle A of Figure (3.10). Thus, the time required to broadcast when cycle A contains an odd number of nodes is $\lfloor \frac{n}{2} \rfloor + 1$.

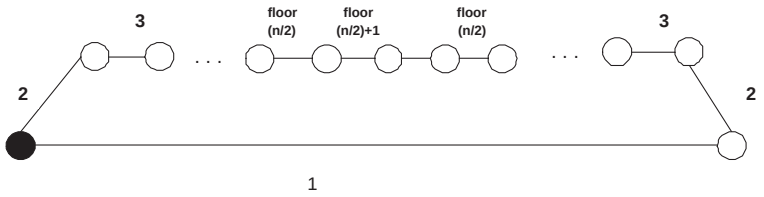

Figure 3.9: Case 2: Broadcasting in C_n, where n is odd, using the chordless cycle algorithm.

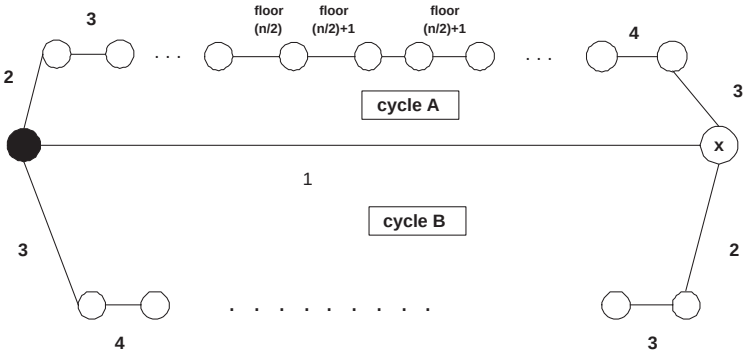

Figure 3.10: Case 2: Broadcasting in $C_{n+(n-2)}$, where n is odd, using Algorithm (3.3.1).

□

3.5 The Unequal Split Chord Originator Algorithm

Our second algorithm considers the case in which the chord divides the cycle into two cycles which differ in length by more than one node. Again, we assume that the originator is a node that forms one of the endpoints of the chord. We call this the Unequal Split Chord Originator (USCO) algorithm. When the chord is added to the cycle, it creates two cycles which share a single edge; we will call the larger cycle A and the smaller cycle B. USCO proceeds by first sending the message

across the chord and then both of the informed nodes send the message to cycle A first and, then, to cycle B. Algorithm (3.5.1) is a formal description of our technique. See Figure (3.11) for an example of broadcasting using Algorithm (3.5.1).

Function USCO($\mathbf{G}, n$)

1: The originator sends across the chord.
2: The originator and the node informed in step 1 both send to their neighbor in cycle A.
3: The originator and the node informed in step 1 both send to their neighbor in cycle B. The nodes informed in step 2 send to their uninformed neighbor.
4: **while** Not all nodes are informed. **do**
5: Nodes informed in the previous step send to their uninformed neighbor.
6: **end while**

Algorithm 3.5.1: Unequal Split Chord Originator Algorithm

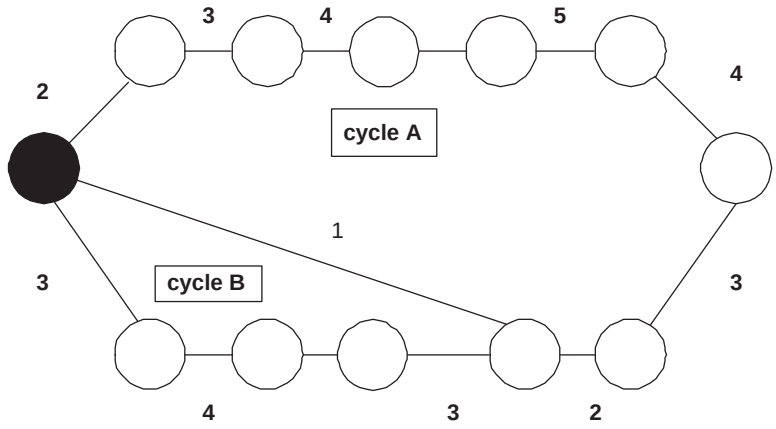

Figure 3.11: Broadcasting using Algorithm (3.5.1).

3.6 Correctness and Analysis of USCO

The correctness of Algorithm (3.5.1) follows from the observations below.

1. The message arrives at a node before that node can pass the message on to another node. This follows clearly from the way the algorithm is defined.

2. At any given time, any node may act as either a sender or receiver of a message, but not both. This occurs since each node is used at most once in each step.

3. At the end of the broadcast algorithm, every node in the network has received the message. This is obvious from the condition of the while loop in the algorithm.

Lemma 3.6.1 *The time required to broadcast using Algorithm (3.5.1) is* $\lceil \frac{n}{2} \rceil$*, where* n *represents the number of nodes in the larger cycle formed by the chord (cycle* A*).*

Proof: In the first step of the algorithm, we send across the chord. In the second step, both informed nodes send to their neighbors in cycle A. In the third step, the originator and the node informed in step 1 send to their neighbors in cycle B, while the nodes informed in step 2 simply pass the message on in their respective cycle. In the remaining steps, broadcasting completes following the pattern used for broadcasting in a cycle without chords. Using this algorithm, broadcasting in cycle A proceeds in the exact same manner as broadcasting in a cycle without chords (i.e., using the chordless cycle algorithm); see Figure (3.12). Thus, the broadcast time for cycle A is equal to $\lfloor \frac{n}{2} \rfloor$, if the cycle is of even length and $\lfloor \frac{n}{2} \rfloor + 1$, if the cycle is of odd length, which is equal to $\lceil \frac{n}{2} \rceil$ for both cases.

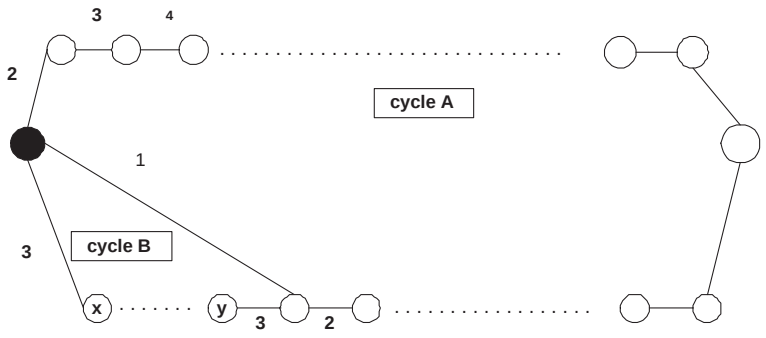

Figure 3.12: Broadcasting in a cycle with a chord using Algorithm (3.5.1).

Cycle B has at least 2 fewer nodes than cycle A. If we assume that cycle A contains m nodes, then cycle B can contain no more than $m-2$ nodes. Comparing the times to broadcast in a cycle of size m and a cycle of size $m-2$, we have $\lceil \frac{m}{2} \rceil$ and $\lceil \frac{m-2}{2} \rceil$, respectively. Thus, regardless of m, cycle A will always require at least one more unit of time than cycle B. Broadcasting in cycle B is identical to broadcasting using the chordless cycle algorithm, except that nodes x and y are informed a time unit later in cycle B than they are using the chordless cycle algorithm; see Figure (3.13). Thus, broadcasting in cycle B will require one extra time step than the chordless cycle algorithm. Since we have shown that cycle B will always finish at least one time unit before cycle A, adding an extra time unit to the time needed to broadcast in cycle B will cause cycle B to take time at most equal to that of cycle A. Therefore, the time to broadcast using Algorithm (3.5.1) is equal to the time needed to broadcast in cycle A (i.e., $\lceil \frac{n}{2} \rceil$, where n represents the number of nodes in cycle A).

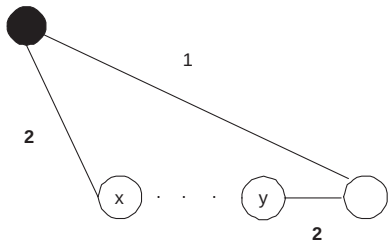

Figure 3.13: Broadcasting in a cycle using the chordless cycle algorithm.

□

3.7 Results

In this chapter, we developed 2 algorithms for broadcasting in a cycle with a single chord (with one of its endpoints being the originator). In such a graph, there are only 2 possibilities for the location of the chord. Either:

1. It splits the cycle in half or nearly half, in the case of an odd length cycle, or
2. It splits the cycle into two smaller cycles, one of which is 2 or more nodes larger than the other.

Algorithm (3.3.1) deals with case 1 and has a running time equal to $\lfloor \frac{n}{2} \rfloor + 1$, where n is the number of nodes in the larger cycle formed by the chord (cycle A). Algorithm (3.5.1) deals with case 2 and has a running time equal to $\lceil \frac{n}{2} \rceil$, where n represents the number of nodes in the larger cycle formed by the chord (cycle A). Since the algorithm for case 2 always considers the time to broadcast in a cycle that is 2 or more nodes larger than the largest cycle in case 1, the time taken by the algorithm for case 1 is always as good or better than the time taken by the algorithm for case 2. Thus, the best running time that we are able to achieve when broadcasting in a cycle with a single

chord occurs when the chord divides the cycle in half and the originator is one of the endpoints of the chord and is $\lfloor \frac{n}{2} \rfloor + 1$, where n is the number of nodes in the larger cycle formed by the chord (cycle A).

The chord placement and algorithms presented in this chapter assume that we know the location of the originator. If the location of the originator is unknown, then the best that we can do is to arbitrarily choose a node as the originator and draw a chord from this node that splits the cycle in half. Since the originator is chosen arbitrarily, it could be the case that the actual originator is a node in cycle A that is located halfway between each endpoint of the chord (see Figure (3.14)). In this case, the chord will produce no benefit to the cycle and the time to broadcast will be equal to the time to broadcast in a chordless cycle (i.e., a chordless cycle consisting of n nodes takes time equal to $\lfloor \frac{n}{2} \rfloor$ when n is even and $\lfloor \frac{n}{2} \rfloor + 1$ when n is odd).

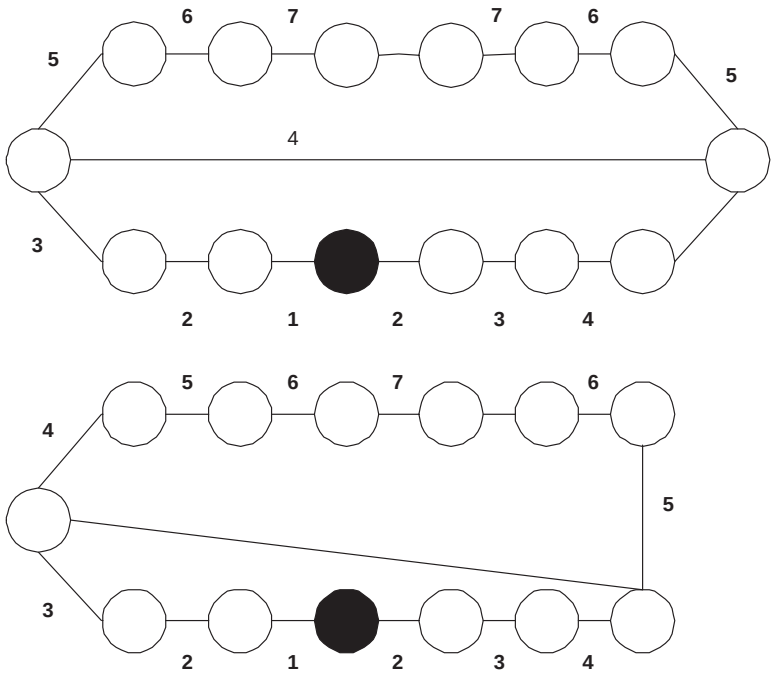

Figure 3.14: Worst case scenario for the location of the chord and the originator.

Chapter 4

Broadcasting in Cycles with Two Chords

4.1 Background

Chapter 3 dealt with broadcasting in cycles with a single chord. Using Algorithm(3.3.1), we are able to broadcast in minimum time for cycles containing between 7 and 12 nodes inclusive (see Figure (4.1)). The running time of Algorithm (3.3.1) is equal to the diameter of the larger cycle formed by the chord, D_m, plus 1. Both the function representing the running time of Algorithm (3.3.1) and the function representing the minimum broadcast time in a graph (i.e., $\lceil \log_2 n \rceil$) are monotonically increasing and for cycles containing more than 12 nodes (i.e., $n > 12$), the latter function grows slower. Thus, minimum time broadcasting cannot be obtained for cycles containing more than 12 nodes and a single chord (see Table(4.1)). Therefore, more chords must be added. In this chapter, we first discuss the placement of each of the chords in the cycle. We then develop two algorithms for the placement of the chords and we give algorithms for broadcasting in such cycles.

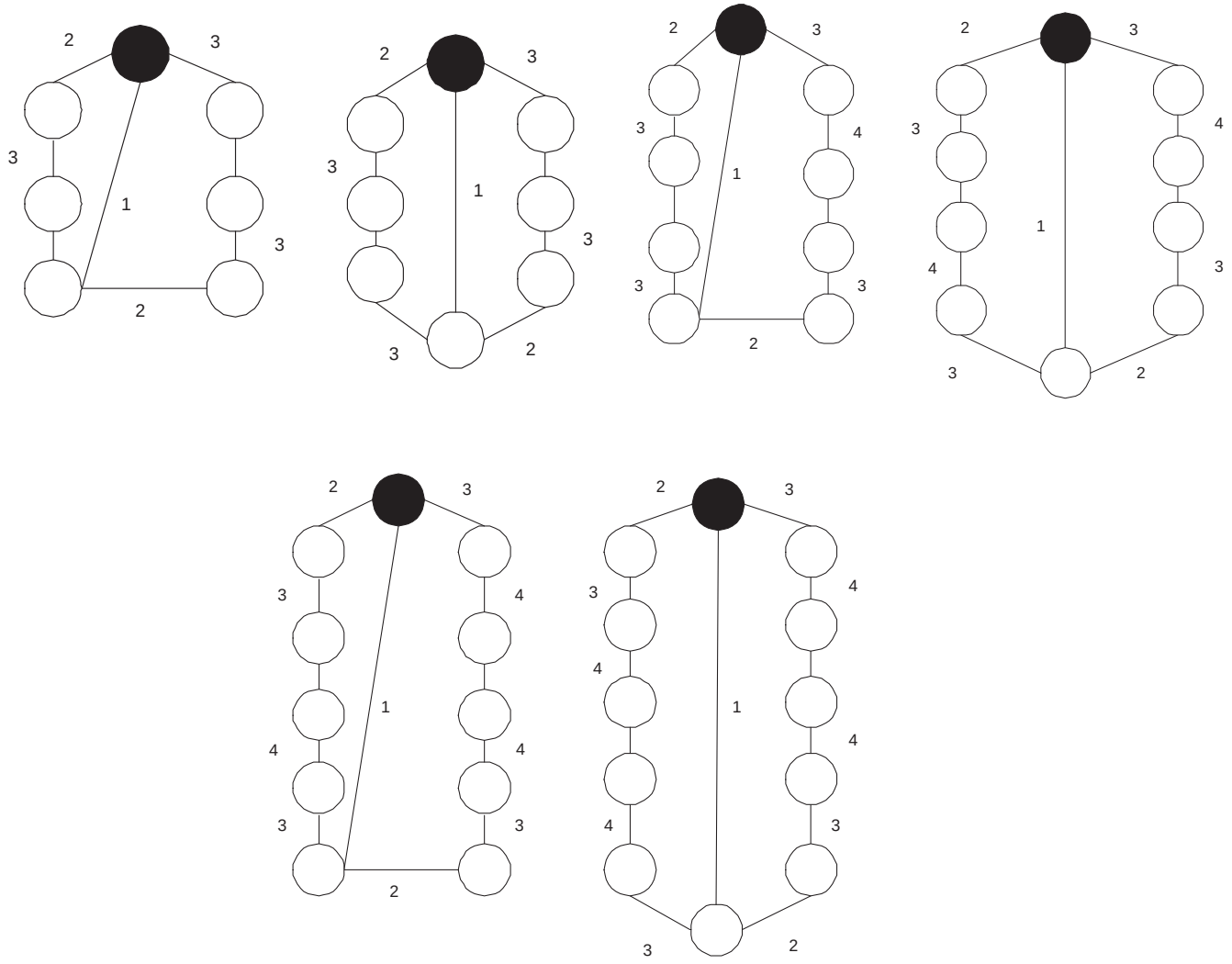

Figure 4.1: Broadcasting in C_7, C_8, C_9, C_{10}, C_{11} and C_{12}.

Table 4.1: Comparing the time to broadcast using Algorithm(3.3.1) to the minimum time to broadcast in a graph.

Number of nodes	Time to broadcast using Algorithm(3.3.1)	Minimum time to broadcast in a graph
8	3	3
9	4	4
10	4	4
11	4	4
12	4	4
13	5	4
14	5	4

4.2 Placement of the Chords

As shown in Chapter 3, in order to increase the number of nodes that can be informed at each time step after time 2, we want to begin by increasing the number of nodes informed at time 3, since this is the first time step that the number of nodes informed by the chordless cycle algorithm differs from the maximum number of nodes that can be informed in a graph. In Chapter 3, we showed that we can inform the maximum number of nodes at time 3 by branching at the originator at time 1. In the single chord case, this increased the number of nodes informed at time 3 and all times after time 3 from 2 nodes to a maximum of 4 nodes. Thus, we will use the first of the two chords to branch at the originator. Since the addition of the first chord maximizes the number of nodes that are informed at time 3, we now want to position the second chord so that we increase the number of nodes informed at time 4. Branching at time 2 will allow this to happen. After time 1, only 2 nodes have the message (the originator and the node at the other endpoint of the first chord). Thus, in order to branch at time 2, one of the endpoints of the first chord must also be an endpoint of the second chord (see Figure (4.2)).

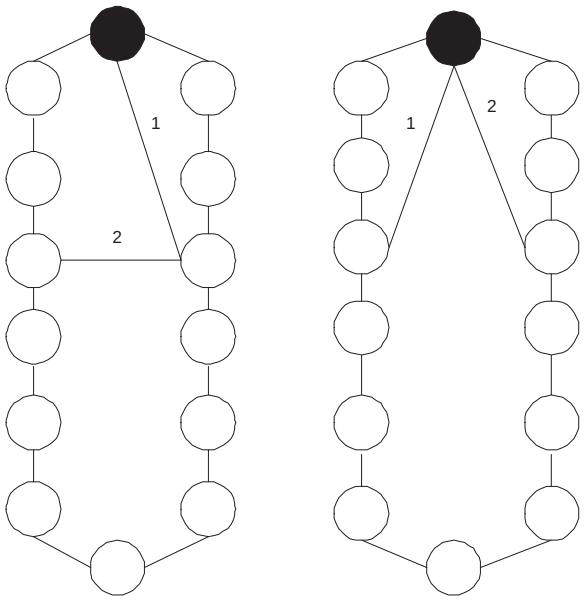

Figure 4.2: Placement of a second chord.

In the next four sections, we develop an algorithm for the placement of each of the chords in a cycle when the originator is the shared endpoint and we develop an algorithm for broadcasting in such a cycle. Then, we develop an algorithm for the placement of each of the chords in a cycle when the node informed at time 1 is the shared endpoint and we develop an algorithm for broadcasting in such a cycle.

4.3 Adding Chords to the Cycle when the Originator is the Shared Endpoint

Now that we have determined that the first chord must use the originator as one of its endpoints and that the second chord must share an endpoint with the first chord, we need to determine where to place the other endpoints of the chords. The shared endpoint must be either the originator or the node informed at time 1. We will first consider the case in which the originator is the shared

endpoint. Adding two chords to a cycle, in the manner that we have explained, creates three smaller cycles that share some endpoints. When analyzing the running time of a broadcast scheme such a graph, we must take 2 factors into consideration.

1. The time to broadcast in each of the smaller cycles that has been formed.

2. The time, at which, broadcasting begins in each of the smaller cycles.

If we position the endpoints of the chords so that we create 3 equal size (or as close to equal size as possible) cycles, each of these cycles should be able to complete broadcasting in approximately the same amount of time, depending upon when the first node in the cycle is informed. Using Algorithm (4.3.1), we are able to calculate the positions for all endpoints of the chords; in this algorithm, we assume that all of the nodes of the cycle have been consecutively numbered clockwise beginning with the originator numbered 0 and that n represents the total number of nodes in the graph. Figure (4.3) shows the results of using Algorithm (4.3.1) to add 2 chords to a cycle containing 16 nodes.

Function CHORDPLACEMENTALG$(\mathbf{G}, n)$
1: Divide $(n+4)$ by 3 in order to obtain a quotient, w, and a remainder, r.
2: **if** $(r > 0)$ **then**
3: $\quad c1 = w + 1$
4: **else**
5: $\quad c1 = w$
6: **end if**
7: $c2 = w$
8: Draw a chord from the originator to node $c1 - 1$.
9: Draw a chord from the originator to node $n - c2 + 1$.

Algorithm 4.3.1: Algorithm to place 2 chords into a cycle.

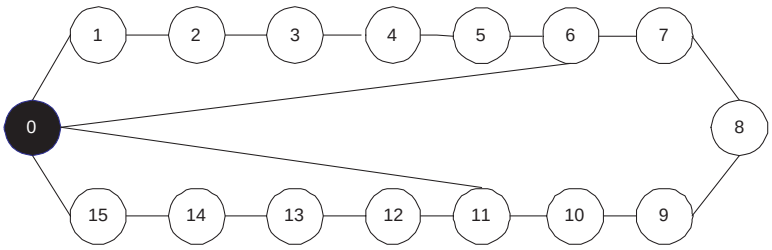

Figure 4.3: Adding 2 chords to a cycle using Algorithm (4.3.1).

4.4 The Two Chord Originator Algorithm

Algorithm (4.4.1) considers broadcasting in a cycle with 2 chords, such that both chords share the originator as a common endpoint and the chords are drawn using Algorithm (4.3.1). We call this the Two Chord Originator (TCO) algorithm. When the chords are added to the cycle, they create three cycles which share some edges. Working in a clockwise fashion around the original cycle, we will call the first small cycle that we encounter, cycle A, the second, cycle B, and the third, cycle C. TCO proceeds by first sending the message across the chord forming cycle A. Then, the message is sent across the other chord, while the node informed at time 1 sends to its neighbor in cycle A. Algorithm (4.4.1) is a formal description of our technique. See Figure (4.4) for an example of broadcasting using Algorithm (4.4.1).

Function $\mathrm{TCO}(\mathbf{G}, n)$

1: The originator sends across the chord forming cycle A.
2: The originator sends across the other chord and the node informed in step 1 sends to its neighbor in cycle A.
3: The originator sends to its neighbor in cycle C, the node informed in step 1 sends to its neighbor in cycle B, the node informed by the originator in step 2 sends to its neighbor in cycle C and the other node informed in step 2 sends to its uninformed neighbor.
4: The originator sends to its neighbor in cycle A, the node informed by the originator in step 2 sends to its neighbor in cycle B and all nodes informed in step 3 send to their uninformed neighbor.
5: **while** Not all nodes are informed. **do**
6: Nodes informed in the previous step send to their uninformed neighbor.
7: **end while**

Algorithm 4.4.1: Two Chord Originator Algorithm

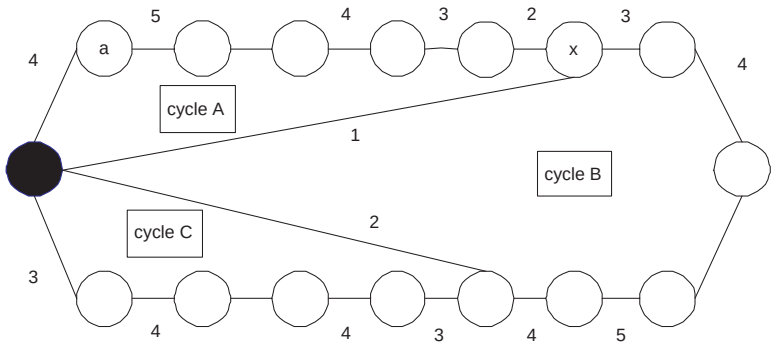

Figure 4.4: Broadcasting in C_{16} with two chords using Algorithm (4.4.1).

4.5 Correctness and Analysis of TCO

The correctness of Algorithm (4.4.1) follows from the observations below.

1. The message arrives at a node before that node can pass the message on to another node.

   This follows clearly from the way the algorithm is defined.

2. At any given time, any node may act as either a sender or receiver of a message, but not both.

   This occurs since each node is used at most once in each step.

3. At the end of the broadcast algorithm, every node in the network has received the message. This is obvious from the condition of the while loop in the algorithm.

Lemma 4.5.1 *The time required to broadcast in Algorithm (4.4.1) is equal to* $\lceil \frac{n}{2} \rceil + 1$ *where* n *represents the number of nodes in the largest cycle formed by the chords (cycle* A*).*

Proof: When analyzing the running time of such an algorithm, we must take 2 factors into consideration.

1. The time to broadcast in each of the smaller cycles that has been formed.

2. The time, at which, broadcasting begins in each of the smaller cycles.

The time to broadcast in a cycle is determined by the size of the cycle. According to Algorithm (4.3.1), cycle A will always be larger than or equal in size to cycle B and cycle C. Next, we must consider the time at which broadcasting begins in each of the smaller cycles. Referring to Algorithm (4.4.1) and Figure (4.4), we find that after the message is passed along the chords, it will eventually begin moving through the small cycles using the chordless cycle algorithm (i.e., each informed node simply passes the message along to its uninformed neighbor, if such a neighbor exists). We define this as the time at which broadcasting begins in each of the smaller cycles. Again, referring to Algorithm (4.4.1) and Figure (4.4), we find that after time 2, we do not pass along either of the chords and after time 3, each small cycle contains the same number, 4, of informed nodes. Thus, after time 3, broadcasting proceeds in each of the smaller cycles using the chordless cycle algorithm. This means that at every step after time 3, all of the smaller cycles will have the same number of nodes informed. Therefore, when calculating the running time of Algorithm (4.4.1), we can simply concentrate on the largest of the small cycles (cycle A). Further

analysis of cycle A requires us to consider the case in which cycle A contains an even number of nodes and the case in which cycle A contains an odd number of nodes.

Case 1**:** Cycle A contains an even number of nodes.

Farley's lower bound is realized by such cycles; this bound states that the minimum broadcast time is equal to $2 \cdot (M-1) + D$, where M represents the number of messages being broadcast and D represents the diameter of the graph [F80]. Since we are dealing with a cycle, $D = \lfloor \frac{n}{2} \rfloor$ and since we are only interested in broadcasting one message, $M = 1$. Using this information, we calculate the minimum broadcast time for the cycle to be $\lfloor \frac{n}{2} \rfloor$. Figure (4.5) shows a broadcast algorithm on C_n where n is even, which obtains this bound. Figure (4.6) shows Algorithm (4.4.1) on C_{3n-4} with 2 chords dividing the cycle into 3 smaller cycles of equal length (i.e., each smaller cycle contains n nodes). In Figure (4.5), the last nodes to be informed are informed at time $\lfloor \frac{n}{2} \rfloor$. Since Algorithm (4.4.1) begins sending to node a 2 time units later than the chordless cycle algorithm, by time $\lfloor \frac{n}{2} \rfloor$, we have 2 nodes, x and y, that have not yet been informed (refer to Figure (4.6)). In the next time step (i.e., at time $\lfloor \frac{n}{2} \rfloor + 1$), node y can be informed via node z and node x can be informed via node w, which completes the broadcast. Therefore, the time required to broadcast in cycle A, when cycle A contains an even number of nodes is $\lfloor \frac{n}{2} \rfloor + 1$, which is equal to $\lceil \frac{n}{2} \rceil + 1$.

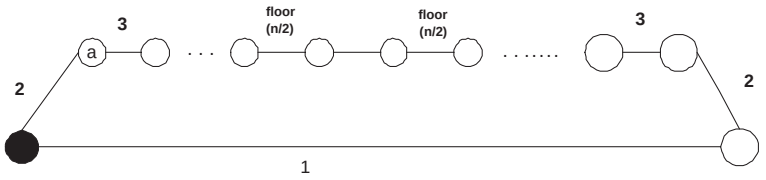

Figure 4.5: Case 1: Broadcasting in C_n where n is even, using the chordless cycle algorithm.

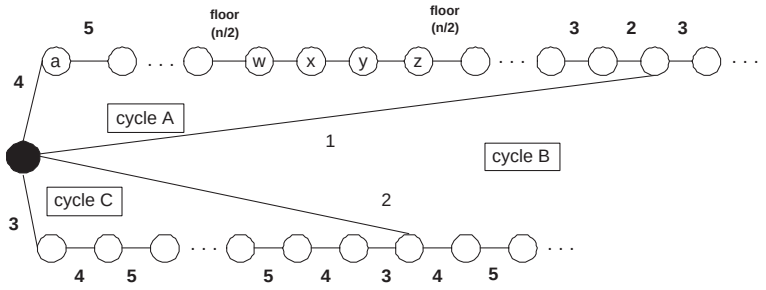

Figure 4.6: Case 1: Broadcasting in C_{3n-4} where n is even, using Algorithm (4.4.1).

Case 2**:** Cycle A contains an odd number of nodes.

Farley's lower bound cannot be achieved in such cycles; since there are two nodes which are both the maximum distance from the originator. The time required to broadcast is $2 \cdot (M-1) + D + 1$. Once again, $D = \lfloor \frac{n}{2} \rfloor$ and $M = 1$; thus the time required to broadcast becomes $\lfloor \frac{n}{2} \rfloor + 1$. Figure (4.7) shows the chordless cycle algorithm on C_n where n is odd, which achieves this bound. Figure (4.8) shows Algorithm (4.4.1) on C_{3n-4} with two chords dividing the cycle into 3 smaller cycles of equal length (i.e., each smaller cycle contains n nodes). In Figure (4.7), the last node to be informed is informed at time $\lfloor \frac{n}{2} \rfloor + 1$. Since Algorithm (4.4.1) begins sending to node a 2 time units later than the chordless cycle algorithm, by time $\lfloor \frac{n}{2} \rfloor + 1$, we have 1 node, x, that has not yet been informed (refer to Figure (4.8)). In the next time step (i.e., at time $\lfloor \frac{n}{2} \rfloor + 1 + 1 = \lfloor \frac{n}{2} \rfloor + 2$), node x can be informed by either of its neighbors, which completes the broadcast. Therefore, the time to broadcast in cycle A, when cycle A contains an odd number of nodes is $\lfloor \frac{n}{2} \rfloor + 2$, which is equal to $\lceil \frac{n}{2} \rceil + 1$.

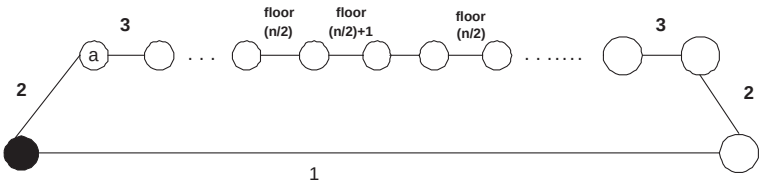

Figure 4.7: Case 2: Broadcasting in C_n where n is odd, using the chordless cycle algorithm.

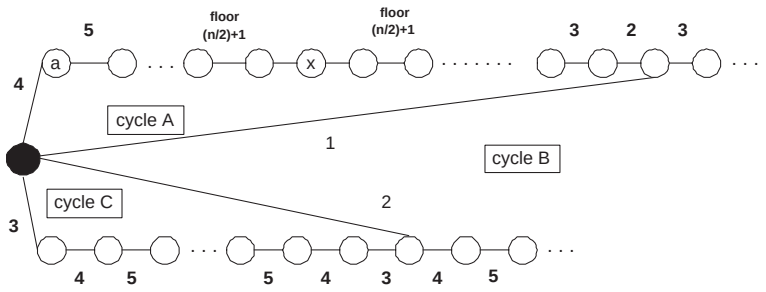

Figure 4.8: Case 2: Broadcasting in C_{3n-4} where n is odd, using Algorithm (4.4.1).

□

4.6 Alternative Placement of the Chords

In Section (4.2), we determined that the first chord added to the cycle must use the originator as one of its endpoints and that the second chord must share an endpoint with the first chord. Thus, the shared endpoint must be either the originator or the node informed at time 1. In Section (4.3), we considered the case in which the originator was the shared endpoint. We now consider the case in which the node informed at time 1 is the shared endpoint. Adding two chords to a cycle, in the manner that we have explained, creates three smaller cycles that share some endpoints. As discussed in Section (4.3), we want to position the endpoints of the chords so that we create 3 equal size (or as close to equal size as possible) cycles, so that each of these cycles has the ability to complete broadcasting in approximately the same amount of time, depending upon when the first node in the cycle is informed. Using Algorithm (4.6.1), we are able to calculate the positions for all endpoints of the chords; in this algorithm, we assume that all of the nodes of the cycle have been consecutively numbered clockwise beginning with the originator numbered 0 and that n represents the total number of nodes in the graph. Figure (4.9) shows the results of using Algorithm

(4.6.1) to add 2 chords to a cycle containing 16 nodes.

Function ALTERNATIVECHORDPLACEMENTALG$(\mathbf{G}, n)$
1: Divide $(n+4)$ by 3 in order to obtain a quotient, w, and a remainder, r.
2: **if** $(r > 0)$ **then**
3: $\quad c1 = w + 1$
4: $\quad r = r - 1$
5: **else**
6: $\quad c1 = w$
7: **end if**
8: **if** $r > 0$ **then**
9: $\quad c2 = w + 1$
10: **else**
11: $\quad c2 = w$
12: **end if**
13: Draw a chord from the originator to node $c1 - 1$.
14: Draw a chord from the node $c1 - 1$ to node $c1 - 1 + c2 - 1$.

Algorithm 4.6.1: Algorithm to place 2 chords into a cycle.

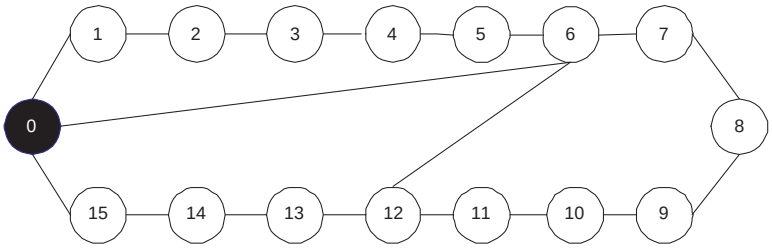

Figure 4.9: Adding 2 chords to a cycle using Algorithm (4.6.1).

4.7 The Two Chord Non-Originator Algorithm

Algorithm (4.7.1) considers broadcasting in a cycle with 2 chords, such that both chords share the node informed at time 1 as a common endpoint. We call this the Two Chord Non-Originator (TCNO) algorithm. When the chords are added to the cycle, they create three cycles which share some edges. Working in a clockwise fashion around the original cycle, we will call the first small cycle that we encounter, cycle A, the second, cycle B, and the third, cycle C. TCNO proceeds

by the originator first sending the message across the chord. Then the message is sent across the other chord, while the originator sends to its neighbor in cycle A. Algorithm (4.7.1) is a formal description of our technique. See Figure (4.10) for an example of broadcasting using Algorithm (4.7.1).

Function TCNO$(\mathbf{G}, n)$

1: The originator sends across the chord.
2: The originator sends to its neighbor in cycle A, while the node informed in step 1 sends across the other chord.
3: The originator sends to its neighbor in cycle C, the node informed in step 1 sends to its neighbor in cycle B, the node informed by the chord in step 2 sends to its neighbor in cycle B and the other node informed in step 2 sends to its uninformed neighbor.
4: The node informed in step 1 sends to its neighbor in cycle A, the node informed by the chord in step 2 sends to its neighbor in cycle C and all nodes informed in step 3 send to their uninformed neighbor.
5: **while** Not all nodes are informed. **do**
6: Nodes informed in the previous step send to their uninformed neighbor.
7: **end while**

Algorithm 4.7.1: Two Chord Non-Originator Algorithm

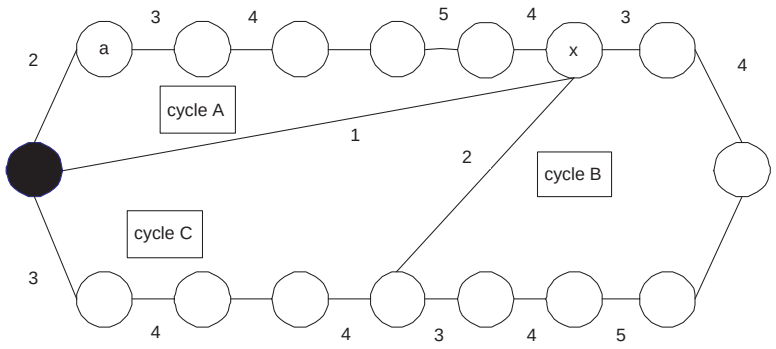

Figure 4.10: Broadcasting in C_{16} with two chords using Algorithm (4.7.1).

4.8 Correctness and Analysis of TCNO

The correctness of Algorithm (4.7.1) follows from the observations below.

1. The message arrives at a node before that node can pass the message on to another node. This follows clearly from the way the algorithm is defined.

2. At any given time, any node may act as either a sender or receiver of a message, but not both. This occurs since each node is used at most once in each step.

3. At the end of the broadcast algorithm, every node in the network has received the message. This is obvious from the condition of the while loop in the algorithm.

Lemma 4.8.1 *The time required to broadcast in Algorithm (4.7.1) is equal to* $\lceil \frac{n}{2} \rceil + 1$*, where* n *represents the number of nodes in the largest cycle formed by the chords (cycle* A*).*

Proof: According to Algorithm (4.6.1), cycle A will always be as large or larger than cycle B and cycle C. Both Algorithm (4.3.1) and Algorithm (4.6.1) place the first chord in the same position in a given cycle (i.e., no matter which algorithm we use, cycle A will be exactly the same) and after time 3, each small cycle contains the same number, 4, of informed nodes. After time 3, broadcasting proceeds in each of the smaller cycles using the chordless cycle algorithm. This is precisely the same scenario that occurred using Algorithm (4.4.1). Thus, both Algorithm (4.4.1) and Algorithm (4.7.1) will complete broadcasting at the same time (i.e., $\lceil \frac{n}{2} \rceil + 1$.) □

4.9 Results

In this chapter, we developed 2 algorithms for the placement of 2 chords into a cycle. The first algorithm dealt with the case when the originator was the shared endpoint of the chords, while the second algorithm dealt with the case when the node informed at time 1 was the shared endpoint of the chords. We developed and analyzed broadcasting algorithms for each of these cases. We found

that the running time of both broadcasting algorithms was equal; therefore, it makes no difference whether one chooses to make the originator the shared endpoint of the chords or the node informed at time 1 the shared endpoint of the chords.

Chapter 5

Multiple Message Broadcasting in Cycles with Chords

5.1 Background

Thus far, we have presented several algorithms for broadcasting a single message in a cycle with one and two chords, respectively. Often times, it is necessary to broadcast a large file over a computer network. Assuming that each message (file) to be passed requires only one call taking a single time unit is not very realistic. Considering a large file, it is usually the case that the file is broken up into many small pieces (messages) by the sending computer and then reassembled by the receiving computer.

Multiple message broadcasting occurs when one node has m messages which must be sent to all other nodes in a network, $G = (V, E)$, subject to the following constraints.

1. Each call involves two nodes.

2. Each call requires only one unit of time.

3. A node can only call a node with which it shares an edge.

4. Only one message is sent during a single call.

5. A message must arrive at a node before that node can pass the message on to another node.

6. At any given time unit, any node may act as either a sender or a receiver of a message, but not both.

7. At the end of a multiple message broadcast algorithm, every node in the network has received every message.

The following are two obvious approaches to the problem of broadcasting multiple messages.

1. Repeat a single message algorithm m times.

2. Modify a single message algorithm, so that each call now uses m time units.

Each of these approaches takes $m \cdot t$ time, where m is the number of messages to be sent and t is the time taken to broadcast a single message using the single message algorithm. Thus, simple modifications to a single message broadcasting scheme are not efficient enough to broadcast multiple messages.

Farley was the first to study multiple message broadcasting in general connected graphs [F80]. He defined $b_m(v)$ for $v \in V$ to be the minimum number of time units required to broadcast m messages from node v throughout a connected graph $G = (V, E)$, when $|V| = n$. He also defined

the broadcast time of a graph G, $B_m(G)$, as follows.

$$B_m(G) = \overset{max}{v \in V}\ b_m(v)$$

Farley then developed the following bounds for $B_m(G)$ for any connected graph, G, with diameter, D, and the maximum degree of any node, d_{max}.

$$2 \cdot (m-1) + D \leq B_m(G) \leq d_{max} \cdot (m-1) + (n-1)$$

Farley's lower bound is realized in cycles with an even number of nodes (see Figure (5.1)). The upper bound is realized in paths and stars (see Figure (5.2)).

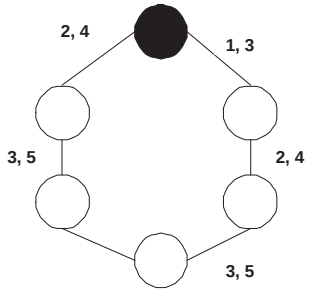

Figure 5.1: Multiple message broadcasting in C_6 using the chordless cycle algorithm.

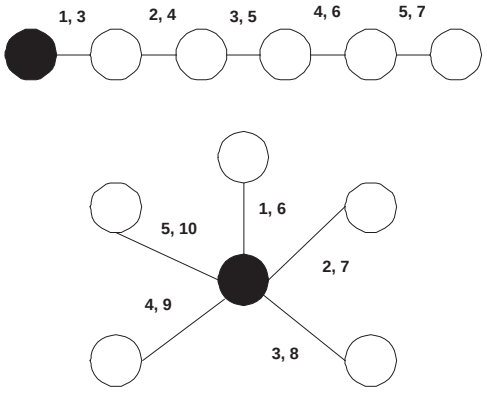

Figure 5.2: Multiple message broadcasting in a path with 6 nodes and a star with 6 nodes.

Farley's lower bound cannot be achieved in a cycle with an odd number of nodes, since there are 2 nodes at the maximum distance from the originator, an additional time unit is required (see

Figure (5.3)). The time required to broadcast in a cycle with an odd number of nodes is equal to $2 \cdot (m-1) + \lfloor \frac{n}{2} \rfloor + 1$, where 2 represents the time delay between a node receiving consecutive messages, $(m-1)$ represents the number of additional messages to be sent (after the first message) and $\lfloor \frac{n}{2} \rfloor + 1$ represents the time taken to broadcast the first message to all n nodes in the graph.

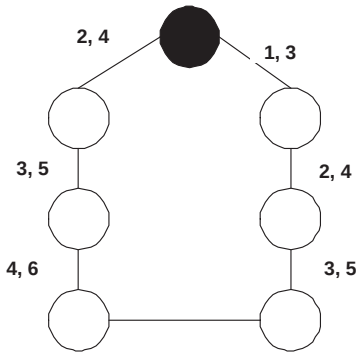

Figure 5.3: Multiple message broadcasting in C_7 using the chordless cycle algorithm.

In this chapter, we will examine multiple message broadcasting in cycles with one and two chords, respectively.

5.2 Multiple Message Broadcasting in a Cycle with a Single Chord

Does the addition of a single chord decrease the time required to broadcast multiple messages in a cycle? We start by comparing Farley's upper and lower bounds on a chordless cycle with Farley's upper and lower bounds on a cycle with a single chord. All nodes in a chordless cycle have degree 2 and that the diameter of such a cycle is $\lfloor \frac{n}{2} \rfloor$, where n is the number of nodes in the cycle. This gives us the following bounds for a chordless cycle.

$$2 \cdot (m-1) + \lfloor \frac{n}{2} \rfloor \leq B_m(G) \leq 2 \cdot (m-1) + (n-1)$$

On the other hand, the maximum degree of any node in a cycle with a single chord is 3, while the diameter of such a cycle remains $\lfloor \frac{n}{2} \rfloor$, which is realized by choosing two nodes in the cycle that are the maximum distance from each other and do not benefit from the use of the chord (see Figure (5.4)). This gives us the following bounds for a cycle with a single chord.

$$2 \cdot (m-1) + \lfloor \frac{n}{2} \rfloor \leq B_m(G) \leq 3 \cdot (m-1) + (n-1)$$

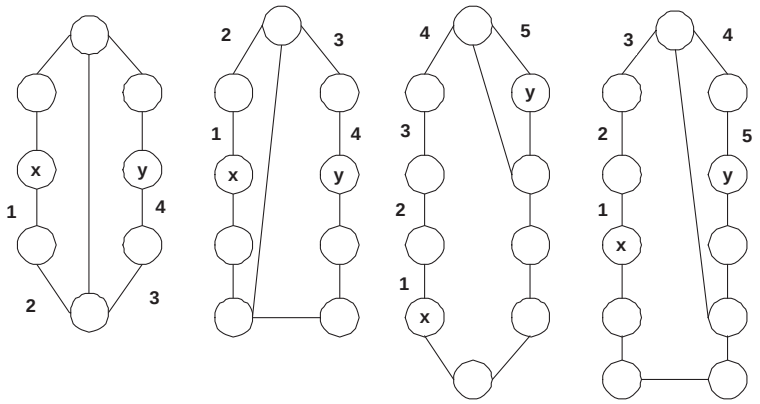

Figure 5.4: The diameter of cycles with a single chord remains $\lfloor \frac{n}{2} \rfloor$.

The lower bound of a chordless cycle is equal to the lower bound of a cycle with a single chord. Farley's lower bound can be achieved in a chordless cycle with an even number of nodes using the chordless cycle multiple message broadcasting algorithm; therefore, Farley's lower bound can be achieved in a cycle with a single chord and an even number of nodes (i.e., we would simply ignore the chord and use the chordless cycle multiple message broadcasting algorithm). However, in the case of a chordless cycle with an odd number of nodes, Farley's lower bound cannot be achieved, since there are two nodes distance D from the originator, one additional time unit is required for broadcasting. So, we ask the question: Can Farley's lower bound be achieved in a cycle with a single chord and an odd number of nodes? The location of the originator is important in this investigation; if the originator happens to be a node halfway between the endpoints of the chord,

the chord will provide no benefit to the graph. As shown in Chapter 3, positioning the chord so that it cuts the cycle in half and the originator forms one endpoint of the chord allows us to inform the maximum number of nodes in the minimum amount of time. Therefore, we will assume that the originator forms one of the endpoints of the chord and we will use the chord to divide the cycle in half. A simple modification to our algorithm for broadcasting a single message in a cycle with a single chord, such as repeating the single message broadcast scheme m times or having each call take m time units, is not efficient. If we treat the messages as a single set, sending each message out from the originator following the same broadcast scheme, and the nodes with three edges do not make use of all three of their edges (i.e., they do not make use of branching), we would simply be broadcasting in a chordless cycle (see Figure (5.5)). However, when all 3 of a node's edges are used, a delay of 3 time units is introduced between messages leaving the originator, since at least one node in the graph is busy for 3 time units (see Figure (5.6)). The time required to broadcast a single set of multiple messages in such a graph is $3 \cdot (m-1) + t$, where 3 represents the delay between messages (i.e., the coefficient of the message term), m represents the number of messages to be broadcast and t represents the time required to broadcast the first message. According to Farley's lower bound, t can be no less than the diameter of the graph, D, which is equal to $\lfloor \frac{n}{2} \rfloor$. Thus, a lower bound of the expression is $3 \cdot (m-1) + \lfloor \frac{n}{2} \rfloor$. Comparing this expression to the time required to broadcast multiple messages in a chordless cycle with an odd number of nodes (i.e., $2 \cdot (m-1) + \lfloor \frac{n}{2} \rfloor + 1$), it is obvious that the latter algorithm is more efficient. Therefore, adding a single chord to a cycle does not decrease the time required to broadcast a single set of multiple messages.

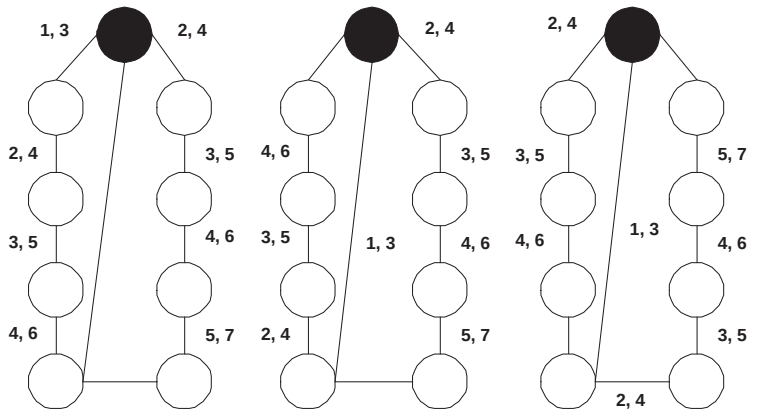

Figure 5.5: Performing multiple message broadcasting using a maximum of 2 of any node's edges.

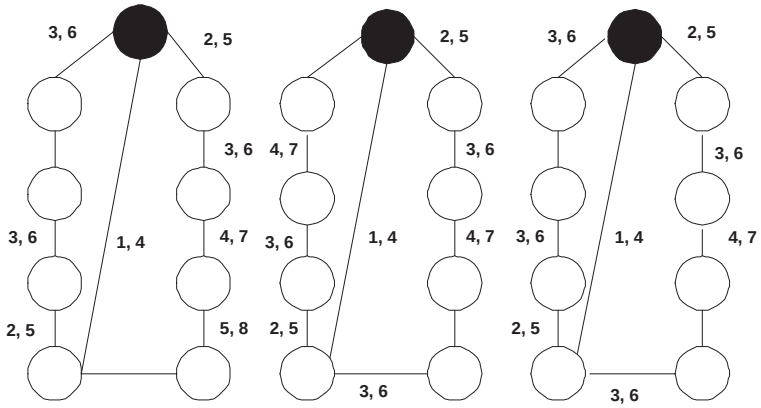

Figure 5.6: Performing multiple message broadcasting using all 3 edges of a node.

In order to improve upon the time taken to broadcast multiple messages in a chordless cycle with an odd number of nodes, we must keep the coefficient of the message term equal to 2. The coefficient of the message term correlates to the delay between messages and in order to keep the coefficient small, we must keep branching to a minimum, since each time we branch, we increase the number of time units during which a node is busy. As an example, refer to Figure (5.6). Considering the graphs in the example from left to right, in the first graph, the originator is busy sending the first message for 3 time units; thus, it cannot begin to send the second message until time 4. Although, the originator is only busy for 2 time units in the second graph, the node

informed at time 1 is busy for 3 time units, which causes the originator to wait 3 time units before sending the second message. Since, at time 3, if the originator sent a second message to the node informed at time 1, this node would be receiving the second message from the originator at the same time that it was sending the first message to one of its neighbors and such an action is not permitted, according to the definition of multiple message broadcasting (i.e., at any given time unit, any node may act as either a sender or a receiver of a message, but not both). In the third graph, both the originator and the node informed at time 1 are busy for 3 time units; thus, the originator cannot begin to send the second message until time 4.

In an attempt to decrease the coefficient of the message term, thereby making multiple message broadcasting more efficient, several authors including [G06, VB94, V79, WV96] and [W99] have developed methods for multiple message broadcasting, wherein, the messages are divided into multiple sets (such as a set of odd messages and a set of even messages) and each set uses a different broadcast scheme in order to deliver a message to every node in the graph. As an example, Figure (5.7) depicts a multiple message algorithm developed by Wojciechowska in [W99] which broadcasts multiple messages in a grid by dividing the messages into an odd set and an even set.

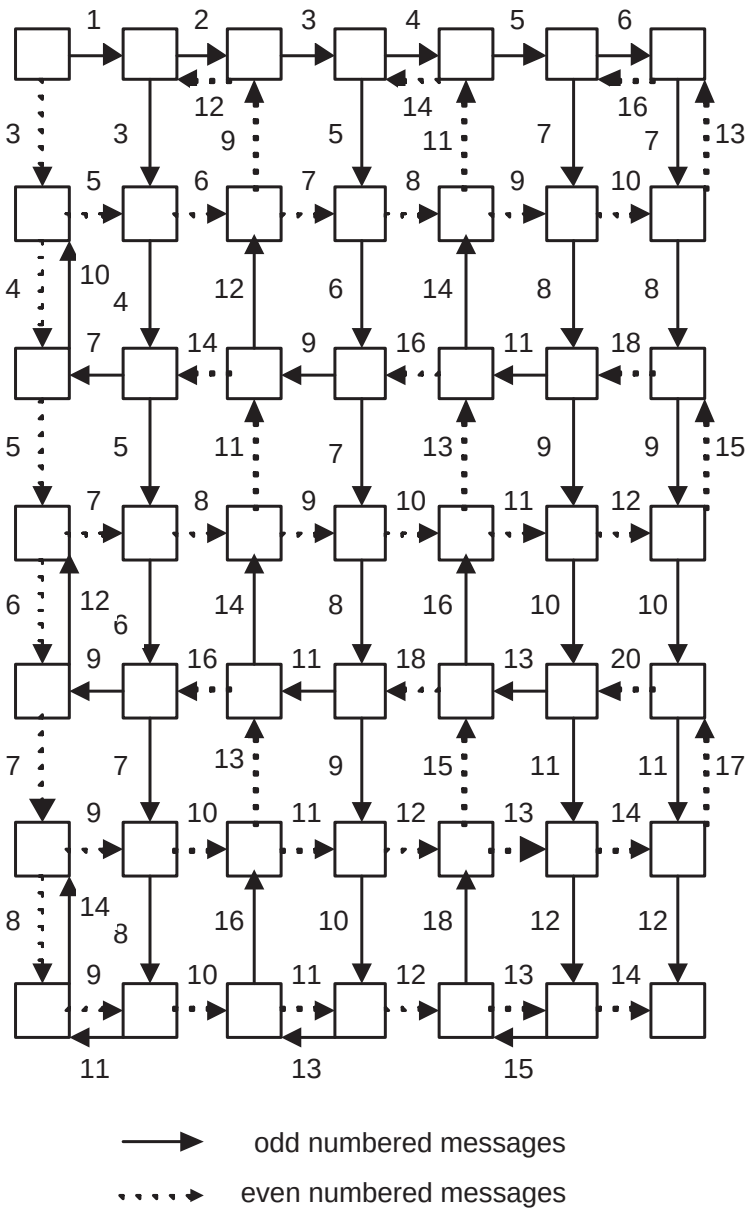

Figure 5.7: Performing multiple message broadcasting in a grid by dividing the messages into two sets [W99].

In order to calculate the coefficient of the message term, when using multiple sets of messages, we must first locate the node that requires the largest number of time units to broadcast one message from each of the k sets of messages (let us call this node b). We then calculate the total number of time units that node b requires to broadcast one message from each of the k sets and divide this

number by the number of sets of messages (i.e., k). For example, let us say that a given broadcast algorithm divides its messages into an odd set and an even set and that the busiest node (node b) requires 1 time unit to broadcast an odd message and 3 time units to broadcast an even message. Then, node b requires a total of 4 time units to complete broadcasting one message from each set. Since this is the busiest node and it requires 4 time units to broadcast 2 messages, the coefficient of the message term is $\frac{4}{2} = 2$.

In order to keep the coefficient of the message term equal to 2, every node of the graph must use no more than $2 \cdot k$ time units to complete broadcasting a single message from each of k sets of messages.

Referring to Figure (5.8), all nodes, except the originator and the node forming the other endpoint of the chord (let us call this node y), have exactly 2 edges connecting the node to other nodes of the graph. These non-chord endpoint nodes can be busy for at most 2 time units per message, since, according to the definition of multiple message broadcasting, a node can only send to an uninformed neighbor (i.e., a node can only receive each message once). Obviously, none of these non-chord endpoint nodes will increase the coefficient of the message term beyond 2. Therefore, the originator and node y are the only nodes that we will consider, when attempting to limit the coefficient of the message term to 2. We will refer to these nodes as branching nodes, since they each have the ability to inform more than one node of a message.

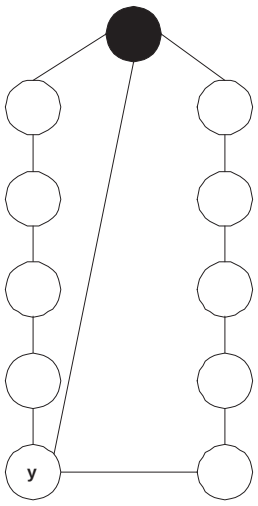

Figure 5.8: C_{11} with a single chord dividing the cycle in half.

According to the definition of multiple message broadcasting, at the end of the broadcast algorithm, every node in the network has received every message; thus, each node, except the originator, must require at least k time units to complete broadcasting all k messages (i.e., each node requires 1 time unit to receive each of the k messages). The originator knows all k messages at the beginning of the broadcast algorithm; thus, it will never spend any time units receiving messages (i.e., it can spend all of its $2 \cdot k$ time units sending messages). The originator is connected to the rest of the graph via exactly 3 edges; therefore, the originator may inform at most 3 uninformed nodes of a message (i.e., the nodes connected to it via the 3 edges - its neighbors). However, the originator cannot make use of all 3 of its edges when sending every message, since this would cause the coefficient of the message term to be $\frac{6}{2} = 3$ rather than 2.

We now consider node y; this node, like the originator, is connected to the rest of the graph via exactly 3 edges. However, unlike the originator, node y must receive every message. Thus, for any given message, node y may inform at most 2 uninformed neighbors, since it must use the third edge to receive the message. Node y cannot make use of both edges when sending each message,

since this would cause the coefficient of the message term to be 3 rather than 2. Furthermore, if node y is not informed via the originator, it may inform at most 1 uninformed neighbor, since the originator already knows all of the messages.

The chordless cycle algorithm is able to broadcast multiple messages in odd cycles in time equal to $2 \cdot (m-1) + \lfloor \frac{n}{2} \rfloor + 1$, where m is equal to the number of messages and n is equal to the number of nodes in the cycle. According to this equation, the coefficient of the message term is 2; the first message completes broadcasting at time equal to $\lfloor \frac{n}{2} \rfloor + 1$ (let us call this time t) and consecutive messages complete broadcasting every 2 time units later. In order to improve upon the efficiency of the chordless cycle algorithm for broadcasting multiple messages, while keeping the coefficient of the message term equal to 2, we must limit t. Since we are broadcasting messages from multiple sets, t is now the maximum amount of time it takes to broadcast the first message from a set. For example, let us say that we divide the messages into an odd set and an even set and that the first odd message completes broadcasting in time equal to $2 \cdot n + 1$ and the first even message completes broadcasting in time equal to $2 \cdot n + 2$. Then, $t = 2 \cdot n + 2$, since this the maximum time taken by a set to complete broadcasting of the first message from the set. Farley's lower bound for broadcasting in a chordless cycle and Farley's lower bound for broadcasting in a cycle with a single chord are equal (i.e., $2 \cdot (m-1) + \lfloor \frac{n}{2} \rfloor$, where m is equal to the number of messages and n is equal to the number of nodes in the cycle); therefore, the best we can do is to decrease the amount of time taken to broadcast the first message by a single time unit, which will, in turn, decrease the amount of time taken to broadcast each additional message by a single time unit.

It is not possible to broadcast a message in an odd length chordless cycle in time less than $\lfloor \frac{n}{2} \rfloor + 1$, since there are exactly 2 nodes as far away as possible from the originator. In other words, if each node makes use of exactly 2 edges, we cannot broadcast a message in an odd length cycle

in time less than $\lfloor \frac{n}{2} \rfloor + 1$. When attempting to broadcast a message in an odd length cycle in time equal to $\lfloor \frac{n}{2} \rfloor$, we must add a chord and make use of the chord. In order to keep the coefficient of the message term at 2, while improving upon the efficiency of the chordless cycle algorithm for broadcasting multiple messages, we want to divide the messages into multiple sets and use a different branching node for each set of messages.

Since we must use different branching nodes for each set of messages and we have only 2 branching nodes from which to choose, we will divide the messages into 2 sets (an odd set and an even set). One set of messages will use the originator as its branching node and the other set of messages will use node y as its branching node.

In order to improve upon the efficiency of the chordless cycle multiple message broadcasting algorithm, the first message from each set must complete broadcasting within $\lfloor \frac{n}{2} \rfloor$ time units. Broadcasting using a maximum of 2 edges of each node will not achieve this time bound, since when we make use of only 2 edges of each node, every message must follow one of just 2 possible paths (see Figure (5.9)). In order to decrease the number of time units required to broadcast the first message, we must make use of at least 3 paths. The only nodes that have the ability to create 3 paths are the 2 branching nodes (i.e., the originator and node y), each of which has 3 neighbors. The originator may send to all three of its neighbors, allowing it to create 3 paths; however, node y must receive the message through one of its neighbors, which leaves it a maximum of 2 neighbors that it may inform, allowing it to create a maximum of 2 paths (see Figure (5.10)). In order for node y to inform 2 of its uninformed neighbors, it must receive the message from the originator.

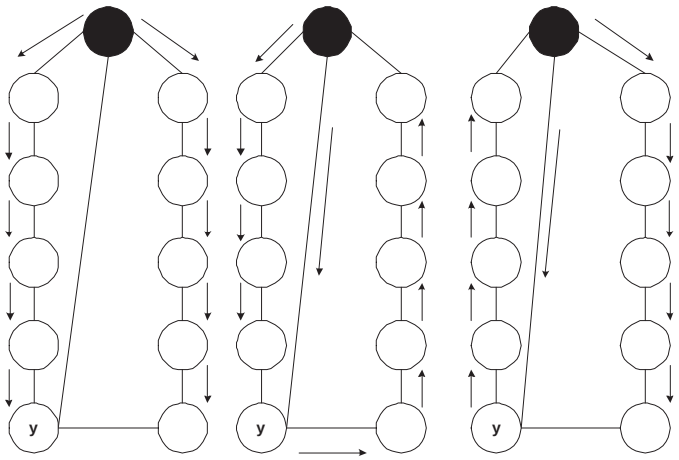

Figure 5.9: When each node uses a maximum of 2 of its edges for broadcasting, all nodes are informed via 1 of only 2 paths.

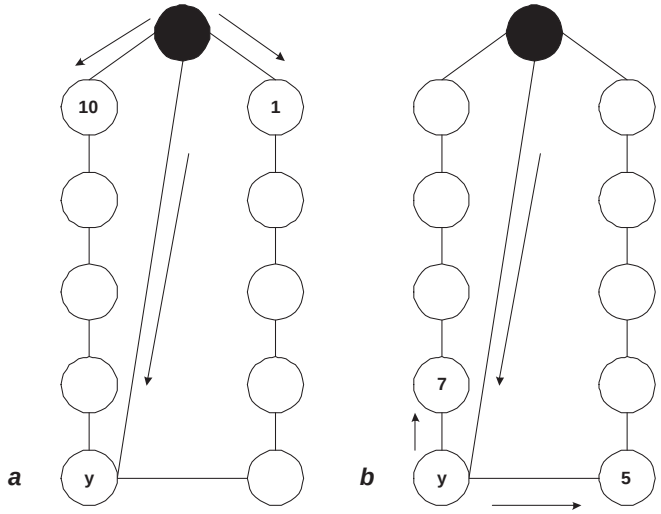

Figure 5.10: (*a*) The movement of messages out of the originator. (*b*) The movement of messages out of node y.

Without loss of generality, we will assume that odd messages will use the originator as their branching node; when sending an odd message, the originator will use all 3 of its neighbors. Node y is one of the originator's neighbors; therefore, node y will receive odd messages from the originator via the chord. However, once node y receives an odd message, it may not send the message to either of its neighbors, since node y may not be busy for more than 4 time units for every 2 messages (i.e., the coefficient of the message term must remain 2) and node y is used

as a branching node for even messages (i.e., it is busy for 3 time units for each even message). According to the definition of a multiple message broadcast algorithm, every node in the network must receive every message; since node y may not send an odd message to any of its neighbors, all nodes except the originator and its neighbors must receive odd messages from either node 1 or node $n-1$, when the nodes are numbered clockwise, beginning with the originator numbered 0. More specifically, if we label the larger cycle formed by the chord cycle A and the smaller cycle formed by the chord cycle B, all nodes in cycle A, except the originator and its neighbors, will be informed of an odd message via a path through node 1. Likewise, all nodes in cycle B, except the originator and its neighbors, will be informed of an odd message via a path through node $n-1$. See Figure (5.11) for an example.

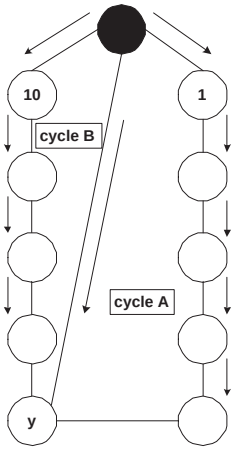

Figure 5.11: The movement of odd messages through C_{11} with a single chord dividing the cycle in half.

Using the broadcast scheme for odd messages presented in Figure (5.11), it is possible to broadcast the first message odd message in time equal to $\lfloor \frac{n}{2} \rfloor$. Figure (5.12) gives an example of broadcasting an odd message in C_{11} using this scheme.

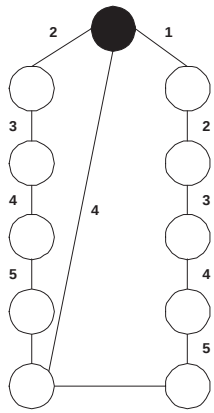

Figure 5.12: Broadcasting an odd message in C_{11} in time equal to $\lfloor \frac{n}{2} \rfloor$.

Now that we have an algorithm that achieves the $\lfloor \frac{n}{2} \rfloor$ time bound for sending odd messages, we will attempt to find a second algorithm that uses node y as its branching node and also achieves this time bound. Since the originator is busy sending odd messages at time step 1 and time step 2, this second algorithm will begin broadcasting 2 time units later than the first algorithm and will send even messages.

When sending an even message, node y will act as the branching node, using all 3 of its neighbors. We have already shown that in order for node y to inform the maximum number of uninformed nodes, it must receive each even message from the originator. Node y will then inform node $\lfloor \frac{n}{2} \rfloor$ and node $\lfloor \frac{n}{2} \rfloor + 2$ of the message. Once the originator sends an even message to node y, it may not send the even message to any of its other neighbors, since it is busy for 3 time units for every odd message and it may not be busy for more than 4 time units for every 2 messages. Therefore, all nodes except node y and its neighbors must receive even messages from either node $\lfloor \frac{n}{2} \rfloor$ or node $\lfloor \frac{n}{2} \rfloor + 2$. More specifically, all nodes in cycle A, except node y and its neighbors, must be informed via a path through node $\lfloor \frac{n}{2} \rfloor$. Likewise, all nodes in cycle B, except node y and its neighbors, must be informed via a path through node $\lfloor \frac{n}{2} \rfloor + 2$. See Figure (5.13) for an example.

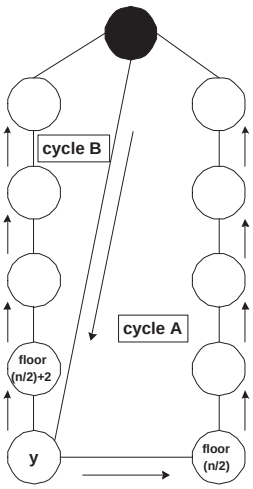

Figure 5.13: The movement of even messages through C_{11} with a single chord dividing the cycle in half.

As shown in Figure (5.13), when broadcasting using node y as the branching node, every even message must follow one of just 2 possible paths. We have already shown that it is not possible to complete broadcasting in time equal to $\lfloor \frac{n}{2} \rfloor$, when just 2 paths are used. Therefore, we conclude that it is not possible to broadcast multiple messages in an odd length cycle with a single chord in time less than $2 \cdot (m-1) + \lfloor \frac{n}{2} \rfloor + 1$.

5.3 Multiple Message Broadcasting in a Cycle with Two Chords

Does the addition of a second chord decrease the time required to broadcast multiple messages in a cycle? We would like to begin our investigation by comparing Farley's upper and lower bounds on a chordless cycle with Farley's upper and lower bounds on a cycle with two chords; however, before we can do this, we must calculate the diameter of a cycle with two chords. As shown in Chapter 4, positioning the chords so that they share an endpoint and cut the cycle into three smaller, equal size cycles and the originator forms the shared endpoint, allows us to inform the maximum

number of nodes in the minimum amount of time. Therefore, we will assume that the originator is the shared endpoint and we will use the chords to divide the cycle into three smaller, equal size cycles. This can be done using Algorithm (4.3.1); see Figure (4.3) for an example.

In order to calculate the diameter for this graph, we must locate two nodes (x and y) in the graph that are the maximum distance apart; then, calculate the distance between these two nodes. Obviously, these nodes must be in different cycles. According to Algorithm (4.3.1), cycle A is always as large or larger than cycle B and cycle C; therefore, one of the nodes (let us say node x) will be located in cycle A. If node x is located in cycle A, then node y must be located in either cycle B or cycle C and will be as far away as possible from node x. Let us say that node y_1 is the node in cycle B that is located as far away as possible from any node in cycle A and that node y_2 is the node in cycle C that is located as far away as possible from any node in cycle A. We must determine which node (y_1 or y_2) is furthest away from any node in cycle A. According to Algorithm (4.3.1), either cycle B and cycle C are the same size, or cycle B is one node larger than cycle C. Focusing on the number of nodes in these cycles that are also located in cycle A, we find that 2 nodes of cycle B are also located in cycle A, whereas only a single node of cycle C is also located in cycle A. The faster a message can move from node y into cycle A, the faster the message can move from node y to node x. Thus, node y_2 is at least as far away from any node in cycle A as is node y_1; thus, node y will be located in cycle C.

Now that we know the general location of node x and node y, we will begin by locating the node in cycle A that is as far away as possible from any node in cycle C. Since the shared endpoint is the only node that is a member of both cycle A and cycle C, the shared endpoint will be the first node from cycle C that we encounter when moving from node x to node y. Therefore, we will label the node in cycle A that is as far away as possible from the shared endpoint, node x. We can

use the following formula to locate node x: $\lceil \frac{c1}{2} \rceil$, where $c1$ is defined by Algorithm (4.3.1). Figure (5.14) uses this formula to locate node x (remember, the nodes of the cycle have been consecutively numbered clockwise beginning with the shared endpoint, which is numbered node 0).

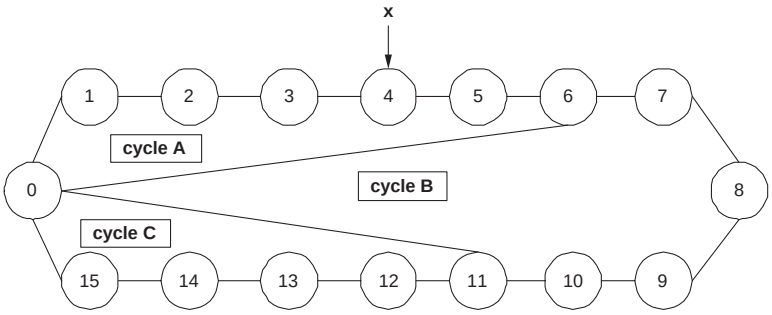

Figure 5.14: Locating node x in C_{16} with two chords.

We must now locate a node, y, that is as far away as possible from node x. This node will be located in cycle C and will be as far away as possible from the shared endpoint. The following formula can be used to calculate the position of node y: $n - \lfloor \frac{w}{2} \rfloor$, where n represents the number of nodes in the cycle and w is defined by Algorithm (4.3.1). Figure (5.15) uses this formula to locate node y.

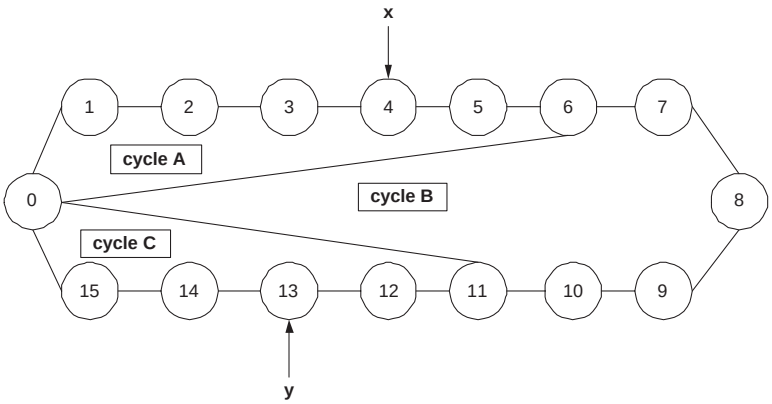

Figure 5.15: Locating node y in C_{16} with two chords.

Now that we have established the location of node x and node y, we must find the fastest way to move from node x to node y. The shared endpoint (node 0) is contained in both cycle A and cycle

C; our goal is to arrive at node 0 as quickly as possible (i.e., we want to move the message from one cycle to the other as quickly as possible). Examining the location of node x ($\lceil \frac{c1}{2} \rceil$), we find that the shortest path from node x to node 0 is by way of the first chord. Focusing on cycle A, node x is as far away as possible from node 0. Since cycle A is itself a chordless cycle of length $c1$, the distance between node 0 and node x is $\lfloor \frac{c1}{2} \rfloor$ (i.e., the diameter of chordless cycle A). Examining the location of node y (i.e., $n - \lfloor \frac{w}{2} \rfloor$), we find that the shortest path from node 0 to node y is by way of the edges of the original chordless cycle. Focusing on cycle C, node y is as far away as possible from node 0. Since cycle C is itself a chordless cycle of length w, the distance between node 0 and node y is $\lfloor \frac{w}{2} \rfloor$ (i.e., the diameter of chordless cycle C). Combining the distances between node 0 and node x and node 0 and node y, we have $\lfloor \frac{c1}{2} \rfloor + \lfloor \frac{w}{2} \rfloor$, which is the diameter of the graph.

Cycle A and cycle C each contain approximately one-third of the total nodes, n in the graph. Furthermore, according to Algorithm (4.3.1), cycle A will either be the same size as cycle C or cycle A will contain one more node than cycle C. Thus, the worst case for the diameter occurs when cycle A and cycle C both contain the same number of nodes (i.e., $\frac{n+4}{3}$ nodes), which gives us a diameter for the graph of $2 \cdot \lfloor \frac{\frac{n+4}{3}}{2} \rfloor$.

Now that we have calculated the diameter for a cycle with two chords that cut the cycle into three smaller, equal size cycles and share an endpoint, we can compare Farley's upper and lower bounds on a chordless cycle with Farley's upper and lower bounds on a cycle with two chords.

All nodes in a chordless cycle have degree 2 and the diameter of such a cycle is $\lfloor \frac{n}{2} \rfloor$, where n is the number of nodes in the cycle. This gives us the following bounds for a chordless cycle.

$$2 \cdot (m-1) + \lfloor \frac{n}{2} \rfloor \leq B_m(G) \leq 2 \cdot (m-1) + (n-1)$$

The chordless cycle multiple message broadcasting algorithm is able to achieve Farley's lower

bound in even cycles and requires only a single additional time unit to broadcast in odd cycles (i.e., $2 \cdot (m-1) + \lfloor \frac{n}{2} \rfloor + 1$).

On the other hand, the maximum degree of any node in a cycle with two chords that share an endpoint is 4; while we have shown that, in the worst case, the diameter of such a cycle is $2 \cdot \lfloor \frac{\frac{n+4}{3}}{2} \rfloor$. This gives us the following bounds for a cycle with two chords.

$$2 \cdot (m-1) + 2 \cdot \lfloor \frac{\frac{n+4}{3}}{2} \rfloor \leq B_m(G) \leq 4 \cdot (m-1) + (n-1)$$

The lower bound of a cycle with two chords that share an endpoint is better than the lower bound of an equal size chordless cycle. Therefore, we will attempt to find an algorithm that will achieve this lower bound.

We already know that a simple modification to our algorithm for broadcasting a single message in a cycle with two chords that share an endpoint, such as repeating the single message broadcast scheme m times or having each call take m time units, is not efficient.

When attempting to find an efficient algorithm for broadcasting in a cycle with two chords, in order to be at least as efficient as the chordless cycle multiple message broadcasting algorithm, we must keep the coefficient of the message term equal to 2. The coefficient of the message term correlates to the delay between messages and a delay is introduced for each time unit that a node is busy broadcasting a message. In order to keep the coefficient of the message term equal to 2, we must keep branching to a minimum.

If we treat the messages as a single set, sending each message out from the originator following the same broadcast scheme, then, in order to limit the coefficient of the message term to 2, a node can be busy for no more than 2 time units (allowing for a delay of 2 between messages). This means that both the originator and the node at the other end of each chord can use at most 2 of

their edges. The originator has 4 edges; in determining which edges to use, we could choose both chords, both original chordless cycle edges or one chord and one original chordless cycle edge (see Figure (5.16)).

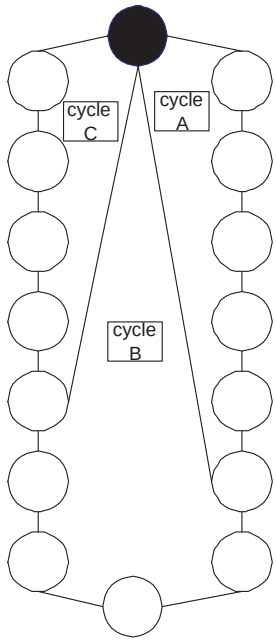

Figure 5.16: C_{16} with two chords, which share an endpoint, dividing the cycle into three smaller, equal size cycles.

If we choose to use both chords as the originator's broadcasting edges, then each non-originator node at the end of the chord could send in at most one direction, since it can be busy for at most 2 time units and one of those time units is needed for receiving the message from the originator. Since each non-originator endpoint can only choose to send in a single direction, there will always be one segment of the graph that will not receive the message. For example, let us label the non-originator endpoint of the chord separating cycle A and cycle B, node a and the non-originator endpoint of the chord separating cycle B and cycle C, node b and let node a receive the message and send to its neighbor in cycle A. Then, let node b receive the message and send to its neighbor in cycle C. There is no way for the non-chord endpoint nodes of cycle B to receive the message

without introducing a delay of 3 (see Figure (5.17)). Therefore, it is not feasible to use both chords when sending a single set of messages and limiting the coefficient of the message term to 2.

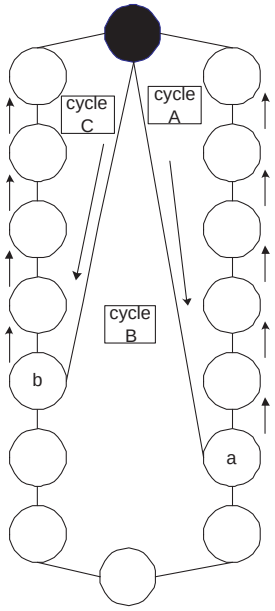

Figure 5.17: Broadcasting in C_{16}, using just two of each node's edges, when the originator uses its chords as broadcasting edges.

If we choose both original chordless cycle edges as the originator's broadcasting edges, then we are simply broadcasting in a chordless cycle. An efficient algorithm already exists for broadcasting multiple messages in a chordless cycle.

Our final option is to make use of one chord and one original chordless cycle edge as the originator's broadcasting edges. Again, in order to limit the coefficient of the message term to 2, the non-originator chord endpoint can only send in a single direction, since it must receive the message from the originator. In order to inform all of the nodes in the graph, the non-originator chord endpoint must send in the opposite direction of the chordless cycle edge that is being used (see Figure (5.18)).

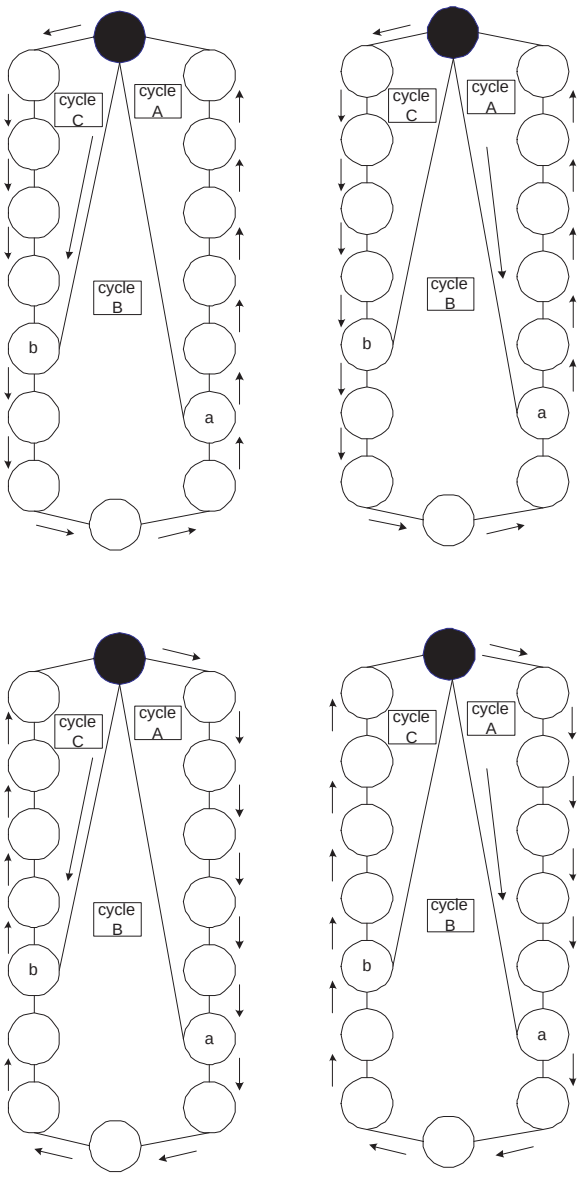

Figure 5.18: Broadcasting in C_{16}, when the originator uses one chord and one original chordless cycle edge as broadcasting edges.

Regardless of the chordless cycle edge and chord that we choose, either the original chordless cycle edge will need to inform approximately one-third of the nodes via a single path and the non-originator chord endpoint will need to inform the remaining nodes (i.e., approximately two-thirds of the nodes) via a single path or vice versa. Although, we have limited the coefficient of the message term to 2, we have increased the time required to send the first message from, at most,

$\lfloor \frac{n}{2} \rfloor + 1$, which can be achieved using the chordless cycle multiple message broadcasting algorithm, to approximately $\frac{2 \cdot n}{3}$. Therefore, using one chord and one original chordless cycle edge will not improve upon the amount of time needed to broadcast multiple messages.

We conclude that, when performing multiple message broadcasting using a single set of messages in a cycle with two chords, using the chordless cycle algorithm for multiple message broadcasting is more efficient than using an algorithm that makes use of one or more of the chords. We will now attempt to improve upon the efficiency of the chordless cycle multiple message broadcasting algorithm by using multiple sets of messages, in order to develop a multiple message broadcasting algorithm that makes use of one or more of the cycle's chords.

When broadcasting using multiple sets of messages, we hope to limit the coefficient of the message term by requiring each set of messages to use a different broadcast scheme to deliver the message to every node in the graph. In order to be at least as efficient as the chordless cycle algorithm, we need to keep the coefficient of the message term equal to 2. Therefore, every node of the graph must use no more than $2 \cdot k$ time units to complete broadcasting a single message from each of k sets of messages.

Referring to Figure (5.16), all nodes, except the originator and the nodes forming the other endpoint of the chords (let us call these nodes a and b), have exactly 2 edges connecting the node to other nodes of the graph. These non-chord endpoint nodes will be busy for at most 2 time units per message (one time unit receiving the message and possibly, another time unit sending to an uninformed neighbor). Obviously, none of these non-chord endpoint nodes will increase the coefficient of the message term beyond 2; therefore, the originator, node a and node b are the only nodes that we will consider when attempting to limit the coefficient of the message term to 2. We will refer to these nodes as branching nodes, since they each have the ability to inform more than

one node.

The chordless cycle algorithm is able to broadcast multiple messages in time equal to $2 \cdot (m - 1) + \lfloor \frac{n}{2} \rfloor + 1$, in odd cycles, and in time equal to $2 \cdot (m - 1) + \lfloor \frac{n}{2} \rfloor$, in even cycles, where m is equal to the number of messages and n is equal to the number of nodes in the cycle. According to these equations, the coefficient of the message term is 2, the first message completes broadcasting at time equal to $\lfloor \frac{n}{2} \rfloor + 1$, in odd cycles (let us call this time t_1), and at time equal to $\lfloor \frac{n}{2} \rfloor$, in even cycles (let us call this time t_2), and in each case, consecutive messages complete broadcasting every 2 time units later. In order to improve upon the efficiency of the chordless cycle algorithm for multiple messages, while keeping the coefficient of the message term equal to 2, we must limit t_1 and t_2. Since we are broadcasting messages from multiple sets, t_1 is now the maximum amount of time it takes to broadcast the first message from a set in an odd cycle and t_2 is the maximum amount of time it takes to broadcast the first message from a set in an even cycle. For example, let us say that we are working with an even cycle and we divide the messages into an odd set and an even set and that the first odd message completes broadcasting in time equal to $2 \cdot n + 2$ and the first even message completes broadcasting in time equal to $2 \cdot n + 1$. Then, $t_2 = 2 \cdot n + 2$, since this the maximum time taken by a set to complete broadcasting of the first message from the set in a even cycle. In the above example, it is important to realize the difference between an odd/even cycle and an odd/even set. An odd/even cycle refers to the number of nodes in the original cycle; if the number of nodes is odd, then the cycle is referred to as an odd cycle. Likewise, if the number of nodes is even, then the cycle is referred to as an even cycle. Regardless of whether the original cycle is odd or even, the messages being broadcast through the cycle are divided into two sets (i.e., an odd set and an even set). Farley's lower bound for broadcasting in a chordless cycle is equal to $2 \cdot (m - 1) + \lfloor \frac{n}{2} \rfloor$ and Farley's lower bound for broadcasting in a cycle with two chords is equal

to approximately $2 \cdot (m-1) + 2 \cdot \lfloor \frac{\frac{n+4}{3}}{2} \rfloor$; where m is equal to the number of messages and n is equal to the number of nodes in the cycle, for both bounds. Since the first term in each of the above equations is equal (i.e., $2 \cdot (m-1)$), the best we can do is to decrease the amount of time taken to broadcast the first message, which will, in turn, decrease the amount of time taken to broadcast each additional message.

If each node makes use of exactly 2 edges, we cannot broadcast a message in a cycle with two chords in time less than that taken by the chordless cycle algorithm, since when we complete broadcasting making use of only 2 edges of each node, every message must follow one of just 2 possible paths (see Figure (5.19)). In order to decrease the number of time units required to broadcast the first message, we must make use of at least 3 paths. The only nodes that have the ability to create multiple paths are the branching nodes (i.e., the originator, node a and node b). The originator may send to all 4 of its neighbors, allowing it to create 4 paths. Node a and node b each have 3 neighbors; however, they must receive each message through one of their neighbors, which leaves them each a maximum of 2 neighbors that they may inform, allowing them each to create a maximum of 2 paths. Since every node must receive every message and the originator knows all messages at the beginning of the multiple message broadcasting algorithm, in order for node a and node b to each inform 2 of their neighbors, they must each receive the message from the originator.

We will begin by dividing the messages into 2 sets (an odd set and an even set). Since we must limit the coefficient of the message term to 2 and the originator must send out both odd and even messages, the originator can be busy for no more than 3 time units when sending the first message (i.e., the odd message). Thus, the originator can send an odd message to one, two or three of its

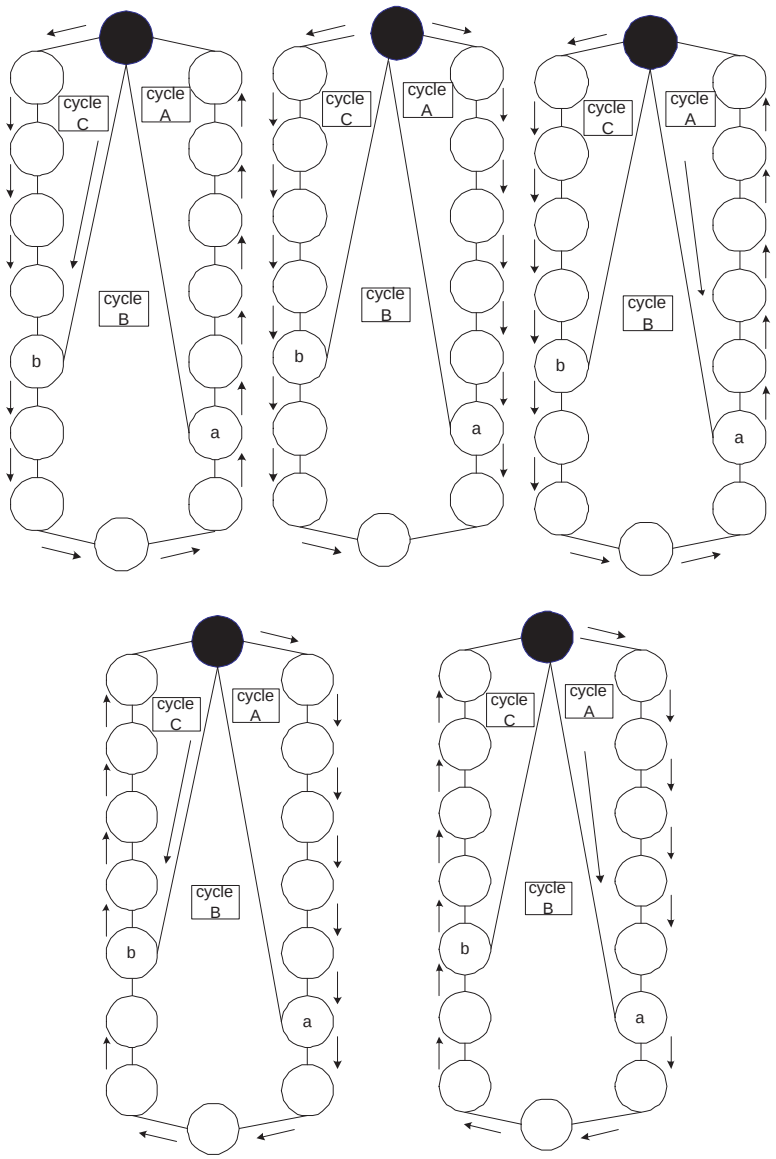

Figure 5.19: Completing broadcasting in C_{16}, making use of just 2 of each node's edges.

neighbors.

If the originator sends an odd message to a single neighbor, either it will send the message to one of its original chordless cycle neighbors (let us call these neighbors node x and node y), or it will send the message to node a or node b. If the originator sends to node x or node y, then all nodes will receive the message via a single path; on the other hand, if the originator sends to node a or node b, then all nodes will receive the message via one of just 2 paths (see Figure (5.20)). Since

we must make use of at least 3 paths, in order to improve upon the time taken by the chordless cycle algorithm, allowing the originator to send to a single node will not provide an algorithm that is more efficient than the chordless cycle multiple message broadcasting algorithm.

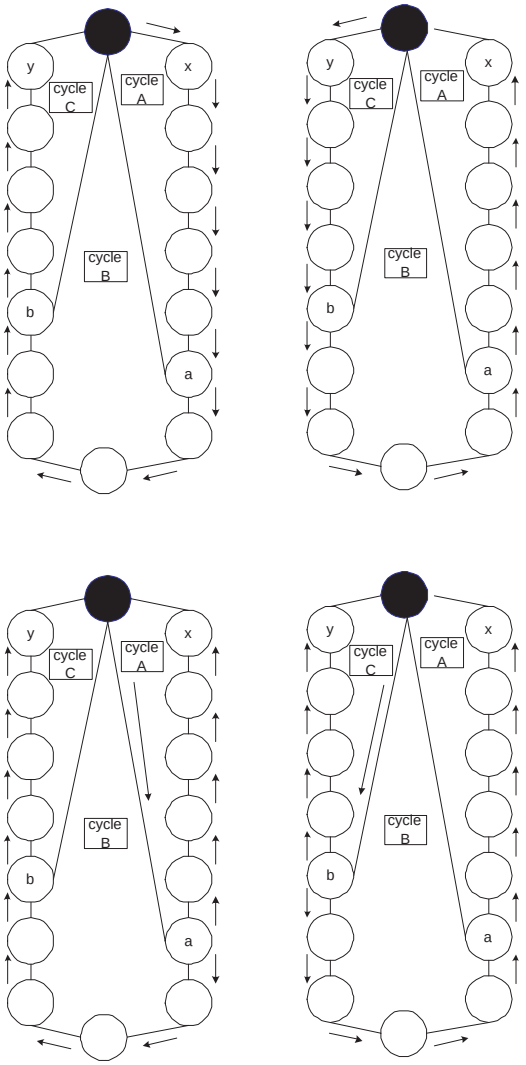

Figure 5.20: Broadcasting an odd message in C_{16}, when the originator sends to a single neighbor.

In order to improve upon the time taken by the chordless cycle multiple message broadcasting algorithm, the maximum time taken to broadcast the first message from any set (odd or even) must be less than $\lfloor \frac{n}{2} \rfloor$. Thus, if the originator sends an odd message to 3 of its neighbors, in order to limit the message coefficient to 2, it must send an even message to a single neighbor. However, we have

already determined that allowing the originator to send to a single neighbor will create at most 2 paths for the message to follow, which will not improve upon the time taken by the chordless cycle multiple message broadcasting algorithm.

We will now investigate allowing the originator to make use of 2 of its edges to send a message from each set. When sending an odd message, the originator may send to one of the following pairs of neighbors xy, xa, by, ab, xb or ay. We will not allow the originator to send to node x and node y, because this would result in broadcasting using the chordless cycle algorithm and we are attempting to find a more efficient algorithm.

If the originator sends an odd message to node x and node a, in order to complete broadcasting, node a would be responsible for informing approximately two-thirds of the nodes via a single path (see Figure (5.21)). The most nodes informed via a single path using the chordless cycle multiple message broadcasting algorithm is approximately one-half; therefore, allowing the originator to send an odd message to node x and node a would not decrease the time taken to send a message. Similarly, allowing the originator to send an odd message to node b and node y would cause node b to be responsible for informing approximately two-thirds of the nodes via a single path and would not improve upon the efficiency of the chordless cycle multiple message broadcasting algorithm.

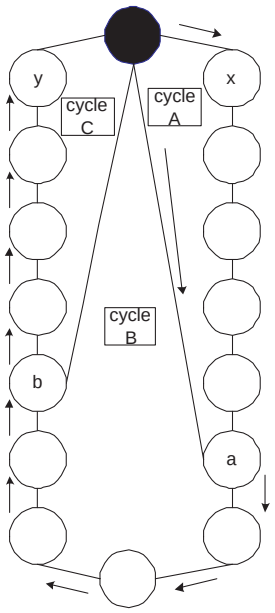

Figure 5.21: Broadcasting an odd message in C_{16}, when the originator informs node x and node a.

If the originator sends an odd message to node a and node b and these nodes both send to each of their uninformed neighbors, they will each be busy with an odd message for 3 time units, leaving them each just a single time unit to spend on an even message. In order to spend a single time unit on an even message, both nodes must receive the even message; however, neither of them would be permitted to pass the even message on to either of their neighbors, since this would increase the coefficient of the message term beyond 2.

If the originator informs node a and node b of an even message, then, it may not send an even message to any other node, since it can send to at most 2 of its neighbors. Thus, if the originator sends an even message to node a and node b, who were each busy for 3 time units with an odd message, the originator, node a and node b would be the only nodes that would ever receive an even message. This would not be a legal broadcast scheme, since, according to the definition of multiple message broadcasting, at the end of a multiple message broadcasting algorithm, every node in the network has received every message.

If the originator does not send an even message to node a or node b, then it must send an even message to node x and node y and both node a and node b must receive an even message from one of their non-originator neighbors; however, neither node a nor node b will be permitted to pass the message on to either of their neighbors. In this scenario, node x will inform node a and node y will inform node b; however, none of the non-chord endpoint nodes of cycle B will ever receive the message, resulting in an illegal broadcast scheme (see Figure (5.22)).

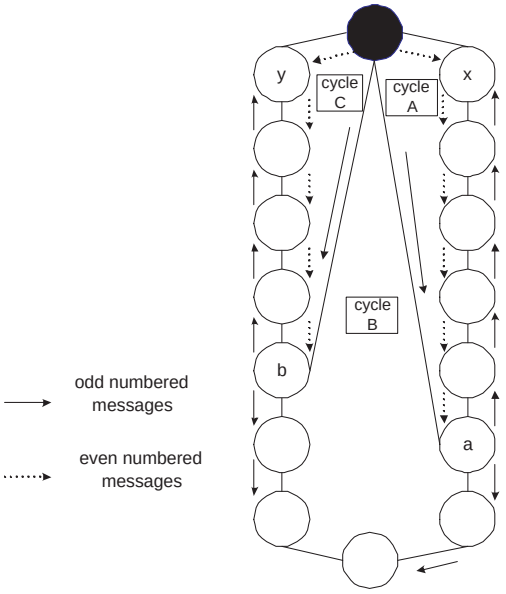

Figure 5.22: Problems that occur with broadcasting when the originator informs node a and node b of an odd message and they each pass the message on to both of their uninformed neighbors; then, the originator sends an even message to node x and node y.

If the originator sends an even message to node a and node x, then, there would exist nodes in cycle B and cycle C that would never receive the message, since node a is not permitted to pass the message on to any other node (see Figure (5.23)). Thus, this scenario would result in an illegal broadcast scheme; a similar argument can be made, if the originator were to send an even message to node a and node y, node b and node x or node b and node y. If the originator sends an even message to node a and node y, then there would exist nodes in cycle A and cycle B that would

never receive the message. If the originator sends an even message to node b and node x, then there would exist nodes in cycle B and cycle C that would never receive the message. Finally, if the originator sends an even message to node b and node y, then there would exist nodes in cycle A and cycle B that would never receive the message.

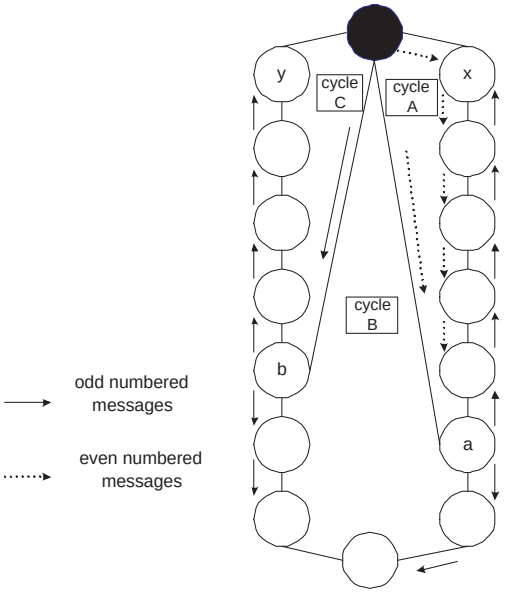

Figure 5.23: Problems that occur with broadcasting when the originator informs node a and node b of an odd message and they each pass the message on to both of their uninformed neighbors; then, the originator sends an even message to node a and node x.

Therefore, if the originator sends an odd message to node a and node b, either node a or node b (not both) may send to both of their uninformed neighbors. In order for all of the nodes to be informed of an odd message, either node a or node b must inform both of their uninformed neighbors. Without loss of generality, let us assume that node a is the node that sends an odd message in both directions. Then node a must be informed of an even message, however, it cannot pass this message on to another node.

If the originator informs node a of an even message, then the originator can only inform one other node of an even message, either node x, node b or node y. If the originator chooses to inform

node a and node x, then there will be nodes in cycle B and cycle C that will never receive an even message (see Figure (5.24)). If the originator informs node a and either node b or node y of an even message, then there will be nodes in cycle A (and either cycle B or cycle C - if node b is used) that will never receive an even message (see Figure (5.25)).

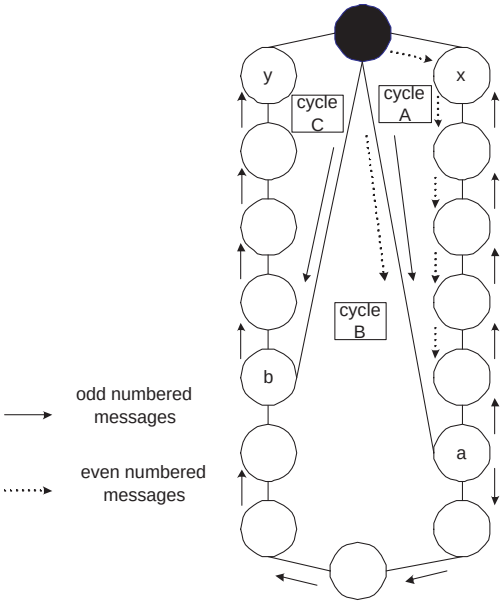

Figure 5.24: Problems with broadcasting when the originator sends an odd message to node a and node b and node a passes the message on to both of its uninformed neighbors; then, the originator sends an even message to node a and node x.

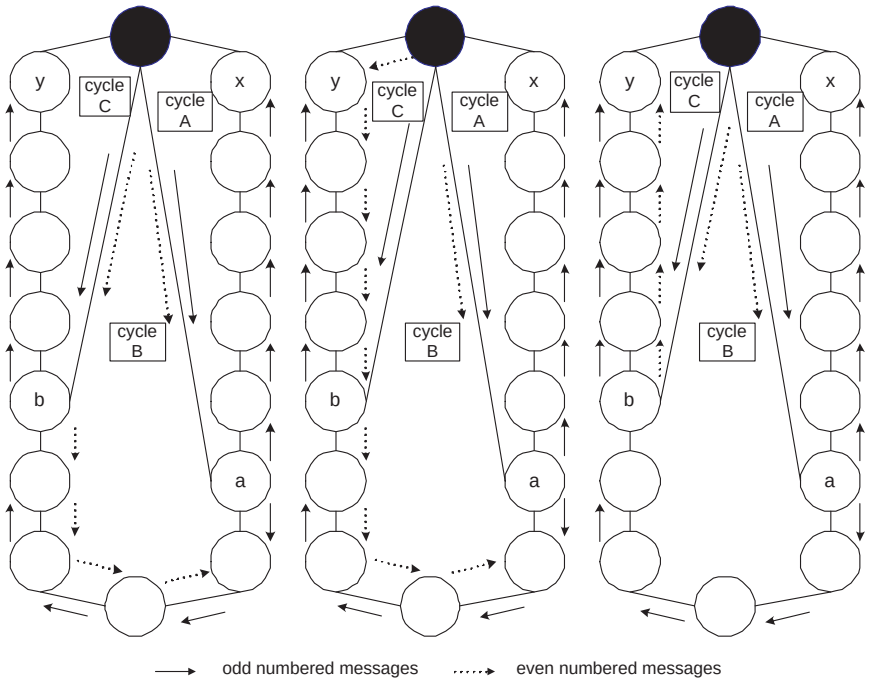

Figure 5.25: Problems with broadcasting when the originator sends an odd message to node a and node b and node a passes the message on to both of its uninformed neighbors; then, the originator sends an even message to node a and either node b or node y.

If the originator does not inform node a of an even message, then, node a must be informed via its neighbor in cycle A or its neighbor in cycle B. Node a cannot pass an even message on to any of its neighbors; therefore, in order for all of the nodes in cycle A to receive an even message, the originator must send to node x. Since the originator is not sending to node a and it must send to 2 of its neighbors, it will send to node x and either node b or node y. Since node b was busy for 2 time units with odd messages, it can only send in a single direction, once it receives an even message; therefore, in order to inform all of the nodes in cycle B and cycle C of an even message, the originator must send to node y. However, if we allow the originator to send even messages out through node x and node y, we have just 2 paths for even messages to follow (see Figure (5.26)); this broadcast scheme would not improve upon the efficiency of the chordless cycle multiple message broadcasting algorithm.

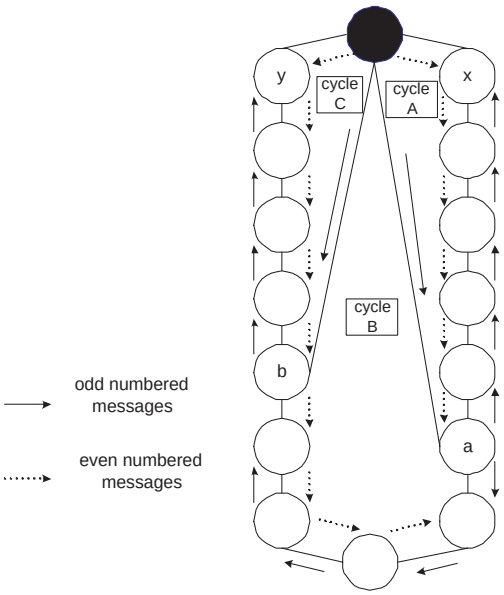

Figure 5.26: Broadcasting when the originator sends an odd message to node a and node b and node a passes the message on to both of its uninformed neighbors; then, the originator sends an even message to node x and node y.

If the originator sends an odd message to node x and node b and node b informs just one of its uninformed neighbors, then, all nodes would be informed via just one of 2 paths (see Figure (5.27)), which would not improve upon the efficiency of the chordless cycle multiple message broadcasting algorithm. On the other hand, if node b informs both of its uninformed neighbors, then node b will receive an even message; however, it will not be permitted to pass the message on to any of its neighbors.

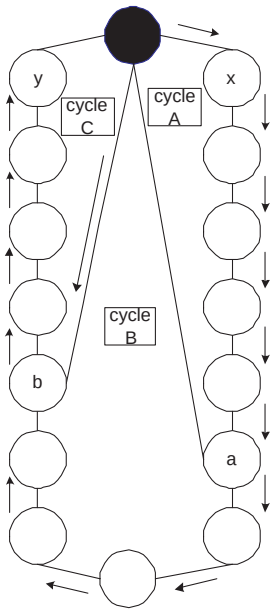

Figure 5.27: Completing broadcasting of an odd message in C_{16}, when the originator sends to node x and node b and node b sends to just one of its uninformed neighbors.

If node b received an even message from the originator, then, the originator could only send to one other node, which would cause nodes from at least one cycle to never receive the message. This scenario is similar to that explained by Figure (5.23).

If node b is not informed of an even message via the originator, then, it must be informed by either its neighbor in cycle C or its neighbor in cycle B and it cannot pass the message on to any of its neighbors. In order for all of the nodes of cycle C to be informed of the even message, node y must receive the message from the originator. Since the originator is not sending to node b and it must send to 2 of its neighbors, it will send to node y and either node x or node a. If the originator sends to node x, then all nodes would be informed via just one of 2 paths (similar to Figure (5.26)), which would not improve upon the efficiency of the chordless cycle multiple message broadcasting algorithm. If the originator sends to node a and node a was busy with an odd message for just a single time unit (i.e., it simply received an odd message and did not pass the message on to one

of its neighbors), then node a can send to both of its uninformed neighbors (see Figure (5.28)). This scenario gives us what appears to be a feasible solution to our multiple message broadcasting problem.

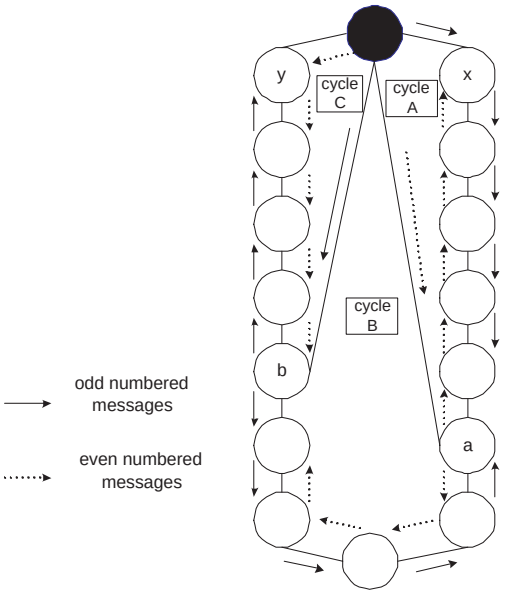

Figure 5.28: A solution to our multiple message broadcasting problem that appears to be feasible.

We will now attempt to assign times to our broadcast schemes, such that, the first message from each set completes broadcasting in time less than that taken by the chordless cycle multiple message broadcasting algorithm (i.e., less than $\lfloor \frac{n}{2} \rfloor$ time units). Figure (5.29) gives an example of broadcasting an odd message in C_{16} using our scheme. Now that we have an algorithm for odd messages that improves upon the time taken by the chordless cycle multiple message broadcasting algorithm, we will attempt to assign times to our even broadcast scheme. In Figure (5.28), even messages travel across each edge in the opposite direction of odd messages and most nodes in the cycle are busy for precisely 4 time units broadcasting one message from each set. In order to decrease the time taken to send an odd message, our odd broadcast scheme requires that we pass an odd message from one node to the next at consecutive time intervals. Since the coefficient of

the message term is $\frac{4}{2} = 2$, each of our broadcast schemes will be repeated every 4 time units (i.e., time units are assigned modulo 4). Referring to Figure (5.30), if odd messages move through the graph from node e through node a at consecutive time intervals, then we must carefully assign the even time intervals, so that at any given time, every node in the graph is idle, is sending a message or is receiving a message. As shown in Figure (5.30), node b is busy receiving an odd message at time 3 mod 4 and sending that same odd message the very next time step (i.e., 0 mod 4). When node b receives an even message, it must pass this even message on to node c at time 1 mod 4, since it is busy with an odd message at time unit 3 mod 4 and time unit 0 mod 4 and node c is busy with an odd message at time unit 2 mod 4 and time unit 3 mod 4. Once node c receives an even message at time unit 1 mod 4, it cannot pass this message on to node d until time unit 0 mod 4; however, this produces a delay of 3 time units between the time that node c receives an odd message and the time that node c can pass an odd message on to node d. Since even messages cannot begin broadcasting until 2 time units after odd messages, we cannot delay the movement of an even message throughout the graph, while improving upon the time taken to broadcast in the chordless cycle multiple message broadcasting algorithm. We could delay the time at which an odd message moves through the graph by at most one time unit; however, this would not be sufficient to help the movement of even messages. A solution to the time assignments could be to send odd messages during odd time steps and even messages during even time steps (see Figure (5.31)); however, this assignment would cause our broadcast scheme to take longer than the chordless cycle multiple message broadcasting algorithm. See Figure (5.32) for an example.

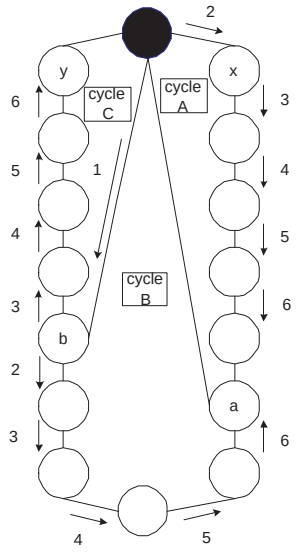

Figure 5.29: Broadcasting an odd message in C_{16} using the scheme shown in Figure (5.28).

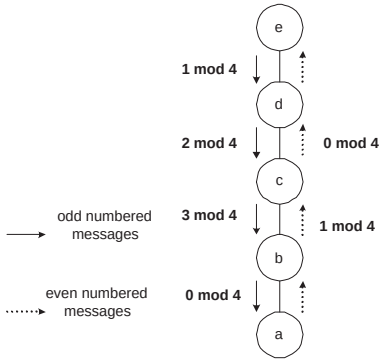

Figure 5.30: An example of why moving odd messages and even messages through the graph at consecutive time intervals is not feasible.

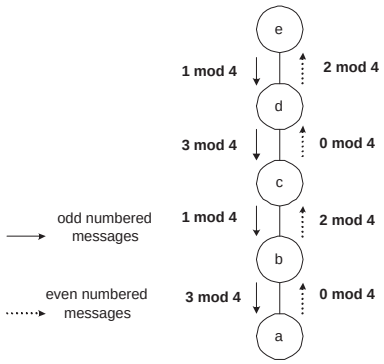

Figure 5.31: Moving odd messages during odd time steps and even messages during even time steps.

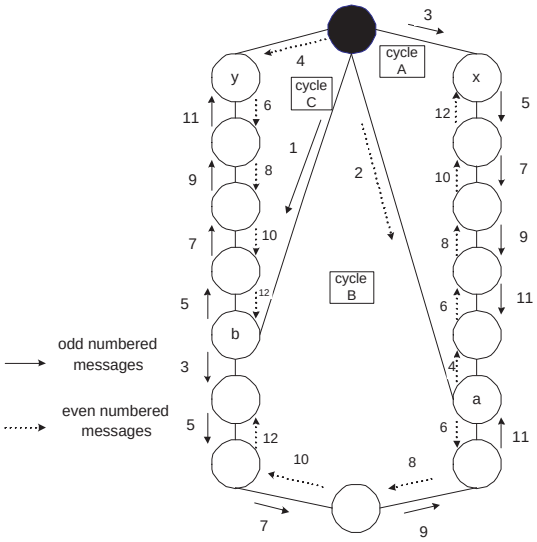

Figure 5.32: A solution to our multiple message broadcasting problem that takes longer than multiple message broadcasting using the chordless cycle algorithm on the same size graph.

If the originator sends to node a and node y, it is obvious that a similar scenario to that, which was just discussed for the originator sending to node x and node b will occur.

We conclude that, it is not possible to broadcast multiple messages using two message sets in a cycle with two chords in time less than that required to broadcast multiple messages in the same size chordless cycle using the chordless cycle algorithm.

5.4 Results

In this chapter, we researched multiple message broadcasting in a cycle with one and two chords, respectively. For each type of graph, we used Farley's bounds to realize both upper and lower bounds for broadcasting; we then compared these bounds to Farley's bounds for broadcasting in a chordless cycle of the same size. When feasible, we attempted to develop multiple message broadcast algorithms for cycles with one and two chords that made use of the chords and improved upon the time taken to broadcast multiple messages using the chordless cycle algorithm on chordless cycles of the same size. We studied broadcasting multiple messages using both a single set of messages and multiple sets of messages. Unfortunately, we were not able to broadcast in time better than the chordless cycle multiple message broadcasting algorithm by adding one or two chords to a cycle. We conjecture that there does exist some number of chords which, if used, will improve upon the time taken to broadcast multiple messages in a cycle beyond that of the chordless cycle algorithm.

Chapter 6

Line Broadcasting in Cycles with Chords

6.1 Background

Thus far, we have studied a form of broadcasting known as local broadcasting. Using a local broadcasting scheme, a node may only send a message to a node with which it shares an edge (i.e., a neighbor). Due to this restriction on broadcasting, a local broadcasting scheme is often defined as one in which each node of a graph is only permitted to make local calls. This type of broadcasting is modeled in packet switched networks.

In this chapter, we study line broadcasting. In line broadcasting, a node may send a message to any other node in the graph as long as a simple path exists between the sending node and the receiving node and every edge along the path is not in use. A line broadcasting scheme is often described as a scheme in which nodes are permitted to make long distance calls. Whereas, in local broadcasting, at any single time unit, each node of a given graph may be either a sender or receiver of a message, but not both, in line broadcasting, during a single time unit, a node may act as any of the following.

1. switching node

2. sending node

3. receiving node

4. switching and sending node

5. switching and receiving node

6. dormant node

Figure (6.1) shows examples of each of the above types of nodes.

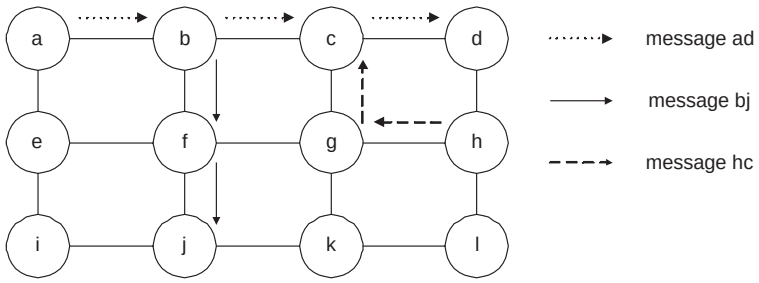

Figure 6.1: An example of line broadcasting.

In the graph depicted in Figure (6.1), node f is referred to as a switching node, since it is not the sender or receiver of a message; however, it is used as a throughway for message bj. Node g is another example of a switching node. We refer to node a and node h as sending nodes, since their sole task is to send a message. Similarly, node d and node j are referred to as receiving nodes, since they simply receive a message. Node b is referred to as a switching and sending node, since it is used as a throughway in moving message ad from node a to node d and it sends message bj. Node c is known as a switching and receiving node, since it is used as a throughway in moving message ad from node a to node d and it is the receiver of message hc. Finally, nodes e, i, k and l

are referred to as dormant nodes, since they are not busy during the time unit depicted in the graph of Figure (6.1).

It is known that the minimum time required to complete broadcasting when using only local calls is equal to $\lceil \log_2 n \rceil$, since, at each time step, the number of informed nodes can, at most, double. In [F80], Farley showed that broadcasting can be completed in time equal to $\lceil \log_2 n \rceil$ in any connected graph and from any originator, using line broadcasting. Thus, as long as a graph is connected, it is possible to complete line broadcasting in minimum time. However, minimum time line broadcasting does not come without a fee. Whereas, in local broadcasting, each call uses only a single edge, in line broadcasting, multiple edges may be used in order to place a single call. If we associate each edge that a message must cross with a distance cost of 1 unit, we find that every call in local broadcasting costs the same amount (1 unit per call). However, when line broadcasting is used, calls have varying distance costs ranging from 1 (a local call) to D, where D is the diameter of the graph. Thus, when creating line broadcasting algorithms, our goal is to broadcast in minimum time while keeping the sum of the distances of all of the calls required of the broadcast algorithm (i.e., cumulative cost) as low as possible.

In local broadcasting, cumulative cost is always $n - 1$, where n represents the number of nodes in the connected graph, since each node may receive the message only once and all calls require a single unit of time. Thus, $n - 1$ is a minimum bound on the distance cost of line broadcasting. In [F80], Farley determined an upper bound of $(n - 1) \cdot \lceil \log_2 n \rceil$ on the distance cost required to complete line broadcasting in minimum time on any connected graph with n nodes.

Kane and Peters studied line broadcasting in cycles. In [KP98], they determine the distance 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 such schemes.

Kane and Peters develop minimum time, optimal line broadcasting schemes for cycles containing 2^k nodes, where k is an integer and $k > 1$, using a constructive approach. They begin by developing a scheme for a cycle containing 4 nodes (i.e., C_4). This scheme is constructed by removing a single edge from the cycle and laying the nodes out flat, so that they produce a path. If the nodes of the cycle are numbered beginning with the leftmost node labeled 0, then node 1 is denoted as the originator. During the first step of their line broadcast scheme, the originator sends the message to its neighbor to the left. During the second and final step, the originator sends to its neighbor to the right, while the node informed at time step 1 sends to its uninformed neighbor. An example of this line broadcast scheme is shown in Figure (6.2).

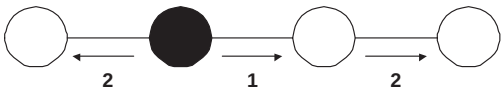

Figure 6.2: Line broadcasting in C_4 using the scheme developed by Kane and Peters.

In order to construct a minimum time, optimal line broadcasting scheme for C_8, [KP98] places two mirror images of the line broadcasting scheme for a cycle containing 4 nodes next to one another (though existent, the edge connecting node 3 to node 4 is not shown, since it is not used in the line broadcast scheme). Each of the mirror images has its own originator. We will label the originator of the mirror image on the left as the "actual" originator, whereas, the originator of the mirror image on the right will be labeled as the "faux" originator. At time step 1, the "actual" originator places a long distance call to the "faux" originator; then, each mirror image completes line broadcasting following the scheme for C_4. Figure (6.3) depicts Kane and Peters' line broadcast scheme on C_8.

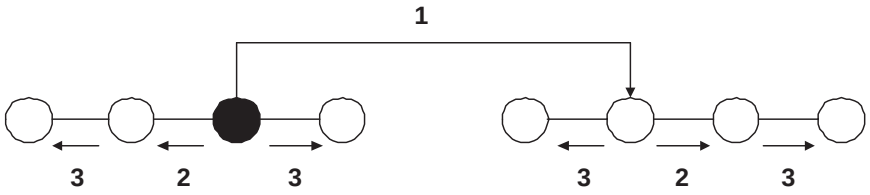

Figure 6.3: Line broadcasting in C_8 using the scheme developed by Kane and Peters.

Using the techniques of mirror images and long distance calls, Kane and Peters are able to construct a line broadcast scheme for any cycle containing 2^{k+1} nodes by placing two mirror images of 2^k line broadcast schemes next to one another and joining them with a long distance call between the "actual" originator and the "faux" originator at time step 1. After time step 1, each mirror image will complete line broadcasting following the scheme used for a cycle containing 2^k nodes. Figure (6.4) depicts line broadcast schemes for C_{16} and C_{32}, respectively.

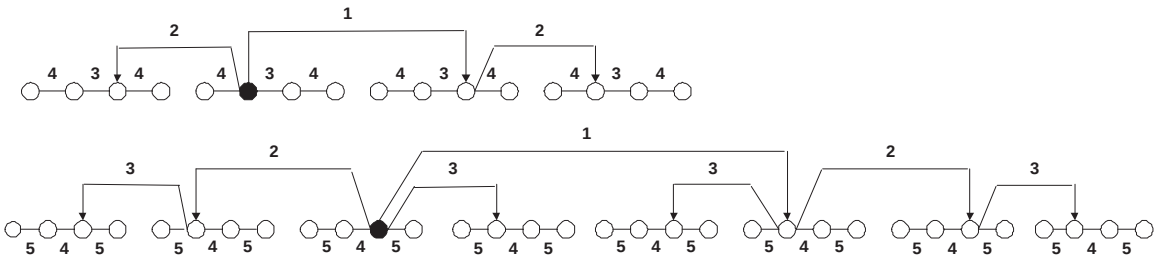

Figure 6.4: Line broadcasting in C_{16} and C_{32} using the scheme developed by Kane and Peters.

In [KP98], Kane and Peters proved that their line broadcast scheme for cycles containing 2^k nodes, where k is an integer and $k > 1$, is optimal. In other words, they show that the cumulative cost of line broadcasting using their algorithm is as small as possible, while achieving minimum time broadcasting.

6.2 Line Broadcasting versus Local Broadcasting

If we view the long distance calls of the line broadcast scheme developed by Kane and Peters as chords of a cycle, rather than calls, their construction method can be adapted to solve the problem of adding a sufficient number of chords to a given cycle, in order to perform local broadcasting in minimum time. In this section, we modify the construction method developed by Kane and Peters in order to determine the number of chords required to perform minimum time local broadcasting in a given cycle.

Kane and Peters' construction method arranges nodes into sets, each of which contains 4 nodes. Each set is then connected to the adjacent set through a long distance call. Using this scheme, it is possible to line broadcast in minimum time. If we modify Kane and Peters' line broadcast scheme by replacing a long distance call with a chord in the cycle, we are able to perform local broadcasting in minimum time. We will call this the Alternative Kane and Peters (AKP) local broadcast scheme.

Lemma 6.2.1 *The total number of chords required to perform minimum time local broadcasting in a cycle containing 2^k nodes, where k is an integer and $k > 1$, using the AKP local broadcast scheme is equal to $2^{k-2} - 1$.*

Proof: The construction method used to create the AKP local broadcast scheme arranges nodes into sets, each containing 4 nodes. It then places a chord between adjacent sets of nodes. Since there are a total of 2^k nodes in the graph, there are $\frac{2^k}{4}$ sets. If we place the sets in a single line with the leftmost set being the first set, each of these sets, except the last, is connected to the set to its right by a single chord. Therefore, $\frac{2^k}{4} - 1 = 2^{k-2} - 1$ chords are created. □

Lemma 6.2.2 *The minimum number of chords required to perform local broadcasting in minimum time in a cycle containing 2^k nodes, where k is an integer and $k > 1$, is equal to $2^{k-2} - 1$.*

Proof: The minimum time required to broadcast in a graph with n nodes is $\lceil \log_2 n \rceil$. If $n = 2^k$, the minimum time to broadcast becomes k. In such a graph, this minimum time bound is achieved by doubling the number of nodes informed at each time step (see Table (3.3)). In order to double the number of informed nodes at each time step, every informed node must send to an uninformed neighbor at each time step until all nodes are informed. However, in a chordless cycle, all nodes have degree 2 (i.e., each node is connected via an edge, to exactly 2 other nodes). Therefore, unless chords are added, all nodes, except the originator, may send to at most one uninformed node (the originator can send to at most 2 uninformed nodes). The addition of a chord connecting an informed node, w, with an uninformed node, y, increases the number of nodes that node w can inform by 1. In order to make the best use of the addition of a chord, we will choose node y such that both of y's neighbors are uninformed nodes (see Figure (6.5)). As shown in Chapter 3, this choice will then allow each node informed via a chord to inform at least one extra node. In order to maximize the number of nodes informed at each step, we want to make use of chords as early as possible. Since an informed node must send to an uninformed neighbor at every time step, the originator must send at every time step. In a chordless cycle, the originator is connected to exactly 2 uninformed nodes; therefore, it must use chords to inform nodes at every time step except steps k and $k-1$, where k is the time required to complete broadcasting. Thus, for all cycles containing 2^k nodes, such that $k > 2$, the node informed at time step 1 will be informed by the originator via a chord. For all cycles containing 2^k nodes, such that $k > 3$, the originator and the node informed at time step 1 must each make use of chords at time step 2. Continuing this idea, we see that

all nodes, except those informed during the last 3 time steps, must make use of chords in order for every informed node to send to an uninformed node at each time step (see Table (6.1) for an example).

Figure 6.5: Making the best use of the addition of a chord to a cycle.

Table 6.1: Analyzing the number of chords needed to perform minimum time local broadcasting in a cycle containing 2^5 nodes.

Time step	Number of newly informed nodes	Number of chords needed for each newly informed node
0		3
1	1	2
2	2	1
3	4	0
4	8	0
5	16	0

The first 2^k node cycle to make use of a chord is C_8 (i.e., $k = 3$); this cycle will require a single chord, since, the only node which needs to be informed via a chord is the node informed at time step 1. When $k = 4$, both the originator and the node informed at time step 1 must make use of chords, which requires a cycle with 2^4 nodes to make use of 3 chords. Table (6.2) shows the number of chords required to perform local broadcasting in minimum time in cycles containing 2^k nodes when $2 < k < 7$.

Table 6.2: The number of chords needed to perform local broadcasting in minimum time in a cycle containing 2^k nodes.

k	Number of chords needed
3	1
4	3
5	7
6	15

The following recurrence can be used in order to calculate the total number of chords required to keep every informed node busy at every time step.

$$\begin{aligned} T(k) &= 1 && \text{, when } k = 3 \\ &= T(k-1) + 2^{k-3} && \text{, otherwise} \end{aligned}$$

A simple proof by induction can be used to show that $T(k) = 2^{k-2} - 1$. □

Lemma 6.2.3 *The AKP local broadcast scheme uses the minimum number of chords necessary to complete local broadcasting in a cycle containing 2^k nodes in minimum time.*

Proof: This result follows immediately from Lemma (6.2.1) and Lemma (6.2.2). □

By replacing Kane and Peters' long distance calls with chords, we can construct minimum time local broadcast schemes for cycles containing 2^k nodes, such that k is an integer and $k > 1$, using the minimum number of chords necessary. Thus far, we have only given results for cycles whose total number of nodes (i.e., size) is a power of 2. The remainder of this chapter will consider cycles whose size is not a power of 2.

6.3 Local Broadcasting in Various Size Cycles

Kane and Peters have developed a method by which to construct a line broadcast scheme for cycles containing between 2^{k-1} and 2^k nodes, where k is an integer and $k > 2$, by first constructing a line broadcast scheme for a cycle containing 2^k nodes and then eliminating the necessary number of nodes from the deepest layer of the broadcast scheme. Our method of constructing a local broadcast scheme using chords closely follows that of Kane and Peters; however, we must be more specific in the selection of nodes to delete in order to ensure the minimum number of chords are being utilized.

Before describing the elimination process, we define the term layer as follows. Kane and Peters group the nodes of a cycle into layers which are determined by the location of each node in relation to the long distance calls utilized by the line broadcast scheme. They define layer 0 to include all nodes who never take on the role of a switching node. Perhaps, this situation is best explained through an example. Consider Kane and Peters' line broadcast scheme for a cycle containing 8 nodes (see Figure (6.3)). In this line broadcast scheme, all nodes, except node 3 and node 4, are considered layer 0 nodes (remember, nodes are numbered from left to right beginning with 0). Layer 0 nodes never appear under a long distance call. The remaining nodes are grouped into layers based on the number of long distance calls under which they appear. In Figure (6.3), node 3 and node 4 are classified as layer 1 nodes, since they appear under a single long distance call.

In order to create a line broadcast scheme for a cycle containing n nodes, where $2^{k-1} < n < 2^k$, Kane and Peters construct a line broadcast scheme for a cycle containing 2^k nodes; then, they repeatedly delete nodes and their incident edges from the deepest layer until the desired number of nodes remain. The deepest layer is the name given to the largest numbered layer. In [KP98], Kane

and Peters prove that the elimination method described above produces optimal line broadcasting schemes for cycles in which the number of nodes contained in the cycle is not a power of 2. Since Kane and Peters optimality is defined as keeping the cumulative cost of the line broadcast scheme as low as possible while broadcasting in minimum time and all nodes in the deepest layer contribute the same cost to all long distance calls appearing above them, they may arbitrarily choose nodes to delete from the deepest layer.

When working with local broadcasting, cumulative cost is always the same (i.e., $n-1$, where n is the number of nodes in the graph). Thus, when deleting nodes, instead of focusing on lowering cumulative cost, we focus on lowering the number of chords required to complete local broadcasting in minimum time. Each time we delete a node, we will combine its edges to form a single edge. For example, node 3, in Figure (6.4), has two edges; one edge connects node 2 to node 3 and the other edge connects node 3 to node 4 (this edge is not shown on the graph, since it is not used in the local broadcast scheme). If we delete node 3 from this graph, we will remove both edges and replace them with a single edge connecting node 2 to node 4. If we arbitrarily delete nodes from the deepest layer, as done in Kane and Peters' construction method, we cannot guarantee that we will remove the largest number of chords while still broadcasting in minimum time. As an example, let us say that we want to construct a local broadcast scheme for a cycle containing 14 nodes. We will begin by constructing a local broadcast scheme for a cycle containing 16 nodes, C_{16}, as depicted in Figure (6.4). In C_{16}, node 3, node 4, nodes 6 through 9, node 11 and node 12 are all classified as layer 1 nodes and layer 1 is the deepest layer. If we arbitrarily delete nodes from layer 1, it could be the case that we choose to delete nodes located under different chords. Since we are deleting a total of 2 nodes, if these 2 nodes are located under different chords, or if both nodes are located under the chord connecting node 5 to node 10, then the resulting 14 node cycle

will contain the same number of chords as C_{16}. However, if we take care to delete both nodes from the shortest chord, then we can, in fact, lower the number of chords by 1, while still broadcasting in minimum time (see Figure (6.6)).

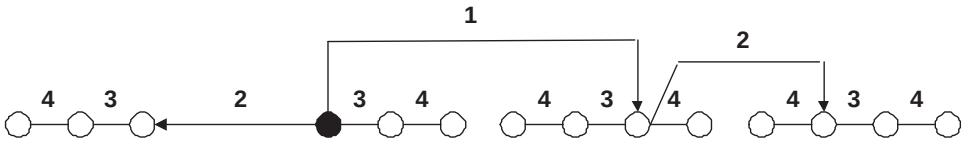

Figure 6.6: A minimum time broadcast scheme for C_{14}.

All nodes, except layer 0 nodes, are located between the endpoints of a chord. Our modification to Kane and Peters' construction method, which we refer to as the Alternate Kane and Peters Deletion method (AKPD), simply involves requiring the nodes to be deleted on a per chord basis beginning with one of the shortest chords. In other words, all the nodes occurring between the endpoints of one of the shortest chords will be deleted first. Once these nodes have been deleted, if we still have nodes to remove, we will begin deleting them from the next shortest chord. We will continue this elimination method until the required number of nodes have been removed from the cycle. Figure (6.7) depicts the results of the AKPD method of constructing local broadcast schemes for C_{17} and C_{25}.

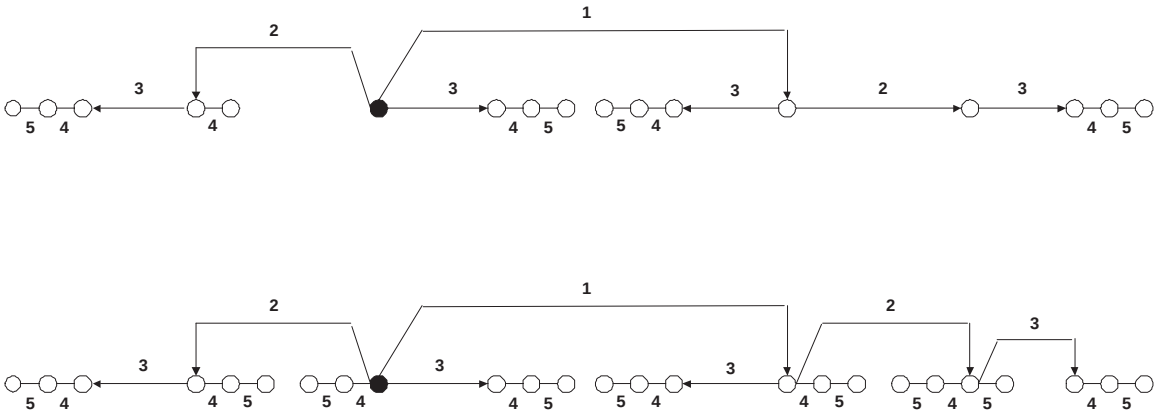

Figure 6.7: Minimum time local broadcast schemes for C_{17} and C_{25}

.

Now that we have a method for deleting nodes, we want to determine the exact number of chords required to perform local broadcasting in minimum time, using the AKPD method, in cycles whose size is not a power of 2. Our construction of a local broadcast scheme for cycles containing n nodes, where $k-1 < n < k$, always begins with the construction of a local broadcast scheme for a cycle containing 2^k nodes and then involves deletion of the extra nodes. Therefore, we will begin by determining the number and size of chords required in a local broadcast scheme for a cycle containing 2^k nodes. The first local broadcast scheme for a 2^k node cycle that requires a chord is the broadcast scheme for the cycle containing 2^3 nodes; this broadcast scheme requires a single chord of size 2. We will refer to the size of a chord as the number of nodes that are contained between the two endpoints of the chord. Since the local broadcast scheme for a cycle containing 2^k nodes is constructed by placing 2 mirror images of the local broadcast scheme for a 2^{k-1} cycle next to one another and joining them with a chord connecting the "actual" originator to the "faux" originator, when creating a local broadcast scheme for a 2^k node cycle, we are doubling the number of chords required in the broadcast scheme for the 2^{k-1} node cycle and we are adding an additional chord to join the "actual" originator with the "faux" originator. For example, consider

the construction of a local broadcast scheme for C_{16}. The local broadcast scheme for this cycle will be formed by placing 2 mirror images of the local broadcast scheme for C_8 next to one another and joining their originators. Each local broadcast scheme for C_8 includes a single chord of size 2 and an additional chord is required to join the two mirror images, which gives us a total of 3 chords required to construct a local broadcast scheme for C_{16} (see Figures (6.3) and (6.4)).

A local broadcast scheme for C_{32} is created from two local broadcast schemes for C_{16}. Each local broadcast scheme for C_{16} makes use of 2 chords of size 2 and one chord of size 4. Therefore, the local broadcast scheme for C_{32} will make use of 4 chords of size 2, 2 chords of size 4 and one additional chord. Using this construction method, the number of chords of each size (i.e., 2, 4, 10, etc.) is a function of k. Specifically, a cycle containing 2^k nodes, where $k > 2$, makes use of exactly 2^{k-3} chords of size 2 and 2^{k-4} chords of size 4, when $k > 3$ (see Table (6.3)).

Table 6.3: The number of chords of size 2 and size 4 contained in a local broadcast scheme for a cycle containing $n = 2^k$ nodes.

n	Number of chords of size 2	Number of chords of size 4
2^3	1	0
2^4	2	1
2^5	4	2
2^6	8	4

Since our method for constructing a local broadcast scheme for a cycle containing n nodes with $2^{k-1} < n < 2^k$ begins by constructing a local broadcast scheme for a cycle containing 2^k nodes, we will never need to delete more than 2^{k-1} nodes. According to the local broadcast scheme for C_8, only a single chord of size 2 exists. When deleting nodes from the local broadcast scheme for C_8, we will begin by removing the nodes that are located between the endpoints of the chord; if additional nodes need to be deleted, they may be arbitrarily removed, since no chords remain in the cycle.

When deleting nodes from the local broadcast scheme for cycles containing 2^k nodes where $k > 3$, we will begin by removing nodes located between the endpoints of the chords of size 2, on a per chord basis. Once all chords of size 2 have been deleted, if additional nodes need to be removed, we will begin removing nodes located between the endpoints of the chords of size 4, on a per chord basis. By removing all of the chords of size 2, we will remove $2^{k-3} \cdot 2$ nodes and by removing all of the chords of size 4, we will remove $2^{k-4} \cdot 4$ additional nodes. Thus, by removing all of the chords of size 2 and size 4 from a local broadcast scheme, we will remove $2^{k-3} \cdot 2 + 2^{k-4} \cdot 4 = 2^{k-1}$ nodes, which is the largest number of nodes that we will ever need to delete. Therefore, we will only ever need to remove nodes located between chords of size 2 or size 4. Using the above information, we are able to develop Algorithm (6.3.1) to calculate the number of chords necessary to perform local broadcasting in minimum time in a cycle that contains n nodes, where $2^{k-1} < n < 2^k$.

Function $\mathrm{CCA}(n)$
1: $k = \lceil \log_2 n \rceil$
2: $nodesToRemove = 2^k - n$
3: $chordsToRemoveFromFirstChords = \lfloor \frac{nodesToRemove}{2} \rfloor$
4: $chordsToRemoveFromSecondChords = 0$
5: **if** ($chordsToRemoveFromFirstChords > 2^{k-3}$) **then**
6: $\quad chordsToRemove = 2^{k-3}$
7: $\quad chordsToRemoveFromSecondChords = \lfloor \frac{nodesToRemove - 2^{k-3} \cdot 2}{4} \rfloor$
8: **else**
9: $\quad chordsToRemove = chordsToRemoveFromFirstChords$
10: **end if**
11: **if** ($chordsToRemoveFromSecondChords > 0$) **then**
12: $\quad chordsToRemove = chordsToRemove + chordsToRemoveFromSecondChords$
13: **end if**
14: **return** $2^{k-2} - 1 - chordsToRemove$

Algorithm 6.3.1: Chord Calculation Algorithm

6.4 Results

In this chapter, we modified the line broadcasting results obtained by Kane and Peters in [KP98] in order to create minimum time local broadcast schemes for cycles through the use of chords. We proved that the number of chords added to a cycle containing 2^k nodes, where k is an integer and $k > 2$, using our local broadcast scheme is the minimum number of chords required to broadcast in minimum time in such a cycle. Then, we developed a scheme for performing local broadcasting in cycles whose size is not a power of 2, through the use of chords. Finally, we developed an algorithm to calculate the number of chords needed to broadcast using our local broadcasting scheme for cycles containing n nodes, where $2^{k-1} < n < 2^k$ and $k > 2$.

Chapter 7

Conclusion

7.1 Broadcast Schemes for Cycles with Chords

We have researched cycles with chords in order to develop efficient broadcast schemes for such graphs. We considered local broadcasting, multiple message broadcasting and line broadcasting schemes.

We began by studying the effects of adding a single chord to a cycle. We developed two algorithms for broadcasting in such a graph. Both algorithms assumed that the originator formed one of the endpoints of the chord. The first algorithm, ESCO, considered the case when the chord divided the cycle into two smaller equal size cycles and completed broadcasting in time equal to $\lfloor \frac{n}{2} \rfloor + 1$, where n represents the number of nodes in the larger cycle formed by the chord. The second algorithm, USCO, considered the case when the chord divided the cycle into two smaller cycles which differed in size by more than a single node and completed broadcasting in time equal to $\lceil \frac{n}{2} \rceil$, where n represents the number of nodes in the larger cycle formed by the chord. We concluded that the time taken to complete broadcasting using ESCO is always as good as or better than that taken by

USCO.

Next, we studied the effects of adding two chords to a cycle. Again, we developed two algorithms for broadcasting in such a graph. Both algorithms considered the case when the addition of chords created three smaller equal size cycles. The first algorithm, TCO, considered the case when the chords shared the originator as an endpoint. The second algorithm, TCNO, considered the case when the chords shared a non-originator node as an endpoint and the originator formed the endpoint of a single chord. Both of these algorithms completed broadcasting in time equal to $\lceil \frac{n}{2} \rceil + 1$, where n represents the number of nodes in the largest cycle formed by the chords. We concluded that the choice of the node forming the shared endpoint does not affect the running time of the broadcast algorithm.

Then, we studied the effects of adding chords to a cycle in an attempt to improve upon the time required to perform multiple message broadcasting in cycles. We discovered that the addition of either one or two chords to a cycle does not improve upon the time taken to broadcast multiple messages in the cycle.

Finally, we used techniques developed for creating minimum time line broadcast schemes for cycles in order to develop minimum time local broadcast schemes for cycles through the use of chords. We proved that the number of chords added to a cycle containing 2^k nodes, where k is an integer and $k > 2$, using our local broadcast scheme is the minimum number of chords required to broadcast in minimum time in such a cycle. We developed a scheme to perform local broadcasting in minimum time in cycles whose size is not a power of 2, through the use of chords. Then, we developed an algorithm for calculating the number of chords needed to broadcast using our local broadcasting scheme for cycles containing n nodes, where $2^{k-1} < n < 2^k$.

Bibliography

[AGR01] A. Averbuch, I. Gaber, and Y.Roditty Low-cost minimum-time line-broadcasting schemes in complete binary trees. *Networks*, 38:189–193, 2001.

[BK92] Amotz Bar-Noy and Shlomo Kipnis. Designing broadcasting algorithms in the postal model for message-passing systems. *In Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 13–22, San Diego, California, June 29-July 1, 1992.

[FHMP78] A. Farley, S. Hedetniemi, S. Mitchell, and A. Proskurowski. Minimum broadcast graphs. *Discr. Math.* 25:189–193, 1979.

[FH78] A. Farley and S. Hedetniemi. Broadcasting in grid graphs. In *Proc. Ninth SE Conf. on Combinatorics, Graph Theory and Computing*, pages 275–288, Winnipeg, 1978. Utilitas Mathematica.

[FL94] P. Fraigniaud and E. Lazard. Methods and problems of communication in unusual networks, *Discrete Applied Mathematics* 53:79–133, 1994.

[F79] A. Farley. Minimal broadcast networks. *Networks*, 9:313–332, 1979.

[F80] A. Farley. Broadcast time in communication networks. *SIAM Journal of Applied Mathematics*, 39:385–390, 1980.

[GLPR94] Luisa Gargano, Author L. Leistman, Joseph G. Peters, and Dana Richards. *Discrete Applied Mathematics*, 53:135–148, 1994.

[GRV98] Luisa Gargano, Adele A. Rescigno, and Ugo Vaccaro. Minimum time broadcast in faulty star networks. *Discrete Applied Mathematics*, 83:97–119, 1998.

[GS99] M. Golin and Assaf Schuster. Optimal point-to-point broadcast algorithms via lopsided trees. *Discrete Applied Mathematics*, 93:233–263, 1999.

[G06] G. Goltukhchyan. Multiple message broadcasting and gossiping in the dynamically orientable graphs. Dissertation, *West Virginia University*, 2006.

[HHL88] S. M. Hedetniemi, S. T. Hedetniemi, and A. Liestman. A survey of broadcasting and gossiping in communication networks. *Networks*, 18:319–349, 1988.

[HKMP96] Juraj Hromkovic, Ralf Klasing, Burkhard Monien, and Regine Peine. Dissemination of information in interconnection networks (Broadcasting and Gossiping). In Du, D.-Z. and Hsu, D.F. (Eds.). *Combinatorial Network Theory*. Kluwer Academic Publishers, Netherlands, 1996, 125–212.

[JG79] D. Johnson and M. Garey. Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisco, CA 1979.

[JW01] Z. Jiang and J. Wu. A fault-tolerant broadcasting in 2-D wormhole-routed meshes. *2001 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'2001)*, June 2001, 2028–2034.

[KC95] O. H. Kwon and K. Y. Chwa. Multiple message broadcasting in communication networks. *Networks*, 26:253–261, 1995.

[KP98] Jave O. Kane and Joseph G. Peters. Line broadcasting in cycles. *Discrete Applied Mathematics*, 83:207–228, 1998.

[K79] C. Ko. On a conjecture concerning broadcasting in grid graphs, preliminary report. *Notices Amer. Math. Soc.* 26:A-196, 1979.

[MBHI96] Aohan Mei, Feng Boa, Yukihiro Hamada, and Yoshihide Igarashi. "Optimal Time Broadcasting in Faulty Star Networks," Proc. 10th International Workshop on Distributed Algorithms, LNCS, vol. 1151, pp. 175–190, Springer-Verlag, 1996.

[M92] P.D. MacKenzie. A lower bound for order preserving broadcast in the postal model. *Manuscript*, University of Texas at Austin, December 1992.

[P80] G. W. Peck. Optimal spreading in an n-dimensional rectilinear grid. *Stud. Appl. Math.* 62:69–74, 1980.

[P81] Andrzej Proskurowski. Minimum broadcast trees *IEEE Trans. on Comput.* 30:363–366, 1981.

[QA97] Ke Qiu and Selim G. Akl. Novel data communication algorithms on hypercubes and related interconnection networks and their applications in computational geometry. Technical Report No. 97-415, Queens University, Ontario 1997.

[SCH81] P. J.Slater, E. Cockayne and S. T. Hedetniemi. Information dissemination in trees. *SIAM J. Comput.*, 10:692–701, 1981.

[SW84] P. Scheuermann and G. Wu. Heuristic algorithms for broadcasting in point-to-point computer networks. *IEEE Trans. on Comp.*, 33:804–811, 1984.

[S99] Matthew Suderman. Multiple message broadcasting. MS Thesis, *Simon Fraser University*, 1999.

[VB94] F. Van Scoy and J. Brooks. Broadcasting multiple messages in grids. *Discrete Applied Mathematics*, 53:321–336, 1994.

[V79] F. Van Scoy. Broadcasting a small number of messages in a square grid graph. *Proc. Seventeenth Allerton Conf. on Communication, Control and Computing*, 1979.

[WGLM00] J. Wu, F. Gao, Z. Li, and Y. Min. Optimal fault-tolerant routing in hypercubes using extended safety vectors. *Proc. of 7th International Conf. on Parallel and Distributed Systesm (ICPADS)*. July 2000, 264–271.

[WV96] I. Wojciechowska and F. Van Scoy. Broadcasting multiple messages in grid graphs. *6th Lakeview Conference on Computational Research and Materials*, Morgantown, West Virginia, May 8-10, 1996.

[W99] I. Wojciechowska. Broadcasting in grid graphs. Dissertation, *West Virginia University*, 1999. Available: https://eidr.wvu.edu/etd/documentdata.eTD?documentid=877.