k-Port Line Broadcasting in Trees

A. Averbuch * R. Hollander Shabtai † Y. Roditty ‡

Abstract

Broadcasting is the process of message transmission in a communication network. The communication network is modeled by a graph $G = (V, E)$, where the set of vertices V represents the network members and the set of edges E represents the communication links between two given vertices. We assume that G is connected and undirected. One vertex, called the *originator* of the graph holds a message that has to be transmitted to all vertices of the network by placing a series of calls over the network.

A **k- port line broadcasting** in G is a model in which an informed vertex can call, at each time unit, at most k vertices and transmit a message through a path, as long as, two transmissions do not use the same edge at the same time. In case k is not bounded the model is called **all-port line model.**

In this paper, we extend Cohen's work [6], that handles the all-port line model.

1 Introduction

Broadcasting is the process of message transmission in a communication network. The communication network is modeled by a graph $G = (V, E)$, $|V| = n$, where the set of vertices V represents the network members and the set

*School of Computer Sciences Tel Aviv University, Tel Aviv 69978, Israel

†School of Computer Sciences Tel Aviv University, Tel Aviv 69978, Israel

‡School of Computer Sciences, Tel Aviv University, Tel Aviv 69978, Israel and School of Computer Sciences, The Academic College of Tel-Aviv-Yaffo, Tel-Aviv 61161, Israel. email: jr@post.tau.ac.il

JCMCC 77 (2011), pp. 125-160

of edges E represents the communication links between two given vertices. We assume that G is connected and undirected. One vertex, called the *originator* of the graph holds a message that has to be transmitted to all vertices of the network by placing a series of calls over the network. A *k-port broadcasting* is the process in which each vertex transmits the message to at most k of its neighbours at each time unit, where each call requires one time unit. Each call involves only two vertices - the sender and the receiver. In case k is not bounded the model is called *all-port model.* In the k-port model we assume that $k \leq \Delta$ where Δ is the maximal degree in the graph (for $k \geq \Delta$, the problem is equal to the all-port problem). *Line broadcasting* is the process in which a vertex transmits the message to any vertex in the graph through a path in any length in just one time unit. Two calls can not use the same edge during a specific time unit, i.e., the paths used by two calls must be edge disjoint. The cost of a call is the number of edges used by the call, which is the number of edges in the path between the call's transmitter and receiver. Two paths may intersect during a given time unit only in vertices. A *broadcasting scheme* is a specification of which calls are scheduled at each time unit (in any broadcasting model) and which paths are used in each call.

The line model is related to circuit-switched networks, wormhole routing, optical networks, ATM switching and networks supporting connected mode routing protocols.

We count the *total time* and the *cumulative cost* of the broadcasting scheme. The total time of the broadcasting scheme is equal to the number of time units the broadcasting scheme needs to complete the broadcasting. The cumulative cost is the sum of the number of edges used by all calls at each time unit cumulated on all time units.

In this work, we introduce an algorithm for the *k-port line broadcasting* problem in trees. This extends Cohen's algorithm [6], that handles the all-port model.

It is easily observed that the lower bound on the number of time units

needed to broadcast (using the k-port model) in a graph G is $\lceil log_{k+1} n \rceil$, $1 \leq k \leq \Delta + 1$. The possibility to reach this lower bound depends on the tree topology.

In this work we present the exact broadcasting time in stars and in complete trees, and give an upper bound for any tree. For stars the total time of the k-port line broadcasting scheme is $\lceil \log_2 \left(\frac{n-1}{k} + 1\right) \rceil + 1$ and the cumulative cost of the scheme is $2(n-1) - k \lceil \log_2 \left(\frac{n-1}{k} + 1\right) \rceil$. In case of complete tree we have that, the total time of the k-port line broadcasting scheme is $h \lceil log_2(\frac{p}{k} + 1) \rceil$, where h is the tree height and p is the number of children of each non-leaf vertex in the tree. This total time meets the lower bound $O(log_{\Delta+1} n)$ on the broadcasting scheme total time. The cumulative cost of the scheme in a complete tree is $\frac{p^h - 1}{p-1} \left[2p - k \lceil \log_2 \left(\frac{p}{k} + 1\right) \rceil\right]$. The cumulative cost of the scheme is bounded by $2(n-1)$.

The paper is organized as follows: In chapter 2 we summarize the related work. Chapter 3 includes the notations and definitions that are relevant to this work. In chapter 4 we give an optimal k-port line broadcasting scheme in stars and complete trees, and a general algorithm for any tree. We prove the algorithm's correctness, analyze the total time of the algorithm and calculate the total time and its cumulative cost in the star and in the complete tree topologies.

2 Related Work

Broadcasting in communication networks has been investigated in the literature since the early 1950's (see the survey on broadcasting under various models and different topologies [13],[14]). With the growing interest in parallel and telecommunication systems, a vast literature has been devoted to specific group of communication setups on specific network topologies.

The general broadcasting problem under the single-port local model has been shown to be NP-complete, however, if the graph is a tree, finding an optimal broadcasting scheme was shown to be polynomial [18, 20].

Other topologies that have been investigated are complete graph, torus

graph, ring, grid, hypercube, shuffle-exchange and butterfly graph, with a recent generalization of weighted trees in [2].

Analysis of broadcasting in grids was first investigated by Farley and Hedetniemi [10]. Van-Scoy and Brooks [21] extended their result to broadcasting of m messages from a corner of a 2- and 3-dimensional grid. These results were extended to a d-dimensional grid by Roditty and Shoham [19], where they also showed an efficient broadcasting algorithm from any originator in a d-dimensional grid.

The line-broadcasting model is an approximation model of the *wormhole* and *cut-through* communication protocols. It was introduced by Farley [8]. In his work, he studied the problem of line broadcasting in general trees under the edge-disjoint single-port model. He proved that a minimum-time line broadcasting scheme, that runs in $\lceil \log_2 n \rceil$ time units, always exists in any undirected graph by presenting a minimum-time line broadcasting scheme on any given tree. The scheme is generated in polynomial time.

Cohen, Fraigniaud, and Mitjana [5] summarized the known results and proposed new schemes how to achieve minimum-time line broadcasting in trees and directed trees. In the single-port edge-disjoint path model, the result is that every undirected graph (and hence a tree) has a broadcasting time $\lceil \log_2 n \rceil$ ([8]). For directed trees, Cohen, Fraigniaud, and Mitjana [5] showed a $O(n^2)$-time algorithm in which, given any directed tree T rooted at s, returns an optimal broadcasting protocol from s to the vertices of T. In the all-port edge-disjoint model they showed that there is an $O(n \log n)$-time algorithm which, given any directed tree T rooted at s, returns an optimal broadcasting protocol from s to the vertices of T. In the same model, they also presented an $O(n \log n)$-time algorithm which, given any undirected tree T rooted at s, returns an optimal broadcasting protocol from s to T.

A restriction to the general single-port model in which the length of the calls is limited by a given parameter k was proposed by Fujita and Farley [9] and investigated by Gaber in [12]. Gaber introduced an algorithm where a given tree and a parameter k which is a limit of the length of a

call, the algorithm produces an efficient line-broadcasting scheme. This is a generalization of the line model of Farley ([8]) for $k = n - 1$.

When a minimum-time single-port line broadcasting scheme is explored, the question is how to minimize the cumulative cost of such a scheme. The cost of Farley's scheme in [8] is at most $(n-1)\lceil log_2 n \rceil$ where n is the number of vertices in the graph. Following his work, other researchers focused on specific graphs whose structure was known in advance, and presented minimum-time schemes that minimize the cumulative cost.

Kane and Peters [17] determined the value of the minimum cost for a minimum-time line broadcasting in any cycle with n vertices. For $n = 2^k$ they gave an exact value, while for other choices of n an upper bound was presented. In each case the cost is about $\frac{1}{3}$ of Farley's upper bound.

Fujita and Farley [9] discussed minimum-time line broadcasting in paths. The cost of their scheme depends on the position of the originator in the path. They proved that the end-vertex of a path has the greatest cost of any source vertex, and that a minimum-cost line broadcasting scheme from any source vertex in a path P_n has cost that is not more than that from an end vertex and not less than that cost minus $n - 2$.

Averbuch, Roditty and Shoham [3] obtained efficient line broadcasting algorithms in a d-dimensional grid, which produce a linear cost as a function of the number of vertices in the graph.

Averbuch, Gaber and Roditty [1] studied the line broadcasting in complete binary trees. They provided a minimum-time line broadcasting scheme that minimizes the cumulative cost.

Another model that was investigated by Cohen in [6] was the *all-port* model in [6]. Although it clearly yields faster schemes than those derived for the single-port model, it was shown that the decision problem for general graphs is NP-complete (see [7]). Few specific topologies have been investigated. Efficient schemes for trees were given.

3 Notation and Definitions

In this section we present the notations and definitions that are relevant to our work. Graphs here are finite, simple and undirected. To the sequel we denote by G a graph $G = (V, E)$, $|V| = n$ and by $T = (V, E)$ a tree. For other Graph Theoretical definitions we refer to [21].

1. Let $G = (V, E), |V| = n$ be a graph.

   (a) For each $v \epsilon V$, we denote by $d(v)$ the degree of v in G.

   (b) For each $x, y \epsilon V$, $d(x, y)$ denotes the distance between x and y, which is the length of a shortest path between them.

   The next notations and definitions refer to a tree $T = (V, E), |V| = n$, rooted at u.

2. The *height*, h, or h_u of T (related to u) is the number of edges on a path from u to the farthest leaf.

3. For each $v \epsilon V$, $l(v)$ or l is the level of v in T. Observe that $l = d(u, v)$ since there is a unique path between u and v in T.

4. For each $v \epsilon V$, let $\overline{l(v)}$ be the complementary level of v. Then,

$$\overline{l(v)} = \begin{cases} h_u & \text{if } v \text{ is the root} \\ h_u - d(u, v) & \text{otherwise} \end{cases}$$

   For example: $\overline{l(u)} = h, \overline{l(v)} = h - 1$, for any child v of u, v and $\overline{l(v)} = 0$ if v is a farthest leaf from the root u.

5. For each $v \epsilon V$, denote by P_v its parent.

6. Let T_v and $\overline{T}_v$ be the two trees obtained by deleting from T the edge $e = (P_v, v)$ where T_v is the subtree rooted at v.

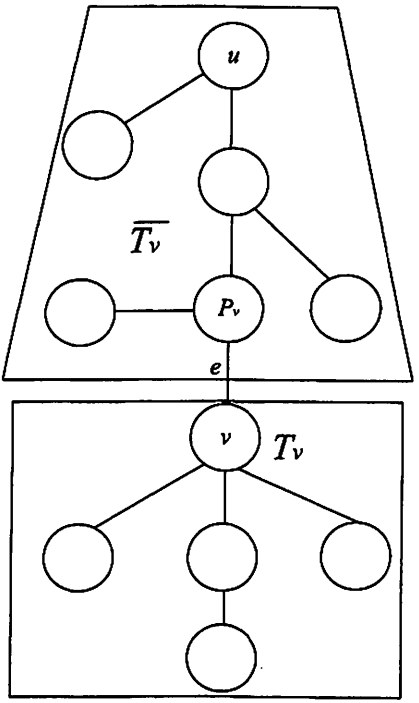

Figure 1: Example for Definition 6.

7. Let $T = (V, E)$ be a tree with $V = \{v_1, v_2, ..., v_n\}$. For $t = \lceil \log_2 n \rceil, n = |V|$ let $M_{t \times n}$ be the matrix where the values of its entries are from the set $\{0, 1, \ldots, k\}$, where k refers to k-port line broadcasting. At the beginning all entries of $M_{t \times n}$ are 0. At each step of the algorithm, each entry $M_{i,j}$, $1 \leq i \leq \lceil \log_2 n \rceil$, $1 \leq j \leq n$, contains the number of vertices that vertex v_j calls at time unit i. For example, $M_{i,j} = p$ iff the vertex v_j broadcasts to $p, 0 \leq p \leq k$, vertices at time unit i.

8. Denote by $\{x \rightarrow y\}$ a call from a vertex x to a vertex y in T.

   Define two types of non-particular calls: $\{S \rightarrow x\}$ is a call from a non-particular informed source S to a particular vertex $x \epsilon V$. $\{x \rightarrow D\}$ is a call from a particular informed vertex x in T to some destination D.

9. $b_T(k)$ is the number of time units needed to complete k-port line broad-

casting in the tree $T = (V, E)$ where $|V| = n$, from any originator.

10. Define a CpT to be a complete tree, where each internal vertex is of degree p.

4 A k-Port Line Broadcasting Algorithm in Trees

In this section, an algorithm for k-Port Line broadcasting in general trees is presented.

4.1 k-Port Line Broadcasting in Trees Algorithm Description

4.1.1 General Description

The algorithm input is the tree T rooted in an arbitrary vertex u called, the *originator*. The output is a k-Port Line broadcasting scheme from u. The broadcasting scheme here is a list of calls for each time unit. Each call specifies the call sender and receiver. We deduce from the scheme the total broadcasting time.

The broadcasting scheme construction is recursive from the leaves to the root u of T. For any vertex v in T, the algorithm constructs a broadcast scheme, A_v, in T_v.

At the beginning of this algorithm all the schemes of all vertices are empty. During the scheme construction, the algorithm adds either particular calls or non-particular calls as to the scheme of the relevant vertex. The algorithm also merges couples of non-particular calls $\{x \rightarrow D\}$ and $\{S \rightarrow v\}$ into one particular call $\{x \rightarrow v\}$.

To simplify the notation, the time units are counted from the end of the broadcast scheme. i.e., time unit 1 is counted first, but in fact it is the last time unit.

The k-port line broadcasting algorithm is divided into two phases:

Phase 1: Scheme construction of the leaves of T. For each leaf v in T the broadcasting scheme A_v is trivial and contains only one non-

particular call $\{S \rightarrow v\}$ at time unit 1 (the last time unit). This is a dummy call that stands for a call where v receives the message from some vertex in $\overline{T}_v$ which is not specified at this step of the construction. The time unit where a leaf v is informed may be changed by the algorithm in future steps.

Phase 2: Scheme construction of the non-leaf vertices of T. Given all schemes A_y for all the leaves of T, the algorithm constructs a broadcasting scheme for all the internal (non-leaf) vertices in T. For each internal vertex v in T, the construction of the scheme is based on all communication schemes A_y where y is a child of v in T_v. That is, assuming that for all vertices y of level of at most $\overline{l(v)}$ (see definition 4) the communication scheme A_y is known, the algorithm constructs a communication scheme A_v where v is at level $\overline{l(v)} + 1$.

The algorithm merges not yet specified calls of two types as defined in definition 8 in section 3:

1. The non-particular calls of type $\{x \rightarrow D\}$ means that x is already informed and x calls a not yet specified vertex D in $\overline{T}_x$ at a specific time unit.
2. Calls of type $\{S \rightarrow x\}$ means that x is called by a not yet specified vertex S in $\overline{T}_x$ at a specific time unit.

4.2 The k-port Line Broadcasting in Trees Scheme Construction

Let $A_v[i]$ be the set of calls of the broadcast scheme A_v at time unit i. Three cases are considered below: v is a leaf, v is an internal vertex ($d(v) \neq 1, v \neq u$) and $v = u$, is the root.

1. **v is a leaf.** Let S_{P_v} be a star rooted at P_v where the leaves are the vertices which are adjacent to P_v in T. The star, S_{P_v}, is a subtree of T. As described above, at the beginning of the algorithm, v receives

the message at time unit $t = 1$. When the algorithm constructs the scheme for P_v, the time unit where v receives the message may change, if $d(P_v) - 1 > k$, and v is informed at time unit t, $1 < t \leq b_{S_{P_v}}(k)$. In this case, the vertex that informs v may be either P_v, one of v's siblings or another vertex from one of v's siblings branches in T.

2. **v is an internal vertex,** i.e., $d(v) \neq 1, v \neq u$. At this step, the broadcasting scheme A_y for every child y of v is known. The algorithm merges the schemes $\{A_y | y \text{ is a child of } v\}$ into the scheme A_v.

   Notice that at this step of the algorithm, the scheme may include non-particular calls of both types: $\{S \rightarrow x\}$ and $\{x \rightarrow D\}$. The calls of type $\{x \rightarrow D\}$ may or may not become specified calls by the end of the algorithm.

   Next, we describe the scheme construction for the internal vertices:

   Let

   $$t = \max_{\text{y is a child of } v} \{\text{number of time units of the broadcast schemes } A_y\}.$$

   Let τ_v be the time unit in which v is informed.

   The algorithm computes the scheme for v by scanning the time units from the last time unit 1, to time unit $max\{\tau_v, t\}$. $\tau_v > t$, means that during the scheme construction of v the algorithm adds at least one time unit to the scheme. This is done only if necessary. Otherwise, $\tau_v \leq t$ and no time unit is added to the scheme at this step.

3. **v is the root u** (i.e., the originator). All the broadcasting schemes A_y, where y is a child of u, are merged into the scheme A_u. A similar process as computed for the internal vertices, is computed for u, except that at each time unit all non-particular calls of type $\{S \rightarrow x\}$ are deleted from A_u. Thus, in the end of the construction there are no uninformed vertices.

4.3 The Procedures of the Algorithm

In this section we introduce the procedures that the algorithm uses. We introduce first the four supporting procedures and then the main procedure of the algorithm.

4.3.1 The Algorithm Supporting Procedures

Following is the pseudo codes and descriptions of the four procedures that the algorithm uses. The procedures can access the stored data in t, in the matrix M and in the sets R_i and Q_i.

- The procedure: **make_calls_from_v**.

  Input: The tree T, the vertex v, current time unit τ_v and k, the k-port constraint.

  Process: The procedure schedules up to k calls from v to its predecessors at time unit τ_v. The vertices that v informs are the vertices r, where a call $\{S \rightarrow r\}$ at time unit $j, j \leq \tau_v$, was scheduled in previous steps of the algorithm. If $j < \tau_v$, i.e., r receives the message before the desired time unit. A non-particular call $\{r \rightarrow D\}$ is added to the scheme for each time unit m, where $j < m < \tau_v - 1$.

  Output: The scheme A_v with additional calls. The procedure also updates the matrix M and deletes vertices from the sets R_i and Q_i.

make_calls_from_v(T,v,τ_v ,k)

1. $i := 1$
2. *// add calls from v to up to k of its children*
   while $(i \leq k)$ and $M[\tau_v, v] < k$ and $(j \neq 0$ where
   $j = \begin{cases} \min\{0 \downarrow 1 j \tau_v | R_j \neq \phi\} & if \bigcup_{j=1}^{\tau_v} R_j \neq \phi \\ 0 & otherwise \end{cases}$) do
3. *// v transmits the message to at most k of its uninformed children.*
   Let r be the first vertex in R_j where if r is not a child of v then the ancestor of r, P_r, is a child of v and follows:$\{v \rightarrow P_r\} \notin A_v[\tau_v]$
4. $A_v[\tau_v] = A_v[\tau_v] \cup \{v \rightarrow r\}$, delete call $\{S \rightarrow r\}$ from A_v
5. inc $M[\tau_v, v]$

6. inc i
7. delete r from R_j
8. *// update the new informed child as a potential sender in later time units.*
   If ($\tau_v > j$) or (r is a leaf) then
9. for $m := \tau_v - 1$ down to j do
10. add r to Q_m and $A_v[m] = A_v[m] \cup \{r \rightarrow D\}$
11. endWhile

- The procedure: **make_calls_from_v_predecessors.**

  Input: The tree T, the vertex v, and the current time unit τ_v.

  Process: The procedure schedules calls from v's informed predecessors that are available to transmit the message at time unit τ_v to v's uninformed predecessors. The informed vertices are r vertices where a call $\{S \rightarrow r\}$ was scheduled at time unit $m, m \leq \tau_v$ in previous steps of the algorithm. If $m < \tau_v$ i.e., r receives the message before the desired time unit, an unparticular call $\{r \rightarrow D\}$ is added to the scheme for each time unit j, where $m < j < \tau_v - 1$.

  Output: A_v with additional calls. The procedure also updates the matrix M and deletes members from the sets R_i and Q_i.

make_calls_from_v_predecessors(T, v, τ_v)

1. for $m := \tau_v$ down to 1 do
2. while ($Q_m \neq \phi$) and ($\bigcup_{n=1}^{m} R_n \neq \phi$) do
3. Let q be the first vertex in Q_m
4. Let r be the first vertex in R_p where $p = \max\{\{1 \leq n \leq m | R_n \neq \phi\}$
5. if $M[\tau_v, v] < k$
6. $A_v[m] = A_v[m] \cup \{q \rightarrow r\}$ delete call $\{S \rightarrow r\}$ and delete call $\{q \rightarrow D\}$ from A_v
7. delete all calls $\{y \rightarrow D\}$ where $t \in T_q$ at time unit m
8. if r is a leaf then
9. for $j := m - 1$ down to p do
10. add r to Q_m
11. $A_v[j] = A_v[j] \cup \{r \rightarrow D\}$
12. endFor

```
13.            inc M[m, q], delete q from Q_m, delete r from R_p
14.          endIf
15.        endWhile
16.  endFor
```

- The procedure: **more_time_units.**

  Input: The tree T, the vertex v, the time unit τ_v, t and $\rho[\tau_v]$.

  Process: The procedure checks first if v is active as a transmitter at time unit τ_v. If v is active, then an additional time unit is needed and the procedure returns true. Otherwise, the procedure checks if at each time unit m, $\tau_v + 1 \leq m \leq t$, there is more than one vertex in T_v that must receive the message from a vertex in $\overline{T}_v$.

  Output: True, when additional time units are needed to complete broadcasting in minimal time and false otherwise.

  more_time_units$(T, v, \tau_v, t, \rho[\tau_v])$

```
1.  not_end = true
2.  if (ρ[τ_v] ≤ 0) and (d(v) ≤ k) and (v ≠ u) then
3.    not_end := false
4.    for m := τ_v + 1 up to t do
5.      if (ρ[m] > 1) then
6.        not_end := true
7.    endFor
8.  endIf
9.  return not_end
```

- The procedure: **inform_v_and_clear_unnecessary_calls**

  Input: The tree T, the vertex v, the time unit τ_v and t as defined in the algorithm.

  Process: The procedure first checks if v receives the message from a vertex in T_v. In this case, a call is scheduled from the closest vertex in T_v to v at time unit τ_v. Otherwise, an unparticular call is scheduled

from an unknown vertex in $\overline{T}_v$ to v at time unit τ_v and the procedure deletes all the unparticular calls from vertices in T_v at time unit τ_v.

The last operation executed by this procedure is to delete calls in time units later than τ_v; the procedure scans all time units which are later than time unit τ_v. At each such time unit where v informs less than k vertices, it can inform one more vertex in $\overline{T}_v$. Therefore, an unparticular call $\{v \rightarrow D\}$ from v to yet unknown vertex is added at the relevant time unit. In such case v may use the edge that connects it to $\overline{T}_v$ and therefore all other calls from vertices in T_v are deleted in order to avoid edge conflicts. If v can't inform a vertex in $\overline{T}_v$, all calls are deleted except for one call from the highest vertex in T_v (this is done to reduce the total cost).

Output: A_v with additional scheduled calls and after the deletion of other calls. The procedure also updates the matrix M.

inform_v_and_clear_unnecessary_calls(T, v, τ_v, t)

1. *// if the time unit for v to receive the message < t then v receives*
   // the message from a vertex in T_v. Then, add a call to v from a vertex in T_v.
   // Otherwise, add an unparticular call to v from a vertex in $\overline{T}_v$
   if $(\tau_v \leq t)$ and $(Q_{\tau_v} \neq \phi)$ and $M[\tau_v, x] < k$ then
2. //there exist a call $\{x \rightarrow D\}$ in $A_v[\tau_v]$
   $A_v[\tau_v] := A_v[\tau_v] \bigcup \{x \rightarrow y\}$ where x is the first element in Q_{τ_v} delete call $\{x \rightarrow D\}$
3. inc $M[\tau_v, x]$
4. else
5. $A_v[\tau_v] := A_v[\tau_v] \bigcup \{S \rightarrow v\}$
6. delete all calls $\{y \rightarrow D\}$ from $A_v[\tau_v]$
7. endif
8. *// if v is available to send the message to a vertex in $\overline{T}_v$,*
   // clear all unparticular calls from v predecessors.
   // Otherwise, delete all unparticular calls in T_v
   //except one unparticular call from v's highest predecessor.
   For each time unit $i := \tau_v - 1$ to 1 do
9. if $(M[i, v] < k)$ then
10. delete all calls $\{y \rightarrow D\}$ where $y \in T_v$ at time unit i
11. $A_v[i] := A_v[i] \bigcup \{v \rightarrow D\}$

12. else

13. delete all calls $\{y \rightarrow D\}$ where $y \in T_v$ at time unit i,
except one call of highest vertex in T_v

14. EndFor

4.3.2 The Main Procedure of the Algorithm

make k-port line broadcasting scheme(T, u, k)

1. Let h be the height of a tree $T = (V, E)$ rooted at u.

2. *// M is a matrix that indicates for each vertex v in T*
// at each time unit $1 \leq i \leq \log|V|$, the number of
// vertices that receive the message from v
Let $M_{\lceil \log|V| \rceil \times |V|}$ be a matrix of integers $(0 \ldots k)$. Initialize each entry in M to 0

3. *//This loop writes for each leaf in T a non-particular call in time unit 1*
For each leaf v in T do

4. $A_v[1] := \{S \rightarrow v\}$

5. End For

6. *// A loop that scans the tree internal vertices level by level from level 1 to the originator children*
For $\overline{l} := 1$ to h do

7. For each vertex v of level $\overline{l}$ do

8. Let t be max number of time units of A_y where $y \in T_v$

9. Let $p = b_{S_v}(k) + a$, *where*

$$a = \begin{cases} 0, & \text{if all v children are leaves} \\ \text{max num of time units in } A_y \text{ of non-leaf } y \in \tau_{T_v} & \text{otherwise} \end{cases}$$

10. Let $R_1, R_2, \ldots, R_p$ be sorted linked lists of predecessors of v,
where $y \in R_i$ iff $\{S \rightarrow y\} \in A_y[i]$.
The vertices in R_i are sorted by the decreasing order of level $\overline{l(v)}$.

11. Let $Q_1, Q_2, \ldots, Q_p$ be sorted linked lists of predecessors of v, where $y \in Q_i$ iff $\{y \rightarrow D\} \in A_y[i]$.
The vertices in Q_i are sorted by the decreasing order of level $\overline{l(v)}$.

12. $\tau_v := 1$, not_end := true

13. *// This is the main loop that constructs the k-Line broadcasting scheme A_v and finds*
// the best time unit for v for receiving the message
while $(\bigcup_{j=1}^{i} R_j \neq \phi)$ and (not_end = true) do

14. $\rho[\tau_v] = |R\tau_v| - |Q\tau_v| + |C|$, where $C = \{\{S \rightarrow x\} | x \epsilon T_v \bigwedge d(x) = 1 \bigwedge \tau_v \neq 1\}$

15. *// add calls from v to up to k of its predecessors*
if ($\rho[\tau_v] \geq 1$) or ($d(v) > k$) then

16. make_calls_from_v (T, v, τ_v)

17. *// add calls from free predecessors of v to other nodes in* T_v
make_calls_from_v_predecessors(T, v, τ_v)

18. *//check if v can be informed at current time unit or more time units are needed*
not_end=more_time_units$(T, v, \tau_v, t, \rho[\tau_v])$

19. if (not_end=true) then

20. inc τ_v

21. endWhile

22. *// if the time unit that receives the message* $< t$, *add a call to v from a vertex in* T_v.
// Otherwise, add an unparticular call to v from a vertex in $\overline{T}_v$.
if ($v \neq u$) then

23. inform _v_and_clear_unnecessary_calls(T, v, τ_v, t)

24. if not_end = false then

25. for $m = \tau_v + 1$ to t do

26. make_calls_from_v_predecessors(T, v, m)

27. EndFor

28. EndFor

// End of pseudo code.

4.4 Example

In this section we demonstrate the k-port line broadcasting in trees for $k = 2$. In table 1 we describe the k-port line broadcasting scheme produced by the algorithm i.e. the list of calls that the algorithm schedules at each time unit i, $1 \leq i \leq t$. In table 2 we describe the values of the matrix M at the end of the algorithm. Each column represents a vertex and each row represents a time unit. The value in each entry $M_{i,j} \leq k$, $1 \leq i \leq t$, $1 \leq j \leq n$, is the number of vertices the vertex v_j calls at time unit i. Table 3 includes the

values of the variables used in the main procedure of the algorithm **Make k-port line broadcasting scheme**. Each column refers to an iteration that handles a specific vertex. Some of the table entries include more than one value. Such entries describe the changes that the algorithm makes to a value of a variable (the values should be read from left to right). When a variable does not get a value in a specific time unit, the relevant entry in the table is empty.

4.4.1 Example: k-Port Line Broadcasting in a Tree

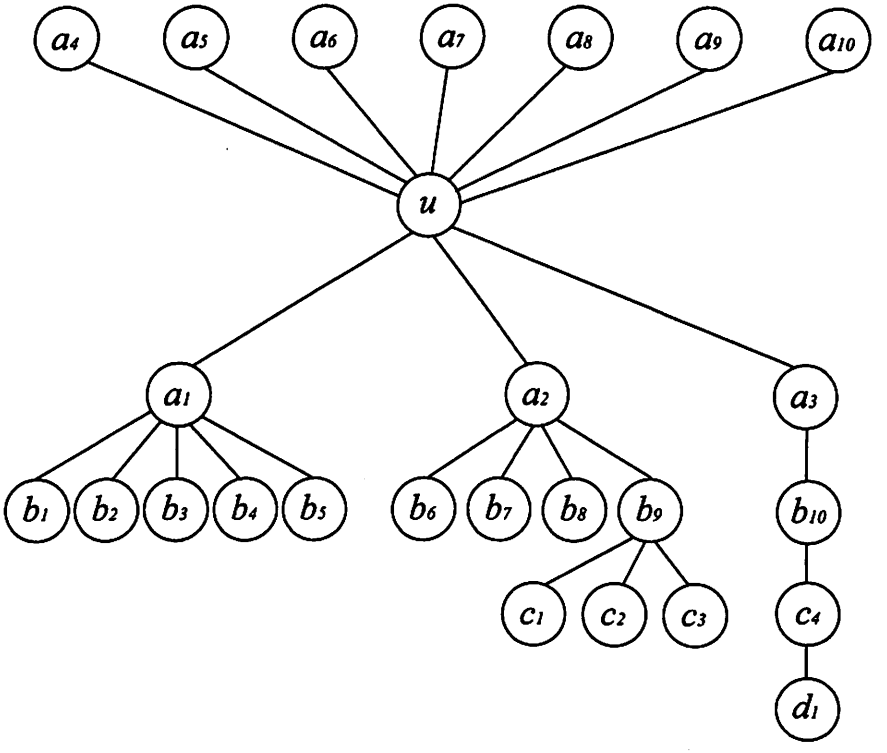

Figure 2: k-Port Line Broadcasting in a General Tree.

In this example, $k = 2$ and the scheme total time is $t = 4$.

t	calls
1	$\{c_4 \to d_1\}\{c_4 \to b_{10}\}\{b_9 \to c_1\}\{b_9 \to c_2\}$ $\{a_2 \to b_6\}\{a_2 \to b_7\}$ $\{a_1 \to b_1\}\{a_1 \to b_2\}\{b_3 \to b_5\}\{c_3 \to b_8\}$ $\{u \to a_3\}\{u \to a_4\}\{a_6 \to a_8\}\{a_7 \to a_9\}$ $\{b_4 \to a_5\}$
2	$\{a_1 \to b_3\}\{a_1 \to b_4\}$ $\{b_9 \to c_3\}\{b_9 \to a_2\}$ $\{u \to a_6\}\{u \to a_7\}$
3	$\{u \to a_{10}\}\{u \to c_4\}$
4	$\{u \to a_1\}\{u \to b_9\}$

Table 1: The k-port line broadcasting scheme

						v									
t	u	$a1$	$a2$	$a3$	$a4$	$a5$	$a6$	$a7$	$a8$	$a9$	$a10$				
1	2	2	2	0	0	0	1	1	0	0	0				
2	2	2	0	0	0	0	0	0	0	0	0				
3	2	0	0	0	0	0	0	0	0	0	0				
4	2	0	0	0	0	0	0	0	0	0	0				
t	$b1$	$b2$	$b3$	$b4$	$b5$	$b6$	$b7$	$b8$	$b9$	$b10$	$c1$	$c2$	$c3$	$c4$	$d1$
1	0	0	1	1	0	0	0	0	2	0	0	0	1	2	0
2	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2: The entries of the matrix M

l	1	2	3	3	3	4
v	c_4	b_9	a_1	a_2	a_3	u
t	1	1	2	3	2	4
p	1	1	2	5	3	7
τ_v	1 2	1 2 3	1 2 3	1 2 3 4	1	1 2 3 4 5 6
not_end	True	True	True	True	True	True
R_1	d_1	$c_1 - c_3$	$b_1 - b_5$	$b_6 - b_8$	ϕ	$a_4 - a_{10}$
Q_1	ϕ	ϕ	ϕ	c_3	ϕ	ϕ
R_2			ϕ	ϕ	ϕ	ϕ
Q_2			ϕ	b_9	ϕ	ϕ
R_3				b_9		
Q_3						
R_4						
Q_4						
$\rho[1]$	1	3	5	2		
$\rho[2]$	0	1				
$\rho[3]$				1		

Table 3: Instance of the variables of the algorithm when T is a general tree.

4.5 Algorithm Correctness

In this section we prove the correctness of the k-port line broadcasting in trees algorithm presented in this section. In lemma 4.1 we prove that all vertices are informed in the k-port line broadcasting scheme produced by the algorithm. In lemma 4.2 we prove that the algorithm respects the paths edge disjoint constraint.

Let T be a tree, u the originator, A_u the broadcasting scheme produced by the algorithm and let

$$t = \max_{\text{v is a child of } u} \{\text{No. of time units of the broadcast schemes } A_v\}.$$

Lemma 4.1. *After t time units the algorithm completes broadcasting in T.*

Proof. Let $V = L \bigcup A$, where, $L = \{v \epsilon V | d(v) = 1\}$, $A = \{v \epsilon V | d(v) > 1\}$.

case 1: $v \epsilon L$. For each leaf v in T, lines 3-5 in the algorithm determines temporarily that v is informed at time unit 1 and writes a non-particular call $\{S \rightarrow v\}$ in $A_v[1]$. In line 10, v is inserted to R_1. In lines 13-21 the main loop of the algorithm constructs the scheme for the internal vertices. We claim that v is informed either by one of its ancestors or by the originator u.

Indeed, the algorithm does not exit from the loop in lines 13-21 until one of the two following conditions is accomplished: The first condition is $\bigcup_{j=1}^{t} R_j = \phi$. If this condition is satisfied then $v \in R_1$. v is deleted from R_1 only after v is informed by u (procedure call in line 16) or by a vertex from T_u (procedure call in line 18). The second condition, where the loop in lines 13-21 ends is not_end=False. Only in the procedure **more time units** and only if $v = u$, not_end is assigned to false.

Thus, in this case the condition is not satisfied and therefore v is informed.

case 2: $v \epsilon A$. For each internal vertex in T, the procedure **inform v and delete unnecessary calls** (described in section 4.3) adds a call that informs v. When there is a non-particular call from a vertex x in T_v, the procedure adds the particular call $\{x \rightarrow v\}$ at time unit τ_v(in line 2) and thus v is informed. Otherwise, the procedure adds a non-particular call $\{S \rightarrow v\}$ at time unit τ_v (in line 5). In the next iterations of the main loop the algorithm constructs the scheme of the ancestors of v. In each iteration until the call $\{S \rightarrow v\}$ is deleted, $v \in R_{\tau_v}$. From this step of the proof, this case is similar to the case where v is a leaf. Thus, by the end of the algorithm either v is informed by a vertex in $\overline{T}_v$ other then the root, or v is informed by the root u.

□

Lemma 4.2. *The algorithm keeps the edge disjoint constraint.*

Proof. We will prove that at each time unit, each edge in T is used by exactly one call. Let $e = (P_v, v)$ be the edge that connects the vertex v to $\overline{T}_v$, where P_v is the parent of v. Let τ_v be the time unit where v is informed.

- **If v is a leaf** then v is the only vertex in T_v. At time unit τ_v the message passes from P_v to v and there are no edge conflicts in e and no other call passes from P_v to v in other time units. At each time unit p, $1 \leq p < \tau_v$, the algorithm adds a single non-particular call $\{v \rightarrow D\}$. At these time units v is already informed and is available to call a single vertex in $\overline{T}_v$. No other call where v is the sender or receiver, is scheduled at other time units and therefore, the algorithm respects the edge disjoint constraint for the edge e.

- **If v is an internal vertex** then v is informed at time unit τ_v. We divide the proof into three parts:

  1. **Time unit τ_v:**

     v is informed by a vertex in T_v:

     If v is informed by a vertex in T_v then the call that informs v does not use the edge e. Moreover, all calls between vertices in T_v which are scheduled by the algorithm at time unit τ_v, does not use the edge e. If there are calls of type $\{x \rightarrow D\}$ in $A_v[\tau_v]$, then it is possible that a vertex in $x \epsilon T_v$ will inform a vertex in $y \epsilon \overline{T}_v$ at time unit τ_v. Line 4 in the procedure **inform v and clear unnecessary calls** verifies that at most one call will be scheduled at time unit τ_v where the sender is in T_v and the receiver is in $\overline{T}_v$. The procedure deletes non-particular calls in T_v, except one call in which the sender is the vertex in the highest level in T_v. In this case, calls of type $\{S \rightarrow x\}$ do not exist in A_v and therefore, no call passes from $\overline{T}_v$ to T_v.

     v is informed by a vertex in $\overline{T}_v$:

The call that informs v uses the edge e. The algorithm decides that v receives the message at time unit τ_v in the procedure **more time units**. In this procedure the condition $\rho[\tau_v] \leq 0$ is respected. This condition means that the number of potential callers in T_v is equal to or greater than the number of uninformed vertices in T_v. When the procedure **more time units** ends, the algorithm calls the procedure **make calls from v predecessors**. In this procedure all uninformed vertices in T_v are informed at time unit τ_v by vertices in T_v. Thus, at time unit τ_v there are no other calls from $\overline{T}_v$ to T_v. Moreover, after scheduling all calls in T_v by the procedure **make calls from v predecessors**, the algorithm deletes all non-particular calls $\{x \rightarrow D\}$ from A_v (line 7 in **inform v and delete unnecessary calls**). Therefore no call from T_v to $\overline{T}_v$ is scheduled or will be scheduled later. Thus, there are no edge conflicts in the edge e.

2. **At each time unit p, $1 \leq p < \tau_v$:** The algorithm exits the loop in line 13 when either $\bigcup_{j=1}^{t} R = \phi$ or not_end $\neq$ true, is respected. If $\bigcup_{j=1}^{t} R = \phi$, then all vertices in T_v are informed by vertices in T_v and therefore, the edge e is not used at these time units by calls from $\overline{T}_v$ to T_v. If not_end $\neq$ true, then conditions $(\rho[\tau_v] \leq 0)$, $(d(v) \leq k)$ and $(v \neq u)$ (in line 2 in the procedure **more time units**) are respected. The algorithm changes not_end to *false* and in later time units, $\tau_v < p \leq t$, $\rho[\tau_v] > 1$. In this case all vertices in T_v are informed by vertices in T_v. Thus in this case, there will be no conflicts on the edge e.

3. **At each time unit p, $\tau_v < p \leq t$:** Let t_m be the max number of time units of A_y where $y \in \tau_{T_v(v)}$. According to the procedure **more time units**, there is at most one call of type $\{S \rightarrow x\}$ where $x \in T_v$, at each time unit p, $\tau_v < p \leq t_m$. Therefore, at most one call can pass from $\overline{T}_v$ to T_v at each of these time units. No call passes from T_v to $\overline{T}_v$ as well. Line 5 in the proce-

dure **more time units** checks that at each of these time units, p, $\rho[p] \leq 0$ and therefore, all vertices in T_v are informed by vertices in T_v.

No call passes from $\overline{T}_v$ to T_v or from T_v to $\overline{T}_v$ at each time unit p, $t_m < p \leq t$.

□

Lemma 4.3. *At each time unit in the broadcasting scheme each informed vertex calls no more that k vertices.*

Proof. It is easily observed that the algorithm adds a call from a vertex v at time unit i iff $M[i,v] < k$. Meaning that v informs less than k vertices at time unit i and therefore at least one more call from v can be added at time unit i. □

4.6 Algorithm Analysis

In the following section we give the lower bound for the broadcasting time on stars and on CpTs. In addition, we show that when the general algorithm described in section 4 is operated on a star or on a CpT, the broadcasting scheme generated by the algorithm meets the optimal broadcasting time for each of these topologies.

4.6.1 A Lower Bound in a Star

Lemma 4.4. *Given a star S of n vertices then*

$$b_S(k) \geq \begin{cases} 1, & k \geq n-1 \\ \lceil \log_2 (\frac{n-1}{k} + 1) \rceil + 1, & otherwise \end{cases}$$

Proof. Assume that the originator is the root. When $k \geq n-1$, the broadcasting completes within 1 time unit. Therefore, we assume $k < n-1$. Observe that the root can transmit the message to at most k vertices at

each time unit and each of the leaves can transmit the message to at most one vertex at each time unit.

If there are m informed vertices after time unit i (including the root), then after time unit $i+1$ there are at most $m+k+m-1 = 2m+k-1$ informed vertices: since the root informs at most k vertices and each of the other $m-1$ informed vertices informs at most one vertex.
Denote by a_i the number of informed vertices after i time units. Thus we have the recurrence relation: $a_0 = 1$, $a_i = 2a_{i-1} + k - 1$ for $i \geq 1$, where its solution is $a_i = (2^i - 1)k + 1$, $0 \leq i \leq b_S(k)$.
Assume that after t time units all vertices are informed. Thus,

$$a_t = (2^t - 1)k + 1 \geq n.$$

Then,

$$b_S(k) = t \geq \left\lceil \log_2 \left(\frac{n-1}{k} + 1 \right) \right\rceil.$$

Note that when the originator is a leaf, broadcasting takes at most one additional time unit, e.g., $\left\lceil \log_2 \left(\frac{n-1}{k} + 1 \right) \right\rceil + 1$, since the leaf informs first the root and than we proceed as before. Notice, when $k = 1$ the leaf may broadcast at the first time unit to any vertex in the star. □

4.6.2 Analysis of the Total Time of the k-Port Line Broadcasting Scheme in Stars

Given a star of n vertices, assume the originator is the star root. The algorithm schedules a call for all $n-1$ star leaves at time unit 1. Then, the algorithm constructs the broadcasting scheme for the originator. The algorithm begins the construction at time unit 1 and adds more time units if necessary. At each time unit, the algorithm calls first the procedure **make calls from v**, which schedules k calls from the root to uninformed leaves. Then, for each informed leaf v the algorithm schedules an open call $\{v \rightarrow D\}$ at each later time unit. Then the algorithm calls the procedure **make calls from v predecessors** which schedules calls from each of the informed vertices to an uninformed vertex, v (a leaf in which there is an open call

$\{S \rightarrow v\}$) at each later time unit. Thus, the algorithm constructs the same broadcasting scheme as the optimal scheme described in section 4 and therefor, the total time of the scheme and the cumulative cost are the same as in the optimal scheme.

4.6.3 Analysis of the Total Cost of the Algorithm in stars

Lemma 4.5. *Given a star with n vertices, the total cost of the k-port broadcasting scheme is:*

$$2(n-1) - k\left\lceil \log_2\left(\frac{n-1}{k}+1\right)\right\rceil$$

Proof. Each transmission from the root to a leaf uses one edge and each transmission from a leaf to another leaf uses two edges. Therefore, at each time unit, i (possibly not including the last one), the cost of all calls from the root is k. The cost of all calls from the informed leaves is at most $2(2^{i-1}-1)k$, where $2 \leq i \leq b_S(k)$ and $(2^{i-1}-1)k$ is the number of informed leaves after time unit $i-1$ (again, possibly not including the last time unit). Then, the total cost for each time unit i is $k+2(2^{i-1}-1)k = k(2^i-1)$. Thus, the total cost of the star broadcasting scheme is at most By substituting the bound of lemma 4.3 we obtain,

$$\begin{aligned}
\sum_{i=1}^{b_S(k)} k(2^i-1) &= k\sum_{i=1}^{b_S(k)} 2^i - kb_S(k) \\
&\geq 2k\left(\frac{2^{\lceil \log_2(\frac{n-1}{k}+1)\rceil}-1}{2-1}\right) - k\left\lceil \log_2\left(\frac{n-1}{k}+1\right)\right\rceil \\
&\geq 2k\left(\frac{n-1}{k}+1-1\right) - k\left\lceil \log_2\left(\frac{n-1}{k}+1\right)\right\rceil \\
&= 2(n-1) - k\left\lceil \log_2\left(\frac{n-1}{k}+1\right)\right\rceil .
\end{aligned}$$

□

4.6.4 A Lower Bound in a Complete Tree

Lemma 4.6. *Given a CpT on n vertices and of height h. Assume the originator is root,u. Then, for $k > 1$,*

$$b_{CpT}(k) \geq \left\lceil \log_2 \left(\frac{p}{k} + 1\right)\right\rceil + (h-1)\left\lceil \log_2 \left(\frac{p}{k+1} + 1\right)\right\rceil + 1$$

where,

$$n = \frac{p^{h+1} - 1}{p - 1}.$$

Notice, that if the originator is not u, then at the first time unit the originator broadcasts to u and as from the second time unit the broadcasting continues as described in the case where the originator is u.

Proof. Let T be a CpT and let u be the root of T. Suppose that the children of each non leaf vertex in T are numbered from 1 to p. For each vertex v in level l, $1 \leq l \leq h$, in T, we define a vector $(i_1, ..., i_l)$ of length l, where $1 \leq i_j \leq p$, $1 \leq j \leq l$. The i_j^{th} coordinate in each vector is the index of the ancestor of v in level j. We denote each vertex v in level l in T by $v(i_1, ..., i_l)$. Similarly, we denote by $T(v(i_1, ..., i_l))$ the branch in T that is rooted at $v(i_1, ..., i_l)$.

The branches of u are $T(j), 1 \leq j \leq p$, and $\overline{T}(j), 1 \leq j \leq p$ are the trees obtained by deleting the edge that connects the branch $T(j)$ to T. Since there is a single edge that connects $T(j)$ to $\overline{T}(j)$, and since the broadcasting model is an edge path disjoint, at each time unit, then each of the branches $T(j)$ can receive only a single message from $\overline{T}(j)$. Therefore, the transmissions from a vertex in $T(j)$ to a vertex in $\overline{T}(j)$ and from u to a vertex in $T(j)$ are the same as in a star rooted at u with p leaves; u is acting as the star root, which can transmit the message to at most k vertices at each time unit. Moreover, in each branch only one vertex can transmit the message to only one vertex in another branch. Suppose, W.L.O.G, that the originator is not in $T(p)$ and that $T(p)$ is the last branch that receives the message of all branches of u. Therefore, the first time unit that a vertex from $T(p)$ receives the message is the minimum time needed to broadcast

a message in a star where the star root is u, that is $\lceil \log_2 (\frac{p}{k} + 1) \rceil$. Similarly, consider for each vertex $v(i_1, ..., i_l)$ in level l, the transmissions from a vertex in $T(v(i_1, ..., i_l))$ to a vertex in $\overline{T}(v(i_1, ..., i_l))$ and from u to a vertex in $T(v(i_1, ..., i_l))$. In this case, there is one edge that connects the branch $T(v(i_1, ..., i_l))$ to $\overline{T}(v(i_1, ..., i_l))$ where a single message can be transmitted to or from $T(v(i_1, ..., i_l))$ at each time unit. Therefore, for each vertex $v(i_1, ..., i_l)$ in level $l, 2 \leq l \leq h$ the transmissions from $T(v(i_1, ..., i_l))$ to $\overline{T}(v(i_1, ..., i_l))$ and vice versa, are the same as in a star composed of $v(i_1, ..., i_l)$ (the star root) and $p + 1$ vertices, acting as the star leaves, as follows, p vertices belong to $v(i_1, ..., i_l)$ branches and one vertex from each of the branches of v and one additional vertex from $\overline{T}(v(i_1, ..., i_l))$. The vertex that belongs to $\overline{T}(v(i_1, ..., i_l))$ is informed and all other vertices that belong to $T(v(i_1, ..., i_l))$ are not informed. That is, before the first time unit there are two informed vertices including the vertex $v(i_1, ..., i_l)$ in that star, and p uninformed vertices.

Denote by a_i the number of informed vertices after i time units. Thus, we have the recurrence relation: $a_0 = 2, a_i = 2a_{i-1} + k - 1$ for $i > 1$, where its solution is:

$$a_i = (2^i - 1)k + 2^i + 1, \ 0 \leq i \leq b_S(k)$$

Let t be the number of time units the algorithm needs to complete broadcasting in a star where the root and one additional vertex are informed at the beginning of the broadcasting. Then,

$$a_t = (2^t - 1)k + 2^t + 1 = p + 2$$

and therefore,

$$t = \left\lceil \log_2 \left(\frac{p}{k+1} + 1 \right) \right\rceil$$

This calculation holds for each j, $2 \leq j \leq h$, that is $(h-1) \left\lceil \log_2 \left(\frac{p}{k+1} + 1 \right) \right\rceil$ time units are needed for $2 \leq j \leq h$. Thus, $b_{CpT}(k)$, the minimum total time for broadcasting in T is

$$\left\lceil \log_2 \left(\frac{p}{k} + 1 \right) \right\rceil + (h-1) \left\lceil \log_2 \left(\frac{p}{k+1} + 1 \right) \right\rceil .$$

The lower bound for $k \geq p+1$ is the tree height h:

$$\left\lceil \log_2 \left(\frac{p}{p+1} + 1 \right) \right\rceil + (h-1) \left\lceil \log_2 \left(\frac{p}{p+2} + 1 \right) \right\rceil = 1 + (h-1) = h.$$

□

4.6.5 Analysis of the Total Time of the k-Port Line Broadcasting Scheme in Complete Trees

Lemma 4.7. *The total time of the scheme generated by the algorithm on a CpT is* $F(T,p,h,k) = h \left\lceil \log_2 \left(\frac{p}{k} + 1 \right) \right\rceil$.

Proof. Let T be a CpT with height h. In the first phase out of the two phases of the general algorithm, an open call is scheduled for all T leaves at time unit 1. Then, the algorithm constructs the scheme for the internal vertices and for the originator u. For each vertex v, the algorithm constructs a star scheme as described in section 4.1; At each time unit, at most k calls are scheduled from v to its uninformed children. A single call is scheduled from each informed child y of v that receives the message at time unit i, to one of its siblings. Such call is scheduled at each time unit $1 \leq j < i$. No calls from v ancestors to v predecessors are scheduled, since the star scheme construction for v completes only when all v children are informed (see procedure **more time units**). Thus, the time unit where a vertex in level $l = 1$ receives the message is: $b_s(k) = \left\lceil \log_2 \left(\frac{p}{k} + 1 \right) \right\rceil$, where S stands for the star composed from the originator and its children. Thus, for each vertex v in level i, where $1 \leq i \leq h$, the time unit in which v receives the message is: $i \left\lceil \log_2 \left(\frac{p}{k} + 1 \right) \right\rceil$, yields that the total time of the k-port broadcasting scheme in T to be: $F(T,p,h,k)$. □

In the following paragraph, we evaluate the total time of the scheme generated by this algorithm on a complete p-tree in relation to $b_{CpT}(k)$, as described in section 4.1.

Lemma 4.8. *The total broadcasting time of the algorithm needs at most* $h-1$ *more time units than the optimal time.*

Proof. The optimal scheme total time is,

$$b_{C_pT}(k) = \left\lceil \log_2 \left(\frac{p}{k}+1\right)\right\rceil + (h-1)\left\lceil \log_2 \left(\frac{p}{k+1}+1\right)\right\rceil .$$

The difference between the total times is derived from the fact that our algorithm does not schedule calls between some vertex in the complete tree to its indirect predecessor. By subtracting the optimal total time from the total time of our algorithm we receive that:

$$F(T,p,h,k) - b_{C_pT}(k) =$$

$$h\left\lceil \log_2 \left(\frac{p}{k}+1\right)\right\rceil - \left\lceil \log_2 \left(\frac{p}{k}+1\right)\right\rceil - (h-1)\left\lceil \log_2 \left(\frac{p}{k+1}+1\right)\right\rceil =$$

$$(h-1)\left[\left\lceil \log_2 \left(\frac{p}{k}+1\right)\right\rceil - \left\lceil \log_2 \left(\frac{p}{k+1}+1\right)\right\rceil\right] \leq$$

$$(h-1)\left\lceil \log_2 \frac{k+1}{k}\right\rceil = h-1$$

□

The algorithm is optimal for star topology and for most of the complete trees. $h \leq log_p n$ and we showed that at most $h-1$ more time unit are needed to complete the broadcasting. Thus, for other complete trees the algorithm is asymptotically optimal in time.

Note that u is the tree root. If u is not the root, the originator u transmits the message to the root at the first time unit. Thus, at most one more time unit is needed to complete the broadcasting. For some k and p the additional time unit is not needed.

Following are a few examples. In each internal vertex we wrote the vertex name. Under the name we wrote the time unit the vertex receives the information and the vertex that sent the message. For example: $2/a_3$ meaning that the vertex receives the information at time unit 2 from vertex a_3.

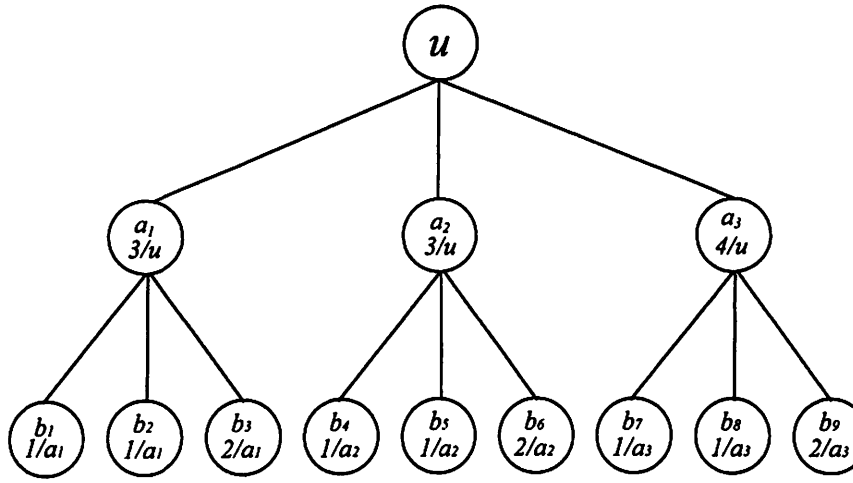

Figure 3: The Algorithm k-Port Line Broadcasting in a Complete Trinary Tree.

The scheme generated by our algorithm for a complete trinary tree of height 2, where $k = 2$. The total time is $t = 4$. The first time unit is time unit 4.

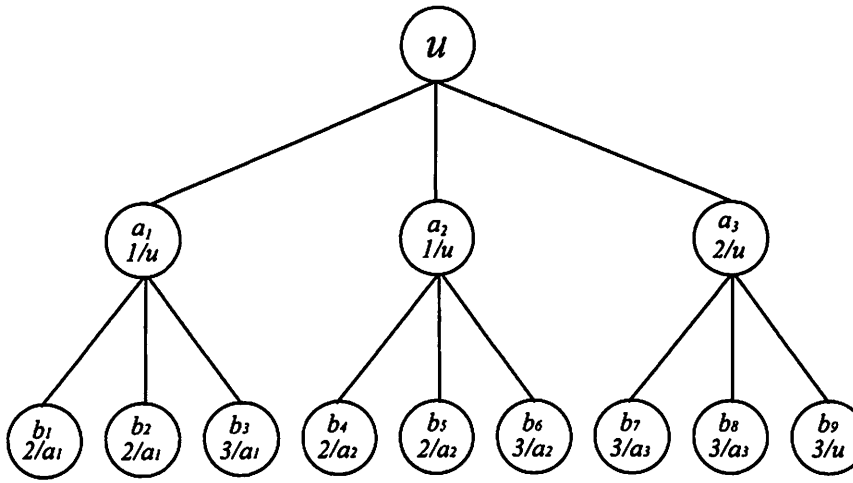

Figure 4: Optimal Scheme of a k-Port Line Broadcasting in a Complete Trinary Tree.

An optimal scheme for the complete trinary tree of height 2, where $k = 2$. The total time is $t = 3$. The first time unit is time unit 1.

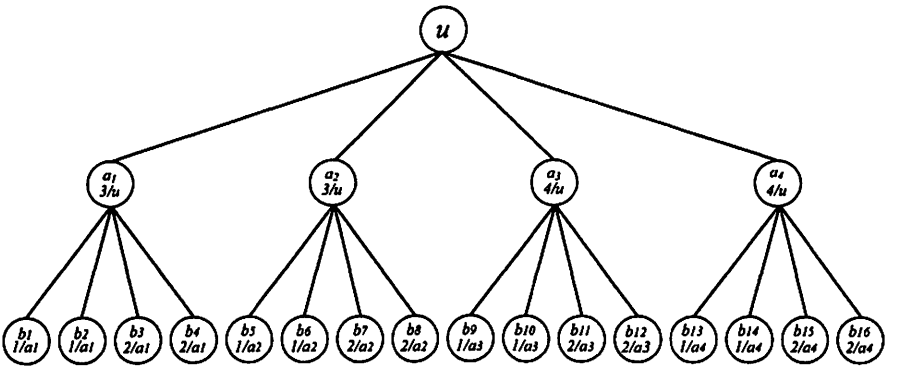

Figure 5: The Algorithm k-Port Line Broadcasting in a Complete 4-Tree.

The scheme generated by our algorithm for the complete 4-tree of height 2, where $k = 2$. The total time of the algorithm is $t = 4$ and meets the optimal time. The first time unit is time unit 4.

4.6.6 Analysis of the Cumulative Cost of the k-Port Line Broadcasting Scheme in Complete Trees

Lemma 4.9. *The cumulative cost of the broadcasting scheme is at most*

$$\frac{p^h - 1}{p - 1}\left[2p - k\left\lceil \log_2\left(\frac{p}{k} + 1\right)\right\rceil\right].$$

Proof. For each internal vertex in a complete tree, the algorithm constructs a star scheme. Therefore, the cumulative cost of all calls scheduled from each internal vertex is the cumulative cost of a star scheme. Each call uses at most 2 edges and there are exactly $n - 1$ calls and therefore, the cumulative cost is bounded by: $2(n-1)$. More precisely, the number of internal vertices in a complete p-tree of hight h is at most:

$$n - p^h = \frac{p^{h+1} - 1}{p - 1} - p^h = \frac{p^h - 1}{p - 1}$$

Using lemma 4.4, the total cost for a star rooted at each internal vertex with $p+1$ vertices the total cost of the scheme is:

$$\frac{p^h-1}{p-1}\left[2p-k\left\lceil\log_2\left(\frac{p}{k}+1\right)\right\rceil\right].$$

□

4.6.7 Analysis of the Algorithm Complexity

In this section an analysis of the algorithm complexity analysis on a tree with n vertices is presented, i.e., an upper bound of the time of the scheme construction is given. The complexity of each of the procedures is written first and followed by the complexity analysis of the algorithm. The total time of the scheme construction is $O(n^3 \log^2 n)$.

In these tables we analyze the algorithms procedures pseudo code lines.

Line	Complexity	Remark
1	$O(1)$	Constant time operations
2	$O(t)$	At most t lists to check, each list in $O(1)$ time
3,4,7	$O(n)$	Insert/delete from a sorted list of size at most n
5-7,8-11	$O(1)$	Constant time operations
Total	$O(tn)$	$t = O(\log n) \Rightarrow total = O(n \log n)$

Table 4: **Make calls from v** procedure complexity analysis

Line	Complexity	Remark
1	$O(t)$	At most t iterations
2	$O(t)$	At most t lists to check, each list in $O(1)$ time
3,7	$O(1)$	Constant time operations
4-6	$O(n)$	Insert/delete from a sorted list size of at most n
8	$O(t)$	At most t lists to check, each list in $O(1)$ time
9,10	$O(1)$	Constant time operations
Total	$O(t^2 n)$	$t = O(\log n) \Rightarrow total = O(n \log^2 n)$

Table 5: **Make calls from v predecessors** procedure complexity analysis

Line	Complexity	Remark
1,3	$O(1)$	Constant time operations
2	$O(n)$	At most n elements to count
4	$O(t)$	At most t iterations
5	$O(n)$	At most n vertices to scan
6-9	$O(1)$	Constant time operations
Total	$O(tn^2)$	$t = O(\log n) \Rightarrow total = O(n^2 \log n)$

Table 6: **More time units** procedure complexity analysis

Line	Complexity	Remark
1-5	$O(1)$	Constant time operations
6	$O(n)$	Deletion from a sorted list of size at most n
8	$O(t)$	At most t iterations
9,11	$O(1)$	Constant time operations
10,13	$O(n)$	Deletion from a sorted list of size at most n
Total	$O(tn)$	$t = O(\log n) \Rightarrow total = O(n \log n)$

Table 7: **Inform v and clear unnecessary calls** procedure complexity analysis

Line	Complexity	Remark
1	$O(n)$	Calculating the tree height
2	$O(n \log n)$	Initialization of $n \log n$ entries
3-5	$O(n)$	A single constant operation on at most n-1 leaves
6-7	$O(n)$	n iterations of both loops
8	$O(n)$	In each call to the function the vertex v calculates the maximum number of time units of the scheme s of $d(v)$ v's branches. The totsl cost of this line in all calls to this function is n-number of leaves in T
9	$O(n)$	At most n vertices to scan
10-11	$O(n \log n + t)$	At most $2p$ lists are allocated in each iteration. There are at most $2n$ vertices in all lists. The vertices will be sorted in $O(n \log n)$ time.
12	$O(1)$	Constant time operations
13	$O(t)$	At most t iterations
14	$O(n)$	At most n vertices to scan
15,17	$O(1)$	Constant time operation
16	$O(n \log n)$	Shown in table above
17	$O(n \log^2 n)$	Shown in table above
18	$O(n^2 \log n)$	Shown in table above
19-22	$O(1)$	Constant time operation
23	$O(n \log n)$	Shown in table above
24	$O(1)$	Constant time operation
25	$O(t)$	At most t iteration
26	$O(n \log^2 n)$	Shown in table above
Total	$O(n^3 \log^2 n)$	

Table 8: **Make k-Port Line Broadcasting Algorithm** complexity analysis

References

[1] A. Averbuch, I. Gaber, Y. Roditty. Low cost minimum-time line broadcasting in complete binary trees. *Networks* 38(2001), 189-193.

[2] A. Averbuch, Y. Roditty, B. Shoham. Computation of broadcasting multiple messages in a positive weighted tree. *J. of Combinatorial Mathematics, Combinatorical Computation* 35(2000),161-184.

[3] A. Averbuch, Y. Roditty, B. Shoham. Efficient line broadcast in a $d-$dimensional grid. *Discrete Applied Mathematics* 113 (2001), 129-141.

[4] A. Averbuch, Y. Roditty, B. Shoham. Construction of minimum time-relaxed broadcasting communication networks. *J. of Combinatorial Mathematics, Combinatorial Computation* 43(2002), 123-134.

[5] J. Cohen, P. Fraigniaud, M. Mitjana. Polynomial time algorithms for minimum-time broadcast in trees. *Theory of Computing Systems*, 35(6), 2002, 641-665.

[6] J. Cohen. Broadcasting, multicasting and gossiping in trees under the all-port line model. *In 10th ACM Symposium on Parallel Algorithms and Architectures*, SPAA 1998, 164-171.

[7] J. Cohen. Communications multipoints dans la modele commute. *PhD thesis, LRI, Universite Paris-Sud, Orsay, France*, 1999.

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

[9] S. Fujita, A. Farley. Minimum-cost line broadcasting in paths. *Discrete Applied Mathematics* 75(1997), 255-268.

[10] A. Farley, S. Hedetniemi. Broadcasting in grid graphs. *In proceedings of the Ninth SE Conference on Combinatorics, Graph Theory and Computing Winnipeg*, 1978, 175-288.

[11] A. Farley, S. Hedetniemi, S. Mitchell, A. Proskurowski. Minimum broadcast graphs. *Discrete Math.* 25(1979), 189-193.

[12] I. Gaber. Minimal-time k-line broadcasting. *SIAM, J. on Descrete Math.* 18(2005), 769-777.

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

[14] P. Fraigniaud and E. Lazard. Methods and Problems of Communication in Usual Networks. *Discrete Applied Mathematics* 53 (1994), p. 79-133.

[15] H.A.Harutyunyan, A.L.Leistman. k-Broadcasting in Trees. *Networks* 38(2001) No. 3, 163-168.

[16] H.A.Harutyunyan, A.L.Leistman. Improved Upper and Lower Bounds for k-Broadcasting. *Networks* 37(2001) No. 2, 94-101.

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

[18] A. Proskurowski. Minimum broadcast trees. *IEEE Transactions on Computers* c-30(1981), 363-366.

[19] Y. Roditty, B. Shoham. On broadcasting multiple messages in a d–dimensional grid. *Discrete Applied Mathematics*, 75(1997), 277-284.

[20] P.J. Slater, E.J. Cockayne, S.T. Hedetniemi. Information dissemination in trees. *SIAM J. Comput.* 10(1981) No. 4, 692-701.

[21] F.L. Van Scoy, J.A. Brooks. Broadcasting multiple messages in a grid. *Discrete Applied Mathematics* 53(1994), 321-336.

[22] Douglas B. West. Introduction to Graph Theory. *Prentice Hall* (1996).