1999

Broadcasting in grid graphs

Iwona Wojciechowska
West Virginia University

Broadcasting in Grid Graphs

Iwona Wojciechowska

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

Doctor of Philosophy
in
Computer Science

Frances L. Van Scoy, Ph.D., Chair
John Atkins, Ph.D.
Michael Henry, Ph.D.
William F. Klostermeyer, Ph.D.
Department of Computer Science and Electrical Engineering
Cun-Quan Zhang, Ph.D.
Department of Mathematics

Morgantown, West Virginia
1999

ABSTRACT

Broadcasting in Grid Graphs

Iwona Wojciechowska

This work consists of two separate parts. The first part deals with the problem of multiple message broadcasting, and the topic of the second part is line broadcasting. Broadcasting is a process in which one vertex in a graph knows one or more messages. The goal is to inform all remaining vertices as fast as possible. In this work we consider a special kind of graphs, grids.

In 1980 A. M. Farley showed that the minimum time required to broadcast a set of M messages in any connected graph with diameter d is $d + 2(M - 1)$. This work presents an approach to broadcasting multiple messages from the corner vertex of a 2-dimensional grid. This approach gives us a broadcasting scheme that differs only by 2 (and in the case of an $even \times even$ grid by only 1) from the above lower bound.

Line broadcasting describes a different variant of the broadcasting process. A. M. Farley showed that line broadcasting can always be completed in $\lceil \log n \rceil$ time units in any connected graph on n vertices. He defined three different cost measures for line broadcasting. This work presents strategies for minimizing those costs for various grid sizes.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The need to obtain faster algorithms leads us to use parallel architectures. Several general purpose processors are connected to build such a system. Processors communicate by exchanging messages. Communication schemes are very dependent on the problem to be solved. Fortunately, the study of classical algorithms shows some "generic communications" that appear very often. Some of them are:

- *broadcasting:* one processor has to send a message to all the others;
- *gossiping:* a message must be sent from each processor to all the others;
- *scattering:* one processor has to send a different message to each of the other processors;
- *gathering:* a processor receives a message from each of the other processors (exact inverse of scattering);
- *multiple message broadcasting:* one processor has to send a set of messages to all the others.

In order to better utilize parallel computers, we need fast and effective algorithms for frequently occurring communication problems for different parallel architectures.

Bavelas [1] was the first person to study the effectiveness of different communication patterns in helping small groups of people solve common tasks. In studied tasks the subjects could communicate with one another according to a given communication pattern by writing messages. Bavelas considered such measures as the number of messages and the time required to complete the task.

One of the typical tasks studied was the following: each of five people has five playing cards; the cards may not be passed around, but players can communicate with one another according to a given communication pattern by writing messages; the task is completed when each player selects one of their five cards so that the cards selected represent the highest ranking poker hand that can be made by selecting one card from each person.

He showed that for any communication pattern of a certain type among p people, $2(p-1)$ messages are required to solve a given task. He also showed that, if any communication pattern is allowed and each message takes unit time, then the time required to complete the task is no more than $\lceil \log_2 p \rceil$.

In 1972, Hajnal, Milner and Szemeredi [12] attributed to Boyd the following problem: "There are n ladies, and each one of them knows an item of scandal which is not known to any of the others. They communicate by telephone, and during each call two ladies pass on to each other as much scandal as they know at the time. How many calls are needed before all ladies know all the scandal?" This problem is known as the Gossip Problem, or the Telephone Problem. It has been the source of dozens of research papers that have studied problems concerning the spread of information among a set of people using telephone calls, conference calls, letters or computer networks.

The first papers on gossiping dealt with counting the calls. Let $f(n)$ be the minimum number of calls necessary to complete gossiping among n people where any pair of people may call each other. Hajnal, Milner and Szemeredi [12] showed that

$$f(1) = 0; \qquad f(2) = 1; \qquad f(3) = 3;$$

$$f(n) = 2n - 4, \qquad \text{for } n \geq 4.$$

Another variant of the Gossip Problem is obtained by considering the time it takes to complete the given task. We assume that there are n people, each of them knowing one unique piece of information. Information is exchanged by making calls. Each call is assumed to take one unit of time. Each call involves exactly two people. During a call people involved exchange all information known to them. At any given time a person can participate in at most one call. A non-conflicting call can take place in the same time unit.

Let $t(n)$ be the minimum amount of time required to complete gossiping among n people when all possible calls are allowed. Bavelas [1], Landau [19] and Knodel [17] showed that

$$t(n) = \begin{cases} \lceil \log_2 n \rceil & \text{for } n \text{ even,} \\ \lceil \log_2 n \rceil + 1 & \text{for } n \text{ odd.} \end{cases}$$

Broadcasting, as a major variant of the Gossip Problem, was introduced in 1977 by Slater, Cockayne and Hedetniemi [24, 25], when they studied the minimum time required for one person to transmit one piece of information to everyone in a communication network.

A survey of the early results in gossiping and broadcasting was presented by Hedetniemi, Hedetniemi and Liestman [13]. Recent results were published by Fraigniaud and Lazard [9].

Chapter 2

Problem description and previous results

Broadcasting refers to a process of information dissemination in a communication network whereby one member of the network has one or more messages that are to be transmitted to all members of the network. We model a communication network by a *connected graph* $G = (V, E)$, where V is the set of *vertices* (members) and E is the set of *edges* (communication lines). A message is transmitted by placing a series of *calls* between members. Calls can be placed according to the following rules:

(i) each call involves only two vertices, one sender and one receiver;

(ii) each call requires one unit of time;

(iii) at a given time, a vertex may participate as sender or receiver in at most one call;

(iv) calls can be placed between adjacent vertices only.

This process is to be completed as quickly as possible. Because the calls can be placed between adjacent vertices only this type of broadcasting is called *local broadcasting*.

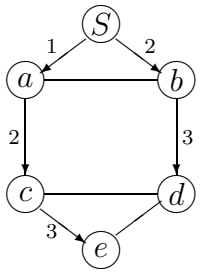


Figure 2.1: Local broadcasting

Figure 2.1 shows an example of local broadcasting. The graph is undirected, and the edges that are used in the broadcasting process are represented as arrows. Vertex S is the source. It has one message that has to be sent to all vertices in the graph. An edge label indicates the time a call is placed along it. At time 1 vertex S sends its message to vertex a. At time 2 vertices S and a send the same message to vertices b and c respectively. At time 3 vertex b calls vertex d and vertex c calls vertex e, thus completing the broadcast process.

Given a connected graph G and a vertex v, we can ask "what is the minimum number of time units required to complete broadcasting from vertex v?"

We define the *broadcast time of vertex* v, $b(v)$ to equal the minimum number of time units required to broadcast one message from vertex v to all vertices in the graph G.

2.1 Previous results

It is easy to see that for any vertex v in a connected graph $G = (V, E)$, $|V| = p$, $b(v) \geq \lceil \log_2 p \rceil$, since during each time unit the number of informed vertices can at most double.

We define the *broadcast time of a graph* G, $b(G)$ to equal the maximum broadcast time of any vertex v in G, i.e.

$$b(G) = \max_{v \in V} b(v).$$

For the complete graph K_p (with $p \geq 2$ vertices), since during a given time unit every informed vertex can call some uninformed vertex, $b(K_p) = \lceil \log_2 p \rceil$.

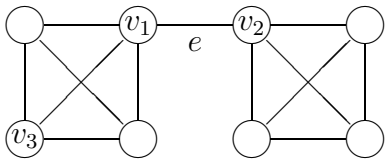


Figure 2.2: Two K_4 graphs connected by a single edge

The graph G in Figure 2.2 consists of two complete graphs K_4 connected by a single edge e. The time required to broadcast a single message from any vertex in K_4 is $\log_2 4 = 2$. What is the time necessary to broadcast one message from vertex v_1 in graph G? In the first time unit the message is sent from v_1 to vertex v_2. Then vertices v_1 and v_2 serve as the sources for broadcasting in the complete graphs K_4. The same holds for vertex v_2. Thus

$$b(v_1) = b(v_2) = 3.$$

For any other vertex other than v_1 or v_2, one additional time unit is required to complete broadcasting. For example, when vertex v_3 is the source, it takes one time unit to transmit the message to vertex v_1, a second time unit to send the message to vertex v_2, and two additional time units to complete broadcasting in K_4 from vertex v_2. It gives us that $b(v_3) = 4$. Thus

$$b(G) = \max_{v \in G} b(v) = b(v_3) = 4.$$

D. S. Johnson has shown that the problem of determining whether $b(v) \geq k$, for an arbitrary vertex v in an arbitrary graph G for fixed $k \geq 4$, is NP-complete. This result was presented by Slater, Cockayne and Hedetniemi [25]. They showed that the more general problem:

Given a graph $G = (V, E)$ with a specified set of vertices $V_0 \subseteq V$ and a positive integer k, is there a sequence

$$V_0, E_1, V_1, E_2, V_2, \ldots, E_k, V_k,$$

where $V_i \subseteq V$, $E_i \subseteq E$, E_i consists only of edges with exactly one vertex in V_{i-1}, no two edges in E_i share a common endpoint,

$$V_i = V_{i-1} \cup \{v : (u, v) \in E_i\},$$

and $V_k = V$? (V_i is the set of vertices that are informed at time i with calls along the edges in E_i.)

is NP-complete. They reduced the broadcast problem for $k = 4$ to the three-dimensional matching (3DM) which is known to be NP-complete (see Garey and Johnson [10]).

The general problem remains NP-complete for any $k \geq 4$, but is solvable in polynomial time by matching for $k = 1$.

The special case where $|V_0| = 1$ (determining whether $b(v) = k$) remains NP-complete, but is solvable in polynomial time for trees (see [25]).

Scheuermann and Wu [22] presented a dynamic programming formulation for determining $b(v)$ and a corresponding broadcasting scheme for a vertex v in an arbitrary graph. Scheuermann and Edelberg [21] implemented a backtracking algorithm based on this formulation. Since this exact algorithm is not efficient for large graphs, Scheuermann and Wu also presented several heuristics for achieving efficient near-optimal schemes.

2.2 Multiple message broadcasting

Early work on broadcasting was based on the assumption that each call requires one unit of time. This assumption is not very realistic if one is broadcasting large files of information over the lines of a computer network. It was realized that if more than

one message is to be broadcast, then simple modifications of one-message broadcast schemes would be too inefficient. Neither repeating a one-message broadcast scheme M times, nor having each call take M time units produces a very efficient M-message broadcast scheme. It is possible to construct "mixed" calling schemes which are more efficient for M-message broadcasting than either of these two schemes.

Chinn, Hedetniemi and Mitchell [3] defined the following functions:

$\mathcal{P}(M,t)$ — the maximum number of processors who can be informed of M messages in t time units; (the message originator is included in that number)

$\mathcal{M}(p,t)$ — the maximum number of messages which can be broadcast to p processors in t time units;

$\mathcal{T}(M,p)$ — the minimum number of time units necessary to broadcast M messages to p processors.

They were the first to study multiple-message broadcasting in complete graphs. They determined the values of $\mathcal{P}(M,t)$ for $1 \leq M \leq 8$ and $1 \leq t \leq 9$. They also obtained some general results for small values of M.

Farley [5] determined the value for $\mathcal{T}(M,p)$ for odd values of p.

$$\mathcal{T}(M,p) = 2M - 1 + \lfloor \log_2 p \rfloor$$

The problem of multiple-message broadcasting for p even was settled by Cockayne and Thomason [4], as follows:

$$\mathcal{T}(M,p) = 2M + \lfloor \log_2 p \rfloor - \lfloor \frac{M - 1 + 2^{log_2 p}}{p/2} \rfloor.$$

For p odd

$$\mathcal{M}(p,t) = \lfloor \frac{t - q + 1}{2} \rfloor,$$

where $q = \lfloor \log_2 p \rfloor$. For p even

$$\mathcal{M}(p,t) = \lfloor \frac{p^{t-q} + 2^{q+1} - 2}{2(p-1)} \rfloor.$$

For p odd, M messages can be broadcast to p processors in time t if and only if $p \leq 2^{t-2(M-1)} - 1$.

For p even, M messages can be broadcast to p processors in time t if and only if $p \leq \mathcal{P}(M,t)$, where $\mathcal{P}(M,t)$ is the largest even integer u such that

$$t \geq 2M + q - \lfloor \frac{M - 1 + 2^q}{u/2} \rfloor,$$

where $q = \lfloor \log_2 u \rfloor$.

Farley [5] was the first to study multiple message broadcasting in general connected graphs. He defined $b_M(v)$, $v \in V$, to equal the minimum number of time units required to broadcast a set of M messages from vertex v throughout a connected graph G. He also defined $B_M(G)$, the *broadcast time of graph* G as follows

$$B_M(G) = \max_{v \in V} b_M(v).$$

Farley showed that for any connected graph $G = (V, E)$, $|V| = p$, with diameter D and the maximum degree of any vertex d_{max} the following holds

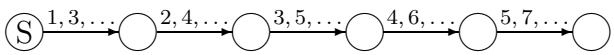
$$2(M-1) + D \leq B_M(G) \leq d_{max}(M-1) + (p-1). \tag{2.1}$$

The lower bound in the above inequality is realized in cycle graphs, and the upper bound is realized in path and star graphs.

For path P_p with p vertices ($D = p - 1$, $d_{max} = 2$) from 2.1 we have

$$2(M-1) + p - 1 \leq B_M(P_p) \leq 2(M-1) + p - 1.$$

Thus lower and upper bounds on the broadcast time for broadcasting M messages in any path graph are equal.



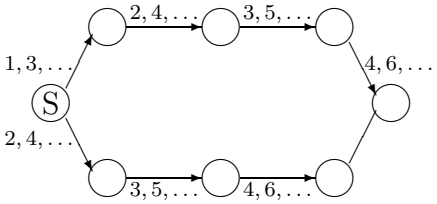
The above picture shows an optimal pattern for broadcasting multiple messages from the end vertex of path graph P_6. The first message leaves the source vertex at time one, and consecutive messages follow every two time units.

For cycle graph C_p with p vertices, $D = \lfloor p/2 \rfloor$, $d_{max} = 2$. From 2.1 we have

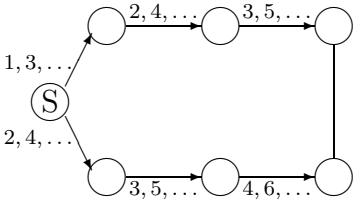
$$2(M-1) + \lfloor p/2 \rfloor \leq B_M(C_p). \tag{2.2}$$

which for p even gives us

$$2(M-1) + p/2 \leq B_M(C_p).$$

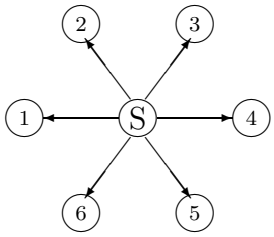


For C_8 the optimal broadcasting pattern is presented above. However, for p odd, the lower bound in 2.2 can not achieved. Since there are two vertices that are distance D from the source, one additional time unit is required to complete broadcasting.



The above picture shows a broadcasting scheme for the cycle graph C_7 ($D = 3$, $d_{max} = 2$). The time required to broadcast M messages is $2(M-1)+4$.

For star graph S_p with p vertices ($D = p-1$, $d_{max} = p-1$), only one call can take place during any time unit (each call has to involve the center vertex of the star). Since there are M messages that have to be sent to $p-1$ vertices each, $M(p-1)$ time units are required.



The figure above presents a broadcasting scheme for the star graph S_7, with the center vertex as the source. Node labels (other than the source) represent the time the first message was received. Consecutive messages arrive every $d_{max} = p - 1 = 6$ time units.

If $G_{m,n}$ is a two-dimensional grid with mn vertices, then Farley's bounds on $B_M(G)$ are:

$$2(M-1) + m + n - 2 \leq B_M(G_{m,n}) \leq 4(M-1) + (mn-1).$$

In particular,

$$m + n + 2M - 4 \leq B_M(G_{M,n}).$$

Chapter 3

Multiple message broadcasting in grid graphs

A d-dimensional grid graph $G_{n_1,n_2,\ldots,n_d}$ is a graph with $n_1 n_2 \ldots n_d$ vertices, each labeled with a unique element of $\{(i_1, i_2, \ldots, i_d) \,|\, 1 \leq i_j \leq n_j, 1 \leq j \leq d\}$, and with edges connecting vertices which differ by 1 in exactly one label component.

3.1 Previous results

3.1.1 Broadcasting one message

Farley and Hedetniemi [7] were the first to study broadcasting in grid graphs. They demonstrated that the broadcast times of vertices of a grid $G_{m,n}$ are as follows:

source vertex location	broadcast time
corner	$n+m-2$
vertex $(i,1)$ or (i,n),	$n+m-i+1$ if $i < \frac{n+1}{2}$
$i \neq 1, i \neq n$	$m+i-1$ if $i = \frac{n+1}{2}$
	$m+i-2$ if $i > \frac{n+1}{2}$
vertex $(1,j)$ or (n,j),	$n+m-j-1$ if $j < \frac{m+1}{2}$
$j \neq 1, j \neq m$	$n+j-1$ if $j = \frac{m+1}{2}$
	$n+j-2$ if $j > \frac{m+1}{2}$
vertex (i,j),	maximum distance from vertex (i,j)
$i \neq 1, i \neq n, j \neq 1, j \neq m$	to corner vertex
	$+1$ if $j = m-j+1$
	$+2$ if $j = m-j+1$ and $i = n-i+1$

They also considered broadcasting in an infinite 2-dimensional grid. Let $f(d,t)$ be the maximum number of vertices which can be informed of a single message originated by a given "cell" in the infinite d-dimensional grid, after t time units, by any local broadcasting scheme. They presented a scheme which yielded the following upper bound on $f(d,t)$ and conjectured equality.

$$f(2,t) \geq 2t^2 - 6t + 8, \qquad \text{for } t \geq 2.$$

The 2-dimensional conjecture of Farley and Hedetniemi was independently verified by Ko [18] and Peck [20].

3.1.2 Broadcasting a few messages

Van Scoy [26] found an algorithm for broadcasting a small number of messages from a corner vertex of a square $n \times n$ grid in the time predicted by Farley's lower bound. Her algorithm divides the messages into two sets of (approximately) equal size: black (odd numbered) messages and gray (even numbered) messages. In Figure 3.1 the edges along which black messages move are represented by solid arrows and the edges along which gray messages move by dashed arrows. The black messages are sent

along the same edges as in the Farley-Hedetniemi algorithm and the gray messages along edges that are the transpose of the edges in their algorithm. (That is, if Farley and Hedetniemi send a message over an edge from (i,j) to (k,l), Van Scoy sends the black messages from (i, j) to (k, l) and the gray messages from (j, i) to (l, k).) Black messages leave the source corner, moving along row 1, at time 1 and every time $4t+1$ afterwards, and gray edges leave the source corner, moving down column 1, at time 3 and at every time $4t + 3$ afterwards. Once the first message has arrived at a point which is not in rows 1 or n or columns 1 or n, that point is continually busy until it has received and sent all M messages. Her algorithm requires time $2n + 2M - 4$ and succeeds for $M \leq n - 2$ if n is odd and $M \leq n - 1$ if n is even. The pattern of edges used is shown in Figure 3.1.

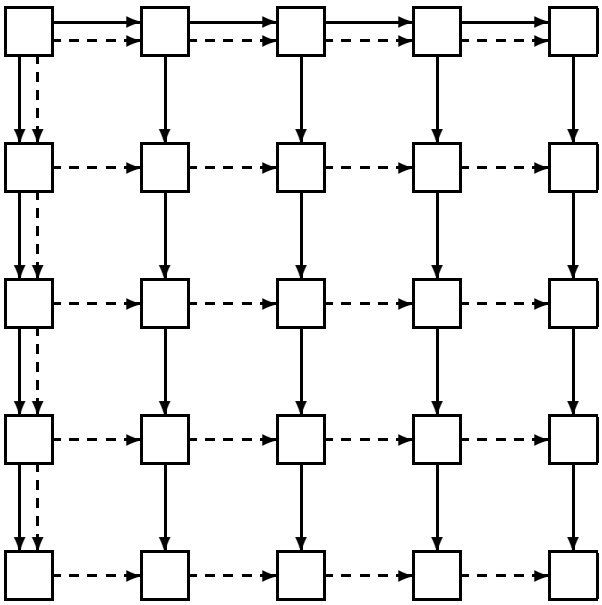


Figure 3.1: Broadcasting a few messages (Van Scoy, 1979)

Her algorithm fails for larger numbers of messages because it moves the gray messages along row 1 during the one time out of every four that a non corner point of row 1 is not busy passing a black message as well as the final few times when all black

messages have moved through row 1 but have not reached all points (and similarly for black messages along column 1). There are simply not enough times at which a point in row 1 is not busy transmitting a black message to support transmitting $\frac{M}{2}$ or more gray messages as well.

3.1.3 Broadcasting many messages in non optimal time

Brooks [2] found an algorithm which broadcasts arbitrarily many messages but which requires time $2n + \frac{5}{2}M - 1$. His algorithm differs from Van Scoy's in that gray messages are sent to most points in row 1 by their neighbors in row 2, and similarly black messages are sent to most points in column 1 by their neighbors in column 2. Because most points in row (column) 2 must receive each of the M messages and must send gray (black) messages to two neighbors and black (gray) messages to one neighbor, the times at which a new black (gray) message can leave the source corner are separated by 5 rather than by 4 as in Van Scoy's algorithm. The pattern of edges used is shown in Figure 3.2.

Van Scoy and Brooks [27] published another time $2n+\frac{5}{2}M-constant$ broadcasting algorithm. This algorithm required time $2n + \frac{5}{2}M - 4$. It has an advantage over Brooks's algorithm in that while his $\frac{5}{2}M$ algorithm requires some points in row 2 or column 2 to store $\frac{M}{2}$ messages, their algorithm requires storage of at most two messages at a point. The pattern of edges used is shown in Figure 3.3.

3.1.4 Broadcasting many messages in higher dimension grids

Van Scoy and Brooks [27] constructed an algorithm for broadcasting M messages in an $n \times n \times n$ grid (all messages originate in vertex $(1, 1, 1)$) in time $3n + \frac{10}{3}M + 2$.

They also conjectured that there exists an algorithm for broadcasting M messages from a corner vertex of an equilateral d-dimensional grid with n points per edge in time $dn + \frac{d^2+1}{d}M + constant$.

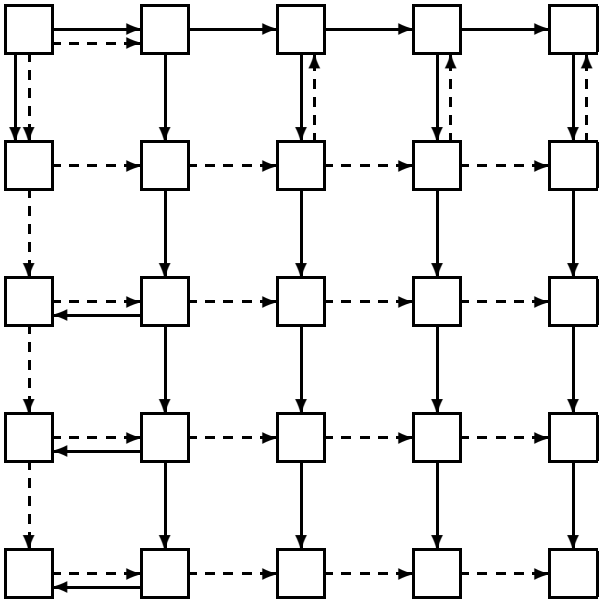


Figure 3.2: Broadcasting many messages in time $2n + \frac{5}{2}M + constant$ (Brooks, 1989)

3.2 Broadcasting multiple messages in 2-dimensional grids—approach

The edges used in broadcasting a particular message throughout a graph form a (directed) spanning tree rooted at the source of the message. If we split the set of messages into several sets each of which uses a different (but not necessarily disjoint) set of edges for broadcasting, we are constructing a set of spanning trees which share the same root. Since we are seeking an algorithm that executes in time $m+n+2M-4$, then the height of any such spanning tree must be no more than approximately $m+n$.

To achieve the $2M$ portion of the desired execution time we must have at least two spanning trees in the algorithm. This is because any node v (other than the root) in the spanning tree with $outdegree > 1$ introduces a delay of at least 3 between the time two successive messages can be completely transmitted through this node. Such

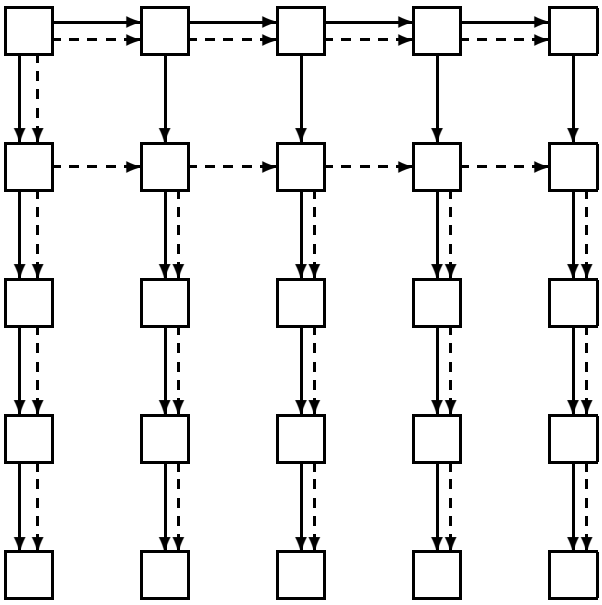


Figure 3.3: Broadcasting many messages in time $2n + \frac{5}{2}M + constant$ (Van Scoy and Brooks, 1994)

a node v is busy with a single message for time

$$indegree(v) + outdegree(v) \geq 1 + outdegree(v) \geq 1 + 2 = 3.$$

Since the longest path allowed in the spanning tree is approximately $m+n$ but there are mn nodes in the grid, there must be several branches in the spanning tree. This means that an algorithm that uses a single spanning tree for broadcasting multiple messages from the corner of a grid will require time proportional to at least $3M$. This is too slow to meet Farley's lower bound, so we must partition the messages into two or more sets and construct a spanning tree for each set.

For simplicity of discussion when each of the two spanning trees contains the same edge e, we shall think of edge e as two parallel edges e_1 and e_2, one in each of the spanning trees.

To participate in an algorithm (using two spanning trees) that gives us a coefficient of 2 associated with the number of messages M, for every vertex v, the sum $indegree(v) + outdegree(v)$ must be ≤ 4. For every vertex except the source $indegree(v) = 2$, because each vertex receives approximately half of the messages from the edges belonging to each of the two spanning trees. Each vertex (except the source) then has $outdegree \leq 2$. These two outgoing edges may be members of the same spanning tree or of different spanning trees. We call vertices with two outgoing edges belonging to the same spanning tree *branching vertices* and vertices with two outgoing edges belonging to different spanning trees *crossing vertices*.

We observe that branching and crossing cannot take place at the same vertex. Figure 3.4 shows examples of branching and crossing vertices.

In the case of an odd by odd grid, our approach restricts branching vertices to vertex positions whose indices sum to an odd number and crossing vertices to vertex positions whose indices sum to an even number (with the exception of the source, vertex $(1, 1)$, and vertices in rows 1 and m or columns 1 and n).

Odd numbered messages follow the comb-like pattern of Farley and Hedetniemi (see [7]) along row 1 and down even numbered columns. Even numbered messages follow the transpose of the Farley-Hedetniemi comb down column 1 and along even numbered rows. An odd numbered message follows a "curl" when it moves left one

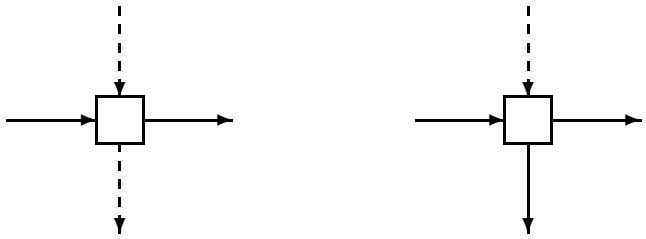


Figure 3.4: Example of crossing and passing vertices

column and then up one row, clockwise, heading back towards the source, and an even numbered message follows a curl when it moves up one row and left one column.

3.3 Broadcasting from a corner vertex in an odd by odd grid

3.3.1 Basic algorithm

The scheme described in the previous section can be used for broadcasting from a corner vertex in any odd by odd grid.

Let $G_{m,n}$ be an $m \times n$ two-dimensional grid. Let m and n be odd integers, $1 < n, m$. The following algorithm for broadcasting multiple messages from the corner $(1, 1)$ of $G_{m,n}$ requires time $D + 2M + 4$, where $D = m + n - 2$ is the diameter of the grid, and M is the number of messages.

Vertex $(1, 1)$ sends odd numbered messages along row 1, then down even numbered columns and down column n. Vertices in odd numbered columns receive odd numbered messages according to the rule:

> vertex $(2i + 1, 2j - 1)$ receives a message from vertex $(2i + 1, 2j)$, and sends it to vertex $(2i, 2j - 1)$, for $1 \leq i \leq \frac{m-1}{2}$, $1 \leq j \leq \frac{n-1}{2}$.

Odd numbered messages are sent from vertex $(1, 1)$ to vertex $(1, 2)$ at times $t \equiv 1 \pmod 4$.

Figure 3.5 shows the pattern for sending odd numbered messages from vertex $(1, 1)$ in 7×7 grid.

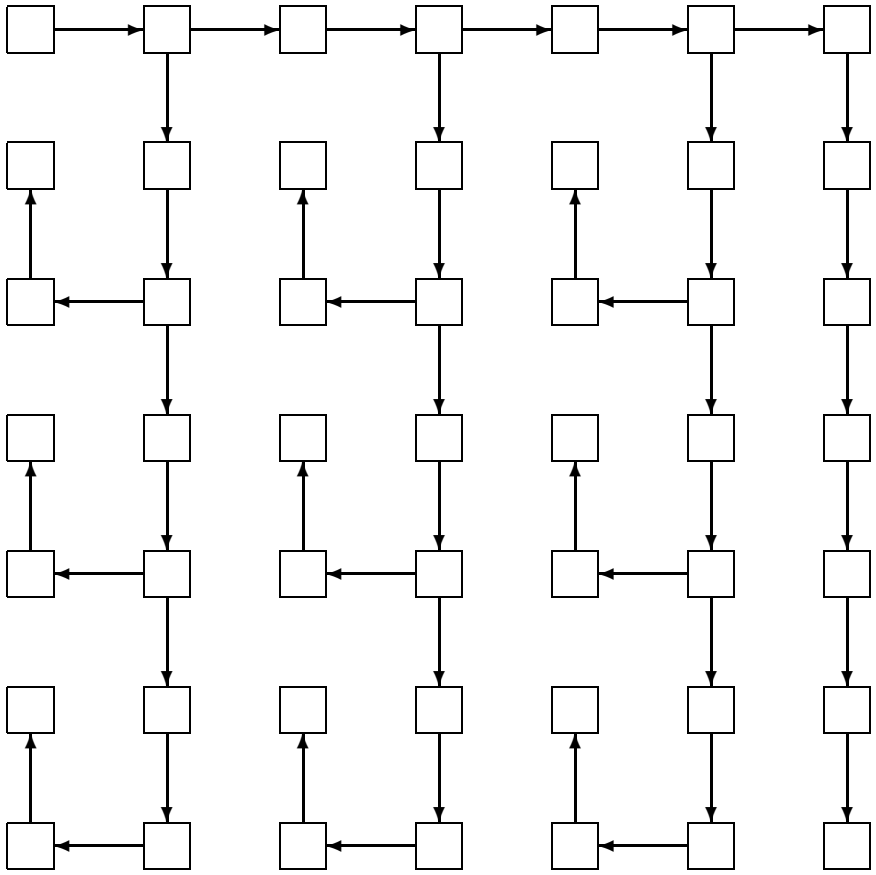


Figure 3.5: Sending odd numbered messages in 7×7 grid.

Even numbered messages are sent from vertex $(1, 1)$ to vertex $(2, 1)$ at times $t \equiv 3 \pmod 4$. They follow a pattern symmetric to the pattern for odd numbered messages.

Let's now consider our algorithm for transmitting odd numbered messages in a grid graph $G_{m,n}$, m, n odd.

Vertex $(1,1)$ sends odd numbered messages to vertex $(1,2)$ at time 1 and every 4 time units afterwards. Vertex $(1,k)$, after receiving a message from its west neighbor, sends it immediately to its east neighbor (if such exists); if k is even, vertex $(1,k)$ sends the same message south as soon as possible. Vertices in column n pass the message down as fast as possible.

Each vertex (i,j), $j < n$, that receives a message from its north neighbor at time t, forwards it south at time $t+1$ and, if j is even, west at time $t+3$. Each vertex that receives at time t an odd numbered message from its east neighbor, sends it north at time $t+3$.

Even numbered messages leave vertex $(1,1)$ at time 3 and every 4 time units afterwards. They follow the pattern that is the transpose of the pattern for odd numbered messages (with times offset by 2).

Figure 3.6 shows message patterns for even and odd numbered messages in a 7×7 grid. Arrows indicate the edges (and directions) used for sending odd numbered messages, dashed arrows edges for even numbered messages. Each edge used to send both odd and even numbered messages is shown as two directed edges. Each edge label indicates the time the given edge is used to send a message for the first time.

Consecutive odd or even numbered messages move along each edge every 4 time units.

Let's consider activity times of each vertex (i,j), $1 \leq i \leq m$ and $1 \leq j \leq n$.

- vertex $(1,1)$:
  - sends message k to $(1,2)$ at time $t = 1 + 2(k-1)$, for k odd;
  - sends message k to $(2,1)$ at time $t = 3 + 2(k-2)$, for k even.
- vertices $(1,j)$, $1 < j < n$:
  - receives message k from $(1,j-1)$ at time $t = j - 1 + 2(k-1)$, for k odd;
  - sends message k to $(1,j+1)$ at time $t = j + 2(k-1)$, for k odd;
  - for j even:

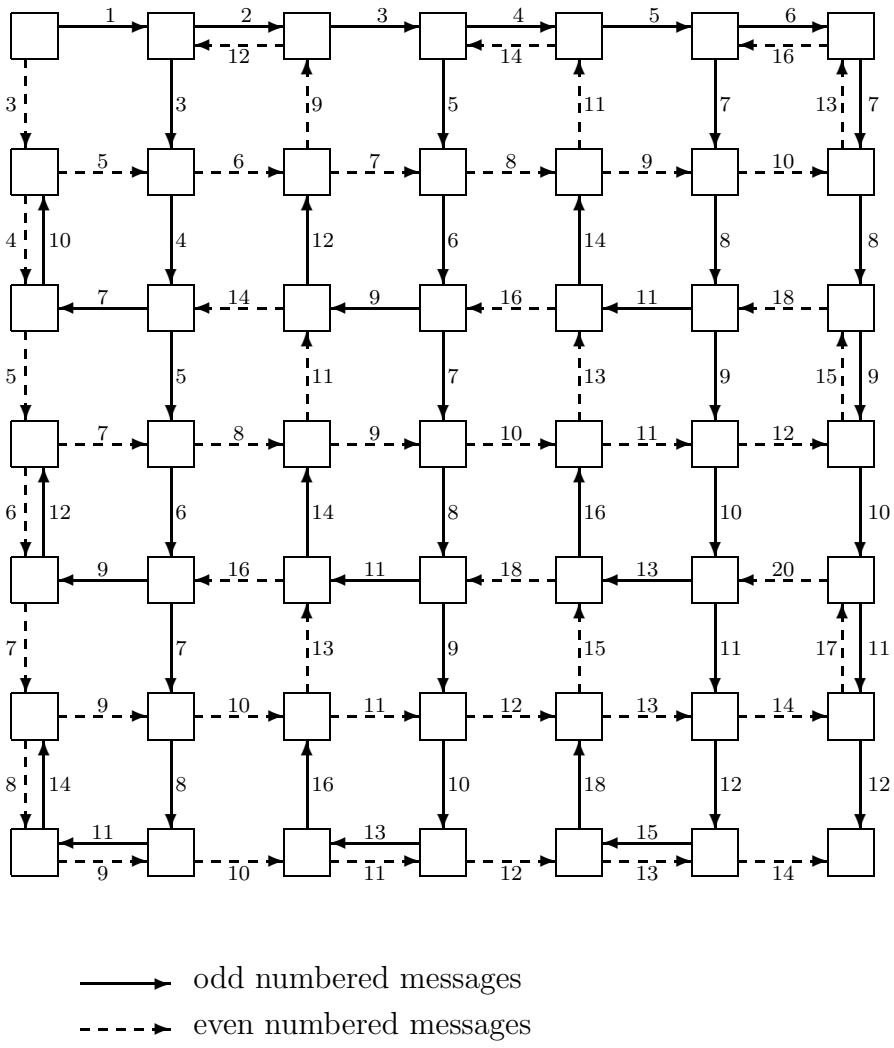


Figure 3.6: Message patterns in 7×7 grid

        * sends message k to $(2, j)$ at time $t = j + 1 + 2(k-1)$, for k odd;
        * receives message k from $(1, j+1)$ at time $t = j + 10 + 2(k-2)$, for k even;
    - for j odd:
        * receives message k from $(2, j)$ at time $t = j + 6 = 2(k-2)$, for k even;
        * sends message k to $(1, j-1)$ at time $t = j + 9 + 2(k-2)$, for k even.

- vertex $(1, n)$:
    - receives message k from $(1, n-1)$ at time $t = n - 1 + 2(k-1)$, for k odd;
    - sends message k to $(2, n)$ at time $t = n + 2(k-1)$, for k odd;
    - receives message k from $(2, n)$ at time $t = n + 6 = 2(k-2)$, for k even;
    - sends message k to $(1, n-1)$ at time $t = n + 9 + 2(k-2)$, for k even.

- vertices $(i, 1)$, $1 < i < m$:
    - receives message k from $(i-1, 1)$ at time $t = i + 1 + 2(k-2)$, for k even;
    - sends message k to $(i+1, 1)$ at time $t = i + 2 + 2(k-2)$, for k even;
    - for i even:
        * sends message k to $(i, 2)$ at time $t = i + 3 + 2(k-2)$, for k even;
        * receives message k from $(i+1, 1)$ at time $t = i + 8 + 2(k-1)$, for k odd;
    - for i odd:
        * receives message k from $(i, 2)$ at time $t = i + 4 + 2(k-1)$, for k odd;
        * sends message k to $(i-1, 1)$ at time $t = i + 7 + 2(k-1)$, for k odd.

- vertex $(m, 1)$:
    - receives message k from $(m-1, 1)$ at time $t = m + 1 + 2(k-2)$, for k even;
    - sends message k to $(m, 2)$ at time $t = m + 2 + 2(k-2)$, for k even;

- receives message k from $(m, 2)$ at time $t = m + 4 + 2(k-1)$, for k odd;
- sends message k to $(m-1, 1)$ at time $t = m + 7 + 2(k-1)$, for k odd.

- vertices (i, j), $1 < i < m$, $1 < j < n$:
  - i even, j even:
    - receives message k from $(i-1, j)$ at time $t = i + j - 1 + 2(k-1)$, for k odd;
    - sends message k to $(i+1, j)$ at time $t = i + j + 2(k-1)$, for k odd;
    - receives message k from $(i, j-1)$ at time $t = i + j + 1 + 2(k-2)$, for k even;
    - sends message k to $(i, j+1)$ at time $t = i + j + 2 + 2(k-2)$, for k even;
  - i even, j odd:
    - receives message k from $(i, j-1)$ at time $t = i + j + 1 + 2(k-2)$, for k even;
    - sends message k to $(i, j+1)$ at time $t = i + j + 2 + 2(k-2)$, for k even;
    - sends message k to $(i-1, j)$ at time $t = i + j + 4 + 2(k-2)$, for k even;
    - receives message k from $(i+1, j)$ at time $t = i + j + 7 + 2(k-1)$, for k odd;
  - i odd, j even:
    - receives message k from $(i-1, j)$ at time $t = i + j - 1 + 2(k-1)$, for k odd;
    - sends message k to $(i+1, j)$ at time $t = i + j + 2(k-1)$, for k odd;
    - sends message k to $(i, j-1)$ at time $t = i + j + 2 + 2(k-1)$, for k odd;
    - receives message k from $(i, j+1)$ at time $t = i + j + 9 + 2(k-2)$, for k even;

- i odd, j odd:
  - receives message k from $(i, j+1)$ at time $t = i + j + 3 + 2(k-1)$, for k odd;
  - sends message k to $(i-1, j)$ at time $t = i + j + 6 + 2(k-1)$, for k odd;
  - receives message k from $(i+1, j)$ at time $t = i + j + 5 + 2(k-2)$, for k even;
  - sends message k to $(i, j-1)$ at time $t = i + j + 8 + 2(k-2)$, for k even.

- vertices (m, j), $1 < j < n$:
  - receives message k from $(m, j-1)$ at time $t = m + j + 2(k-2)$, for k even;
  - sends message k to $(m, j+1)$ at time $t = m + j + 1 + 2(k-2)$, for k even;
  - for j odd:
    - receives message k from $(m, j+1)$ at time $t = m + j + 3 + 2(k-1)$, for k odd;
    - sends message k to $(m-1, j)$ at time $t = m + j + 6 + 2(k-1)$, for k odd;
  - for j even:
    - receives message k from $(m-1, j)$ at time $t = m + j - 1 + 2(k-1)$, for k odd;
    - sends message k to $(m, j-1)$ at time $t = m + j + 2 + 2(k-1)$, for k odd.

- vertices (i, n), $1 < i < m$:
  - receives message k from $(i-1, n)$ at time $t = i + n - 2 + 2(k-1)$, for k odd;
  - sends message k to $(i+1, n)$ at time $t = i + n - 1 + 2(k-1)$, for k odd;
  - for i odd:

        * receives message k from $(i+1, n)$ at time $t = i + n + 5 + 2(k-2)$, for k even;
        * sends message k to $(i, n-1)$ at time $t = i + n + 8 + 2(k-2)$, for k even;
    - for i even:
        * receives message k from $(i, n-1)$ at time $t = i + n + 1 + 2(k-2)$, for k even;
        * sends message k to $(i-1, n)$ at time $t = i + n + 4 + 2(k-2)$, for k even.

- vertex (m, n):
    - receives message k from $(m-1, n)$ at time $t = m + n - 2 + 2(k-1)$, for k odd;
    - receives message k from $(m, n-1)$ at time $t = m + n + 2(k-2)$, for k even.

As we can see from the above algorithm and Figure 3.6, vertex (m, n) receives message k at time $t = m + n + 2(k-2)$, for any k. Let D denote the diameter of grid $G_{m,n}$, and M denote the number of messages. Vertex (m, n) knows all M messages by the time $t = D + 2(M-1)$.

Not all of the vertices receive all of the messages by that time. When M is odd, vertex $(m-1, n-2)$ receives message M at time

$$t = m - 1 + n - 2 + 7 + 2(M-1) = D + 2(M-1) + 6.$$

For M even, the last vertex to receive M is $(m-2, n-1)$. It happens at the time

$$t = m - 2 + n - 1 + 2(M-2) + 9 = D + 2(M-1) + 6.$$

Theorem 1 *Time required by basic algorithm to broadcast M messages in $m \times n$ grid for odd m, n is*

$$t = n + m + 2M + 2.$$

Proof: We observe that if we have two vertices v_1 and v_2 that are in the same row or column and the distance between them is $2l$, then for any message, the time vertex v_1 sends or receives this message differs from the corresponding time for vertex v_2 by $2l$.

Let us now consider for each vertex the latest time it sends a message. We have to consider two cases: the number of messages M is odd or the number of messages M is even.

Case 1: M is odd.

The last time a vertex sends a message is:

- vertex $(1, 1)$:
  - sends message M at time $2M - 1$;
- vertex $(1, j)$, $1 < j < n$:
  - for j even: sends message M at time $t = j + 2M - 1 \leq n + 2M - 2$;
  - for j odd: sends message $M - 1$ at time $t = j + 2M + 3 \leq n + 2M + 1$;
- vertex $(1, n)$:
  - sends message $M - 1$ at time $t = n + 2M + 3$;
- vertex $(i, 1)$, $1 < i < m$:
  - for i even: sends message $M - 1$ at time $t = i + 2M - 3 \leq m + 2M - 4$;
  - for i odd: sends message M at time $t = i + 2M + 5 \leq m + 2M + 3$;
- vertex $(m, 1)$:
  - sends message M at time $t = m + 2M + 5$;
- vertex (i, j), $1 < i < m$, $1 < j < n$:
  - i even, j even: sends message M at time $t = i+j+2M-2 \leq m+n+2M-4$;

- i even, j odd: sends message $M-1$ at time $t = i+j+2M-2 \leq m+n+2M-5$;
- i odd, j even: sends message M at time $t = i+j+2M \leq m+n+2M-3$;
- i odd, j odd: sends message M at time $t = i+j+2M+4 \leq m+n+2M$;

- vertex (m, j), $1 < j < n$:
  - for j odd: sends message M at time $t = m+j+2M+4 \leq m+n+2M+2$;
  - for j even: sends message M at time $t = m+j+2M \leq m+n+2M-1$;

- vertex (i, n), $1 < i < m$:
  - for i odd: sends message $M-1$ at time $t = i+n+2M+2 \leq m+n+2M$;
  - for i even: sends message $M-1$ at time $t = i+n+2M-2 \leq m+n+2M-3$.

As we can see from the above times for sending messages, the last vertex to send a message is $(m, n-2)$. It sends message M at time $t = m+n+2M+2$.

Case 2: M is even.

The last time a vertex sends a message is:

- vertex $(1, 1)$:
  - sends message M at time $t = 2M-1$;

- vertex $(1, j)$, $1 < j < n$:
  - for j even: sends message $M-1$ to at time $t = j+2M-3 \leq n+2M-4$;
  - for j odd: sends message M at time $t = j+2M+5 \leq n+2M+3$;

- vertex $(1, n)$:
  - sends message M at time $t = n+2M+5$;

- vertex $(i, 1)$, $1 < i < m$:

- for i even: sends message M at time $t = i + 2M - 1 \leq m + 2M - 2$;
- for i odd: sends message $M - 1$ at time $t = i + 2M + 3 \leq m + 2M + 1$;

- vertex $(m, 1)$:
  - sends message $M - 1$ at time $t = m + 2M + 3$;

- vertex (i, j), $1 < i < m$, $1 < j < n$:
  - i even, j even: sends message M at time $t = i+j+2M-2 \leq m+n+2M-4$;
  - i even, j odd: sends message M at time $t = i+j+2M \leq m+n+2M-3$;
  - i odd, j even: sends message $M - 1$ at time $t = i + j + 2M - 2 \leq m + n + 2M - 5$;
  - i odd, j odd: sends message M at time $t = i+j+2M+4 \leq m+n+2M$;

- vertex (m, j), $1 < j < n$:
  - for j odd: sends message $M-1$ at time $t = m+j+2M+2 \leq m+n+2M$;
  - for j even: sends message $M-1$ at time $t = m+j+2M-2 \leq m+n+2M-3$;

- vertex (i, n), $1 < i < m$:
  - for i odd: sends message M at time $t = i+n+2M+4 \leq m+n+2M+2$;
  - for i even: sends message M at time $t = i + n + 2M \leq m + n + 2M - 1$.

In this case, the last vertex to send a message is $(m - 2, n)$. It sends message M at time $t = m + n + 2M + 2$.

□

Thus, the presented algorithm for broadcasting M messages from vertex $(1, 1)$ in $n \times m$ grid $G_{n,m}$ requires time

$$t = m + n + 2M + 2. \tag{3.1}$$

This differs from Farley's lower bound of $(D + 2(M - 1))$ by 6.

This result can be improved by observing that in large grids vertices close to the (m, n) corner have some "free" time units, and the last three or four messages don't have to be delayed on "curls".

For grids with $m, n > 6$ we can modify our approach by "speeding up" last messages. Modifications made depend on the number M of messages sent, whether this number is even or odd.

3.3.2 Improved algorithm for odd number of messages M

When M is odd we have one additional odd numbered message that is following the same pattern as the last odd numbered message in Figure 3.6. (Odd numbered messages are represented by solid arrows.)

Since for each vertex sending a message at time t, there is a vertex that receives this message at the same time t, we only list modifications that affect sending times. The following modifications are required:

- vertices (i, j), $1 < i < m$, $1 < j < n$:
  - vertex $(m-3, n-1)$ sends message M to $(m-3, n-2)$ at time $t = m + n + 2M - 3$;
  - vertex $(m-2, n-2)$ does not send message M;
- vertices (m, j), $1 < j < n$:
  - $j = n - 4$ or $j = n - 2$:
    - sends message $M - 2$ to $(m-1, j)$ at time $t = m + j + 2M - 2$;
    - sends message M to $(m-1, j)$ at time $t = m + j + 2M$;
  - $j = n - 3$ or $j = n - 1$:
    - sends message M to $(m, j-1)$ at time $t = m + j + 2M - 2$;
- vertices (i, n), $1 < i < m$:

– vertex $(m-2, n)$ sends message $M-1$ to $(m-2, n-1)$ at time $t = m+n+2M-2$.

Figure 3.7 shows the 6×6 corner (containing vertex (m, n)) sub-grid of our $m \times n$ grid. Edge labels indicate times (relative to some initial time) messages $M-2$, $M-1$, and M are transmitted.

Let $t = m+n+2M-18$ (time unit before vertex $(m-5, n-5)$ receives message $M-2$). Vertex $(m-5, n-5)$ (marked by A) receives message $M-2$ at time $t+1$, message $M-1$ at time $t+3$, and message M at time $t+5$. All earlier messages are sent and received as in the basic algorithm. Edge labels of the form t^* indicate message passing times that differ from the basic algorithm.

Note that the vertex $(m-3, n-2)$ (marked by X in Figure 3.7) receives message M from vertex $(m-3, n-1)$ and not from $(m-2, n-2)$.

Theorem 2 *For M odd, we can broadcast M messages in an $m \times n$ grid, with $m, n > 6$, m, n odd, in time $t = m+n+2M-2$.*

Proof: Let us now consider vertices that in the basic algorithm performed send operations after time $t = m+n+2M-2$. After the modifications, we have the following send times:

- vertex $(m-3, n-1)$ sends message M at time $t = m+n+2M-3$;
- vertex $(m-2, n-2)$ does not send message M; it sends message $M-1$ at time $t = m+n+2M-2$;
- vertex $(m, n-4)$ sends message M at time $t = m+n+2M-4$;
- vertex $(m, n-2)$ sends message M at time $t = m+n+2M-2$;
- vertex $(m, n-1)$ sends message M at time $t = m+n+2M-3$;
- vertex $(m-2, n)$ sends message $M-1$ at time $t = m+n+2M-2$.

The latest time any message is sent in the modified algorithm is $t = m+n+2M-2$.

□

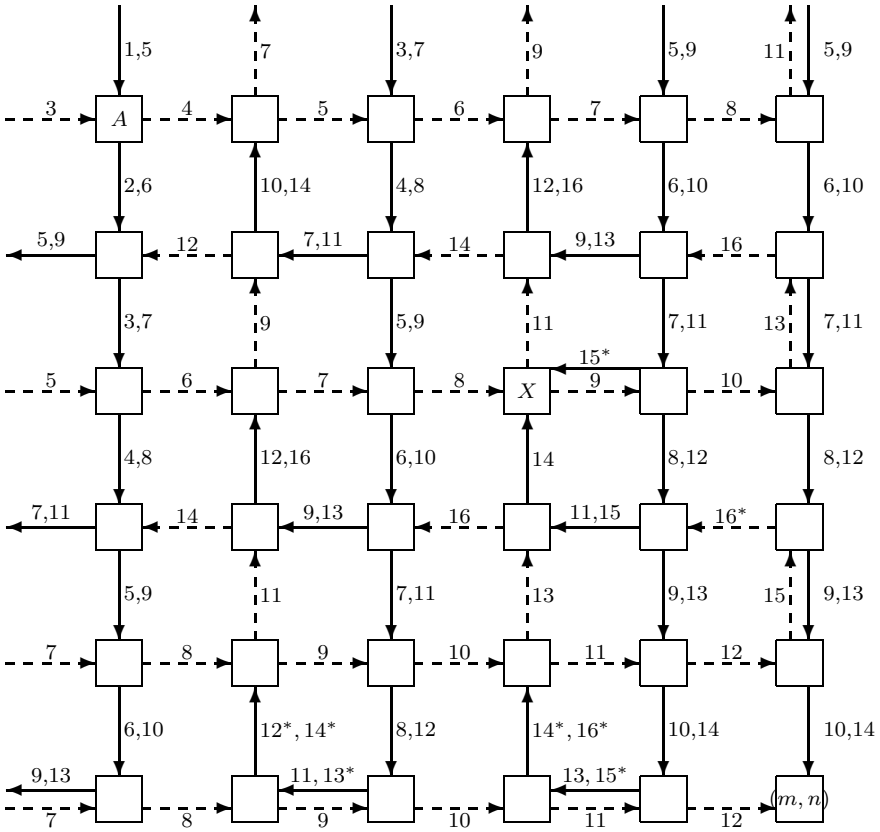


Figure 3.7: Modifications for M odd

3.3.3 Improved algorithm for even number of messages M

When M is even we have two additional messages (one even numbered and one odd numbered) that follow the same pattern as the last two messages in Figure 3.6. (Odd numbered messages are represented by solid arrows, even numbered messages are represented by dash arrows.) The following modifications that affect sending times are required:

- vertices (i, j), $1 < i < m$, $1 < j < n$:
  - vertex $(m-1, n-3)$ sends message M to $(m-2, n-3)$ at time $t = m+n+2M-3$;
  - vertex $(m-2, n-2)$ does not send message M;
- vertices (m, j), $1 < j < n$:
  - vertex $(m, n-2)$ sends message $M-1$ to $(m-1, n-2)$ at time $t = m+n+2M-2$.
- vertices (i, n), $1 < i < m$:
  - $i = m-4$ or $i = m-2$:
    - sends message $M-2$ to $(i, n-1)$ at time $t = n+i+2M-2$;
    - sends message M to $(i, n-1)$ at time $t = n+i+2M$;
  - $i = m-3$ or $i = m-1$:
    - sends message M to $(i-1, n)$ at time $t = n+i+2M-2$;

Figure 3.8 shows the 6×6 corner (containing vertex (m, n)) sub-grid of our $m \times n$ grid. Edge labels indicate times (relative to some initial time) messages $M-3$, $M-2$, $M-1$, and M are transmitted.

Let $t = m+n+2M-18$ (time unit before vertex $(m-5, n-5)$ receives message $M-3$). Vertex $(m-5, n-5)$ (marked by A) receives message $M-3$ at time $t+1$, message $M-2$ at time $t+3$, message $M-1$ at time $t+5$, and message M at time

$t+7$. All earlier messages are sent and received as in the basic algorithm. Edge labels of the form t^* indicate message passing times that differ from the basic algorithm.

Note that the vertex $(m-2, n-3)$ (marked by X in Figure 3.7) receives message M from vertex $(m-1, n-3)$ and not from $(m-2, n-2)$.

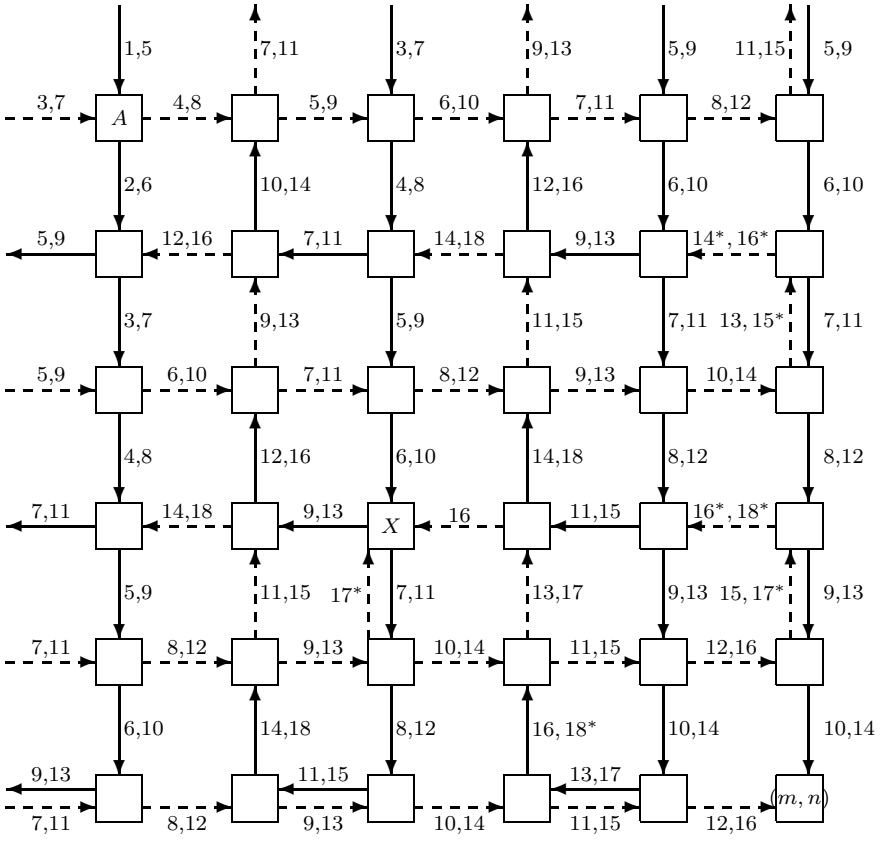


Figure 3.8: Modifications for M even

Since the approach in the case of M even is symmetric to the approach for odd number of messages M, from Theorem 2 we have the following.

Theorem 3 *For M even, we can broadcast M messages in an $m \times n$ grid, with $m, n > 6$, m, n odd, in time $t = m + n + 2M - 2$.*

3.4 Broadcasting from a corner vertex in arbitrary 2-dimensional grid

The algorithm presented in the previous section can be modified to apply to an arbitrary 2-dimensional grid $G_{m,n}$.

Let $G_{m,n}$ be a grid graph with both n and m even, $m, n > 6$. Let $G'_{m,n}$ be the graph obtained from $G_{m,n}$ by removing column 1 and row m. Graph $G'_{m,n}$ is a $(m-1) \times (n-1)$ grid. Let P denote the path consisting of vertices in column 1 and row m of $G_{m,n}$. Path P has length $m+n-1$. Consider the following calling scheme in $G_{m,n}$ (see Figure 3.9):

- vertex $(1,1)$ sends odd and even numbered messages to $(1,2)$ at times $t \equiv 1 \pmod 4$ and $t \equiv 3 \pmod 4$, respectively; vertex $(1,2)$ then serves as a source for broadcasting in $G'_{m,n}$;
- vertex $(1,1)$ serves as a broadcast source for path P; it sends odd and even numbered messages to $(2,1)$ at times $t \equiv 2 \pmod 4$ and $t \equiv 4 \pmod 4$, respectively.

The above scheme accomplishes the broadcasting of M messages in P in time

$$t = (m+n-2) + 2(M-1) + 1 = m + n + 2M - 3 \tag{3.2}$$

Broadcasting in $G'_{m,n}$ can be accomplished in time

$$t = (n-1) + (m-1) + 2M - 2 = n + m + 2M - 4 \tag{3.3}$$

An additional time unit is required to send each message from $(1,1)$ to $(1,2)$. It follows from (3.2) and (3.3), that the above scheme for broadcasting in $m \times n$ grid, n, m even, requires

$$t = n + m + 2M - 3 \tag{3.4}$$

time units. This differs from the lower bound in 2.1 by only one time unit.

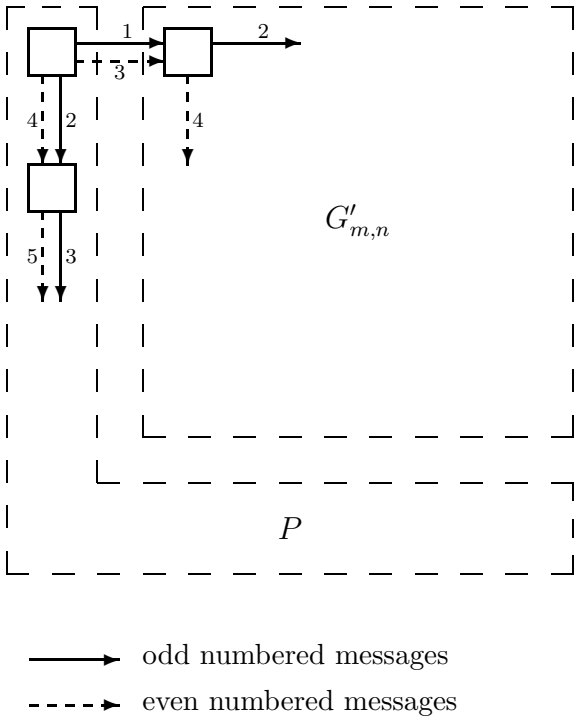


Figure 3.9: Message patterns in *even* × *even* grid

For m odd and n even, $G'_{m,n}$ is the graph obtained from $G_{m,n}$ by removing the first column, P is the first column. Broadcasting can be accomplished in time

$$t = (n-1) + m + 2M - 2 + 1 = n + m + 2M - 2 \tag{3.5}$$

From (2.1),(3.4) and (3.5), we obtain that, the time t required to broadcast M messages from a corner of any two-dimensional grid $G_{m,n}$, $m, n > 6$, satisfies the following inequalities:

$$m + n + 2M - 4 \leq t \leq m + n + 2M - 2.$$

3.5 Arbitrary position in 2-dimensional grid as a source of messages

Let $G_{m,n}$ be an *odd* $\times$ *even* 2-dimensional grid graph. For simplicity, we can assume that n is even and m is odd.

The algorithm described in the previous sections can be modified to work with a vertex $S = (1, k)$, $2 < k < n - 1$, as a source. We observe that, if k is odd, so is $n - k$; if k is even, both $k - 1$ and $n - k + 1$ are odd. Thus, we can divide graph $G_{n,m}$ into two grid graphs $G_1 = G_{l_1,m}$ and $G_2 = G_{l_2,m}$ such that $l_1 + l_2 = n$, both l_1 and l_2 are odd, and $l_1 = k$ or $l_1 = k - 1$. Figure 3.10 shows the broadcasting pattern with vertex $(1, k)$, k even, as a source.

Message patterns inside grids G_1 and G_2 are the same as in Figure 3.7 or 3.8. Times assigned to edges outgoing from S depend on the size of subgraphs. If $k - 1 \geq n - k + 1$, messages from S are sent to G_1 at times 1 and 3 (and every 4 time units thereafter) and inside G_2 at times 2 and 4. In case of $k - 1 < n - k + 1$, they are sent inside G_2 first. If both graphs G_1 and G_2 are lager than 7×7, the time required to broadcast M messages from S in G is

$$b_M(S) = d + 2(M - 1) + 2, \tag{3.6}$$

where d is the largest distance from S to any vertex in G.

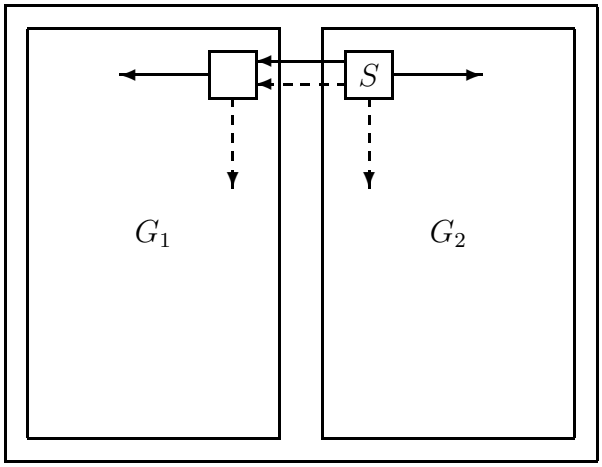


Figure 3.10: Vertex $(1, k)$ as a source

We were unable to extend the above results to an arbitrary vertex as a source. However, an algorithm developed by Brooks [2] can be modified to obtain a pattern for broadcasting multiple messages from any non-corner vertex v of a 2-dimensional grid. The time required to broadcast M messages from any source S by this scheme is

$$b_M(v) = d + \frac{5}{2}M + c \tag{3.7}$$

where d is defined as in 3.6, and c is a constant. Figure 3.11 shows the edges used in broadcasting and the times at which a message is transmitted along a given edge for the first time; the source is an inside vertex. Subsequent messages move along the given edge every 5 time units.

The same approach can be used when the source vertex is an edge vertex. By removing all rows of the grid that are above the source vertex, we can obtain edge patterns for an edge (non-corner) vertex as a source.

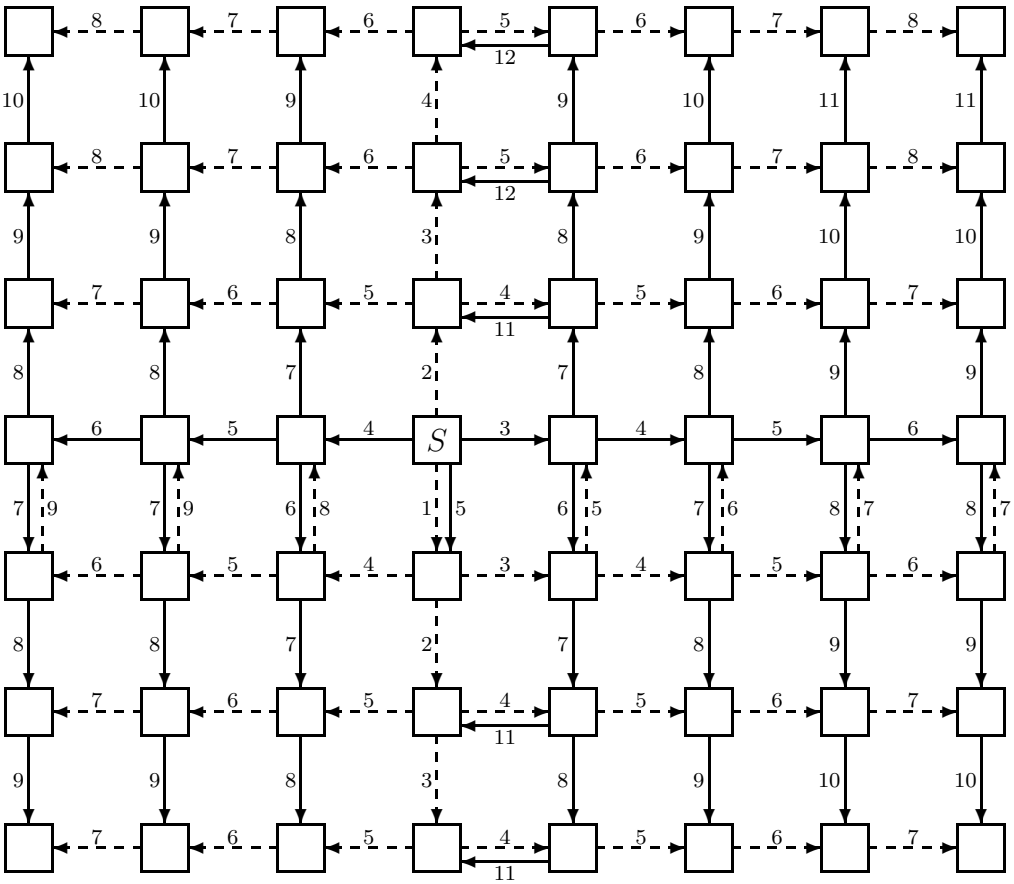


Figure 3.11: Inside vertex as a source

3.6 Arbitrary dimension of a grid

3.6.1 A $3n + 3M + constant$ time algorithm for 3-dimensional grid

We now present an algorithm for broadcasting M messages from a corner vertex of a 3-dimensional grid that requires time $3n + 3M + constant$.

In our algorithm, all of the messages follow the same message pattern. This pattern forms a single directed spanning tree of the 3-dimensional grid. In this tree the maximum degree of any vertex is 3. The message pattern is presented in Figure 3.12.

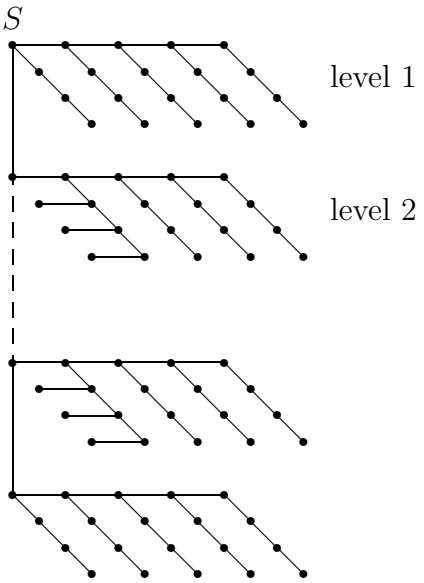


Figure 3.12: Broadcasting in 3-dimensional grid

The time required to broadcast 1 message is $\max(2n+1, 3n-3)$. Subsequent messages are sent every 3 time units. Thus, the time required to broadcast M messages

is

$$\max(2n+1, 3n-3) + 3(M-1).$$

The diameter of an $n \times n \times n$ grid is $D = 3n-3$. For $n \geq 4$, $3n-3 \geq 2n+1$. Thus, for $n \geq 4$, the time required to broadcast m messages is

$$D + 3(M-1) = 3(n-1) + 3(M-1).$$

3.6.2 Higher grid dimensions

An algorithm for higher dimension grids is constructed by induction on d. Let's assume that we have $ln + 3M + constant$ time algorithms for l-dimensional grids for all $l < d$. We now construct algorithm for d-dimensional grid.

Label the dimensions of our grid by $1, 2, \ldots, d$. At time 1, source vertex $(1, 1, \ldots, 1)$ sends a message to its neighbor along dimension 1.

At time 2, it sends the same message to its neighbor along dimension 2. This vertex now broadcasts the received message in a $(d-1)$-dimensional grid of which it is a corner vertex. The broadcast pattern is obtained from an algorithm for a $(d-1)$-dimensional grid by removing level 1.

At time 3, the source sends the message to its neighbor along dimension 3; the receiving vertex serves as a source for broadcasting in a $(d-2)$-dimensional grid.

Every vertex of the form $v = (i, 1, 1, \ldots, 1)$, $1 < i < n$, moves the received message along dimension 1 as fast as possible. Next, it sends the message to its neighbor along dimension 2, who broadcasts it in a $(d-1)$-dimensional grid. Since vertex v has already "used" 3 edges, it cannot forward the message to its dimension 3 neighbor. Uninformed vertices in a $(d-2)$-dimensional subgraph that has v as a corner receive the message from their dimension 2 neighbors.

Vertex $(n, 1, \ldots, 1)$ can forward a received message to its dimension 3 neighbor.

Figure 3.13 shows the generalized layout of broadcasting for a 4-dimensional grid. Figures 3.14, 3.15, 3.16 show broadcasting patterns for top, middle and bottom layers respectively.

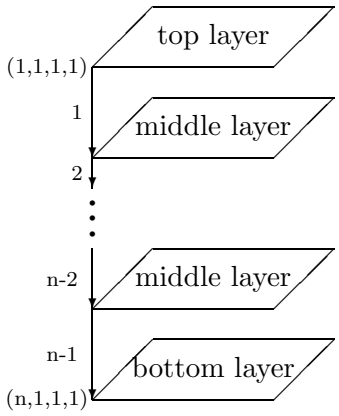


Figure 3.13: Broadcasting in 4D grid

The first message, sent from the source at time 1, reaches vertex $(n, n, \ldots, n)$ in time $d(n-1)$. Some of the vertices receive the first message at time $(d-1)n+d-1$. This value, for a fixed d, can be larger than $d(n-1)$ depending on the value of n. Thus, the time required to broadcast 1 message is

$$b_1((1,1,\ldots,1)) = \max((d-1)n+d-1, d(n-1)).$$

Because consecutive messages follow the same pattern every 3 time units

$$b_M((1,1,\ldots,1)) = \max((d-1)n+d-1, d(n-1)) + 3(M-1).$$

However, for $n \geq 2d-2$, $d(n-1) \geq (d-1)n+d-1$. In such case

$$b_M((1,1,\ldots,1)) = d(n-1) + 3(M-1).$$

3.7 Conclusions

We have shown that broadcasting M messages from the corner vertex in any 2-dimensional $m \times n$ grid G can be accomplished in time $t = m+n+2M+2$.

For $m, n > 6$ a modified approach gives us broadcasting time $t = m+n+2M-2$. If both m, n are even, the time required to broadcast M messages is $t = m+n+2M-3$. This time differs by only 1 from the theoretical lover bound obtained by Farley [5].

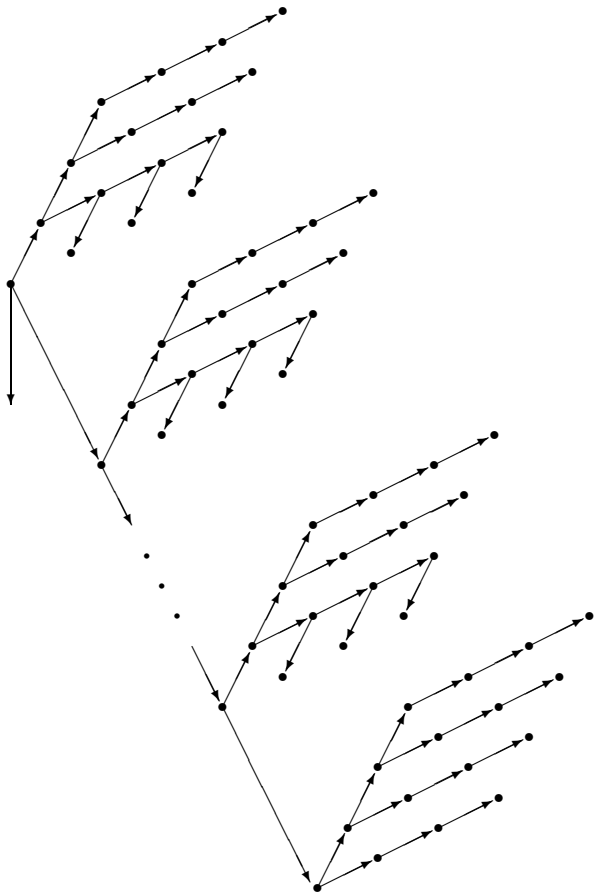


Figure 3.14: 4D - top layer

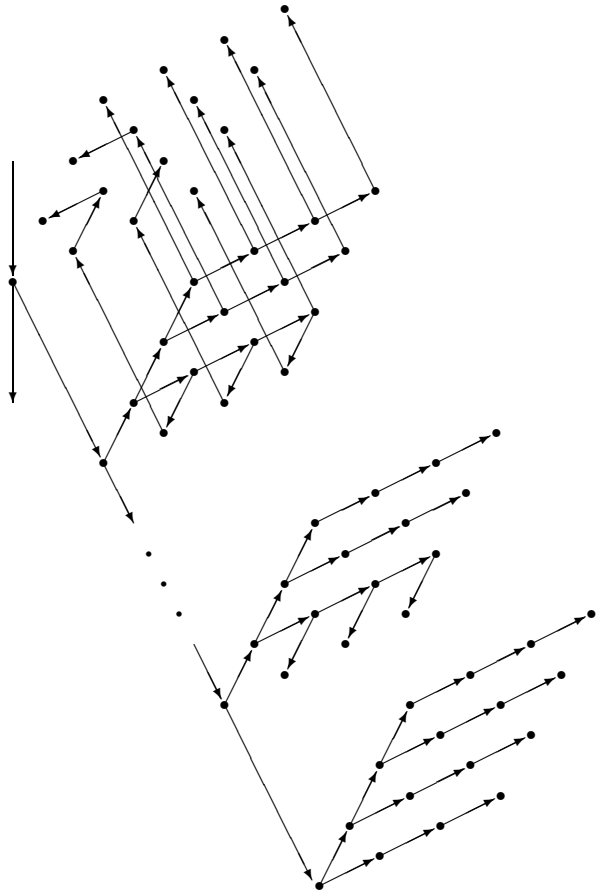


Figure 3.15: 4D - middle layer

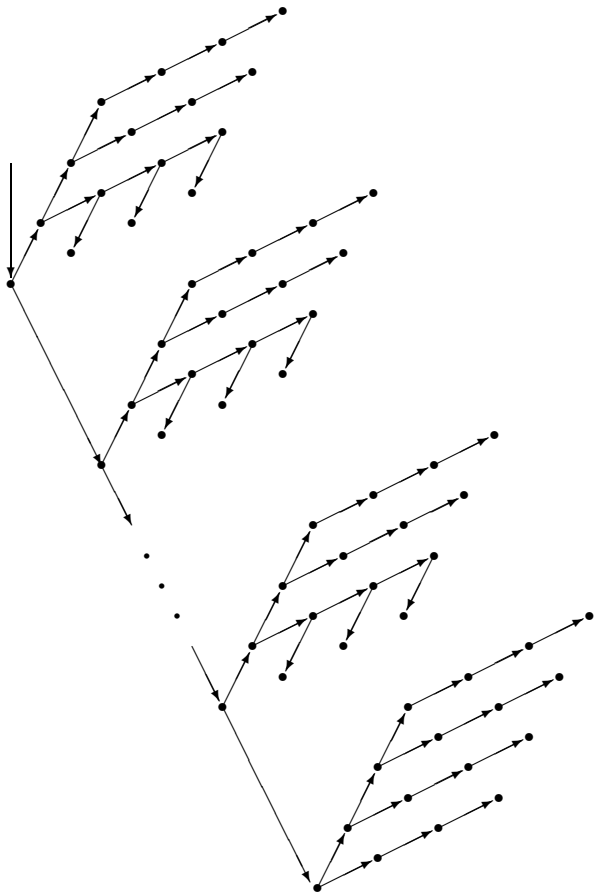


Figure 3.16: 4D - bottom layer

For the source vertex S positioned inside grid G, we can broadcast M messages in time $t = d + \frac{5}{2}M + c$, where d is the largest distance between S and any vertex in G, c is a constant.

For higher dimensions (3- and 4-dimensional grids), we presented a broadcasting strategy that gives us time $t = dn + 3M + c$, where d is the dimension of the grid, n is the number of vertices along each edge, c is a constant.

Chapter 4

Line broadcasting

4.1 Definitions and previous results

Restrictions on types of calls that can be placed during a given time unit determine the type of broadcasting. The most frequently studied type is local broadcasting. However other approaches are also possible. Farley in [6] studies three different approaches.

In *local broadcasting*, a vertex can call another vertex only if there is an edge between them. This situation occurs in message-switched networks.

In *path broadcasting* an informed vertex can call any other vertex, provided there is an open path between them. An *open path* is a path in a graph such that every vertex and edge of the path is not involved in another call during that time unit.

In *line broadcasting* a vertex can call any other vertex, provided there is an open line between them. An *open line* is a path such that every edge in the path is not involved in another call during that time unit. In line broadcasting a vertex may "switch-through" any number of calls. This approximates the situation in circuit-switched networks.

Figure 4.1 shows the differences between possible calls for path and line broadcasting. Dash lines represent edges of the graph (shown in the top picture), and arrows represent lines or paths used to make calls.

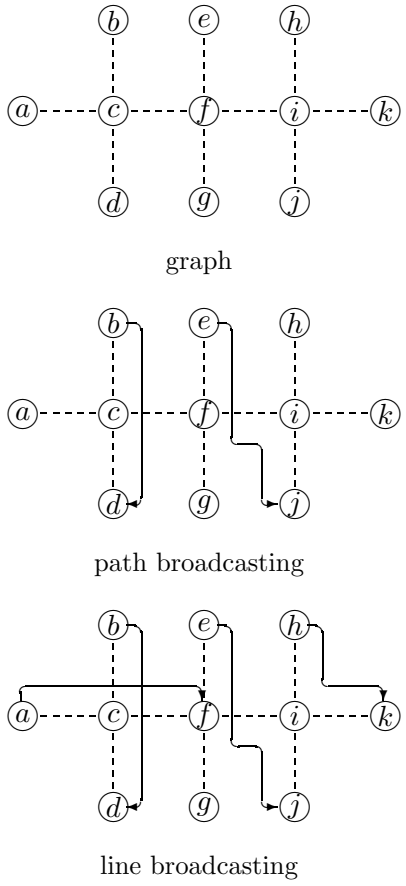


Figure 4.1: Path and line broadcasting

In path broadcasting (middle graph), vertex b can call vertex d (call "goes through" vertex c) and in the same time unit vertex e can call vertex j (using vertices f and i to forward the message). Those two calls do not use the same edges nor vertices.

In line broadcasting (bottom graph), at the same time unit vertex h can call vertex k using vertex i. Also the call from vertex a to f is allowed in line broadcasting but not in path broadcasting. In bottom graph, vertices c and i are involved in "switching-through" two calls each in the same time unit. Vertex f is both the receiving vertex and switching vertex in the same time unit. Those situations are allowed in line broadcasting but not in path broadcasting.

Local broadcasting is a specific case of both path and line broadcasting.

The minimum time required to complete broadcasting in any connected graph on n vertices is $\lceil \log_2 n \rceil$. A *minimum-time broadcast graph* is a graph such that the broadcast can be completed in a minimum number of time units regardless of the message originator. Each type of broadcasting imposes its own constraints upon the architecture of associated minimum-time broadcast graphs.

Farley showed that line broadcasting can be completed in the minimum time $\lceil \log_2 n \rceil$ in any connected graph and for any source vertex. He developed an algorithm that given any tree and a source vertex produces a calling schedule that completes line broadcasting in optimal time.

For any connected graph on n vertices we can always find an optimal time line broadcasting algorithm. However some of the calls are placed along paths of length greater than 1. We "pay" for optimal time by longer call distances. We can define the *cost* of a line broadcasting algorithm in terms of distance, as the number of edges traversed, of the calls involved during a minimum time line broadcast process. There are three such cost measures:

cumulative cost – sum of distances of all calls in a broadcast;

synchronous cost – sum of maximum of distances during each of the $\log_2 n$ time units of broadcast;

asynchronous cost – maximum of the sum of distances of call to each of the $n-1$

recipients during a broadcast.

Cumulative cost corresponds to the total number of edges used in broadcasting. Since some calls can be placed over lines of length greater than 1, some of the edges can be used more than once in the broadcasting process (however it has to take place during different time units). In local broadcasting cumulative cost is always $n - 1$ where n is the number of vertices.

Asynchronous cost is equal to the length of the longest path between the source and any other vertex.

Synchronous cost gives us an idea of how long the broadcasting process will take if we do not assume that all calls take only one time unit regardless of distance, but that a call placed over a line of length l takes l time units.

Although for some line broadcasting algorithms synchronous and asynchronous cost can be equal, in general they are two different cost measures.

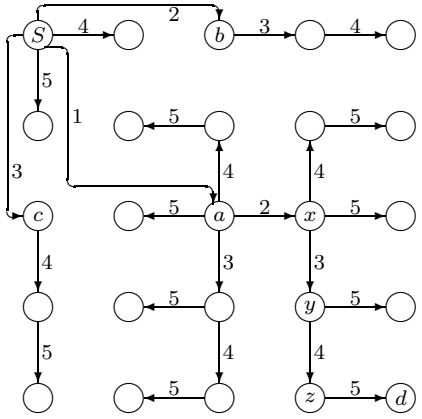


Figure 4.2: Example of line broadcasting

Figure 4.2 shows an example of optimal time line broadcasting scheme. The underlying graph is a 5×5 grid (edges not used in broadcasting are omitted).

In a 5×5 grid there are 25 nodes, thus the time required to complete line broadcasting is $\lceil \log_2 25 \rceil = 5$. Let us now calculate the different costs for this broadcasting process.

Synchronous cost:

time	longest call	cost
1	S calls a	4
2	S calls b	2
3	S calls c	2
4	all calls	1
5	all calls	1
total cost		10

Asynchronous cost: (the longest path is from S to d)

time	call	cost
1	S calls a	4
2	a calls x	1
3	x calls y	1
4	y calls z	1
5	z calls d	1
total cost		8

Cumulative cost:

time	cost
1	4
2	$2+1$
3	$2+3\times 1$
4	7×1
5	10×1
total cost	29

We can ask: "what is the minimum cost, for a given cost measure, of any line broadcasting algorithm for a given graph?".

4.2 Line broadcasting in grid graphs

4.2.1 Algorithms

Let us consider line broadcasting from a corner vertex $(1, 1)$ in a 2-dimensional $n \times n$ grid graph. We observe that since all vertices have to receive the message, both the synchronous and the asynchronous costs can be no less than $2n - 2$, the diameter of the grid.

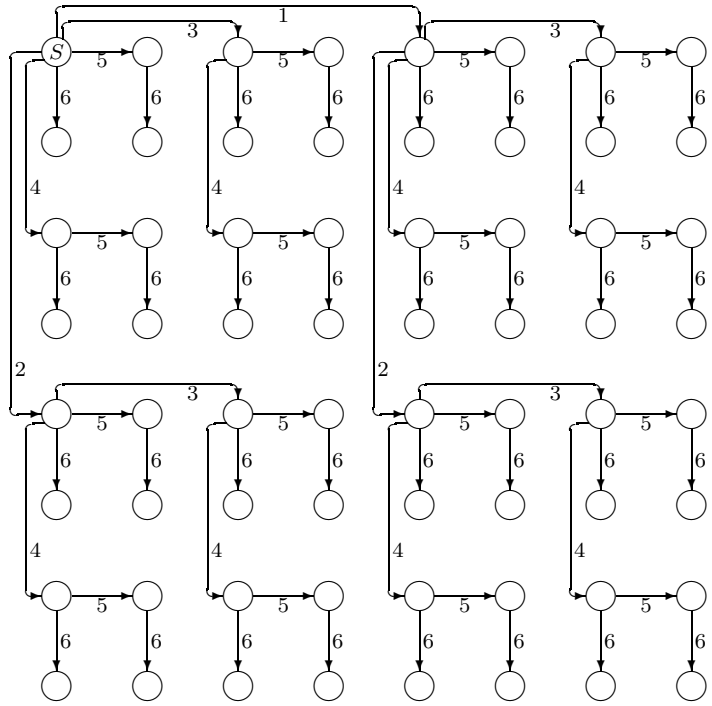


Figure 4.3: Example of line broadcasting in 8×8 grid

This value can be easily achieved in the case when $n = 2^k$. We can divide our grid into four equal square grids. During first two time units, corner vertices of all sub-grids can be informed. We can repeat the above two steps recursively for our sub-grids. This process for 8×8 grid is shown in figure 4.3.

Now we can write formally a "divide and conquer" *Broadcast* algorithm, which

when invoked by $Broadcast(1, 1, n)$, accomplishes line broadcasting from the $(1, 1)$ corner vertex in any $n \times n$ grid, when n is a power of 2. All parameters are of type $Integer$; i, j are coordinates of the broadcast source, $size \geq 2$ is the grid size.

$Broadcast(i, j, size : Integer)$:
begin
(1) send message from vertex (i, j) to $(i, j + size/2)$
(2) **simultaneously**:
    send message from vertex (i, j) to $(i + size/2, j)$
    send message from vertex $(i, j + size/2)$ to $(i + size/2, j + size/2)$
(3) **if** $size > 2$ **then simultaneously**:
    $Broadcast(i, j, size/2)$:
    $Broadcast(i, j + size/2, size/2)$:
    $Broadcast(i + size/2, j, size/2)$:
    $Broadcast(i + size/2, j + size/2, size/2)$:
end

We observe that, in the above algorithm, all the calls made during any single time unit were placed over the same distance. Thus the asynchronous and synchronous costs are equal. Since any vertex is reached from the source vertex along the shortest path connecting them, the cost is equal to $2n - 2$, the diameter of the grid.

In the case when n is not a power of 2, the "divide and conquer" method has to be modified. When the grid is "big enough", i.e. the time required for line broadcasting in the $n \times n$ grid is the same as for the larger grid with size that is a power of 2, we can use the "divide and conquer" strategy for the larger grid, but only include calls that fall within the smaller $n \times n$ grid. This strategy for a the 7×7 grid is presented in Figure 4.4.

This strategy gives us a broadcasting scheme that is optimal in the asynchronous sense. However the synchronous cost for the 7×7 grid is 14 (diameter is 12). To achieve a strategy that has both synchronous and asynchronous costs optimal, we have to partition our grid differently.

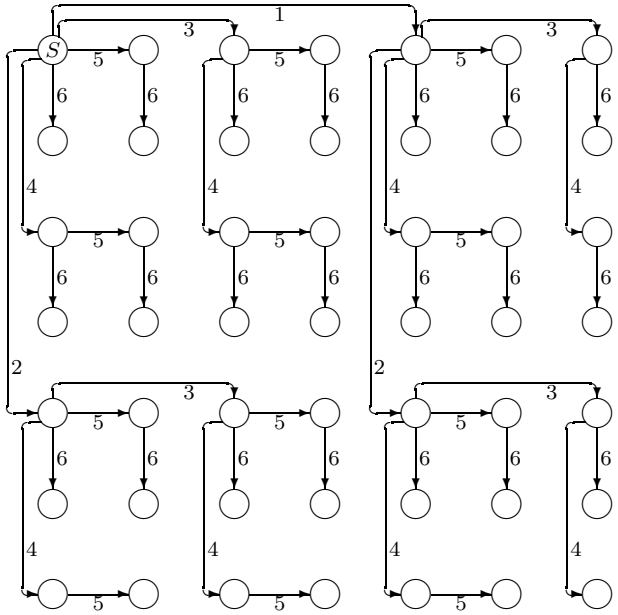


Figure 4.4: Divide and conquer approach for 7×7 grid

Figure 4.5 shows two different partitions of the 7×7 grid. Partition (a) is the one used in Figure 4.4. In this approach the largest (4×4) sub-grid is in the upper left corner of our grid. If we place the largest sub-grid (one that has sides that are powers of 2) in the lower right corner we obtain partition (b) from Figure 4.5.

During the first two time units, the corners of the four sub-grids receive a message. Then we can use the "divide and conquer" strategy for the largest sub-grid. In smaller sub-grids we follow the same strategy, but only include calls that have both sender and receiver within the same sub-grid. The full line broadcasting scheme using this approach for the 7×7 grid is presented in Figure 4.6.

For smaller grid sizes when n is even we can partition our grid into four equal square sub-grids. Since two more time units are required to broadcast in an $n \times n$ grid than are necessary for broadcasting in an $\frac{n}{2} \times \frac{n}{2}$ grid, during the first two time units the corner vertices in our sub-grids can be informed. Then corner vertices can broadcast in their corresponding sub-grids in the remaining time.

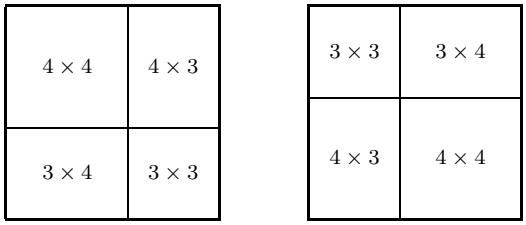


Figure 4.5: Partitions for 7×7 grid

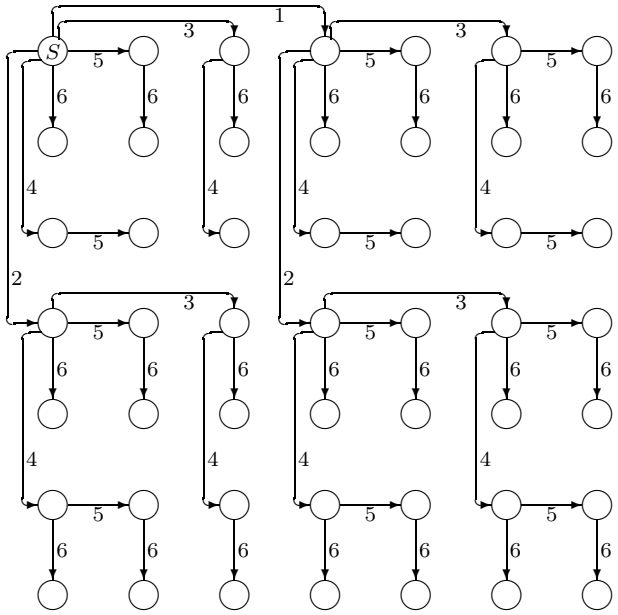


Figure 4.6: Optimal approach for 7×7 grid

For n odd, $n = 2^k + 1$, we can modify the "divide and conquer" method for the $2^k \times 2^k$ grid. We divide our $n \times n$ grid into four equal square sub-grids leaving the middle row and column out. We proceed with our method. The vertices in the "left-out" section are informed during the last two time units. This process for the 9×9 grid is presented in Figure 4.7. Thick arrows represent the calls (made during the last

two time units) that are different from the calls of the "divide and conquer" method.

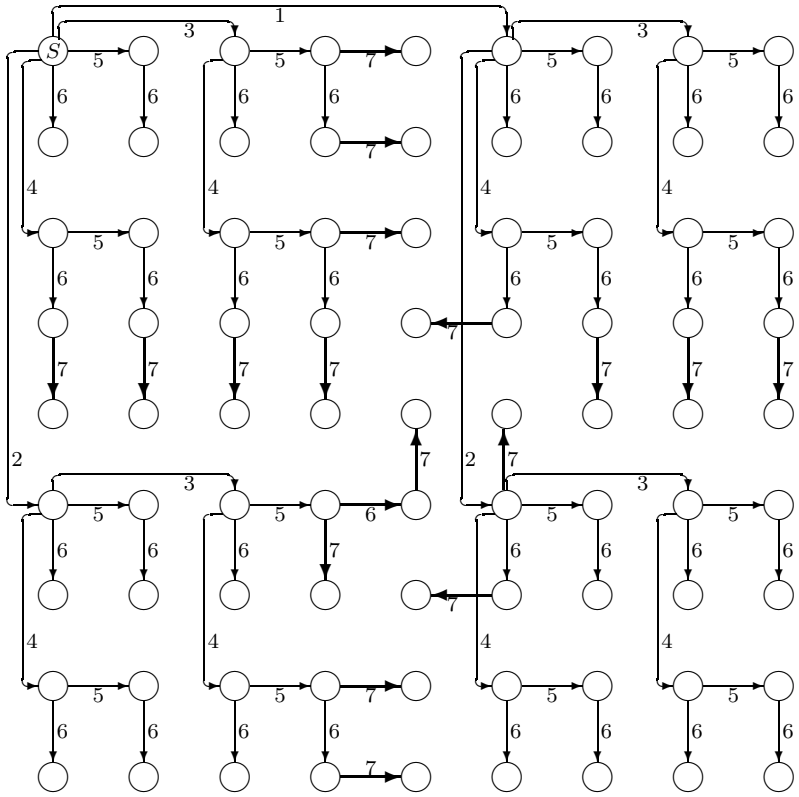


Figure 4.7: Divide and conquer for 9×9 grid

We observe that all calls made during a single time unit are placed over the same distance. The vertices that are not informed by calls placed along the shortest path from the source are away from vertex (n, n), so the distance used by calls is still smaller than the diameter of the grid.

The optimal asynchronous cost strategies for various grid sizes are presented in Table 4.1.

Table 4.1: Optimal asynchronous cost strategies

Strategies for optimal algorithms			
grid size	#vertices	time	approach
2×2	4	2	power of 2
3×3	9	4	like 4×4
4×4	16	4	power of 2
5×5	25	5	$5 = 2^2 + 1$
6×6	36	6	like 8×8
7×7	49	6	like 8×8
8×8	64	6	power of 2
9×9	81	7	$9 = 2^3 + 1$
10×10	100	7	divide into 5×5 grids
11×11	121	7	special case
12×12 to 15×15		8	like 16×16
16×16	256	8	power of 2
17×17	289	9	$17 = 2^4 + 1$
18×18	324	9	divide into 9×9 grids
19×19	361	9	like 20×20
20×20	400	9	divide into 10×10 grids
21×21	441	9	like 22×22
22×22	484	9	divide into 11×11 grids
23×23 to 31×31		10	like 32×32
32×32	1024	10	power of 2
33×33	1089	11	$33 = 2^5 + 1$
34×34	1156	11	divide into 17×17 grids
35×35	1225	11	like 36×36
36×36	1296	11	divide into 18×18 grids
37×37	1369	11	like 38×38
38×38	1444	11	divide into 19×19 grids
39×39	1521	11	like 40×40
40×40	1600	11	divide into 20×20 grids
41×41	1681	11	like 42×42
42×42	1764	11	divide into 21×21 grids
43×43	1849	11	like 44×44
44×44	1934	11	divide into 22×22 grids

continued on next page

continued from the previous page			
grid size	#vertices	time	approach
45×45	2025	11	
46×46 to 63×63		12	like 64×64
64×64	4096	12	power of 2
65×65	4225	13	$65 = 2^6 + 1$

In Table 4.1, the 11×11 grid was left out as a special case. The optimal asynchronous cost broadcasting scheme for this grid is presented in Figure 4.8.

4.2.2 Cumulative cost

Let us now consider the cumulative cost of line broadcasting for the *Broadcast* algorithm from Section 4.2.1. Let c_k denote the cumulative cost of line broadcasting for this method for the $2^k \times 2^k$ grid. We observe that for the 2×2 grid,

$$c_1 = 3.$$

For the $2^{k+1} \times 2^{k+1}$ grid we make 3 calls (during first two time units) of length 2^k each. This accomplishes "division" of our grid into four equal sub-grids of size $2^k \times 2^k$ each. In each of the sub-grids one corner vertex is informed and can serve as a source for line broadcasting. Thus

$$c_{k+1} = 3 * 2^k + 4c_k.$$

From the above, we obtain recursive equations for the cumulative cost:

$$\begin{cases} c_1 &= 3, \\ c_{k+1} &= 3 * 2^k + 4c_k. \end{cases} \tag{4.1}$$

Lemma 1 *The cumulative cost c_k for the divide and conquer method of line broadcasting from the corner vertex in $2^k \times 2^k$ grid is*

$$c_k = 3 * 2^{k-1}(2^k - 1).$$

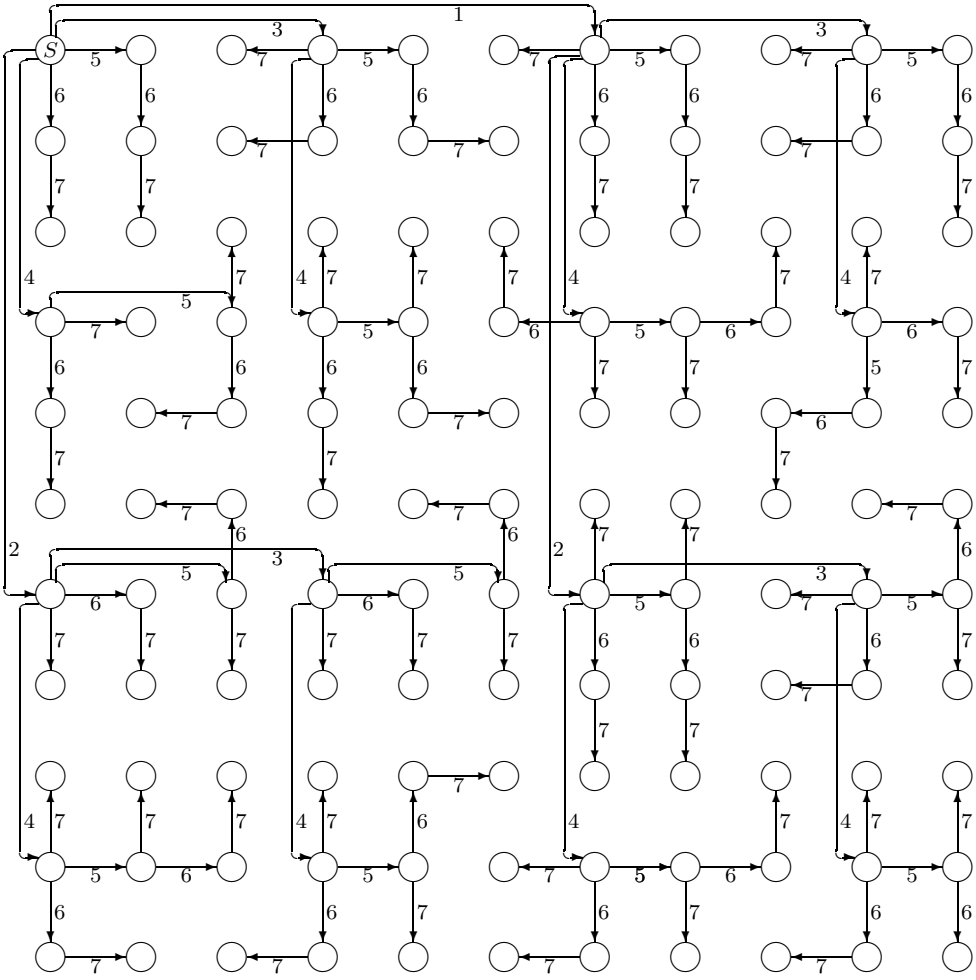


Figure 4.8: Line broadcasting in 11×11 grid

Proof: We can prove the above by induction on k.
For $k = 1$, $c_1 = 3$ (each vertex other than source has to receive the message).
Let us assume that $c_k = 3 * 2^{k-1}(2^k - 1)$. We have to show that $c_{k+1} = 3 * 2^k(2^{k+1} - 1)$.
From 4.1 we know that

$$c_{k+1} = 3 * 2^k + 4c_k.$$

From our inductive assumption we have

$$c_k = 3 * 2^{k-1}(2^k - 1).$$

Thus

$$\begin{aligned} c_{k+1} &= 3 * 2^k + 4 * 3 * 2^{k-1}(2^k - 1) \\ &= 3 * 2^k + 3 * 2^k * 2(2^k - 1) \\ &= 3 * 2^k(1 + 2(2^k - 1)) \\ &= 3 * 2^k(1 + 2^{k+1} - 2) \\ &= 3 * 2^k(2^{k+1} - 1), \end{aligned}$$

which completes our proof.

□

The above algorithm, optimal in the synchronous and asynchronous sense, is not optimal when it comes to cumulative cost as will be shown in the next section.

4.2.3 Lower cumulative cost algorithm

Let us consider the case when n is a power of 2. We have that, during each time step, every informed vertex has to make a call. Each vertex can make at most 4 calls of distance 1 (to inform its neighbors). In a 4×4 grid any inside vertex can be a source for broadcasting a message to all vertices by placing only calls of distance 1. The 4×4 grid is the largest grid in which line broadcasting with calls of distance 1 can be accomplished in minimal time. The cumulative cost is 15. This case is presented in Figure 4.9.

However, when the source is a corner vertex in the 4×4 grid, we can show that the cumulative cost of minimal time line broadcasting is 18 (the divide and conquer method is optimal).

Lemma 2 *The lowest cumulative cost c for line broadcasting from the corner vertex in a* 4×4 *grid is 18.*

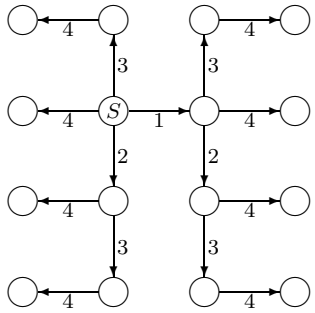


Figure 4.9: Optimal broadcasting in 4×4 grid

Proof: Let us assume that we have a minimal time line broadcasting algorithm from the corner vertex in a 4×4 grid. Since in the 4×4 grid we have 16 vertices and each vertex (other than the source) has to receive the message, $c \geq 15$.

In any grid with a number of vertices that is a power of 2, every informed vertex is busy every time unit. Thus the corner vertex has to make 4 calls. Since it has only two neighbors at distance 1, it has to make at least two calls over distance greater than 1. If there are more than two such calls, or if one of the calls is of length greater than 2, $c \geq 18$.

Let us now assume that the corner vertex makes two calls of length 1 (to vertices $(1, 2)$ and $(2, 1)$) and two calls of length 2. The vertex informed during the first call has to make 3 calls. However it has at most 2 neighbors at distance 1 that are not informed by calls from the source. Thus it has to make at least one call of length greater than 1. This gives us the cumulative cost of our algorithm $c \geq 18$.

From Lemma 1 we have that the cumulative cost c_2 of $Broadcast$ algorithm for a 4×4 grid is $c_2 = 18$. Thus the divide and conquer method is optimal for this case.

□

We can now describe a better (lower cumulative cost) algorithm for line broadcasting from a corner vertex of a 2-dimensional $n \times n$ grid, where n is a power of 2, $n \geq 4$. We will define two broadcasting procedures: $CornerBroadcast(i, j, size : Integer)$ and $Broadcast1(i, j, size : Integer)$. In both cases i and j are the coordinates of the

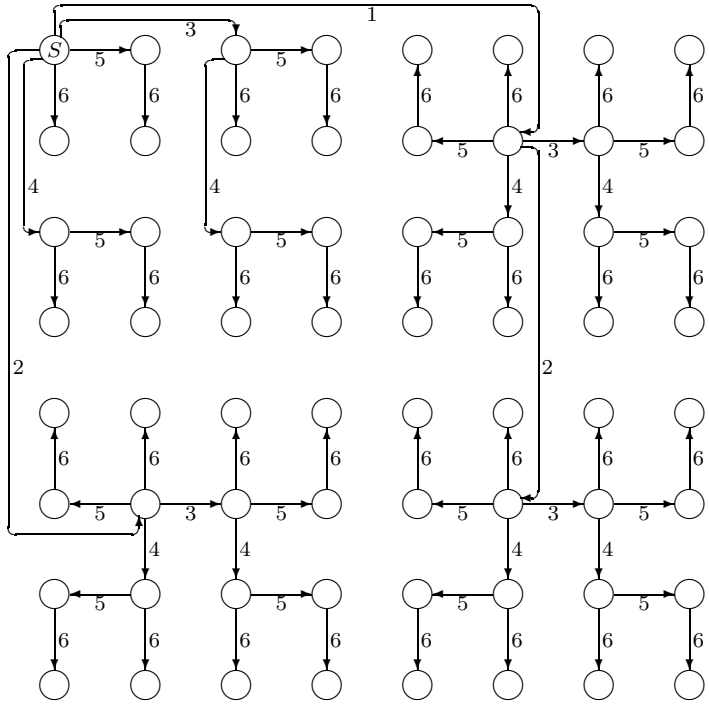


Figure 4.10: Lower cost line broadcasting in 8×8 grid

upper left corner of our grid, and $size = 2^k$ is the number of vertices along the edge.

$CornerBroadcast(i, j, size : Integer)$:
begin
if $size > 4$ **then**

(1) send message from vertex (i, j) to $(i + 1, j + size/2 + 1)$

(2) **simultaneously**:

send message from vertex (i, j) to $(i + size/2 + 1, j + 1)$

send message from vertex $(i+1, j+size/2+1)$ to $(i+size/2+1, j+size/2+1)$

(3) **simultaneously**:

$CornerBroadcast(i, j, size/2)$:

$Broadcast1(i, j + size/2, size/2)$:

$Broadcast1(i + size/2, j, size/2)$:

$Broadcast1(i + size/2, j + size/2, size/2)$:

else

$Broadcast(i, j, size)$;

end

$Broadcast1(i, j, size : Integer)$:

begin

if $size > 4$ **then**

(1) send message from vertex $(i + 1, j + 1)$ to $(i + 1, j + size/2 + 1)$

(2) **simultaneously**:

send message from vertex $(i + 1, j + 1)$ to $(i + size/2 + 1, j + 1)$

send message from vertex $(i+1, j+size/2+1)$ to $(i+size/2+1, j+size/2+1)$

(3) **simultaneously**:

$Broadcast1(i, j, size/2)$:

$Broadcast1(i, j + size/2, size/2)$:

$Broadcast1(i + size/2, j, size/2)$:

$Broadcast1(i + size/2, j + size/2, size/2)$:

else

follow pattern from Figure 4.9

end

This process for an 8×8 grid is presented in Figure 4.10. In this case we have the following calling scheme:

(1) At time 1, vertex $(1, 1)$ (marked as S in the picture) sends the message to vertex $(2, 6)$ (instead of $(1, 5)$ in previous method);
cost: 6 (2 more than in previous method).

(2) At time 2, vertex $(1, 1)$ sends the message to vertex $(6, 2)$, vertex $(2, 6)$ sends the message to vertex $(6, 6)$; cost: $6 + 4$ (2 more than in previous method).

(3) Vertices $(2, 6)$, $(6, 2)$ and $(6, 6)$ are the inside vertices in 4×4 grids; they serve as the sources in optimal line broadcasting;

cost: $3 * 15$ ($3 * 3$ less than in previous method).

(4) Vertex $(1, 1)$ broadcasts in 4×4 grid;
cost: 18.

We observe that, in steps 1 and 2, we "pay" the distance of 4 extra edges. However, in step 3, we "save" the distance of 9 edges. Thus, this approach gives us lower cumulative cost than the simple divide and conquer method.

Generally, for $n = 2^k$, the cost br_k of $Broadcast1$ can be described by the following recursive equations:

$$\begin{cases} br_2 &= 15, \\ br_{k+1} &= 3 * 2^k + 4br_k. \end{cases} \tag{4.2}$$

And the cost cb_k of $CornerBroadcast$ can be described by

$$\begin{cases} cb_2 &= 18, \\ cb_{k+1} &= 2^k + 2(2^k + 2) + cb_k + 3br_k. \end{cases} \tag{4.3}$$

Let us now derive closed form formula for br_k. In procedure $Broadcast1$ we divide our $2^k \times 2^k$ grid into 4 equal sub-grids (steps (1) and (2) of the algorithm). The cost of each of these calls is 2^{k-1}. Then we recursively call $Broadcast1$ for each of the sub-grids. This process ends when all sub-grids are of size 4×4. We call our procedure recursively $k - 2$ times. Then to all 4×4 sub-grids we apply the strategy presented in Figure 4.9.

To calculate the total cost of calls used to divide our initial $2^k \times 2^k$ grid into 4×4 sub-grids we observe that the cost of dividing is:

$3 * 2^{k-1}$	to divide into 4 grids of size $2^{k-1} \times 2^{k-1}$
$4 * 3 * 2^{k-2}$	to divide into 4^2 grids of size $2^{k-2} \times 2^{k-2}$
$4^2 * 3 * 2^{k-3}$	to divide into 4^3 grids of size $2^{k-3} \times 2^{k-3}$
...	
$4^{k-3} * 3 * 2^2$	to divide into 4^{k-2} grids of size $2^2 \times 2^2$

Since in $2^k \times 2^k$ grid ($k \geq 2$) we have 2^{2k-4} sub-grids of size 4×4, the cumulative cost br_k of $Broadcast1$ is

$$\begin{aligned} br_k &= 3 * 2^{k-1} + 4 * 3 * 2^{k-2} + 4^2 * 3 * 2^{k-3} + \ldots + \\ &\quad 4^{k-3} * 3 * 2^2 + 2^{2k-4} * 15 \\ &= 3 * 2^{k-1} + 3 * 2^k + 3 * 2^{k+1} + \ldots + \\ &\quad 3 * 2^{2k-4} + 2^{2k-4} * 15 \\ &= 3 * 2^{k-1}(1 + 2 + 4 + \ldots + 2^{k-3}) + 2^{2k-4} * 15 \\ &= 3 * 2^{k-1}(2^{k-2} - 1) + 2^{2k-4} * 15 \end{aligned}$$

Procedure $CornerBroadcast$ differs from procedure $Broadcast1$, because two of the calls in steps (1) and (2) are of distance $2^{k-1} + 2$ (this adds $4(k-2)$ to the total cost), and in one of the final 4×4 grids we perform divide and conquer algorithm $Broadcast$ (cost is 18) instead of $Broadcast1$ (cost is 15).

Thus, the cumulative cost cb_k for $2^k \times 2^k$ grid is

$$\begin{aligned} cb_k &= 3 * 2^{k-1}(2^{k-2} - 1) + 4(k-2) + (2^{2k-4} - 1) * 15 + 18 \\ &= 3 * 2^{k-1}(2^{k-2} - 1) + 4k + 2^{2k-4} * 15 - 5. \end{aligned}$$

The lower bounds on the synchronous and asynchronous cost of optimal time line broadcasting algorithms from the corner vertex in grid graphs are obvious (the diameter of the graph). The lower bound on cumulative cost is not known.

The case of $n = 2^k$ is easier to analyze, because there is no "idle" time (each informed vertex has to send a message during each time unit). However it is not known if the algorithm presented here is optimal. More work needs to be done in the case when n is not a power of 2.

4.2.4 Lower bound on cumulative cost

Let us now consider the cumulative cost C_n of line broadcasting from a corner vertex in a square grid graph. For any $n \times n$ grid, line broadcasting can be accomplished in $\lceil \log n^2 \rceil = \lceil 2 \log n \rceil$ time units. Because $n^2 - 1$ nodes have to be reached,

$$C_n \geq n^2 - 1.$$

Since the longest path from the source to a node in such grid is $2n-2 \geq \lceil 2\log n \rceil$, $2n-2-\lceil 2\log n \rceil$ extra edges have to be used. Thus, we have that

$$C_n \geq n^2 - 1 + 2n - 2 - \lceil 2\log n \rceil.$$

This lower bound is not tight, because it only accounts for extra edges used to send a message from the source corner to the most distant corner.

We can obtain a better lower bound in the case when $n = 2^k$. In this case, every vertex is busy every time unit after receiving a message.

Each vertex that received a message at time t must send $\lceil 2\log 2^k \rceil - t = 2k - t$ messages. Since each vertex has at most 4 neighbors, if $2k - t \geq 4$, $2k - t - 4$ of those messages must be sent by using calls over distances greater than 1. The source vertex (corner) can only make two calls of length 1.

From the above observations, we can obtain a lower bound on the cumulative cost of line broadcasting in $2^k \times 2^k$ grid. We have that

$$\begin{aligned} C_{2^k} &\geq 2^{2^k} - 1 + (2k-2) + 2(2k-1-4) \\ &\quad +2^2(2k-2-4) + \ldots + 2^{2k-5}(2k-(2k-5)-4) \\ C_{2^k} &\geq 2^{2^k} + 2k - 3 + 2(2k-5) + 2^2(2k-6) + \ldots + 2^{2k-5} \end{aligned}$$

This lower bound also is not tight, because it only takes into account the minimum number of calls of length greater than one that have to be placed, and not the actual lengths of such calls.

4.3 Conclusions

In this chapter we presented different strategies for designing optimal asynchronous cost algorithms for line broadcasting in grid graphs.

Two different approaches for grids with sides that are powers of 2 were presented. These strategies have different cumulative costs.

A method for calculating lower bound on cumulative cost of line broadcasting from the corner vertex in $2^k \times 2^k$ grid was also presented.

Bibliography

[1] Bavelas, A., Communication patterns in task-oriented groups, *J. Acoust. Soc. Amer.* 22 (1950) 725–730.

[2] Brooks, J. A., unpublished term paper, directed research course, West Virginia University, 1989.

[3] Chinn, P., S. T. Hedetniemi, and S. Mitchell, Multiple-message broadcasting in complete graphs, *Proc. Tenth SE Conference on Combinatorics, Graph Theory and Computing*, Utilitas Mathematica, Winnipeg, 1979, 251–260.

[4] Cockayne, E. J., and A. G. Thomason, Optimal multi-message broadcasting in complete graphs, *Utilitas Mathematica* 18 (1980) 181–199.

[5] Farley, A. M., Broadcast time in communication networks, *SIAM J. Appl. Math.* 39 (1980) 385–390.

[6] Farley, A. M. Minimum-time line broadcast networks, *Networks* 10 (1980) 59–70.

[7] Farley, A. M., and S. T. Hedetniemi, Broadcasting in grid graphs, *Proc. 9th S-E Conf. Combinatorics, Graph Theory, and Computing*, 1978, 275–288.

[8] Farley, A., S. Hedetniemi, S. Mitchell, and A. Proskurowski, Note: Minimum broadcast graphs, *Discrete Mathematics* 25 (1979) 189–191.

[9] Fraigniaud, P., and E. Lazard, Methods and problems of communication in usual networks, *Discrete Applied Mathematics* 53 (1994) 79–133.

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

[11] Gentleman, W. M., Some complexity results for matrix computations, *Journal ACM* 25 (1978) 112–115.

[12] Hajnal, A., E. C. Milner, and E. Szemeredi, A cure for the telephone disease, *Canad. Math. Bull.* 15 (1972) 447–450.

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

[14] Kane, J. O., and J. G. Peters, Line broadcasting in cycles, *Discrete Applied Mathematics*, 83 (1998) 207–228.

[15] Klarner, D. A., Mathematical crystal growth 1, *Discrete Applied Mathematics* 3 (1981) 47–52.

[16] Klarner, D. A., Mathematical crystal growth 2, *Discrete Applied Mathematics* 3 (1981) 113–117.

[17] Knodel, W., New gossip and telephones, *Discrete Mathematics 13* (1975) 95.

[18] Ko, C., On a conjecture concerning broadcasting in grid graphs, preliminary report, *Notices American Mathematical Society* 26 (1979) A–196.

[19] Landau, H. G., The distribution of completion times for random communication in a task-oriented group, *Bull. Math. Biophys.* 16 (1954) 187–201.

[20] Peck, G. W., Optimal spreading in an n-dimensional rectilinear grid, *Studies in Applied Mathematics* 62 (1980) 69–74.

[21] Scheuermann, P., and M. Edelberg, Optimal broadcasting in point-to-point computer networks, technical report, Northwestern University, 1981.

[22] Scheuermann, P., and G. Wu, Heuristic algorithms for broadcasting in point-to-point computer networks, *IEEE Transactions on Computers* 33 (1984) 804–811.

[23] Simmen, M., Comments on broadcast algorithms for two-dimensional grids, *Parallel Computing* 17 (1991) 109–112.

[24] Slater, P., E. Cockayne, and S. Hedetniemi, Information dissemination in trees, CS-TR-78-11, Computer Science Department, University of Oregon.

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

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

[27] Van Scoy, F. L., and Jeffrey A. Brooks, Broadcasting multiple messages in a grid, *Discrete Applied Mathematics* 53 (1994) 321–336.

[28] Wojciechowska, I., Broadcasting multiple messages in a grid, technical report TR-94-6, West Virginia University, 1994.

[29] Wojciechowska, I., Line broadcasting in grid graphs, *Congressum Numerantium*, 128 (1997) 129–133.

[30] Wojciechowska, I., and F. L. Van Scoy, A nearly optimal algorithm for broadcasting multiple messages in a two-dimensional grid, technical report TR-95-13, West Virginia University, 1995.

[31] Wojciechowska, I., and F. L. Van Scoy, A nearly optimal algorithm for broadcasting multiple messages in a two-dimensional grid, *Congressum Numerantium*, 118 (1996) 81–96.