POLLING AND RECEIVING IN GRAPHS

by

Daniel Kwai-wah Ling

B.Sc. (Honours), University of Western Ontario, 1983

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

SIMON FRASER UNIVERSITY
August 1985

Approval

Name: Daniel Kwai-wah Ling

Degree: Master of Science

Title of Thesis: Polling and Receiving in Graphs

Examining Committee:

Chairman: Dr. Tiko Kameda

Dr. Arthur Liestman
Senior Supervisor

Dr. Binay Bhattacharya

Dr. Joseph Peters

Dr. Wo Shun Luk
External Examiner
School of Computing Science
Simon Fraser University

Aug. 13, 1985
Date Approved

PARTIAL COPYRIGHT LICENSE

I hereby grant to Simon Fraser University the right to lend my thesis, project or extended essay (the title of which is shown below) to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users. I further agree that permission for multiple copying of this work for scholarly purposes may be granted by me or the Dean of Graduate Studies. It is understood that copying or publication of this work for financial gain shall not be allowed without my written permission.

Title of Thesis/Project/Extended Essay

Polling and Receiving in Graphs

Author:

(signature)

Daniel Kwai-wah Ling

(name)

September 3, 1985

(date)

Abstract

Broadcasting is the process of information dissemination in a communication network in which a message is routed from one special node, called *the originator*, to every other node in the network. **Receiving** is the inverse process, that is, every node has its own unique message that must be routed to a specified node called *the receiver*. **Polling** combines both broadcasting and receiving for a specified node called *the polling station*. The polling station broadcasts a query to every other node and waits to receive a unique response from each of them.

Broadcasting is a well-studied problem, but polling and receiving have only recently been considered. Polling has been investigated only in trees. In this thesis we extend these results to more general graphs. Polling schemes for various classes of graphs are designed and the polling time of an arbitrary polling station is determined. For those graphs with an optimal polling scheme, the polling center is also determined. For receiving, the known results are for trees and 2-connected graphs. We present an optimal receiving scheme for unicyclic graphs and the receiving time of an arbitrary receiver is determined. We also present an $O(|V|^2)$ algorithm to achieve receiving with an arbitrary receiver for general graphs. The resulting receiving time is no worse than 5/4 optimal.

To my parents and girlfriend, Leslie.

Acknowledgement

Special thanks to my supervisor, Dr. Arthur Liestman, for supporting and helping me throughout my work on this research, in particular, his patience to proofread this thesis.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1. Definitions

Broadcasting is the process of information dissemination in a communication network in which a message is routed from one special node, called the originator, to every other node in the network. This is a *one-to-all* information dissemination process. **Receiving** is the inverse process, that is, every node has its own unique message that must be routed to a specified node called the receiver. Thus, it is an *all-to-one* information dissemination process. **Polling** combines both broadcasting and receiving for a specified node called the polling station. The polling station broadcasts a query to every other node and waits to receive a unique response from each of them.

A network is modelled by a simple, connected graph G=(V,E) with the vertices corresponding to nodes and the edges corresponding to communication lines. We assume that neither nodes nor lines will fail during the process. Within the network, (1) only adjacent nodes can communicate, (2) each communication involves only two nodes, and no node can be involved in more than one communication at a time, (3) responses can be accumulated at any internal node, but only one response can be returned at a time, and (4) each communication requires one unit of time.

The receiving time of a node v (denoted **R(G,v)**) is the time required to

complete the receiving process in graph G if v is specified as the receiver. Similarly, **the polling time** of node v (denoted **P(G,v)**) is the time required to complete polling in graph G if v is the polling station. **The receiving center** is the set of nodes with minimum receiving time. Similarly, **the polling center** is the set of nodes with minimum polling time. **The polling time of a graph G** (denoted **P(G)**) is the maximum polling time P(G,v) for any node v in V.

Given a node v in a tree T of n nodes, we define **maxsubtree(v)** to be the size of the largest connected component in T-{v}. That is, if we consider T to be rooted at v, maxsubtree(v) is the size of the largest subtree rooted at a child of v. Kang and Ault [2] have shown that v is in the **centroid** of T if and only if maxsubtree(v) $\leqslant$ n/2. We define a rooted tree T to be a **centroid tree** if its root is in the centroid of T.

1.2. Previous Results in Polling and Receiving

Broadcasting is a well-studied problem [6], but polling and receiving have only recently been considered [4], [5]. So far, only a limited amount of research has been done in polling and receiving. The existing results in polling deal with trees only. Cheston and Hedetniemi [4] have derived a polling scheme for trees and determined its time bound. They also presented an algorithm for finding the polling center of a tree.

A second paper by Cheston and Hedetniemi [5] deals with the related problem of receiving in trees and 2-connected graphs. The receiving center of a tree is determined to be the centroid of the tree network. The receiving time for a node in the centroid of any tree on n vertices is equal to the minimum receiving time of any receiver in any n vertex graph. Since every node of a 2-connected graph is a

centroid of some spanning tree of the graph, an algorithm is developed to construct such a spanning tree for a given receiver. The tree receiving scheme can then be applied to the spanning tree resulting in an optimal receiving scheme for the 2-connected graph. Although the complexity of determining the receiving center of an arbitrary graph is not known, the similar problem of determining the centroid of an arbitrary graph is known to be NP-complete [5]. Determining the receiving time of a given receiver in a general graph is an open problem. There is no known characterization of the receiving center of a general graph.

1.3. Outline of work contained in the thesis

We present polling and receiving schemes for an arbitrary polling station and receiver for various classes of graphs. We also determine the time bounds for these schemes. For those graphs with an optimal polling or receiving scheme, we also determine the polling center or the receiving center.

In Chapter 2, we consider polling in some specific graphs other than trees: simple cycles, complete graphs, complete k-partite graphs and 2-connected graphs.

In Chapter 3, we consider receiving in unicyclic graphs. An optimal scheme is designed and the time is determined.

In Chapter 4, based on the existing results for receiving in trees and 2-connected graphs and the results from Chapter 3, we present an $O(|V|^2)$ algorithm to achieve receiving in general graphs. The performance of the resulting receiving time is no worse than 5/4 optimal.

In Chapter 5, we summarize the results in both polling and receiving, and state a few open problems.

Chapter 2

Polling in Specific Graphs

We begin by investigating polling in a few types of graphs. In Section 2.1, we briefly review the work of Cheston and Hedetniemi [4]. In Section 2.2, we present some preliminary results. In Section 2.3, we present some optimal schemes for some special graphs with polling station of degree 1. The goal is to determine types of graph structures other than paths which can give the best possible polling time. In Section 2.4, we investigate polling in complete k-partite graphs. An optimal scheme is developed and the resulting polling time is determined. Finally in Section 2.5, we investigate polling in 2-connected graphs with the use of a known algorithm from [5], which transforms a 2-connected graph into a tree with the polling station being the centroid. We then use the tree receiving scheme to determine an upper bound on the polling time of the graph.

2.1. Brief Review of Previous Results

An optimal scheme has been designed for polling in trees [4]. Since we will use some terminology and results from [4] in this chapter, we briefly review them here. Figure 2-1 represents part of a polling scheme in a larger tree. The polling scheme may have been generated by the algorithm of [4]. In this scheme, vertex u receives a query at time 1 and sends responses at times 3, 5, 7, 9 and 11. In the Figure, the number to the left of a "slash" represents the time in which the parent node queries its descendant node and the numbers to the right of the "slash" are the times in

which the descendant node returns a response to its parent node. Once u is informed (receives the query), it can proceed to query its children and wait for responses. The scheme shown uses the minimum possible time for polling in this subtree. Note that u returns a response to its parent every second period after it has received the query. In this case, we claim that no "delay" occurs. We say that a "delay" results whenever vertex u cannot return a message to its parent every second period after u has received the query. For example, one time unit of delay (one delay) is introduced at vertex u at time 7 in the example of Figure 2-2 (a).

Gaps(S,u) is defined [4] to be the number of delays occurring in subtree S rooted at u and **ps(S,u)** is defined [4] to be the polling time for subtree S rooted at u. From the subtree polling scheme [4], we can simplify the idea and yield a general formula for Gaps(S,u) and for ps(S,u). This is illustrated by the following example. Suppose that we are given a subtree S rooted at u with $|S| = 5$. We consider all possible cases of the subtree polling scheme [4] in the following figures:

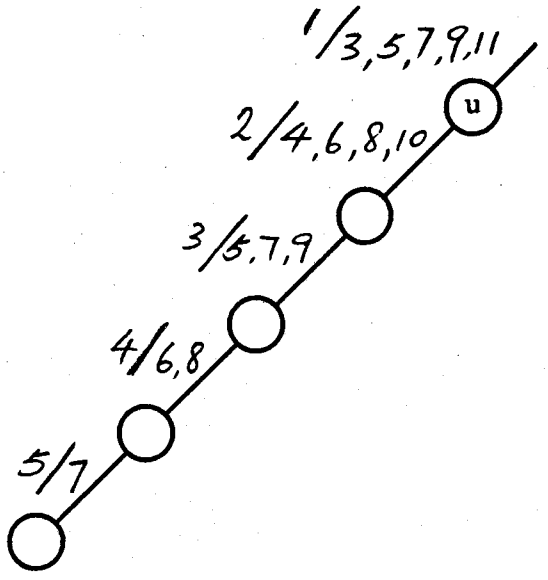
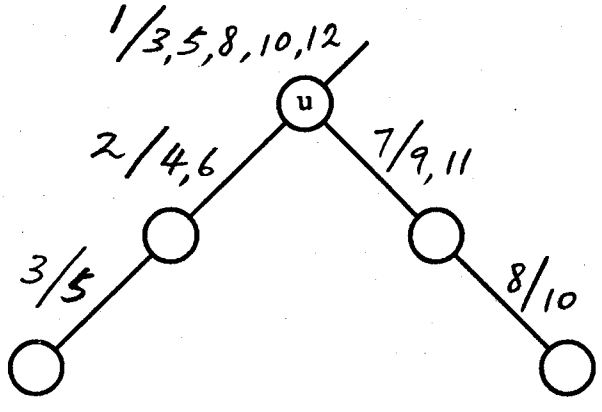
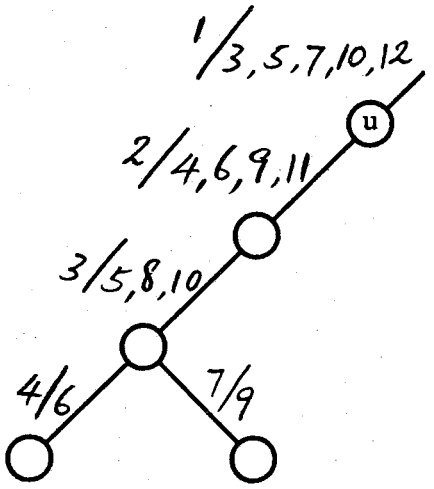


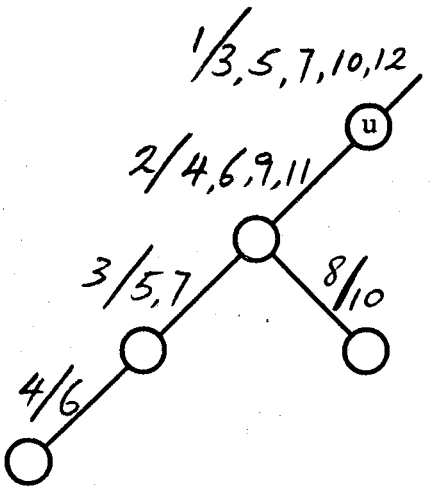
Figure 2-1: Subtree with no delay



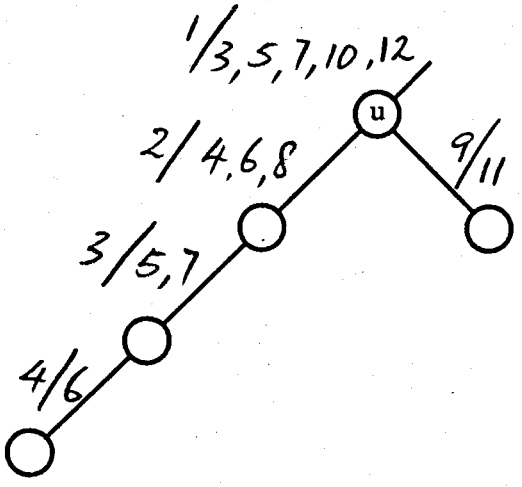
(a)



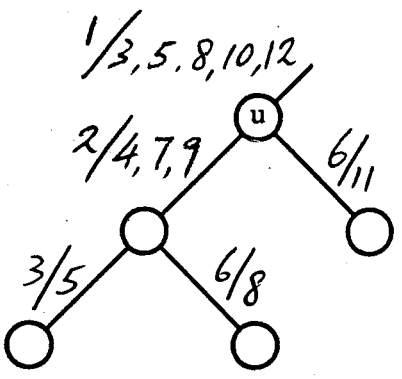
(b)



(c)

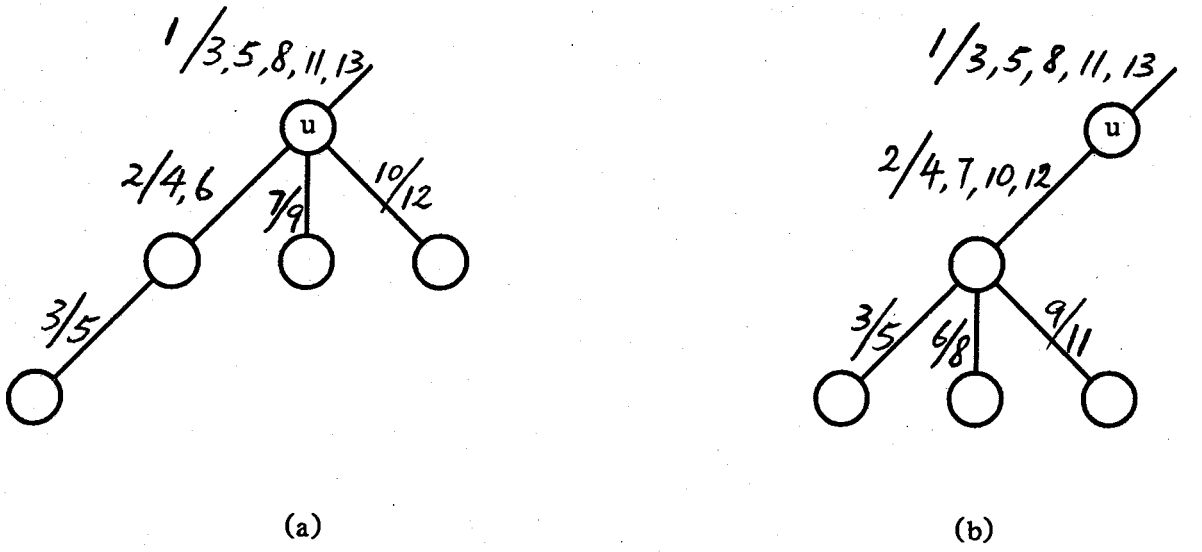


(d)



(e)

Figure 2-2: Subtrees with 1 delay



(a) (b)

Figure 2-3: Subtrees with 2 delays

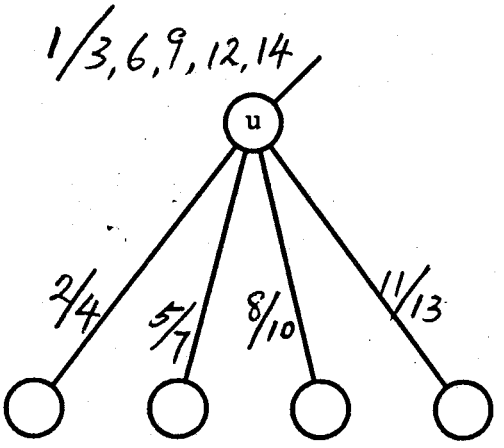


Figure 2-4: Subtree with 3 delays

Note that no delay occurs in Figure 2-1 and thus, Gaps(S,u) = 0 in this case. In Figure 2-2, one delay occurs at time 9 in (b), (c), (d) and at time 7 in (a), (e), and Gaps(S,u) = 1 in this case. In Figure 2-3 two delays occur, at time 7 and 10, and Gaps(S,u) = 2. Finally, in Figure 2-4 three delays occur, at time 5, 8 and 11, and Gaps(S,u) = 3. From this example, we can observe the following facts:

1. When S rooted at u is a path, no delay will occur.

2. When S rooted at u is a star, the maximum delay occurs.

3. In general, Gaps(S,u) = Max{degree(v)-2, v $\in$ S}.

We must determine the time required for polling in subtree S if no delay occurs. We observe that:

1. u must be queried by its parent.

2. u must query one of its children.

3. u must receive all |S| - 1 responses.

4. u must relay all |S| responses (including its own) to its parent.

As a result, 2|S|+1 time units are required. In general, the subtree polling time of Subtree S at root u is:

$$ps(S,u) = 2|S| + 1 + Gaps(S,u)$$

This formula is equivalent to the formula in [4]. We can now determine the polling time of a polling station s in tree T as follows [4]:

$$P(T,s) = \text{Max} \{ \text{degree}(s) + |T| - 1,\ ps(S,u) + i \}$$

where S is the subtree rooted at a child u of s with maximum subtree polling time, and

$i = 1$ if $ps(S,u) \leq ps(R,v) + 1$ for some subtree $R \neq S$.

$= 0$ otherwise.

The first term of the above formula is simply the minimum time required for the tree T if no delay occurs at any subtree rooted at a child of s.

The second term of the above formula tells us that if there is at least one delay occurring in any subtree rooted at a child of s, then we need to determine both the subtree S rooted at a child u of s with the largest subtree polling time and the subtree R rooted at a child v of s with the second largest subtree polling time. If $ps(S,u) - ps(R,v) \leq 1$, then $ps(S,u) = ps(R,v)$ or $ps(S,u) = ps(R,v)+1$. In either case the term $i = 1$. Otherwise $i = 0$. If $ps(S,u)+i$ is greater than the first term in the formula, then $P(T,s) = ps(S,u)+i$. Otherwise $P(T,s) = \text{degree}(s)+|T|-1$.

2.2. Preliminary Results

In this section, we present some simple results for graphs other than trees.

Lemma 2.1: For any connected n vertex graph $G=(V,E)$ with $n > 2$ and a vertex s of degree one, $P(G,s) \geq 2n-1$.

Proof: Let b be s's only neighbour. Recall that each call can convey only one message (either a response or a query) and that each vertex can be involved in at most one call per time unit. Since all messages to and from s must pass through b, the vertex b forms a bottleneck. Consider the calls involving b:

1. b must receive the query from s.

2. b must make at least one call to inform the other members of the query.

3. b must receive each of the n-2 other pieces of information.

4. b must send n-1 pieces of information to s.

Thus, b must be involved in at least 1+1+(n-2)+(n-1) = 2n-1 calls, and $P(G,s) \geq 2n-1$. ■

Lemma 2.2: Given a path P_n of length n-1 with vertices $p_1, p_2, \ldots, p_n$ such that p_i is adjacent to p_{i+1} for $1 \leq i < n$. Let p_k be the polling station.

$$P(P_n,p_k) = \begin{cases} 2(k-1)+1 & \text{if } n-k < k-1 \\ 2(n-k)+1 & \text{if } n-k > k-1 \\ 2(n-k)+2 & \text{if } n = 2k-1 \end{cases}$$

Proof: This follows from the results on polling in trees [4]. ■

Theorem 2.3: Given a connected graph G on n vertices and a vertex s $\in$ V,

$$P(G,s) = \begin{cases} 0 & \text{if } n = 1 \\ 2 & \text{if } n = 2 \end{cases}$$

$P(G,s) \geq n+1$ *if* $n > 2$.

Proof: For n = 1, 2, the result is trivial. For n > 2, consider two cases :

1. If vertex s of degree one is the polling station, then by Lemma 2.1, $P(G,s) \geqslant 2n-1 \geqslant n+1$.

2. If vertex s of degree greater than one is the polling station, then consider a polling scheme at s. At least one time unit must be used to send the query to one of the neighbours of s. A further n-1 time units are required for receiving the n-1 responses. Thus, $P(G,s) \geqslant n$. However, consider two cases :

   a. s queries only one of its neighbours (vertex b).

      If vertex b returns its own message at period two, then s will receive no message and be idle at period three. Thus, $P(G,s) \geqslant n+1$. On the other hand, if b sends the query to one of its neighbours at time period two, then s is idle at period two. Thus, $P(G,s) \geqslant n+1$.

   b. s queries more than one of its neighbours.

      In this case, at least two time periods are used for querying from s in addition to the n-1 time periods required for receiving responses, so $P(G,s) \geqslant n+1$.

Thus, $P(G,s) \geqslant n+1$ for all $s \in V$. ■

Corollary 2.4: Given a connected graph G on n vertices,

$$P(G) \geq \begin{cases} 0 & \text{if } n = 1 \\ 2 & \text{if } n = 2 \\ n+1 & \text{if } n > 2 \end{cases}$$

Theorem 2.5: Given graph G on n vertices which contains a Hamiltonian circuit, $P(G) = n+1$.

Proof: Consider the Hamiltonian circuit $C = \langle v_1, v_2, \ldots, v_n \rangle$. Without loss of generality, let v_1 be the polling station. The tree which results from deleting the edge between $v_{\lfloor n/2 \rfloor}$ and $v_{\lfloor n/2 \rfloor+1}$ can be polled in n+1 time units by using the tree polling algorithm of [4]. From Corollary 2.4, we know that $P(G) \geq n+1$. Thus, $P(G) = n+1$. ■

Corollary 2.6: $P(C_n) = P(K_n) = n+1$.

Corollary 2.7: V is the polling center for any Hamiltonian graph.

2.3. Optimal schemes for some polling stations of degree one

From Lemma 2.1, we know that for any connected graph G with the polling station s of degree 1, $P(G,s) \geq 2n-1$. In this section, we present some graphs G with a fixed polling station s with deg(s) = 1, where $P(G,s) = 2n-1$. The goal is to find polling schemes for graphs other than paths with no delay for the given polling station.

From Lemma 2.2, we know that when the polling station s is the first node or the last node of a path, $s = p_1$ or p_n, then $P(P_n,p_1) = P(P_n,p_n) = 2(n-1)+1 = 2n-1$.

We now consider graphs consisting of paths and cycles with a fixed polling station s with deg(s) = 1. Consider the graph G_1 as shown in Figure 2-5 with polling station p_1. We propose the following polling scheme :

1. p_i queries p_{i+1} at time i, $1 \leq i < k$.

   p_k queries c_1 at time k.

   c_i queries c_{i+1} at time k+i, $1 \leq i < p$.

2. c_i returns a message to c_{i-1} at times k+i+1, k+i+3,...,k+i+2[(p-1)-i]+1, $1 < i < p$.

   c_p returns its message to c_1 at time k+2p-1.

   c_1 returns a message to p_k at times k+2, k+4,..., k+2p.

   p_{i+1} returns a message to p_i at times i+2, i+4,..., 2k-i, 2k-i+2, 2k-i+4,..., 2k-i+2p.

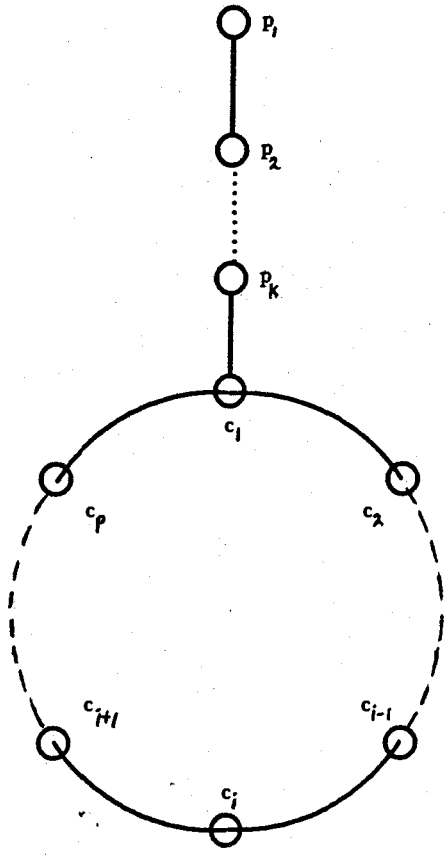


Figure 2-5: G_1

Observe that c_p in G_1 can return a message to c_1 at any time from k+p to k+2p-1, p > 2. Since c_2 returns a response to c_1 at times k+3, k+5,..., and k+2p-3, c_1 can return responses (including its own) to p_k at times k+2, k+4, k+6,...,k+2p-2. As a result, c_1 is busy receiving and returning responses from time k+p to time k+2p-2 inclusive and returns its own message to c_1 at time k+2p-1.

Theorem 2.8: $P(G_1,p_1) = 2n-1$, where $n > 3$.

Proof: The number of vertices in G_1 is $n = k+p$. If $k = 1$, then c_1

returns the last response to p_1 at time $1+2p = 1 + 2(n-1) = 2n-1$ in the above polling scheme. By Lemma 2.1, $P(G_1,p_1) \geq 2n-1$, and therefore $P(G_1,p_1) = 2n-1$.

If $k > 1$, then p_2 returns the last response to p_1 at time $2k-1+2p = 2(k+p)-1 = 2n-1$. Again by Lemma 2.1, $P(G_1,p_1) \geq 2n-1$, therefore $P(G_1,p_1) = 2n-1$. ■

Next we propose a polling scheme for G_2 as shown in Figure 2-6 :

1. p_i queries p_{i+1} at time i, $1 \leq i < k$.

   p_k queries c_1 at time k.

   c_j queries c_{j+1} at times k+j, $1 \leq j < p$.

   c_p queries $L_p(1)$ at time k+p.

   $L_p(i)$ queries $L_p(i+1)$ at times k+p+i, $1 \leq i < q$.

2. $L_p(i+1)$ returns a message to $L_p(i)$ at times k+p+i+2, k+p+i+4,..., k+p+2q-i.

   c_i returns a message to c_{i-1} at times k+j+1, k+j+3,..., k+j+2[(p-1)-j]+1, $1 < j < p$.

   c_1 returns a message to p_k at times k+2, k+4,..., k+2p, k+2p+2,..., k+2p+2q.

   p_{i+1} returns a message to P_i at times i+2, i+4,..., 2k-i, 2k-i+2, 2k-i+4,..2k-i+2p, 2k-i+2p+2, 2k-i+2p+4,..., 2k-i+2p+2q.

Theorem 2.9: $P(G_2,p_1) = 2n-1$, for $n > 4$.

Proof: Here $n = k+p+q$. Note that if $q = 0$, then $G_2 = G_1$. If

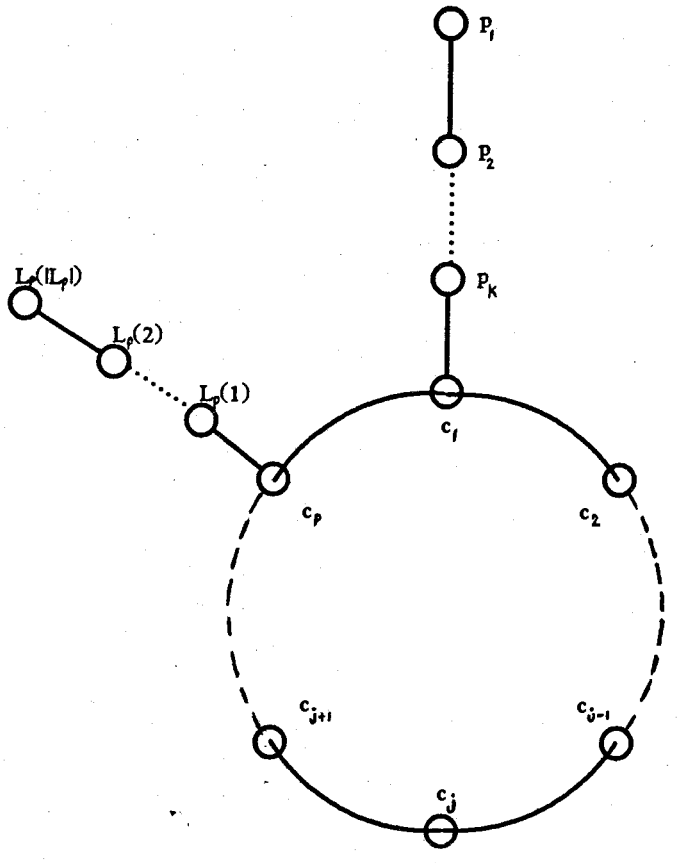


Figure 2-6: G_2

$k = 1$, then c_1 returns the last response to p_1 at time $k+2p+2q = 1 + 2(n-1) = 2n-1$ from the above scheme. By Lemma 2.1, $P(G_2,p_1) \geq 2n-1$ and therefore $P(G_2,p_1) = 2n-1$.

If $k > 1$, for $i = 1$ from the above scheme, p_2 returns the last response to p_1 at time $2k-1+2p+2q = 2(k+p+q)-1 = 2n-1$. Again, by Lemma 2.1, $P(G_2,p_1) \geq 2n-1$ and therefore $P(G_2,p_1) = 2n-1$. ■

We can modify the scheme for G_2 to take care of a more general class of graphs G_3 as shown in Figure 2-7. We propose the modified scheme for G_3 as follows:

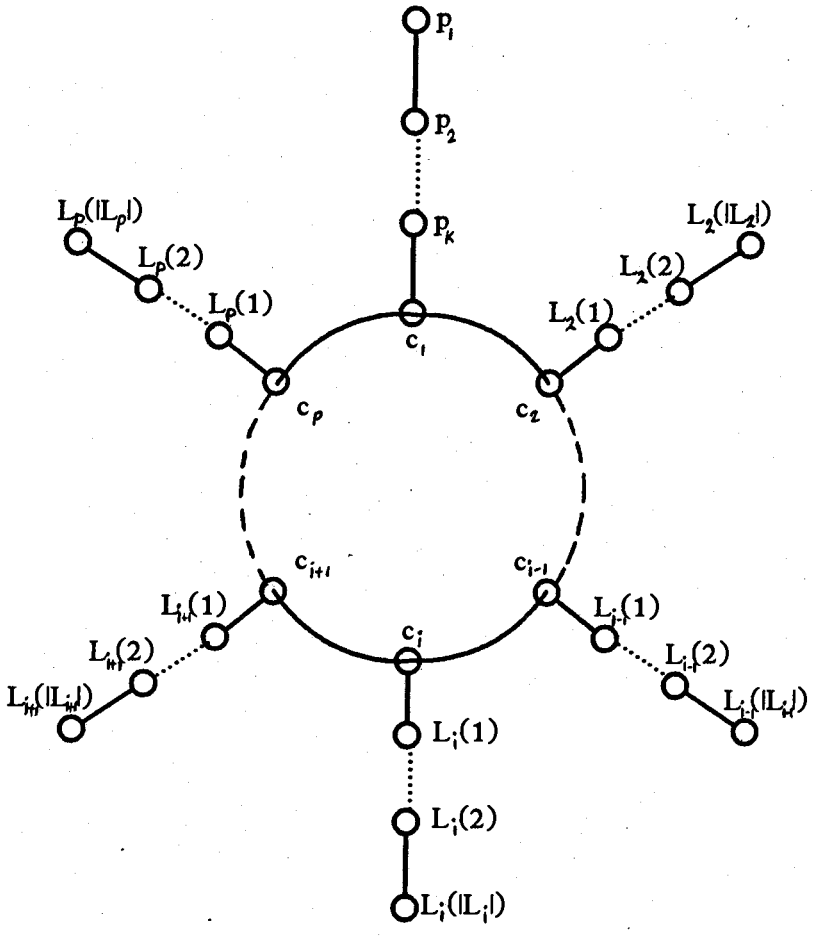


Figure 2-7: G_3

1. p_i queries p_{i+1} at time i, $1 \leqslant i < k$.

   p_k queries c_1 at time k.

   c_i queries c_{i+1} at time k+i, $1 \leqslant i < p$.

If $|L_p| > 0$, then c_p queries $L_p(1)$ at time k+p, and $L_p(j)$ queries $L_p(j+1)$ at time k+p+j, $1 \leq j < |L_p|$.

2. For each c_j in the cycle, $1 < j < p$, c_j returns its response to c_{j-1} during the second time period after it has been queried by c_{j-1}. Then c_j receives (if available) and returns next response to c_{j-1} (if the response is not the last from c_{j+1}) every second period.

3. Let c_i be the vertex with the largest i and $i \neq p$, with $|L_i| > 0$. After c_i has received the last response from all c_j, $i < j < p$, it queries $L_i(1)$ at next time period, say time t. Then c_i continues to relay the last response to c_{i-1} at time t+1 and $L_i(j)$ queries $L_i(j+1)$ at time t+j. Each $L_i(j+1)$ returns a response to $L_i(j)$ at times t+j+2, t+j+4,..., and t+j+2($|L_i|$-j), and $L_i(1)$ returns a response to c_i at times t+2, t+4,...and t+2$|L_i|$. The vertex c_i then relays these responses at times t+3, t+5, ..and 2+2$|L_i|$+1.

4. For each of the remaining vertices c_i with $|L_i| > 0$, if there is a delay time at time t', then c_i queries $L_i(1)$ at this time period. $L_i(j)$ queries $L_i(j+1)$ at time t'+j, $1 \leq j < |L_i|$. The vertex c_i continues to receive the delayed responses from all c_k and L_k, $i < k < p$, at time t'+1 and then relays these responses (if available) to c_{i-1} every second period. When there are no more messages from all these c_k and L_k, the responses are pulled from L_i to continue the message relay.

Theorem 2.10: $P(G_3,p_1) = 2n-1$, for $n > 3$.

Proof: With the scheme above, from time k+p+1 on, there is at least one response waiting at c_p to be returned to c_1 in case a delay occurs at c_2. The vertex c_i defined in Step 3 causes at most a single delay. This delay is propagated through vertices c_j, $i > j > 1$. Each such c_j can make use of this delay to query $L_j(1)$ without causing an additional delay. Thus, only 1 delay will occur at c_2 and c_p can make up this delay by returning a response. As a result, c_1 can receive a response from its descendant vertices and relay it to p_k every second period. This implies that p_2 can also return a response to p_1 every second period and the last call is made at 2n-1. By Lemma 2.1, $P(G_3,p_1) \geq 2n-1$. Thus, $P(G_3,p_1) = 2n-1$. ■

Similarly, we can modify the scheme for G_1 for graphs such as G_4 as shown in Figure 2-8. We propose the modified scheme for G_4 as follows:

1. p_i queries p_{i+1} at time i, $1 \leq i < k$.

   p_k queries c_1 at time k.

   c_i queries c_{i+1} at time k+i, $1 \leq i < p$.

2. For each c_j in the cycle, $1 < j < p$, c_j returns its response to c_{j-1} during the second period after it has been queried by c_{j-1}. c_j then receives a response from c_{j+1} and relays it to c_{j-1} (if the response is not the last response from all c_k, $j < k < p$) every second period.

3. After Step 1, c_p can return its response to c_1 at any time from k+p on.

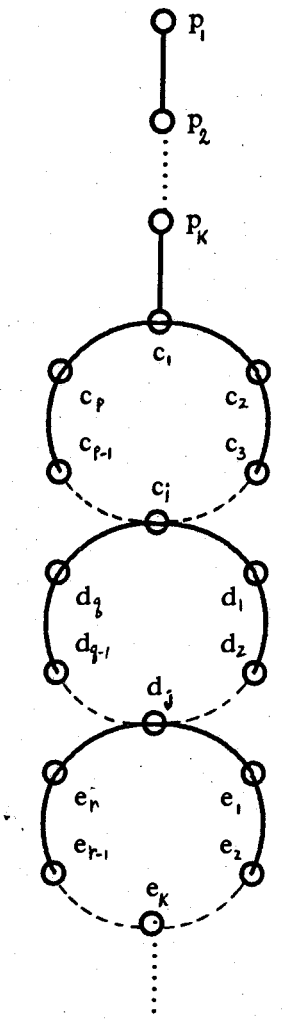


Figure 2-8: G_4

Note that this response stands by to return to c_1 in case of delay occurring at c_2. If there is a cycle $<c_j, d_1, d_2,....d_q, c_j>$ attached at c_j for $1 < j < p$, then c_j queries d_1 right after the response from c_{p-1} arrives at c_j, say at time t. This causes c_j one time delay to relay the response to c_{j-1}, and the delay may occur as early as right after c_p is

queried by c_{p-1}, that is, at time k+p. However, the delay can be filled by the response standing by at c_p.

4. After d_1 is queried by c_i at time t, c_i continues to relay the response of c_{p-1} to c_{i-1} at time t+1. Starting from time t+2, d_1 can return q-1 responses to c_i every second period if there is no other circle attached at d_j for $1 < j < q$. As in Step 3, d_q can return a response to c_i at any time from t+q on and be ready to return the response in case of delay occurring at d_1. Thus if there is another cycle $<d_j, e_1, e_2,....e_r, d_j>$ attached at d_j, as before, the delay caused by querying e_1 from d_j can be filled by the response standing by at d_q.

5. The same technique can be used for additional cycles as shown in G_4.

Theorem 2.11: $P(G_4,p_1) = 2n - 1, n > 3.$

Proof: Without loss of generality, let $<d_j, e_1, \ldots, e_r, d_j>$ be the cycle furthest from p_1. The vertex d_j can receive and return a response to d_{j-1} every second period from all e_k, $1 \leqslant k \leqslant r$. As a result, c_i can also receive and return a response to c_{i-1} every second period from all its descendant vertices. Similarly, c_1 can also receive and return a response to p_k every second period from all its descendant vertices. As a result, p_2 can receive and return a response to p_1 every second period from all its descendant vertices. So p_2 returns the last response to p_1 at time 2n-1. By Lemma 2.1, G_4 requires at least 2n-1 time units, and thus the theorem holds. ■

2.4. Polling in Complete k-partite Graphs

From the results of Section 2.2, we have a lower bound on the polling time of an arbitrary polling station s in general graphs. In this section, we develop polling schemes for complete k-partite graphs using the results developed earlier.

2.4.1. Polling in Complete Bipartite Graphs

Theorem 2.12: Given a complete bipartite graph G on two sets of vertices S_1 and S_2, with $|S_1| \leq |S_2|$ and $|S_1| + |S_2| = n$.

1. If $|S_1| = 1$,

$$P(G,s) = \begin{cases} 2n-2 & \text{if } s \in S_1 \\ 3n-4 & \text{otherwise} \end{cases}$$

2. If $|S_1| = 2$,

$$P(G,s) = \begin{cases} n+1 & \text{if } s \in S_1 \\ n+\lceil (n-3)/2 \rceil & \text{otherwise} \end{cases}$$

3. If $|S_1| > 2$, $P(G,s) = n+1$ for all $s \in V$.

Proof: CASE 1: Consider the diagram as shown in Figure 2-9:

i) If v_{11} is the polling station, then consider the number of calls it is involved in:

1. v_{11} must spend n-1 time units to query every node in S_2.

2. v_{11} must spend another n-1 time units to receive every response from S_2.

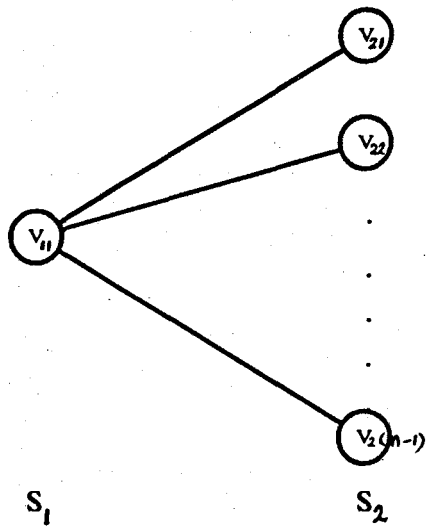


Figure 2-9: A "star" network

Thus $P(G,v_{11}) \geq 2n-2$. With the tree polling scheme, the resulting polling time is $degree(v_{11})+|T|-1 = (n-1)+n-1 = 2n-2$.

ii) If v_{21} is the polling station, then vertex v_{11} acts as a bottleneck for the message transfers. Consider the number of calls in which v_{11} is involved :

1. v_{11} must be queried by s.

2. v_{11} must query the remaining n-2 nodes.

3. v_{11} must receive n-2 responses.

4. v_{11} must return all n-1 responses to s.

Thus, $P(G,s) \geq 3n-4$. With the tree polling scheme, the resulting polling time is $ps(S,v_{11})+i = 2|S|+1+Gap(S,v_{11})+i = 2|S|+1+(degree(v_{11})-2)+i = 3|S|-1 = 3(n-1)-1 = 3n-4$.

CASE 2: Consider the diagram as shown in Figure 2-10:

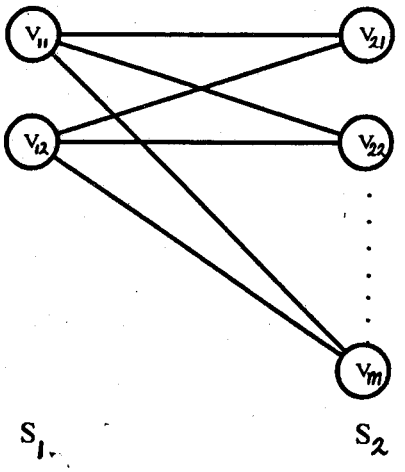


Figure 2-10: $K_{2,m}$, where $m \geq 2$

Let $S_1 = \{ v_{11}, v_{12}\}$, $S_2 = \{ v_{21}, v_{22}, \ldots, v_{2(n-2)}\}$.

i) If $s \in S_1$, say $s = v_{11}$, consider the following polling scheme:

Queries are sent in the following manner :

(Time)

1 : v_{11} queries v_{21}

2 : v_{11} queries v_{22}
v_{21} queries v_{12}

3 : v_{12} queries v_{23}

5 : v_{12} queries v_{24}

6 : v_{12} queries v_{25}

:

n-1 : v_{12} queries $v_{2(n-2)}$

Responses are returned as follows :

(Time)

3 : v_{21} returns a response to v_{11}

4 : v_{22} returns a response to v_{11}
v_{12} returns a response to v_{21}

5 : v_{21} returns a response to v_{11}

6 : v_{23} returns a response to v_{11}

7 : v_{24} returns a response to v_{11}

:

n-1 : $v_{2(n-4)}$ returns a response to v_{11}

n : $v_{2(n-3)}$ returns a response to v_{11}

n+1 : $v_{2(n-2)}$ returns a response to v_{11}

This scheme polls the graph in n+1 time units. By Theorem 2.3, $P(G,s) = n+1$.

ii) If $s \in S_2$, say $s = v_{21}$, consider a spanning tree as shown in Figure 2-11, in which $\deg(v_{11}) = \deg(v_{12})$ or $\deg(v_{12}) = \deg(v_{11}) + 1$.

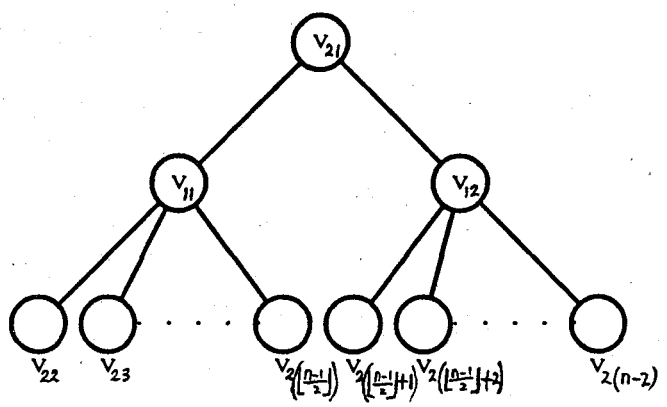


Figure 2-11: A special form of centroid tree in which degree(v_{11}) = degree(v_{12}) or degree(v_{12}) = degree(v_{11})+1

The tree polling scheme [4] can be used to poll the spanning tree in time $n + \lceil (n-3)/2 \rceil$. Thus $\tilde{P}(G,s) \leq n + \lceil (n-3)/2 \rceil$.

To see that this much time is required, note that v_{11} and v_{12} act as bottlenecks for all message transfers. Therefore we can consider the total number of calls in which v_{11} and v_{12} are involved :

1. Both v_{11} and v_{12} receive the query (one call each).

2. Each of the remaining n-3 vertices in $S_2-\{v_{21}\}$ must be queried by either v_{11} or v_{12}.

3. All n-3 responses must be returned to either v_{11} or v_{12}.

4. v_{11} and v_{12} must return a total of n-1 responses to the polling station v_{21}.

Thus, a total of 2 + (n-3) + (n-3) + (n-1) = 3n-5 calls involving v_{11} or v_{12} are required for polling.

At best v_{11} and v_{12} are involved in $\lceil(3n-5)/2\rceil$ and $\lceil(3n-5)/2\rceil$ calls respectively, and thus, at least $\lceil(3n-5)/2\rceil$ time units are required for polling. However, one extra time unit is required. When v_{11} and v_{12} are involved in the same number of calls (n odd), they cannot make their last calls to v_{21} simultaneously. They must finish in consecutive time units.

When the number of calls differs by one (n even), only one of v_{11}, v_{12} can receive the query at time 1. If v_{11} is queried first, then v_{12} must be involved in $\lceil(3n-5)/2\rceil$ calls beginning no earlier than time 2 and will finish no sooner than at time $\lceil(3n-5)/2\rceil$. In this event, as above, the two final calls cannot occur simultaneously at time $\lceil(3n-5)/2\rceil$, but must be completed in consecutive time units.

Thus, at least $\lceil(3n-5)/2\rceil + 1 = n + \lceil(n-3)/2\rceil$ time units are required.

CASE 3 : Let $|S_1| = j > 2$ and $|S_2| = k > 2$. Consider the diagram as shown in Figure 2-12:

i) If $s \in S_1$, say $s = v_{11}$, then consider the following polling scheme:

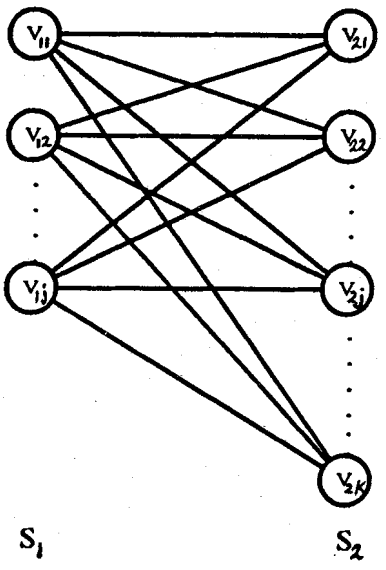


Figure 2-12: $K_{j,k}$, where $k \geq j > 2$

Queries are sent in the following manner:

(Time)

1 : v_{11} queries v_{21}

2 : v_{11} queries v_{22}
v_{21} queries v_{12}

:

i : $v_{1(i-1)}$ queries v_{2i}
$v_{2(i-1)}$ queries v_{1i}

(where $2 \leq i \leq j$)

j+1 : v_{1j} queries $v_{2(j+1)}$

j+3 : v_{1j} queries $v_{2(j+2)}$

j+4 : v_{1j} queries $v_{2(j+3)}$

:

k+1 : v_{1j} queries v_{2k}

Responses are sent as follows:

(Times)

3,5,....,2j+1 : v_{21} returns a response to v_{11}

i+2,i+4,..,2j-i+2 : v_{1i} returns a response to $v_{2(i-1)}$
v_{2i} returns a response to $v_{1(i-1)}$

(where $2 \leqslant i \leqslant j$)

2j+2 : $v_{2(j+1)}$ returns a response to v_{11}

2j+3 : $v_{2(j+2)}$ returns a response to v_{11}

:

k+j+1 = n+1 : v_{2k} returns a response to v_{11}

This polling scheme uses n+1 time units which is optimal by Theorem 2.3.

ii) If $s \in S_2$, say $s = v_{21}$, consider the following polling scheme:

Queries are sent as the following manner:

(Time)

1 : v_{21} queries v_{11}

2 : v_{21} queries v_{12}
v_{11} queries v_{22}

3 : v_{12} queries v_{23}
v_{22} queries v_{13}

:

i : $v_{1(i-1)}$ queries v_{2i}
$v_{2(i-1)}$ queries v_{1i}

(where $2 \leqslant i \leqslant j$)

j+1 : v_{1j} queries $v_{2(j+1)}$

j+3 : v_{1j} queries $v_{2(j+2)}$

j+4 : v_{1j} queries $v_{2(j+3)}$

:

k+1 : v_{1j} queries v_{2k}

Responses are sent as follows :

(Times)

3,5,..2j+1 : v_{11} returns a response to v_{21}

i+2,i+4,..2j-i+2 : v_{2i} returns a response to $v_{1(i-1)}$
v_{1i} returns a response to $v_{2(i-1)}$

(where $2 \leqslant i \leqslant j$)

2j+1 : $v_{2(j+1)}$ returns a response to v_{12}

2j+2 : v_{12} returns a response to v_{21}
$v_{2(j+2)}$ returns a response to v_{11}

2j+3 : v_{11} returns a response to v_{21}
$v_{2(j+3)}$ returns a response to v_{12}

:

k+j : v_{11} returns a response to v_{21}
v_{2k} returns a response to v_{12}
(if k is odd)

or

v_{12} returns a response to v_{21}
v_{2k} returns a response to v_{11}
(if k is even)

$k+j+1 = n+1$: v_{11} returns a response to v_{21}
(if k is even)

or

v_{12} returns a response to v_{21}
(if k is odd)

Again, this polling scheme uses n+1 time units which is optimal by Theorem 2.3. ■

From this Theorem, we know the polling time for each vertex in a complete bipartite graph. The polling time of the graph and the polling center are also known.

Corollary 2.13: For a complete bipartite graph G on two sets of vertices S_1 and S_2 with $|S_1| \leq |S_2|$ and $|S_1| + |S_2| = n$,

$$P(G) = \begin{cases} 3n-4 & \text{if } |S_1| = 1 \\ n+\lceil (n-3)/2 \rceil & \text{if } |S_1| = 2 \\ n+1 & \text{if } |S_1| > 2. \end{cases}$$

Corollary 2.14: The polling center of a complete bipartite graph $K_{p,m}$ is V unless $p = 1$, $m > 1$ or $1 = 2$, $m \geq 4$. The polling center of $K_{1,m}$ with $m > 1$ is the single vertex set S_1. The polling center of $K_{2,m}$ with $m \geq 4$ is the 2 vertex set S_1.

2.4.2. Polling in Complete Tripartite Graphs

Consider a complete tripartite graph on three sets of vertices S_1, S_2 and S_3, with $S_1 = \{v_{11}, v_{12}, \ldots v_{1j}\}$, $S_2 = \{v_{21}, v_{22}, \ldots v_{2k}\}$ and $S_3 = \{v_{31}, v_{32}, \ldots v_{3m}\}$, where $j \leqslant k \leqslant m$ and $j + k + m = n$.

Lemma 2.15: Given a complete tripartite graph G on three sets of vertices S_1, S_2 and S_3, with $|S_1| \leqslant |S_2| \leqslant |S_3|$. If $|S_1| = 1$, $|S_2| = 1$ and $|S_3| > 1$, then the edge (v_{11},v_{21}) can be deleted without increasing the polling time.

Proof: If the edge (v_{11},v_{21}) is deleted, then the resulting graph $G' = G-(v_{11},v_{21})$ is a complete bipartite graph in which $S_1' = S_1 \cup S_2$ and $S_2' = S_3$. If the polling station s is in $S_1 \cup S_2$, the complete bipartite graph polling scheme can be used. The scheme uses only n+1 time units which is optimal. Thus, the polling time is not increased by deleting the edge. If the polling station s is in S_3, then

1. We first show that the edge (v_{11}, v_{21}) need not be used for querying. Without loss of generality, let v_{11} be the vertex being queried by the polling station s at time one. At time two if v_{11} queries v_{21} through (v_{11},v_{21}), then only two vertices v_{11} and v_{21} are queried and s must remain idle during this time period. Thus, s could query v_{21} directly at time 2 with no resulting change to the polling time of the algorithm, and therefore the edge (v_{11}, v_{21}) need not be used for querying.

2. We now show that the edge (v_{11}, v_{21}) need not be used for returning responses. Without loss of generality, assume that v_{11} has a response to return to the polling station s, then consider two cases :

   a. If the polling station s is busy, then it must be communicating with node v_{21}. Hence, v_{11} cannot send the response through (v_{11}, v_{21}) to v_{21} at that time.

   b. If the polling station s is not busy and v_{11} sends the response to v_{21} through (v_{11}, v_{21}) instead of returning to s directly, then the polling station s must be idle at this time period. This call can be replaced by (v_{11}, s).

Thus the edge can be deleted without increasing the polling time. ■

Theorem 2.16: Given a complete tripartite graph G on three sets of vertices S_1 , S_2 and S_3, with $|S_1| \leq |S_2| \leq |S_3|$ and $|S_1| + |S_2| + |S_3| = n$.

$$P(G,s) = \begin{cases} n+\lceil (n-3)/2 \rceil & \text{if } s \in S_3, \text{ and } |S_1| = |S_2| = 1, |S_3| > 1 \\ n+1 & \text{otherwise} \end{cases}$$

Proof: There are four cases to consider :

CASE 1 : If $|S_1| = |S_2| = |S_3| = 1$, then the graph G is a cycle, and P(G) = n+1, and P(G,s) = n+1 for all s.

CASE 2 : If $|S_1| = |S_2| = 1$, and $|S_3| > 1$, then by Lemma 2.15, the edge (v_{11}, v_{21}) can be deleted without increasing the polling time and thus the graph $G-(v_{11},v_{21})$ is equivalent to the case of complete bipartite G' in

which $S_1' = S_1 \cup S_2$ and $S_2' = S_3$. By Theorem 2.12, $P(G',s) = n+1$ if s is in S_1' and $P(G',s) = n + \lceil (n-3)/2 \rceil$ otherwise.

CASE 3 : If $|S_1| = 1$, $|S_2| > 1$ and $|S_3| > 1$, we can convert the graph G into a complete bipartite G' as follows. By removing all edges between the vertices in S_1 and the vertices in S_2, we form a complete bipartite graph on vertex sets $S_1 \cup S_2$ and S_3.

If $|S_2| = |S_3| = 2$, then $|S_1 \cup S_2| = 3$ and $|S_3| = 2$. By Theorem 2.12 we know that using the complete bipartite graph scheme, $P(G',s) = n+1$ if $s \in S_3$ and $P(G',s) = n + \lceil (n-3)/2 \rceil = n+1$ (since n=5) if $s \in S_1 \cup S_2$. Otherwise, $|S_1 \cup S_2| > 2$ and $|S_3| > 2$ and by Theorem 2.12 using the complete bipartite graph scheme, $P(G',s) = n+1$ for any polling station s.

CASE 4 : If $|S_1| > 1$, $|S_2| > 1$ and $|S_3| > 1$, then consider the following cases :

1. If $|S_1| = |S_2| = |S_3| = 2$, then it is easy to obtain a Hamiltonian cycle $H = \langle v_{11}, v_{21}, v_{31}, v_{12}, v_{22}, v_{32}, v_{11} \rangle$ from G, and thus, $P(G,s) = n+1$ for all s.

2. If $|S_1| \geq 2$, $|S_2| \geq 2$ and $|S_3| > 2$, we can convert the graph G into a complete bipartite graph G' as follows. We simply remove all edges between the vertices in S_1 and vertices in S_2. Note that $|S_1 \cup S_2| > 2$ and $|S_3| > 2$. By Theorem 2.12, $P(G',s) = n+1$ for all s. ■

Again, from this Theorem, we know the polling time for each vertex in a complete tripartite graph. The polling time of the graph and the polling center are also known.

Corollary 2.17: Given a complete tripartite graph G on three sets of vertices S_1, S_2 and S_3, with $|S_1| \leq |S_2| \leq |S_3|$ and $|S_1| + |S_2| + |S_3| = n$. $P(G) = n + \lceil (n-3)/2 \rceil$.

Corollary 2.18: The polling center of a complete tripartite graph $K_{k,p,m}$ is V unless $k = p = 1$ and $m \geq 4$. The polling center for $K_{1,1,m}$ with $m \geq 4$ consists of the two single vertex sets S_1 and S_2.

2.4.3. Polling in Complete k-partite Graphs, k > 3

Theorem 2.19: Given a complete k-partite graph $G = (V,E)$, $k > 3$, on k sets of vertices, with $|S_1| \leq |S_2| \leq \ldots \leq |S_k|$ and $|S_1| + |S_2| + \ldots + |S_k| = n$, $P(G,s) = n+1$ for all s.

Proof: Consider the following cases :

1. If $|S_1| \leq 2$, $|S_2| \leq 2$, ..., $|S_k| \leq 2$, a Hamiltonian cycle can be obtained as follows :

   a. If $|S_i| = 1$ for all i, then $H = \langle v_{11}, v_{21}, \ldots, v_{k1}, v_{11} \rangle$ is a Hamiltonian cycle.

   b. Otherwise, let S_j be the first set having 2 elements, $1 \leq j \leq k$.

      If $j = k$, $H = \langle v_{k1}, v_{(k-1)1}, v_{k2}, v_{(k-2)1}, v_{(k-3)1}, \ldots, v_{11}, v_{k1} \rangle$ is a Hamiltonian cycle.

If $j = 1$, $H = \langle v_{k1}, v_{(k-1)1}, \ldots, v_{11}, v_{k2}, v_{(k-1)2}, \ldots, v_{12}, v_{k1} \rangle$ is a Hamiltonian cycle.

If $j \neq k$ and $j \neq 1$, then $H = \langle v_{k1}, v_{(k-1)1}, \ldots, v_{j1}, v_{k2}, v_{(k-1)2}, \ldots, v_{j2}, v_{(j-1)1}, v_{(j-2)1}, \ldots, v_{11}, v_{k1} \rangle$ is a Hamiltonian cycle.

By Corollary 2.6, $P(G,s) = n+1$ for all s.

2. If $|S_i| > 2$ for some i, $1 \leq i \leq k$, convert the graph G into a complete bipartite graph G' as follows. Choose a set S_i such that $|S_i| > 2$. Remove all edges between vertices in $V-S_i$. So G' contains two vertex sets S_i and $V-S_i$ with $|S_i| > 2$ and $|V-S_i| > 2$. By Theorem 2.12, $P(G',s) = n+1$ for all s and thus, $P(G,s) = n+1$ for all s. ■

From this Theorem, we know the polling time for each vertex in a complete k-partite graph, where $k > 3$. The polling time of the graph and the polling center are also known.

Corollary 2.20: Given a complete k-partite graph $G = (V,E)$, $k > 3$, on k sets of vertices, with $|S_1| \leq |S_2| \leq .. \leq |S_k|$ and $|S_1| + |S_2| + .. + |S_k| = n$, $P(G) = n+1$.

Corollary 2.21: The polling center of a complete k-partite graph with $k > 3$ is V.

2.5. Polling in 2-connected Graphs

There is an $O(|V|+|E|)$ algorithm [5] to obtain a centroid tree in a 2-connected graph. We propose to use the algorithm on a 2-connected graph and then to apply the tree polling algorithm [4] on the resulting tree. The resulting polling scheme will give us an upper bound on the polling time P(G,s).

Theorem 2.22: Given a 2-connected graph on n vertices, and any polling station s in G,

$$P(G,s) \leqslant \begin{cases} n+\lfloor n/2 \rfloor -1 & \text{if } n \text{ is odd} \\ 3n/2 & \text{otherwise} \end{cases}$$

Proof: Consider the tree resulting from the centroid tree algorithm [5]. With a specified centroid node s in a 2-connected graph G, the centroid tree algorithm produces a tree with two disjoint subtrees, S_1 rooted at s and S_2 rooted at t (a neighbour of s), which span G. Furthermore, $|S_1| = \lfloor n/2 \rfloor$ and $|S_2| = \lfloor n/2 \rfloor$. The centroid tree T is then formed by joining the two subtrees with the edge (s,t) as shown in Figure 2-13.

We can use the tree polling scheme of [4] on the resulting tree T. The time used by the resulting polling scheme is:

$$P(T,s) = \text{Max}\{\text{degree}(s)+|T|-1,\ ps(S,u)+i\}$$

where S is the subtree rooted at a child u of s
with the maximum subtree polling time, and
i = 1 if $ps(S,u) \leqslant ps(R,v)+1$ for
some subtree $R \neq S$.
= 0 otherwise.

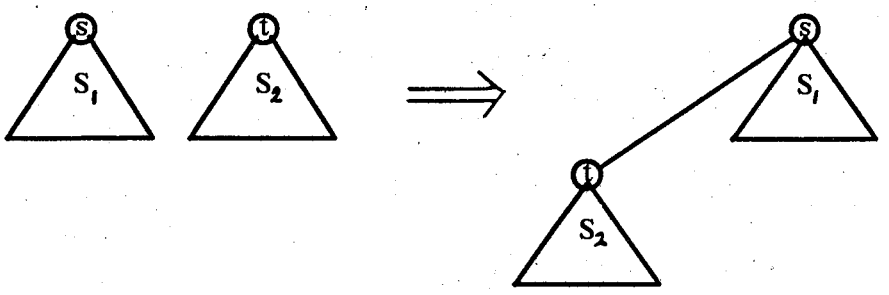


Figure 2-13: A centroid tree in which $|S_1| = \lceil n/2 \rceil$ and $|S_2| = \lfloor n/2 \rfloor$

Note that the first term of the formula is maximized when degree(s) is maximized. In Figure 2-13, the maximum occurs when degree(s) = $\lceil n/2 \rceil$ and T is of the form as shown in Figure 2-14. In this case, the first term of the formula is equal to $\lceil n/2 \rceil + n-1$.

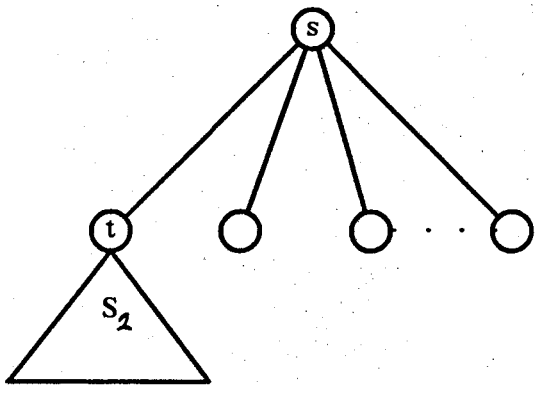


Figure 2-14: A form of trees of Figure 2-13 with degree(s) being maximized

Now consider the second term of the formula, ps(S,u)+i. We will

maximize this term for Figure 2-13. It easy to see that one of the subtrees of s is S_2 rooted at t with subtree polling time $ps(S_2,t) = 2|S_2|+1 + Gaps(S_2,t)$. So $ps(S_2,t)$ is maximized when $Gaps(S_2,t)$ is maximized. This maximum occurs when S_2 is a "star" at t. In this case, $Gaps(S_2,t) = degree(t)-2 = |S_2|-2$ and $ps(S_2,t) = 3|S_2|-1 = 3n/2 -1$. Note that there is no other subtree rooted at a child of s with subtree polling time greater than $ps(S_2,t)$. The second largest subtree R rooted at a child v of s must be in S_1. If $ps(R,v) = ps(S_2,t)$, the tree T must be of the form as shown in Figure 2-15.

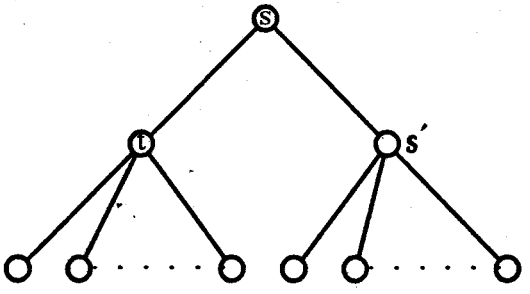


Figure 2-15: A form of trees of Figure 2-13 in which $|degree(t)-degree(s')| \leq 1$

If this is the case, the term $ps(S,u)+i$ is maximized. The value is $ps(S,u)+i = ps(S_2,t)+1 = (3*\lfloor n/2 \rfloor -1)+1 = 3*\lfloor n/2 \rfloor$.

As a result, the maximum possible value for $P(T,s)$ is:

$$P(T,s) = Max\{\lfloor n/2 \rfloor + n - 1,\ 3*\lfloor n/2 \rfloor\}$$
$$= Max\{2*\lfloor n/2 \rfloor + \lfloor n/2 \rfloor - 1,\ 3*\lfloor n/2 \rfloor\}.$$

Note that $2*\lfloor n/2 \rfloor - 1 > 2*\lfloor n/2 \rfloor$ if n is odd because $\lfloor n/2 \rfloor = (n+1)/2$ and $\lfloor n/2 \rfloor = (n-1)/2$ if n is odd. Thus, the maximum possible value of P(T,s) is $n+\lfloor n/2 \rfloor - 1$ if n is odd. If n is even, $\lfloor n/2 \rfloor = n/2$ and $\lfloor n/2 \rfloor = n/2$, and $2*\lfloor n/2 \rfloor - 1 < 2*\lfloor n/2 \rfloor$. Thus, the maximum possible value of P(T,s) is 3n/2 if n is even. ■

Chapter 3

Receiving in Unicyclic Graphs

In this chapter, we study an information dissemination process called **receiving**. As before, a network is modelled by a graph G = (V,E) with vertices corresponding to nodes and edges corresponding to communication lines. In [5], receiving is investigated for trees and 2-connected graphs. No result is known for **separable graphs** that are not trees. Separable graphs are graphs that can be decomposed into biconnected components (blocks). In this chapter, we consider receiving in a particular class of separable graphs called **unicyclic graphs**. Unicyclic graphs are graphs that have exactly one cycle.

We define **Weight(v) of an articulation point v** with respect to the receiver r as follows. When v is removed from a graph G = (V,E), along with its incident edges, the resulting graph consists of several components. Let R be the vertex set of the component containing r. Weight(v) = |V|-|R|. We define **Weight(v) of a non-articulation point v** to be 1, and use **d(v,r)** to denote the distance from vertex v to r, that is, the smallest number of edges in any path from v to r in G. A sequence of time periods 2k, 2(k+1), ..., 2(k+j) is said to have **even time parity** while the sequence 2k+1, 2(k+1)+1, ..., 2(k+j)+1 is said to have **odd time parity**. Recall that **R(G,v)** denotes the receiving time of node v in graph G.

The following results will be useful in proving results in later sections and in Chapter 4.

Lemma 3.1: Given a simple connected graph G on n vertices, if the receiver r is of degree one, then $R(G,r) \geq 2n-3$.

Proof: Consider any receiving scheme at the vertex b which is the only neighbour of the receiver r. Vertex b requires n-2 time units to receive all n-2 messages in addition to another n-1 time units to relay all n-1 messages to r. Thus, a total of 2n-3 time units required and $R(G,r) \geq 2n-3$. ■

Lemma 3.2: Given a graph G=(V,E). If vertex v is an articulation point with Weight(v) with respect to r, then $R(G,r) \geq 2*\text{Weight}(v) + d(v,r)-2$.

Proof: Consider the calls made by v in any receiving scheme. Since v is an articulation point, when v is removed along with all incident edges, the resulting graph consists of several components. Let R be the vertex set containing r and V-R-{v} be another vertex set. Note that v must receive $|V-R-\{v\}| = \text{Weight}(v)-1$ messages from V-R-{v}, and relay a total of Weight(v) messages to r. Thus, it requires a total of $2*\text{Weight}(v)-1$ time units. The last message is sent out by v no earlier than time $2*\text{Weight}(v)-1$ and reaches r no earlier than $d(v,r)-1$ time units later. Thus, the last message is received by r no earlier than at time $2*\text{Weight}(v)-1 + d(v,r)-1$. It follows that $R(G,r) \geq 2*\text{Weight}(v) + d(v,r)-2$. ■

3.1. Rooted Unicyclic Graphs

In this section, we consider receiving in a unicyclic graph with the receiver on the cycle. We define a **rooted unicyclic graph** $G_u(r)$ to be a unicyclic graph with a distinguished vertex r on the cycle. Let this distinguished vertex be the receiver.

Given a rooted unicyclic graph $G_u(r)$ with the cycle of length k+1, we label the cycle vertices (in order) $v_0, v_1, \ldots, v_k$ with $v_0 = r$. For each v_i, let T_{v_i} be tree consisting of v_i and those non-cycle vertices forming a tree rooted at v_i. Also let Short_Path denote the path from v_i to r of length $d(v_i,r)$ and Long_Path denote the other path from v_i to r which has length $\geq d(v_i,r)$.

Lemma 3.3: Given a rooted unicyclic graph $G_u(r)$, the messages from the vertices in T_{v_i} for some $i \neq 0$ can be received by r in $2|T_{v_i}| + d(v_i,r)-2$ time units and cannot be received faster. Thus, $R(G_u(r),r) \geq 2|T_{v_i}| + d(v_i,r) -2$.

Proof: The proof is similar to that of Lemma 3.2. Consider the calls made by v_i in any receiving scheme. v_i requires $|T_{v_i}|-1$ time units to receive all messages from $T_{v_i}-\{v_i\}$ and requires $|T_{v_i}|$ time units to relay all messages from T_{v_i} to either v_{i-1} or v_{i+1}. Thus, v_i requires a total of $2|T_{v_i}|-1$ time units to finish all message transfers for T_{v_i}. The last message from T_{v_i} is sent out from v_i no earlier than time $2|T_{v_i}|-1$ and reaches r no earlier than $d(v_i,r)-1$ time units later. This requires a minimum of $2|T_{v_i}| + d(v_i,r)-2$ time units. It then follows that $R(G_u(r,v_i),r) \geq 2|T_{v_i}| + d(v_i,r)-2$. ■

Lemma 3.4: A rooted unicyclic graph $G_u(r)$ can have at most one subtree T_{v_i} for some $i \neq 0$ such that $2|T_{v_i}| + d(v_i,r)-2 > n-1$.

Proof: By way of contradiction, assume that there exist two distinct vertices v_i, v_j such that $2|T_{v_i}| + d(v_i,r)-2 > n-1$ and $2|T_{v_j}| + d(v_j,r)-2 > n-1$. From these inequalities,

$$|T_{v_i}| > [n-d(v_i,r)+1]/2, \text{ and}$$
$$|T_{v_j}| > [n-d(v_j,r)+1]/2$$

Since there are k+1 cycle vertices. If two such subtrees exist, then

$$n \geq |T_{v_i}| + |T_{v_j}| + (k+1) - 2$$

Since $k+1 \geq 2*d(v_i,r)$ and $k+1 \geq 2*d(v_j,r)$, then $k+1 \geq d(v_i,r)+d(v_j,r)$.

$$\begin{aligned} n &\geq |T_{v_i}| + |T_{v_j}| + d(v_i,r) + d(v_j,r) - 2 \\ &> [n-d(v_i,r)+1]/2 + [n-d(v_j,r)+1]/2 + d(v_i,r) \\ &\quad + d(v_j,r) -2 \end{aligned}$$

$$2n > 2n + d(v_i,r) + d(v_j,r) -2$$

$$\Longrightarrow 2 > d(v_i,r) + d(v_j,r).$$

Since $i \neq 0$, $d(v_i,r), d(v_j,r) \geq 1$. This cannot happen. So, by contradiction, only one such subtree can exist. ■

To find the receiving time of a given rooted unicyclic graph, we proceed as follows. If such a graph $G_u(r)$ admits a spanning tree rooted at r with r being in the centroid of the tree, we simply construct the spanning tree with centroid vertex r and apply the tree receiving scheme [5] to it. If the subtree T_{v_0} rooted at r, consisting of r and those non-cycle vertices, is of size $|T_{v_0}| \geq \lceil n/2 \rceil$, then we

construct an arbitrary spanning tree and apply the tree receiving scheme to it. Note that if $|T_{v_0}| = \lfloor n/2 \rfloor$, a centroid tree can be produced. If $G_u(r)$ contains a subtree S rooted at v consisting of non-cycle vertices and cycle vertex v, where v is adjacent to r, such that $|S| > \lfloor n/2 \rfloor$, then we construct a spanning tree by deleting the edge (v,w), where both v and w are cycle vertex and $w \neq r$. The tree receiving scheme is then applied to it. Otherwise the unicyclic graph has the following properties :

1. There is no spanning tree with centroid r.

2. The subtree T_{v_0} rooted at r is of size $|T_{v_0}| \leq \lfloor n/2 \rfloor = \lfloor (n-1)/2 \rfloor$.

3. There does not exist a subtree S rooted at a child u of r, consisting of non-cycle vertices (except that u can be a cycle vertex), with $|S| > \lfloor n/2 \rfloor = \lfloor (n-1)/2 \rfloor$.

We define this particular class of rooted unicyclic graphs to be rooted **unbalanced unicyclic graphs.**

3.2. Rooted Unbalanced Unicyclic Graphs

Given a rooted unbalanced unicyclic graph with the cycle of length k+1, we label the cycle vertices (in order) $v_0, v_1, \ldots, v_k$ with $v_0 = r$. For each v_i, let T_{v_i} be the tree consisting of v_i and those non-cycle vertices forming a tree rooted at v_i. Also let

$$A_i = \begin{cases} T_{v_1} \cup T_{v_2} \cup \ldots \cup T_{v_i} & \text{if } i \leq k-i+1,\ i \neq 0 \\ T_{v_0} & \text{if } i = 0 \\ T_{v_i} \cup T_{v_{i+1}} \cup \ldots \cup T_{v_k} & \text{if } i > k-i+1,\ i \neq 0 \end{cases}$$

$$B_i = \begin{cases} T_{v_1} \cup T_{v_2} \cup .. \cup T_{v_i} & \text{if } i > k-i+1,\ i \neq 0 \\ T_{v_0} & \text{if } i = 0 \\ T_{v_i} \cup T_{v_{i+1}} \cup .. \cup T_{v_k} & \text{if } i \leq k-i+1,\ i \neq 0 \end{cases}$$

We choose a vertex v_i such that $|A_i| > \lceil (n-1)/2 \rceil$ and $|B_i| > \lceil (n-1)/2 \rceil$. Let $G_u(r,v_i)$ denote this particular graph with root r and the chosen vertex v_i. Note that Short_Path is contained in A_i and Long_Path is contained in B_i. Note that $i \neq 0$ because if $i = 0$, then $A_0 = B_0 = T_{v_0}$ and $|T_{v_0}| > \lceil (n-1)/2 \rceil$, violating property (2) in the previous section. Also note that $i \neq 1$ and $i \neq k$ because of property (3) in the previous section. This implies that $d(v_i,r)$ is always > 1 in $G_u(r,v_i)$. Furthermore, such a v_i always exists in a rooted unbalanced unicyclic graph because of properties (1), (2) and (3) in the previous section. Since a rooted unbalanced unicyclic graph is a rooted unicyclic graph, Lemmas 3.3 and 3.4 hold for rooted unbalanced unicyclic graphs.

3.3. Description of the scheme for $G_u(r,v_i)$

The general strategy is to send roughly half of the messages to r in each direction along the path. In particular, we need to send each message in T_{v_j} to v_j and then to r by the path from v_j to r which does not include v_i. This strategy is followed except for the particular subtree T_{v_i}.

We know that in $G_u(r,v_i)$, $|A_i| > \lceil (n-1)/2 \rceil$ and $|B_i| > \lceil (n-1)/2 \rceil$. Instead of sending all of the messages in T_{v_i} along Short_Path or Long_Path, we split the messages from T_{v_i} into two sets. The first set is sent along Long_Path and the second is sent along Short_Path.

The scheme essentially allows the messages in T_{v_i} rooted at v_i to get out from v_i to Long_Path or Short_Path as soon as possible. The scheme consists of two phases. In phase 1, v_i relays the messages from T_{v_i} by way of Long_Path to r with the highest priority. That is, the messages from T_{v_i} are relayed through the cycle while messages from some other subtree T_{v_k} may have to wait. Meanwhile, for each cycle vertex u in Short_Path, if no message is available from its neighbour cycle vertex w, where $d(w,r) > d(u,r)$, then messages from T_u are continuously relayed by way of Short_Path to r. In phase 2, v_i switches, sending the remaining messages in T_{v_i} to Short_Path. Meanwhile, for each cycle vertex u in Long_Path, if no message is available from its neighbour cycle vertex w, where $d(w,r) > d(u,r)$, then messages from T_u are continuously relayed along Long_Path to r. The time for switching from phase 1 to phase 2 can be determined from the given $G_u(r,v_i)$.

The scheme essentially partitions the n-1 messages into two sets, S_1 and S_2. If S_1 is the set from which r receives at odd time periods, then S_2 is the set from which r receives at even time periods and vice versa. Note that when $|S_1| = \lceil (n-1)/2 \rceil$ and $|S_2| = \lfloor (n-1)/2 \rfloor$ or vice versa, it is easily verified that $R(G_u(r,v_i),r) = n-1$, which is the best possible time. However, whether r receives messages from S_1 at odd or even time periods is not arbitrary but depends on the parities of the lengths of Short_Path and Long_Path. Since v_i gets the messages out of T_{v_i} every odd time period beginning at time 1, then during the switching process from phase 1 to phase 2, it is desirable to arrange the time parities of Short_Path and Long_Path such that both C_{i+1} and C_{i-1} request a message from v_i at odd time periods if possible. Otherwise one time unit of delay may be introduced.

In the scheme, we let $S_1 = A_i - X$ and $S_2 = X \cup B_i - T_{v_i} \cup T_{v_0} - \{r\}$, where X is the set of messages in T_{v_i} that v_i sends to r by way of Long_Path. Therefore,

every message in S_1 must travel through Short_Path to get to r. Every message in S_2 other than those in T_{v_0}-{r} must travel through Long_Path to get to r.

By analyzing the cases, we can determine that if Short_Path is of even length and Long_Path is of odd length, the first message should be received at time 1 from Long_Path. In all other cases, the first message should be received at time 1 from Short_Path.

Note that if r receives messages from S_1 at odd time periods beginning at time 1, then the goal of the scheme is to try to make $|S_1| = \lceil (n-1)/2 \rceil$ and $|S_2| = \lfloor (n-1)/2 \rfloor$. In this case, we would like to send $\lceil (n-1)/2 \rceil - |B_i - T_{v_i} \cup T_{v_0} - \{r\}|$ messages from T_{v_i} along Long_Path. On the other hand, if r receives messages from S_2 at odd time periods beginning at time 1, then the goal of the scheme is to try to make $|S_2| = \lceil (n-1)/2 \rceil$ and $|S_1| = \lfloor (n-1)/2 \rfloor$. In this case, we would like to send $\lceil (n-1)/2 \rceil - |B_i - T_{v_i} \cup T_{v_0} - \{r\}|$ messages from T_{v_i} along Long_Path. In particular, if Short_Path is of even length and Long_Path is of odd length, r starts to receive messages from S_2 beginning at time 1 and we would like to send $\lceil (n-1)/2 \rceil - |B_i - T_{v_i} \cup T_{v_0} - \{r\}|$ messages from T_{v_i} along Long_Path. Otherwise, r starts to receive messages from S_1 beginning at time 1 and we would like to send $\lfloor (n-1)/2 \rfloor - |B_i - T_{v_i} \cup T_{v_0} - \{r\}|$ messages from T_{v_i} along Long_Path.

Thus, if enough messages can be sent to from T_{v_i} along Long_Path, then $|S_1| = \lceil (n-1)/2 \rceil$ and $|S_2| = \lfloor (n-1)/2 \rfloor$ or vice versa, and r can receive the last message at time n-1. However, this is not always the case, since we must also ensure that messages continually arrive at r on both of the paths. We must determine the 'latest time' by which v_i must send the first message in T_{v_i} to Short_Path in order to avoid causing any idle time at r. This time can be determined as the following example shows :

If r starts receiving from S_1 at odd time periods, then the messages in A_i-T_{v_i} can keep r busy at all odd time periods up to time $2|A_i\text{-}T_{v_i}|-1$ in S_1. At time $2|A_i\text{-}T_{v_i}|+1$, r must receive the first message from T_{v_i}. Thus, v_i must send the first message to Short_Path $d(v_i,r)$ time units earlier in order to avoid causing idle time at r. Thus, at time $2|A_i\text{-}T_{v_i}|+1-d(v_i,r)$, v_i must send the first message to Short_Path. Since v_i requires 2 time units to relay a message from one vertex to another, v_i can advance $|A_i\text{-}T_{v_i}| - [d(v_i,r)-1]/2$ messages to Long_Path during phase 1. Thus, by analyzing the cases, we can determine that if both paths are of even length, v_i can advance $|A_i\text{-}T_{v_i}| - \lfloor[d(v_i,r)-1]/2\rfloor$ messages to Long_Path during phase 1. In all other cases, v_i can advance $|A_i\text{-}T_{v_i}| - \lfloor[d(v_i,r)-1]/2\rfloor$ messages to Long_Path during phase 1.

|X| can now be determined. We will send as many messages as possible along Long_Path provided that no idle time is introduced at r. Thus,

1. If both paths are of even length, $|X| = \text{Min}\ \{|A_i\text{-}T_{v_i}| - \lfloor[d(v_i,r)-1]/2\rfloor, \lfloor(n-1)/2\rfloor - (|B_i\text{-}T_{v_i}| + |T_{v_0}\text{-}\{r\}|)\}$

2. If Short_Path is of even length and Long_Path is of odd length, $|X| = \text{Min}\ \{|A_i\text{-}T_{v_i}| - \lfloor[d(v_i,r)-1]/2\rfloor, \lfloor(n-1)/2\rfloor - (|B_i\text{-}T_{v_i}| + |T_{v_0}\text{-}\{r\}|)\}$

3. In the two remaining cases, $|X| = \text{Min}\ \{|A_i\text{-}T_{v_i}| - \lfloor[d(v_i,r)-1]/2\rfloor, \lfloor(n-1)/2\rfloor - (|B_i\text{-}T_{v_i}| + |T_{v_0}\text{-}\{r\}|)\}$.

Receiving Scheme for $G_u(r,v_i)$

Each non-cycle vertex sends a message toward the cycle unless its parent is sending to or receiving from another vertex.

1. If Short_Path is of even length and Long_Path is of odd length, then $|X|$ $= \text{Min } \{|A_i - T_{v_i}| - \lfloor [d(v_i,r)-1]/2 \rfloor, \lfloor (n-1)/2 \rfloor - (|B_i - T_{v_i}| + |T_{v_0} - \{r\}|)\}$

In **phase 1** :

a. r receives messages from Short_Path at even time periods starting at time 2 and from Long_Path at odd time periods starting at time 1.

b. v_i starts sending a message by way of Long_Path to r at time 1 and subsequently sends a message at every odd period until $|X|$ messages have been sent out. Each of these messages is relayed by way of Long_Path to r with the highest priority. That is, the messages from T_{v_i} are relayed through Long_Path while messages from some other subtree T_{v_k} may have to wait.

c. For each cycle vertex u in Short_Path, if no message is available from its neighbour cycle vertex w, where $d(w,r) > d(u,r)$, then it relays a message from T_u along Short_Path to r.

In **phase 2** :

a. At time $2|X|+1$ v_i switches, relaying the rest of the messages in T_{v_i} to Short_Path every odd period. Each of these messages is then

relayed by way of Short_Path to r with the lowest priority. That is, messages from some subtree T_{v_k} are relayed through Short_Path while the messages from T_{v_i} may have to wait.

b. For each cycle vertex u in Long_Path, if no message is available from its neighbour cycle vertex w, where $d(w,r) > d(u,r)$, then it relays a message from T_u along Long_Path to r.

c. After all messages from $X \cup B_i - T_{v_i}$ (in S_2) are received by r at time t' and if $|T_{v_0} - \{r\}| > 0$, messages from $T_{v_0} - \{r\}$ will be received by r every second period starting at time t'+2 until $t' + 2|T_{v_0} - \{r\}|$.

Figure 3-1 and Figure 3-2 are examples to illustrate how the scheme works in case 1. The first example (Figure 3-1) shows a case in which it is possible to balance the sizes of S_1 and S_2. In the second example (Figure 3-2), it is necessary to forward messages along Short_Path beginning at time 5 in order to keep messages arriving at r via Short_Path. Thus, we are not able to balance the sizes of S_1 and S_2.

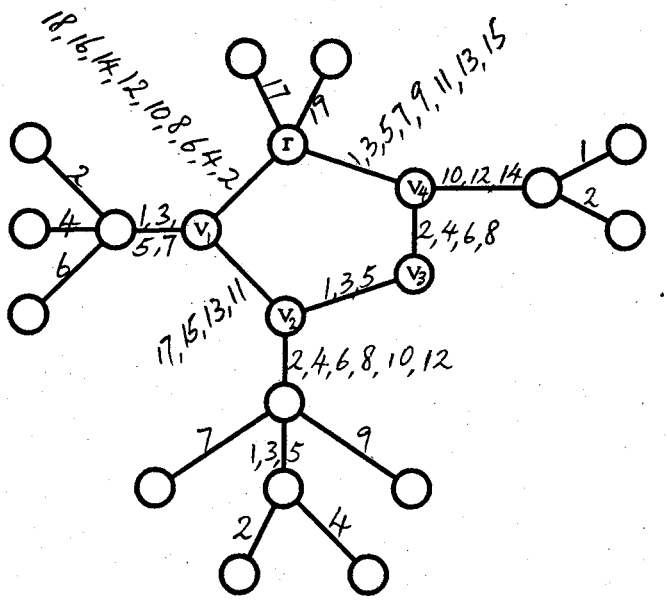


Figure 3-1: An example in which the sizes of S_1 and S_2 can be balanced, and $R(G_u(r,v_2),r) = n-1$

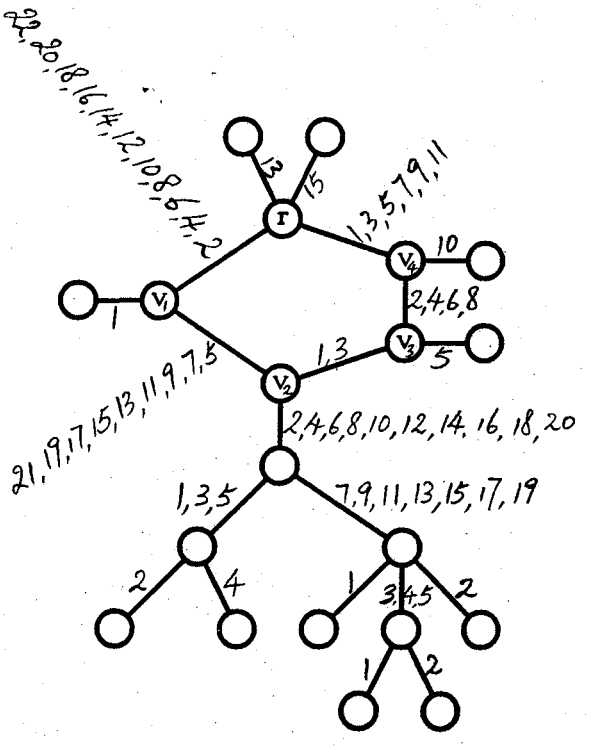


Figure 3-2: An example in which the sizes of S_1 and S_2 cannot be balanced, and $R(G_u(r,v_2),r) = 2|T_{v_2}| + d(v_2,r) - 2$

2. If Short_Path is of odd length and Long_Path is of even length, then $|X|$ = Min $\{|A_i-T_{v_i}| - \lfloor[d(v_i,r)-1]/2\rfloor, \lfloor(n-1)/2\rfloor - (|B_i-T_{v_i}| + |T_{v_0}-\{r\}|)\}$.

In **phase** 1 :

a. r receives messages from Short_Path at odd time periods starting at time 1 and receives messages from Long_Path at even time periods starting at time 2.

b. same as **phase** 1(b) in Case 1

c. same as **phase** 1(c) in Case 1

In **phase** 2 :

Same as **phase** 2 in Case 1.

3. If Short_Path is of odd length and Long_Path is of odd length, then $|X|$ = Min $\{|A_i-T_{v_i}| - \lfloor[d(v_i,r)-1]/2\rfloor, \lfloor(n-1)/2\rfloor - (|B_i-T_{v_i}| + |T_{v_0}-\{r\}|)\}$

In **phase** 1 :

a. r receives messages from Short_Path at odd time periods starting at time 1 and receives messages from Long_Path at even time periods starting at time 2.

b. v_i receives a message from T_{v_i}-{v_i} at time 1 and starts sending a message by way of Long_Path to r at time 2 and subsequently sends a message at every even period until at time 2|X|. Each of these messages is relayed along Long_Path to r with the highest priority. That is, the messages from T_{v_i} are relayed through Long_Path while messages from some other subtree T_{v_k} may have to wait.

c. For each cycle vertex u in Short_Path, if no message is available from its neighbour cycle vertex w, where d(w,r) > d(u,r), then it relays a message from T_u along Short_Path to r.

phase 2 :

a. At time 2|X|+1 v_i sends its own message by way of Short_Path if the neighbour cycle vertex on Short_Path is ready to accept one. If so, all messages in T_{v_i}-{v_i} are deferred by 1 time unit so that v_i is able to receive and send the next message every second period in the other time parity. Each message received by v_i from T_{v_i}-{v_i} is then relayed by way of Short_Path to r with the lowest priority. That is, messages from some subtree T_{v_k} are relayed through Short_Path while the messages from T_{v_i} may have to wait.

b. For each cycle vertex u in Long_Path, if no message is available from its neighbour cycle vertex w, where d(w,r) > d(u,r), then it relays a message from T_u by way of Long_Path to r.

c. After all messages from $X \cup B_i - T_{v_i}$ (in S_2) are received by r at time t' and if $|T_{v_0} - \{r\}| > 0$, then messages from $T_{v_0} - \{r\}$ will be received by r every second period starting at time t'+2 until t' + $2|T_{v_0} - \{r\}|$.

4. If Short_Path is of even length and Long_Path is of even length, then |X| = Min $\{|A_i - T_{v_i}| - \lceil [d(v_i,r)-1]/2 \rceil, \lfloor (n-1)/2 \rfloor - (|B_i - T_{v_i}| + |T_{v_0} - \{r\}|)\}$

In **phase 1** :

a. r receives messages from Short_Path at odd time periods starting at time 1 and receives messages from Long_Path at even time periods starting at time 2.

b. v_i starts sending a message by way of Long_Path to r at time 1 and then next message every odd period until |X| messages have been sent out. Each of these messages is relayed by way of Long_Path to r with the highest priority. That is, the messages from T_{v_i} are relayed through Long_Path while messages from some other subtree T_{v_k} may have to wait.

c. For each cycle vertex u in Short_Path, if no message is available from its neighbour cycle vertex w, where d(w,r) > d(u,r), then it relays a message from T_u along Short_Path to r.

In **phase** 2:

a. If $|X| = |A_i - T_{v_i}| - \lceil [d(v_i,r)-1]/2 \rceil < \lfloor (n-1)/2 \rfloor - (|B_i - T_{v_i}| + |T_{v_0} - \{r\}|)$, then at time $2|X|+2$ or else at time $2|X|+1$, v_i switches sending the rest of the messages in T_{v_i} by way of Short_Path to r every second period. Each of these messages is then relayed along Short_Path to r with the lowest priority. That is, messages from some subtree T_{v_k} are relayed through Short_Path while the messages from T_{v_i} may have to wait.

b. For each cycle vertex u in Long_Path, if no message is available from its neighbour cycle vertex w, where $d(w,r) > d(u,r)$, then it relays a message from T_u Long_Path to r.

c. After all messages from $X \cup B_i - T_{v_i}$ (in S_2) are received by r at time t' and if $|T_{v_0} - \{r\}| > 0$, then messages from $T_{v_0} - \{r\}$ will be received by r every second period starting at time $t'+2$ until $t'+2|T_{v_0} - \{r\}|$.

d. If $|X| = |A_i - T_{v_i}| - \lceil [d(v_i,r)-1]/2 \rceil < \lfloor (n-1)/2 \rfloor - [|B_i - T_{v_i}| + |T_{v_0} - \{r\}|]$ and |Short_Path| = |Long_Path|, then the last message from T_{v_i} is sent by way of Long_Path to r.

Lemma 3.5: In Case 1 of the above scheme, r receives a message from S_1 at even time periods and from S_2 at odd time periods.

Proof: In **phase** 1 Step (a), r receives the first message from Short_Path (S_1) at time 2 and the first message from Long_Path (S_2) at time 1. Thus, the lemma holds for Step (a). For Step (b), r can receive a message from Long_Path every odd period because in addition to the vertices on the Long_Path, v_i keeps providing messages to Long_Path continuously without any delay. For Step (c), it is easily verified that r can receive a message from the Short_Path every even period because messages from the subtree vertices are continuously sent along Short_Path without any delay.

In **phase** 2, for Step (a), X is the set of messages that v_i sends by way of Long_Path to r and thus, during time 1 and time 2|X|-1, v_i is busy sending out the |X| messages. |X| is chosen such that no idle time will occur at r for the switching process from **phase** 1 to **phase** 2 on both Short_Path and Long_Path. From time 1 to 2|X|-1, v_i is busy sending messages by way of Long_Path to r without relaying a message to Short_Path. However, by time 2|X|+1, v_i must relay a message from T_{v_i} to Short_Path and be able to relay a message to the Short_Path every second period in order to avoid causing any idle time at r. Step (a) has done all these and been able to maintain the time parities in both paths started in **phase** 1 without any delay. Thus, the lemma holds for this step. For Step (b), as before, it is easily verified that r can receive a message from Long_Path every odd period because whenever a cycle vertex u on Long_Path runs out of

messages from its neighbour cycle vertex w, where d(w,r) > d(u,r), messages from T_u are continuously relayed along Long_Path without any delay. For Step (c), it is easy to see that r receives a message from T_{v_0}-{r} every second period followed with the same time parity as Long_Path does.

Since no delay occurs in the scheme, r receives a message from S_1 at even time periods and from S_2 at odd time periods. ■

Lemma 3.6: In Case 2 and 3 of the scheme, r receives a message from S_1 at odd time periods and from S_2 at even time periods.

Proof: The proof for Case 2 is similar to that of the previous lemma, except that we have a different time parity for S_1 and S_2.

The proof for Case 3 : In **phase** 1 Step (a), r receives the first message from Short_Path (S_1) at time 1 and receives the first message from Long_Path (S_2) at time 2. Thus, the lemma holds for this step. For Step (b), v_i receives a message from one of its subtrees at time 1 and starts sending a message by way of Long_Path to r at time 2 and subsequently sends a message at every even period to Long_Path. Since r starts receiving from Long_Path at time 2, r can receive all these messages from v_i at even time periods without any delay. For Step (c), same as the previous lemma, r can receive a message from Short_Path every odd period because messages from the subtree vertices are continuously sent along the Short_Path without any delay.

In **phase** 2 Step (a), we choose |X| so that v_i sends |X| messages by

way of Long_Path to r without causing any delay before v_i starts sending messages to Short_Path. Since v_i starts sending the first message to Long_Path at time 2, v_i sends the last message to Long_Path at time 2|X|. We know that v_i must relay a message to Short_Path no later than at time 2|X|+1 in order to avoid any delay. Because there is a message waiting at v_i since time 1, at time 2|X|+1 v_i can send this message and the rest of the messages in T_{v_i} by way of Short_Path to r in the subsequent odd periods. Thus the switching process from **phase** 1 to **phase** 2 causes no delay on either Short_Path or Long_Path and is able to maintain the time parities in both paths started in **phase** 1.

For Step (b) and Step (c), similar arguments can be found in the previous lemma. As a result, no delay occurs in the scheme and r can receive a message from S_1 at odd time periods and receives a message from S_2 at even time periods. ■

Lemma 3.7: In Case 4 of the scheme, if $|A_i-T_{v_i}| - \lceil[d(v_i,r)-1]/2\rceil < \lfloor(n-1)/2\rfloor - (|B_i-T_{v_i}| + |T_{v_0}-\{r\}|)$ and |Short_Path| < |Long_Path|, then one time unit of delay will occur at v_i. Otherwise, no delay will occur.

Proof: If $|X| = \lfloor(n-1)/2\rfloor - (|B_i| + |T_{v_0}-\{r\}|) \leq |A_i-T_{v_i}| - \lceil[d(v_i,r)-1]/2\rceil$, then we can send as many messages to Long_Path as necessary to balance the sizes of S_1 and S_2 and can still ensure that messages continually arrive at r on both of the paths. Therefore, no delay will occur during the switching process from **phase 1** to **phase 2**. No other step will cause a 1 time unit delay as shown in the previous lemmas. Thus, it is true in this case.

Otherwise, we cannot balance the sizes of S_1 and S_2 since we must ensure that messages are continually relayed to r via Short_Path. In this case, at time $2|X|-1$ v_i sends the last message to Long_Path. At time $2|X|$ and $2|X|+1$, v_i requests a total of 2 messages from its subtrees and accumulates an extra message. At time $2|X|+2$ v_i switches, sending messages in T_{v_i} by way of Short_Path to r every even period but still maintains an extra message. If |Short_Path| < |Long_Path|, then the accumulated message will be sent by way of Short_Path to r. Note that the accumulated message at v_i cannot be sent to Short_Path until time $2|X|+2(|T_{v_i}-X|) = 2|T_{v_i}|$. Thus, a 1 time unit delay will occur at v_i in this case. However, if |Short_Path| = |Long_Path|, the accumulated message at v_i will be sent by way of Long_Path to r. This can be done right after v_i sends the second last message in T_{v_i} to Short_Path, that is, at time $2|X|+2(|T_{v_i}-X|)-1 = 2|T_{v_i}|-1$. Thus, no delay will occur at v_i in this case. ■

Now we can determine the resulting receiving times with the above lemmas when the scheme for $G_u(r,v_i)$ is applied.

Theorem 3.8: From the receiving scheme above, the receiver r can receive all n-1 messages in $G_u(r,v_i)$ in the following times:

1. When Short_Path is of even length and Long_Path is of odd length, and if $|X| = \lfloor (n-1)/2 \rfloor - (|B_i-T_{v_i}| + |T_{v_0}-\{r\}|)\} \leq |A_i-T_{v_i}| - \lfloor [d(v_i,r)-1]/2 \rfloor$, then $R(G_u(r,v_i)) \leq n-1$. Otherwise, $R(G_u(r,v_i)) \leq 2|T_{v_i}| + d(v_i,r) - 2$.

2. When Short_Path is of odd length and Long_Path is of even length,

and if $|X| = \lfloor (n-1)/2 \rfloor - (|B_i - T_{v_i}| + |T_{v_0} - \{r\}|)\} \leq |A_i - T_{v_i}| - \lfloor [d(v_i,r)-1]/2 \rfloor$, then $R(G_u(r,v_i)) \leq n-1$. Otherwise, $R(G_u(r,v_i)) \leq 2|T_{v_i}| + d(v_i,r) - 2$.

3. When Short_Path is of odd length and Long_Path is of odd length, and if $|X| = \lfloor (n-1)/2 \rfloor - (|B_i - T_{v_i}| + |T_{v_0} - \{r\}|)\} \leq |A_i - T_{v_i}| - \lfloor [d(v_i,r)-1]/2 \rfloor$, then $R(G_u(r,v_i)) \leq n-1$. Otherwise, $R(G_u(r,v_i)) \leq 2|T_{v_i}| + d(v_i,r) - 2$.

4. When Short_Path is of even length and Long_Path is of even length, and if $|X| = \lfloor (n-1)/2 \rfloor - (|B_i - T_{v_i}| + |T_{v_0} - \{r\}|)\} \leq |A_i - T_{v_i}| - \lfloor [d(v_i,r)-1]/2 \rfloor$, then $R(G_u(r,v_i)) \leq n-1$. Otherwise,

   a. If |Short_Path| = |Long_Path|, then $R(G_u(r,v_i)) \leq 2|T_{v_i}| + d(v_i,r) - 2$.

   b. Otherwise, $R(G_u(r,v_i)) \leq 2|T_{v_i}| + d(v_i,r) - 1$.

Proof:

1. When Short_Path is of even length and Long_Path is of odd length, then by Lemma 3.5, we know that r can receive a message from S_1 at even time periods and from S_2 at odd time periods. If $|X| = \lfloor (n-1)/2 \rfloor - (|B_i - T_{v_i}| + |T_{v_0} - \{r\}|)\} \leq |A_i - T_{v_i}| - \lfloor [d(v_i,r)-1]/2 \rfloor$, then we can balance the sizes of S_1 and S_2 so that $|S_2| = |B_i - T_{v_i}| + |T_{v_0} - \{r\}| + |X| = \lfloor (n-1)/2 \rfloor$ and $|S_1| = \lfloor (n-1)/2 \rfloor$. Thus, $R(G_u(r,v_i),r) \leq n-1$.

Otherwise, the node v_i starts sending out |X| messages by way of

Long_Path to r at time 1 every second period. By time 2|X|-1, the last message of X is relayed to Long_Path by v_i. At time 2|X|, v_i receives a message from T_{v_i} and relays it by way of Short_Path to r at time 2|X|+1. From then on, v_i relays the remaining messages in T_{v_i} to Short_Path every second period with no delay, and the last message received by r is the last message from T_{v_i}-X by way of Short_Path. Thus, at time $2|X|+1 + 2(|T_{v_i}-X|-1) = 2|T_{v_i}|-1$, the last message from T_{v_i} is sent out to Short_Path by v_i. So by time $2|T_{v_i}|-1 + d(v_i,r)-1 = 2|T_{v_i}| + d(v_i,r)-2$, the last message is received by r.

2. When Short_Path is of odd length and Long_Path is of even length, then by Lemma 3.6, we know that r receives a message from S_1 at odd time periods and from S_2 at even time periods. Similarly, If $|X| = \lfloor (n-1)/2 \rfloor - (|B_i-T_{v_i}| + |T_{v_0}-\{r\}|)\} \leq |A_i-T_{v_i}| - \lfloor [d(v_i,r)-1]/2 \rfloor$, then we can balance the sizes of S_1 and S_2 so that $|S_2| = |B_i-T_{v_i}| + |T_{v_0}-\{r\}| + |X| = \lfloor (n-1)/2 \rfloor$ and $|S_1| = \lceil (n-1)/2 \rceil$. Thus $R(G_u(r,v_i),r) \leq n-1$.

Otherwise, the node v_i starts sending out |X| messages by way of Long_Path to r at time 1 every second period. By time 2|X|-1, the last message of X is relayed to Long_Path by v_i. At time 2|X|, v_i receives a message from T_{v_i} and relays it by way of Short_Path to r at time 2|X|+1. From then on, v_i relays the remaining messages in T_{v_i} to Short_Path every second period with no delay, and the last message received by r is the last message from T_{v_i}-X by way of

Short_Path. Thus, at time $2|X|+1 + 2(|T_{v_i}-X|-1) = 2|T_{v_i}|-1$, the last message from T_{v_i} is sent out to Short_Path by v_i. So by time $2|T_{v_i}|-1 + d(v_i,r)-1 = 2|T_{v_i}| + d(v_i,r)-2$, the last message is received by r.

3. When Short_Path is of odd length and Long_Path is of odd length, then by Lemma 3.6, we know that r receives a message from S_1 at odd time periods and from S_2 at even time periods. Similarly, If $|X| = \lfloor(n-1)/2\rfloor - (|B_i-T_{v_i}| + |T_{v_0}-\{r\}|)\} \leq |A_i-T_{v_i}| - \lfloor[d(v_i,r)-1]/2\rfloor$, then we can balance the sizes of S_1 and S_2 so that $|S_2| = |B_i-T_{v_i}| + |T_{v_0}-\{r\}| + |X| = \lfloor(n-1)/2\rfloor$ and $|S_1| = \lceil(n-1)/2\rceil$. Thus $R(G_u(r,v_i),r) \leq n-1$.

Otherwise, the node v_i receives a message from $T_{v_i}-\{v_i\}$ at time 1 and then starts relaying $|X|$ messages to Long_Path at time 2. Note that v_i has now accumulated an extra message other than its own. By time $2|X|$, the last message of X is relayed to Long_Path by v_i and at time $2|X|+1$, v_i will relay the accumulated message to Short_Path. From then on, v_i relays the remaining messages in T_{v_i} to Short_Path every second period with no delay, and the last message received by r is the last message from $T_{v_i}-X$ by way of Short_Path. Thus, at time $2|X|+1 + 2(|T_{v_i}-X|-1) = 2|T_{v_i}|-1$, the last message from T_{v_i} is relayed to Short_Path by v_i. So by time $2|T_{v_i}|-1 + d(v_i,r)-1 = 2|T_{v_i}| + d(v_i,r)-2$, the last message is received by r.

4. When Short_Path is of even length and Long_Path is of even length,

and if $|X| = \lfloor (n-1)/2 \rfloor - (|B_i - T_{v_i}| + |T_{v_0} - \{r\}|) \leq |A_i - T_{v_i}| - \lceil [d(v_i,r)-1]/2 \rceil$, then we can balance the sizes of S_1 and S_2 so that $|S_2| = |B_i - T_{v_i}| + |T_{v_0} - \{r\}| + |X| = \lfloor (n-1)/2 \rfloor$ and $|S_1| = \lceil (n-1)/2 \rceil$. By Lemma 3.7, r receives a message from S_1 at odd time periods and from S_2 at even time periods. Thus, $R(G_u(r,v_i),r) \leq n-1$.

Otherwise, if |Short_Path| = |Long_Path|, then by time $2|X|+2$, v_i sends the remaining $|T_{v_i}|-|X|$ messages by way of Short_Path to r every second period with no delay. Thus, by time $2|X|+2 + 2(|T_{v_i}|-|X|-2) = 2|T_{v_i}|-2$, v_i sends the second last message in T_{v_i} to Short_Path. At time $2|T_{v_i}|-1$, v_i sends the last message, which was accumulated at v_i at time $2|X|+1$, by way of Long_Path to r. Since |Short_Path| = |Long_Path| = $d(v_i,r)$ in this case, by the time $2|T_{v_i}|-1 + d(v_i,r)-1 = 2|T_{v_i}| + d(v_i,r) -2$, the last message is received by r. Thus, $R(G_u(r,v_i),r) \leq 2|T_{v_i}| + d(v_i,r) -2$.

If |Short_Path| < |Long_Path|, then by time $2|X|+2$, v_i sends the remaining $|T_{v_i}|-|X|$ messages in T_{v_i} by way of Short_Path to r every second period with no delay. The last message received by r is the last message of the $T_{v_i}-X$ messages sent by way of Short_Path. Thus, at time $2|X|+2 + 2(|T_{v_i}|-|X|-1) = 2|T_{v_i}|$, the last message is relayed to Short_Path by v_i. So by time $2|T_{v_i}| + d(v_i,r) -1$, the last message is received by r. Thus, $R(G_u(r,v_i),r) \leq 2|T_{v_i}| + d(v_i,r) -1$. ■

Theorem 3.9: The receiving time for $G_u(r,v_i)$ using the above scheme is optimal.

Proof: The node r must receive all n-1 messages and thus, $R(G_u(r,v_i),r) \geq n-1$. Thus, for those cases where we have shown that $R(G_u(r,v_i),r) \leq n-1$, the scheme is optimal.

By Lemma 3.3, $R(G_u(r,v_i),r) \geq 2|T_{v_i}| + d(v_i,r) - 2$ which also implies the scheme is optimal for those cases in which we have shown $R(G_u(r,v_i),r) \leq 2|T_{v_i}| + d(v_i,r) - 2$. As for the case in Case 4 in which we have shown $R(G_u(r,v_i),r) \leq 2|T_{v_i}| + d(v_i,r) - 1$, we will show that this is also optimal. By Lemma 3.3 again, $R(G_u(r,v_i),r) \geq 2|T_{v_i}| + d(v_i,r) - 2$. To show that one extra time unit is required in this case when both Short_Path and Long_Path are of even length and |Short_Path| < |Long_Path|, consider the following cases:

1. If v_i sends all messages in T_{v_i} by way of Short_Path to r, then $R(G_u(r,v_i),r)$ is at least $2|T_{v_i} \cup A_i - T_{v_i}| - 1 > 2|T_{v_i}| + [d(v_i,r)-1]-1$ since $2|A_i - T_{v_i}| > d(v_i,r)-1$. Thus, $R(G_u(r,v_i),r) \geq 2|T_{v_i}| + d(v_i,r)-1$.

2. Similarly, if v_i sends all messages in T_{v_i} by way of Long_Path to r, $R(G_u(r,v_i),r) \geq 2|T_{v_i}| + d(v_i,r)-1$.

3. Now, if v_i sends part of the messages in T_{v_i} by way of Long_Path to r and the remaining by way of Short_Path to r, then let us consider the time parity at v_i and the time parities in Short_Path and

Long_Path. Note that if v_i relays some message in T_{v_i} to one path at each odd time period, these messages will reach r in even time periods from both Short_Path and Long_Path. This cannot happen simultaneously because r has to receive a message from Short_Path in odd time periods and a message from Long_Path in even time periods or vice versa. So one time unit of delay is unavoidable in this case and thus, $R(G_u(r,v_i),r) \geq 2|T_{v_i}| + d(v_i,r) - 1$.

Thus, in this case the time is also optimal. ■

3.4. Receiving in Unicyclic Graphs

Given the tree receiving scheme [5] and rooted unbalanced unicyclic receiving scheme from the previous section, we now consider a unicyclic graph $G_u = (V,E)$.

Theorem 3.10: Given a unicyclic graph G_u on n vertices and a specified receiver r,

1. If r is not on cycle, then $R(G_u,r) = R(T,r) = \text{Max}\ \{n-1, 2*\text{maxsubtree}(v)-1\}$ for any spanning tree T rooted at r, where maxsubtree(v) is the size of a maximum subtree rooted at a child v of r.

2. If r is on cycle and a centroid vertex of some spanning tree T, then $R(G_u,r) = R(T,r) = n-1$.

3. If r is on cycle and not a centroid vertex of any spanning tree but

adjacent to a vertex v with Weight(v) > $\lfloor n/2 \rfloor$, then $R(G_u,r)$ = 2*Weight(v) -1.

4. Otherwise, $R(G_u,r) = R(G_u(r,v_i),r)$, where $i \geq 2$.

Proof: These times are optimal and can be achieved because:

1. If r is a non-cycle vertex, then every neighbour of r is an articulation point. Consider the components obtained by deleting r. All of the messages from each particular component must arrive at r through a single neighbour of r. Thus, any receiving scheme performs no better than the tree receiving scheme [5] on some spanning tree T rooted at r from G_u, and $R(G_u,r) = R(T,r)$ is optimal.

2. If r is a cycle vertex and a centroid tree T rooted at r is constructed, then the tree receiving scheme [5] is applied. Thus, $R(G_u,r) = R(T,r) = n-1$ which is the best possible time. Hence, this is optimal.

3. If r is a cycle vertex and adjacent to a neighbour cycle vertex v of r with the subtree S_v rooted at v, consisting of v and those non-cycle vertices, such that $|S_v| \geq \lfloor n/2 \rfloor$, then a spanning tree T rooted at r is constructed by deleting the edge (u,v), where u is a cycle vertex and $u \neq r$. Note that if $|T_v| = \lfloor n/2 \rfloor$, T is a tree with centroid r and $R(G_u,r) = R(T,r) = n-1$ which is optimal. Otherwise with the tree receiving scheme [5], $R(G_u,r) = R(T,r) = 2*\text{Weight}(v) -1 = 2|S_v| -1$. By Lemma 3.3, $R(G_u(r),r) \geq 2|S_v| + d(v_i,r) -2 = 2|S_v| -1$ (since $d(v_i,r)=1$). Thus, the receiving time is optimal.

If r is a cycle vertex and adjacent to a neighbour non-cycle vertex v of r such that the subtree S_v rooted at v with $|S_v| > \lfloor n/2 \rfloor$, then an arbitrary spanning tree T rooted at r is constructed. With the tree receiving scheme [5], $R(T,r) = 2|S_v| - 1 = 2*\text{Weight}(v) - 1$. By Lemma 3.2, $R(G_u,r) \geq 2*\text{Weight}(v) + d(v,r) - 2 = 2*\text{Weight}(v) - 1$ (since $d(v,r)=1$). Thus, $R(G_u,r) = R(T,r)$ is optimal.

4. Otherwise, the rooted unbalanced unicyclic receiving scheme is used. From the results of the previous section, the resulting receiving time for a rooted unbalanced unicyclic graph is optimal. ■

3.4.1. Time Complexity Analysis

Weight(v) can be determined in $O(|E|)$ time for all points v in the component containing r. To obtain a spanning tree from G_u takes $O(|E|)$ time. The tree receiving algorithm [5] takes $O(|V|^2)$ time as does the rooted unbalanced unicyclic receiving scheme. Thus an optimal receiving scheme for any unicyclic graph can be determined in $O(|V|^2)$ time.

Chapter 4

Receiving in General Graphs

Optimal receiving schemes are known for 2-connected (non-separable) graphs and for trees [5]. An optimal receiving scheme for unicyclic graphs has been presented in the previous chapter. However, no optimal scheme is known for other separable graphs. In this section, we present a receiving scheme for separable graphs with receiving time no worse than 5/4 optimal.

4.1. Description of the algorithm for Separable Graphs

We are given a separable graph G. If G is a tree, we can use the optimal tree receiving algorithm [5]. Otherwise, we preprocess the graph G and obtain : (i) the set of articulation points A [1], (ii) the set of biconnected components B [1] and (iii) the size of each biconnected component. We can then determine Weight(v) of each articulation point v with respect to r.

Consider an articulation point $v \neq r$ in a block containing r. We transform the separable graph G into either a tree or rooted unicyclic graph as follows:

1. If r is not an articulation point, then:

   If the block containing r has only 2 vertices, then we can simply construct a spanning tree T rooted at r and use the tree receiving algorithm [5]. Otherwise:

a. If the block containing r has an articulation point v:

(i) If d(v,r)=1 and Weight(v) $\geq \lfloor (n-1)/2 \rfloor$, then construct a tree T by connecting all vertices outside the block at v as one subtree and the rest of vertices as another subtree at r and then join them together by the edge (r,v). Use the tree receiving algorithm [5] on the centroid tree T. Note that if Weight(v) = $\lfloor (n-1)/2 \rfloor$ or $\lceil (n-1)/2 \rceil$, T is a centroid tree.

(ii) If d(v,r) > 1 and Weight(v) $\geq \lfloor (n-1)/2 \rfloor$, then construct a rooted unbalanced unicyclic graph $G_u(r,v_i)$ from G, where $d(v_i,r) = d(v,r) > 1$, and use the rooted unbalanced unicyclic receiving scheme. To illustrate how to obtain such a rooted unbalanced unicyclic graph, consider Figure 4-1. Let Weight(v_1) = $|B_1|$, Weight(v_2) = $|B_2|$, and Weight(v_3) = $|B_4| + |B_6| + |B_5| + |B_7| - 3$.

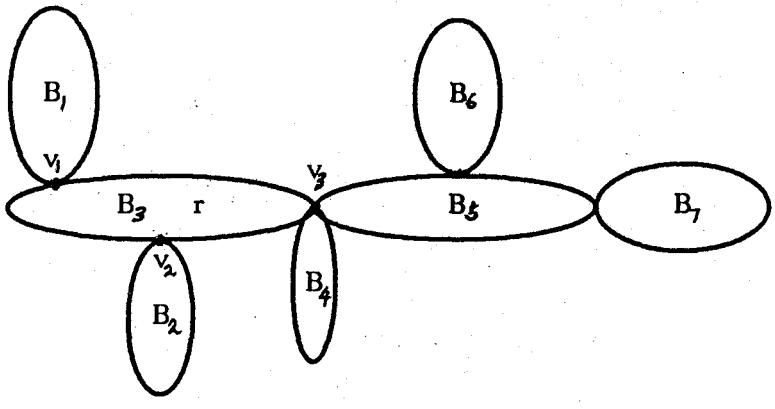


Figure 4-1: An example of separable graphs in which r is not an articulation point

Consider the block B_3. Find Short_Path from v_3 to r of length d(v,r) and then Long_Path from v_3 to r of length $\geq$ d(v,r). Join the remaining vertices in the block to the cycle except at v_3 - forming a unicyclic graph as in Figure 4-2.

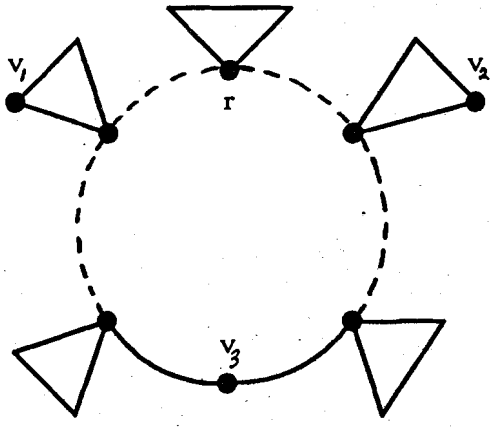


Figure 4-2: The resulting component of B_3 from Figure 4-1

The vertices of the other blocks can be connected to the cycle by choosing an arbitrary spanning tree of each of these blocks. The final rooted unbalanced unicyclic graph $G_u(r,v_i)$, where $d(v_i,r) = d(v_3,r) > 1$, is of the form as shown in Figure 4-3 :

b. If the block containing r has no such an articulation point v such that Weight(v) $\geq \lfloor (n-1)/2 \rfloor$, then we can direct each edge of the block so that the block becomes a directed acyclic graph (dag) with one source (s) and one sink (t), where s is adjacent to t in the

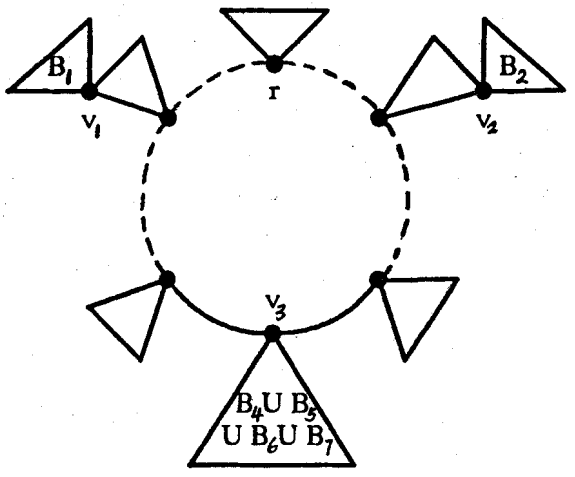


Figure 4-3: The resulting graph from Figure 4-1 after conversion

original block but the edge (s,t) has been removed in the dag [5]. Order the vertices $s=v_1, v_2, \ldots, v_k=t$ by topological sort. Split the dag into two trees $T_s = \{s=v_1, v_2, \ldots, v_m\}$ and $T_t = \{v_{m+1}, v_{m+2}, \ldots, v_k=t\}$, so that $\text{Weight}(T_s) = \sum_{i=1}^{m} \text{Weight}(v_i) \geq \lceil n/2 \rceil$ and $\text{Weight}(T_s) = \sum_{i=1}^{m-1} \text{Weight}(v_i) < \lceil n/2 \rceil$. The vertices of the other blocks can then be connected to T_s or T_t by choosing an arbitrary spanning tree of each of these blocks.

If $\text{Weight}(T_s) = \lceil n/2 \rceil$, then $\text{Weight}(T_t) = \lceil n/2 \rceil$. We construct a centroid tree by joining T_s and T_t with the s-t edge, and then use the optimal tree receiving scheme [5]. If $\text{Weight}(T_s) > \lceil n/2 \rceil$, we construct a rooted unicyclic graph by joining the two trees with the s-t edge in conjunction with another edge (v_m, v_j), where $m < j \leq k$, and use the receiving scheme for unicyclic graphs from Chapter 3.

If r is an articulation point, then:

a. Consider the graph resulting from the removal of r and its incident edges. If there is a component C in this graph such that $|C| > \lfloor n/2 \rfloor$, convert G into a spanning tree T if there is only a single edge from r to the component C. If there are multiple edges from r to C, convert G into G_f with the following operations: (1) obtain a subtree T_0' from G-C, (2) obtain a subgraph G' from C ∪ {r} ∪ {all incident edges from C into r} by using Step 1a and 1b, and (3) $G_f = G' \cup T_0'$. Note that G_f may end up with a tree or a rooted unbalanced unicyclic graph $G_u(r,v_i)$ for some $d(v_i,r) > 1$, with $T_{v_i} < \lfloor (n-1)/2 \rfloor$. To illustrate how it works, consider Figure 4-4.

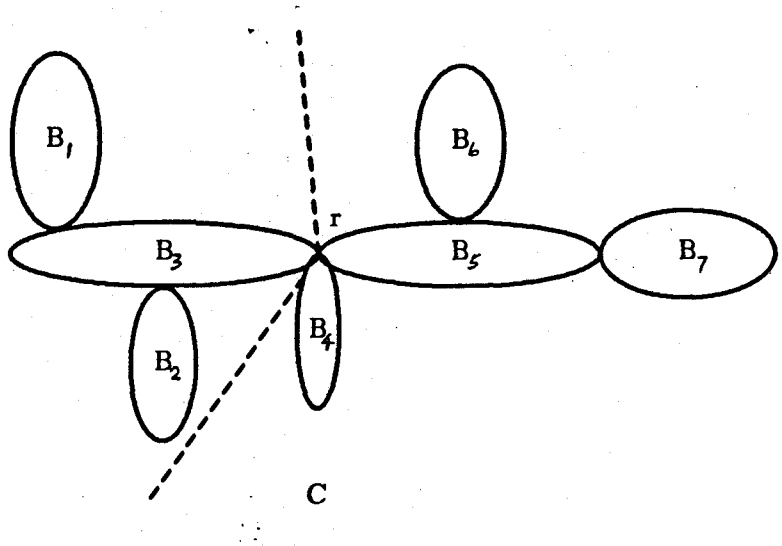


Figure 4-4: An example of separable graphs in which r is an articulation point

Suppose that $C = B_6 \cup B_5 \cup B_7 - \{r\}$ and $|C| > \lfloor n/2 \rfloor$, then we span $B_1 \cup B_2 \cup B_3 \cup B_4$ as the subtree T_0' rooted at r. If B_5 is a single edge, then G' is another spanning tree T_1' rooted at r from $B_5 \cup B_6 \cup B_7$ and $G_f = T_0' \cup T_1'$ is a tree. We then use the tree receiving scheme [5] on G_f. On the other hand, if B_5 is not a single edge, we work on $C \cup \{r\} \cup \{$all incident edges from C into r$\}$ as to obtain a subgraph G' by using step 1a and 1b. Then $G_f = T_0' \cup G'$. If G_f is a tree then use the tree receiving scheme [5]. Otherwise, use the rooted unbalanced unicyclic receiving scheme from Chapter 3 on G_f.

b. If there is no such component C, then we can simply construct an arbitrary centroid tree and use the optimal tree receiving scheme [5].

4.1.1. Proof of correctness of the algorithm

The algorithm converts a given separable graph G into either a tree or a rooted unicyclic graph. Thus, the proof of the correctness for the algorithm is mainly to show that all of the conversions are valid.

1. Initially, the algorithm checks to see whether G is a tree. If G is a tree, no conversion is involved.

2. If the separable graph G is not a tree, the algorithm converts G into either a tree or a rooted unicyclic graph in Step 1(a), 1(b), 2(a) and 2(b).

   a. For Step 1, consider $r \notin A$. If the block containing r has only 2

vertices, then r has only a single edge connecting it to V-{r}. In this case, the algorithm obtains a spanning tree T with root r from G. This conversion from G into T is valid.

If the block containing r has more than 2 vertices, then it is 2-connected, i.e. every two vertices in the block lie in a common cycle.

In Step 1(a), if there exists an articulation point v such that Weight(v) > $\lfloor (n-1)/2 \rfloor$, then the algorithm obtains a spanning tree from G if v is a neighbour of r, and converts G into a rooted unicyclic graph $G_u(r,v_i)$ with d(v,r) > 1 and $|T_{v_i}|$ = Weight(v) otherwise. A rooted unbalanced unicyclic graph is obtained from G by finding the first path of length d(v,r) and the second path of length $\geq$ d(v,r) from the vertex v to the receiver r. These two paths can always be found because the block containing both v and r is a 2-connected component. The next step is to join the remaining vertices in this block to the cycle except at vertex v.

In Step 1(b), if the block does not have such an articulation point v, then the algorithm converts the block into a directed acyclic graph (dag), with one source (s) and one sink (t), where t is adjacent to s in the original block but the edge (s,t) has been removed in the dag [5]. Order the vertices $s=v_1, v_2, \ldots, v_k=t$ in the dag by topological sort [1]. Split the dag into two trees T_s = {$s=v_1$,

$v_2, \ldots, v_m\}$ and $T_t = \{v_{m+1}, v_{m+2}, \ldots, v_k=t\}$ so that $\text{Weight}(T_s) = \sum_{i=1}^{m} \text{Weight}(v_i) \geq \lceil n/2 \rceil$ and $\text{Weight}(T_s) = \sum_{i=1}^{m-1} \text{Weight}(v_i) < \lceil n/2 \rceil$. If $\text{Weight}(T_s) = \lceil n/2 \rceil$ and $\text{Weight}(T_t) = \lceil n/2 \rceil$, then the algorithm constructs a centroid tree by adding the (s,t) edge from T_s to T_t. This is the procedure used to find a centroid tree from a 2-connected graph in [5]. However, if $\text{Weight}(T_s) > \lceil n/2 \rceil$, then the algorithm forms a rooted unicyclic graph by adding two edges from T_s to T_t, namely the (s,t) edge and the (v_m, v_j) edge, where $m < j \leq k$. Note that the (v_m, v_j) edge, where $m < j \leq k$, always exists in the dag. This can be proved as follows : The vertex v_m is of degree at least 2 and the dag has only 1 source (s) and 1 sink (t).

i. If all incident edges of v_m in the dag are incoming edges, then $v_m \neq t$ is a sink. This cannot happen.

ii. If all incident edges of v_m in the dag are outgoing edges, then $v_m \neq s$ is a source. This cannot happen.

Thus, the (v_m, v_j) edge, where $m < j \leq k$, always exists in the dag. As a result, all conversions in Step 1(b) are valid.

b. For Step 2, we are considering $r \in A$. In Steps 2(a) and 2(b), G is converted to either a spanning tree or a rooted unicyclic graph as in Steps 1a and 1b. Hence, all conversions in step 2 are valid as well.

4.2. Algorithm Performance Analysis

If G is a tree, then the tree receiving algorithm [5] achieves the optimal time. Therefore we will look at all possible cases of non-tree G and analyze them.

In Step 1 : we are considering the cases in which r $\notin$ A.

1. If the block containing r has only 2 vertices, then r is of degree 1 and has only a single edge connecting it to V-{r}. The algorithm converts the graph G into a spanning tree T rooted at r and use the tree receiving scheme [5] with R(T,r) = 2n-3. By Lemma 3.1, R(G,r) $\geq$ 2n-3. Thus the conversion yields the optimal receiving time.

2. If the block containing r has more than 2 vertices:

In Step 1(a), the block containing r has an articulation point v. (i) If d(v,r) = 1 and Weight(v) $\geq$ $\lfloor(n-1)/2\rfloor$, then the algorithm constructs a tree T from G. Note that if Weight(v) = $\lfloor(n-1)/2\rfloor$ or $\lceil(n-1)/2\rceil$, T is a centroid tree. With the tree receiving scheme [5] on T, R(G,r) = R(T,r) = n-1 which is optimal. If Weight(v) > $\lfloor(n-1)/2\rfloor$, T is a tree with subtree S rooted at a child v of r such that |S| = Weight(v). With the tree receiving scheme [5] on T, R(T,r) = 2|Maxsubtree(v)| -1 = 2*Weight(v) -1. By Lemma 3.2, R(G,r) $\geq$ 2*Weight(v) + d(v,r) -2 = 2*Weight(v) -1 (since d(v,r)=1). Thus, R(G,r) = R(T,r) = 2*Weight(v) -1 is optimal.

(ii) If d(v,r) > 1 and Weight(v) $\geq$ $\lfloor(n-1)/2\rfloor$, then the algorithm

constructs a rooted unbalanced unicyclic graph $G_u(r,v_i)$ with $d(v_i,r) = d(v,r) > 1$ and $|T_{v_i}| = \text{Weight}(v)$. Observe that $|T_{v_i}| \geq 2$, $d(v_i,r) \geq 2$ in $G_u(r,v_i)$. By Lemma 3.2, we know that $R(G,r) \geq 2*\text{Weight}(v) + d(v,r) - 2 = 2|T_{v_i}| + d(v_i,r) - 2$. So $R(G,r)$ requires at least $2|T_{v_i}| + d(v_i,r) - 2$ time units. Since $R(G_u(r,v_i),r)$ from Section 3.3 is no more than $2|T_{v_i}| + d(v_i,r) - 1$, the resulting time from the scheme obtained by the algorithm is at most greater than the optimal time by 1. In ratio, the performance is :

$$[2|T_{v_i}| + d(v_i,r) - 1]/[2|T_{v_i}| + d(v_i,r) - 2] \text{ optimal}$$

$$\leq 5/4 \text{ optimal, for } |T_{v_i}| \geq 2,\ d(v_i,r) \geq 2.$$

In Step 1(b), when $|T_s| = \lfloor n/2 \rfloor$ and $|T_t| = \lfloor n/2 \rfloor$, then we construct a centroid tree T and use the tree receiving scheme [5] with $R(T,r) = n-1$. Thus, we have an optimal conversion. Otherwise, observe that there is no articulation point v with $\text{Weight}(v) \geq \lfloor (n-1)/2 \rfloor$, and therefore each articulation point v must be of $\text{Weight}(v) < \lfloor (n-1)/2 \rfloor$. In this case, the algorithm constructs a rooted unbalanced unicyclic graph $G_u(r,v_i)$ with $T_{v_i} < \lfloor (n-1)/2 \rfloor$ for some v_i.

$R(G_u(r,v_i),r)$ from Section 3.3 is no more than $2|T_{v_i}| + d(v_i,r) - 1 < 2*\lfloor (n-1)/2 \rfloor + d(v_i,r) - 1$.

Since $|T_{v_i}| = \lfloor (n-1)/2 \rfloor$, the remaining $\lfloor (n-1)/2 \rfloor + 1$ vertices may all contribute to constructing both paths. Note that $\lfloor (n-1)/2 \rfloor \leq \lfloor (n+1)/2 \rfloor$. So $\lfloor (n-1)/2 \rfloor + 1 \leq \lfloor (n+1)/2 \rfloor + 1 = \lfloor (n+3)/2 \rfloor$. Therefore, $d(v_i,r)$ in this particular $G_u(r,v_i)$ can be no more than $\lfloor (n+3)/2 \rfloor / 2 = \lfloor (n+3)/4 \rfloor$, and

$$2*\lfloor (n-1)/2 \rfloor + d(v_i,r) - 1$$

$$\leq 2*\lfloor (n-1)/2 \rfloor + \lfloor (n+3)/4 \rfloor - 1$$

$$\leq n-2 + (n+3)/4$$

Since $\lfloor (n+3)/4 \rfloor \geq 1$, $n \geq 1$. As a result, $R(G_u(r,v_i),r)$ is less than $n-2 + (n+3)/4$ and $R(G,r)$ requires at least $n-1$ time units. Hence, the resulting time is:

$$< [n-2 + (n+3)/4]/(n-1) \text{ optimal, for } n \geq 1$$

$$= (5n-5)/(4n-4) \text{ optimal}$$

$$= 5/4 \text{ optimal, for } n \geq 1.$$

In Step 2, we are considering the remaining cases in which $r \in A$. For Step 2(a), if there is a component C resulting from the removal of r and its incident edges in the graph, then:

1. If there is only a single edge (v,r) joining r to component C, then the

algorithm converts G to a spanning tree T. The tree receiving scheme [5] yields $R(T,r) = 2|C|-1$ because the maximum subtree S rooted at a child v of r is formed from C and $|S| = |C| > \lfloor n/2 \rfloor$. By Lemma 3.2, $R(G,r) \geq 2*\text{Weight}(v) + d(v,r) - 2 = 2|S| - 1$ (since $d(v,r)=1$). Thus $R(G,r) = R(T,r) = 2|S| - 1 = 2|C| - 1$ is optimal.

2. If there are at least 2 edges joining r to the component C, then the algorithm constructs G_f with the following operations: (1) construct spanning tree T_0' from G-C such that $|T_0'| < \lfloor n/2 \rfloor$. (2) construct G' from C ∪ {r} ∪ {all incident edges from r into C} by using Step 1a and 1b. Note that if G' is a tree, G' must be either a centroid tree or a tree T with $R(T,r) = R(G,r)$ as analyzed in Step 1. If G' is a rooted unbalanced unicyclic graph $G_u(r,v_i)$, then $d(v_i,r) > 1$ and $|T_{v_i}| < \lfloor (n-1)/2 \rfloor$ as in Step 1. (3) $G_f = T_0' \cup G'$. The analysis is then the same as for Step 1. That is, if G' is a tree, then G_f is also a tree and $R(G,r) = R(G_f,r)$ is optimal. On the other hand, if G' is a rooted unbalanced unicyclic graph $G_u(r,v_i)$ with $d(v_i,r) > 1$ and $|T_{v_i}| < \lfloor (n-1)/2 \rfloor$, then G_f is also a rooted unbalanced unicyclic graph $G_u(r,v_i)$ for the same v_i, with $d(v_i,r) > 1$ and $|T_{v_i}| < \lfloor (n-1)/2 \rfloor$. By the same analysis as Step 1(b), the performance of the resulting receiving time from the scheme obtained by the algorithm is no worse than 5/4 optimal.

For Step 2(b): Since there does not exist a component C when r is removed, along with all incident edges, such that $|C| > \lfloor n/2 \rfloor$, this implies that no matter how

we obtain a spanning tree T rooted at r, there is no chance for the spanning tree T to have a subtree S rooted at a child of r such that $|S| > \lfloor n/2 \rfloor$. In this case, the algorithm always constructs a centroid tree T and uses the tree receiving scheme [5] with R(G,r) = R(T,r) = n-1 which is the minimum possible time. Thus the conversion is optimal.

As a result of all the conversions in the algorithm, the resulting receiving time is no worse than 5/4 optimal.

4.2.1. Time Complexity Analysis

Preprocessing requires $O(|V|+|E|)$ time to determine the set of articulation points and the set of biconnected components [1], and $O(|V|)$ time to determine the size of each component.

Obtaining a spanning tree T with root r from G takes $O(|E|)$ time and the tree receiving algorithm takes $O(|V|^2)$ time [5].

Besides these, other conversions that need to be considered are :

1. (i) Finding a shortest path from an articulation point v to r, where Weight(v) $\geq \lfloor (n-1)/2 \rfloor$ takes $O(|V|^2)$ time with Dijkstra's Algorithm [3].
   (ii) Similarly, finding the second path from v to r also takes $O(|V|^2)$ time.
   (iii) For those vertices attached to v such that Weight(v) $\geq \lfloor (n-1)/2 \rfloor$, forming a subtree with these vertices attached to v takes $O(|V|)$ time.
   (iv) For the remaining vertices in the block, finding subtrees attached to the cycle except at v takes $O(|V|)$ time. Thus, this conversion requires $O(|V|^2)$ time to obtain a rooted unbalanced unicyclic graph.

2. (i) Directing each edge of the block so that the block becomes a "directed acyclic graph" (dag) with one source (s) and one sink (t) takes $O(|V|+|E|)$ time [5]. (ii) Ordering the dag with topological sort takes $O(|V|+|E|)$ time [1]. (iii) Splitting the dag into two subtrees T_s and T_t takes $O(|V|)$ time. (iv) Joining the two subtrees with (s,t) edge takes $O(1)$ time. Therefore, if $T_s = \lfloor n/2 \rfloor$ and $T_t = \lfloor n/2 \rfloor$, then these steps construct a centroid tree in $O(|V|+|E|)$ time. The tree receiving algorithm [5] requires $O(|V|^2)$ time. If $T_s > \lfloor n/2 \rfloor$ and $T_t < \lfloor n/2 \rfloor$, it constructs a rooted unbalanced unicyclic graph as follows: in addition to step (i) to (iv), join another edge from the vertex with the highest topological order in T_s to any vertex in T_t. This step takes $O(1)$ time. The rooted unbalanced unicyclic scheme requires $O(|V|^2)$ time.

Thus, the algorithm requires $O(|V|^2)$ time to develop a receiving scheme for any separable graph G.

Chapter 5

Summary

An optimal polling scheme for trees was previously known [4]. Optimal polling schemes for Hamiltonian graphs and complete k-partite graphs have been designed and the polling time of an arbitrary polling station in these graphs determined. For 2-connected graphs, a known algorithm [5] is used to obtain a centroid tree. The tree polling scheme can be applied to the resulting tree. The resulting polling time is no more than $n+\lfloor n/2 \rfloor-1$ if n is odd and 3n/2 otherwise. It is not true that the polling time of every 2-connected graph is n+1. $K_{2,m}$ ($m > 3$) is a 2-connected graph that requires more than n+1 time units for polling. It is an open question whether the polling time of every 3-connected graph is n+1. To determine the polling time of an arbitrary graph remains open.

Optimal receiving schemes for trees and 2-connected graphs were previously known [5]. An optimal receiving scheme has been designed for unicyclic graphs and the receiving time of an arbitrary receiver determined. For general graphs, we have presented an $O(|V|^2)$ algorithm to develop a receiving scheme. The resulting receiving time has been shown to be no worse than 5/4 optimal. Finding an optimal receiving scheme for general graphs remains open.

References

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

[2] Kang, A., and Ault, D.
Some properties of a centroid of a free tree.
Information Processing Letters 4:18-20, 1975.

[3] Even, S.
Graph Algorithms.
Computer Science Press, Potomac, Maryland, 1979.

[4] Cheston, G.A., and Hedetniemi, S.T.
Polling in Tree Networks.
In *Proceedings of the Second West Coast Conference on Computing in Graph Theory*, pages 7-20. Eugene, Oregon, 1983.

[5] Cheston, G.A., and Hedetniemi, S.T.
Message Receiving in Communication Networks.
unpublished manuscript.

[6] Hedetniemi, S.T., and Hedetniemi, S.M.
A Survey of Gossiping and Broadcasting in Communication Networks.
Technical Report CIS-TR-81-5, Department of Computer and Information Science, University of Oregon, Eugene, Oregon, 1981.