AN OPTIMAL DISTRIBUTED DEPTH-FIRST-SEARCH ALGORITHM

Mohan B. Sharma ‡, Sitharama S. Iyengar ‡

‡ *Department of Computer Science*
Louisiana State University
Baton Rouge, LA 70803.

Narasimha K. Mandyam †

† *Indian Telephone Industries Ltd.*
Bangalore
India.

ABSTRACT

This paper presents a new distributed depth-first-search algorithm with communication and time complexities of O(|V|). The algorithm is shown to use 2|V|-2 messages and 2|V|-2 units of time and is shown to be optimal in time and message.

keywords: distributed system, distributed algorithm, communication graph

1.Introduction

Many applications in distributed systems require traversal of messages through the underlying communication network. The problem of graph traversal has been studied extensively in a sequential context. In this paper, we present algorithm for depth-first-traversal of a graph in a distributed context. The problem of distributed depth-first-search is defined as follows. Consider a communication network. The aim is to equip the set of processors in the network with a control algorithm which will allow a processor P_i in the network to effect a depth-first-traversal through the graph underlying the network, using messages. The output of the algorithm is a depth-first-search (DFS) tree of the communication network kept in a distributed fashion, i.e., at the end of the algorithm, each node will know its neighbors in the DFS tree [4].

Recent papers [1,5,3] by Awerbuch, Lakshmanan and Cidon present improved algorithms for distributed depth-first-search (DDFS) for graphs representing communication networks. Earlier work of Cheung[2] presents a DDFS algorithm with message and time complexities of 2|E| for both, where E is the set of undirected links in the graph. Awerbuch [1] improves the time complexity of Cheung's algorithm and presents an algorithm with time complexity less than 4|V|, requiring 4|E| messages, where V is the set of nodes in the graph. Lakshmanan et.al., in [5] prove lower bounds for message and time complexities and present an algorithm that is time optimal requiring less than 4|E| - (|V| -1) messages. In [3], both the time and communication complexities are improved with the communication cost shown to be less than 3|E| messages and the time complexity being 2|V|.

In this paper, we present a new algorithm for DDFS with time and communication complexities of O(|V|) for both. The algorithm is shown to use exactly 2|V|-2 messages and terminate after 2|V|-2 units of time. This reduction in message complexity is achieved by the effective use of extended message format in the algorithm and it is later shown that the total number of bits used for communication is less than that used in the earlier algorithms.

2.The model

The communication network is represented by the graph G(V,E) where V and E are respectively the sets of vertices and undirected edges. Vertices and edges of the graph denote the nodes and undirected communication links of the communication network. We make the following assumptions for an asynchronous communication network.

1. No two processors in the network share memory.
2. Any message transmitted from a node i to node j is received by node j unaltered, in finite time.
3. Each node has the knowledge of its neighbors to which it is linked in the network and the node from where a message is received.
4. Each node has a distinct name and the names are taken from the integers 1,2,..,|V|. Nodes do not have the knowledge of total number of nodes in the network.

We evaluate the performance of the algorithm with the following complexity measures as described in the earlier papers. The time complexity is the maximum time elapsed from begining to the termination of the algorithm, assuming that the time for delivering a message over a link is atmost one unit of time and time for receiving a message, local processing and sending it over a link being zero. The communication complexity is the total number of messages sent during the execution of the algorithm. Also, we compare the communication complexity of our algorithm with the previous algorithm [3] in terms of total number of bits used in communication. All through this paper, we use the terms communication complexity and message complexity interchangeably.

3. Proposed Solution

In this section, we describe the extended message format and present the algorithm.

3.1 Message Format:

There are two messages namely START and DISCOVER. The DISCOVER message has the header and an appendage of |V| bits atmost. The appendage in the message is a bit array such that the i^{th} bit represents the state of the i^{th} processor. State of a processor with respect to the appendage is either VISITED or UNVISITED. The START message contains the start header and is used as signal to initiate the algorithm.

3.2 Description of the proposed algorithm

In the earlier algorithms of [1,3,5], the basic idea was to select a node as the son at every 'center of activity' and send the 'VISITED' message to the rest of the unvisited neighboring nodes, in parallel. In [1], each node waits for an ACK message to be received from all the neighbors, after sending VISITED message to them. The ACK message of algorithm in [1] is eliminated in [5] and the time complexity is reduced to 2|E| - 2. The algorithm in [3] eliminates the ACK message and the message complexity is reduced to < 3|E| messages with time

complexity of 2|V|. The high communication cost in all these algorithms is due to the use of VISITED message from a node to some of its neighbors.

Our proposed solution eliminates the need for a separate VISITED message and this information is embedded in the message itself. The algorithm operates as follows. A node is given the START message from the external world, to start with. The START message signals the start of the algorithm and the node that receives the START message becomes the root node of DFS.

The root node forms the DISCOVER message with a bit-array of size equal to the maximum of the root i.d.(name) and the largest integer denoting its neighbor, i.e., if k is the largest i.d. of the nodes adjacent the root and r is the root i.d., then max(k, r) bit-array is created. Every bit in the bit-array is initialized to UNVISITED state. The root node selects one of its neighboring nodes as its son, marks its (root) position in the bit-array of the message as VISITED and sends this message over the link to the selected son. It may be recalled that since each node i is identified as integer i, $1 \leq i \leq |V|$, position of each node in the bit-array is unique. Clearly, the contents of the bit-array depict the prevalent state (VISITED/UNVISITED) of nodes in the network, at any instant of time.

Upon receiving the DISCOVER message for the first time, each node i does the following.

i. It marks its position (i^{th} bit) in the bit-array of the received message as VISITED.

ii. It marks the node from where the DISCOVER message was received, as its father in the DFS.

iii. It extends the bit-array to k bits if k > size of the received bit-array, where k is the largest i.d. of its neighbors and initializes the added bits to UNVISITED state.

iv. The node chooses an UNVISITED node as its son from its adjacency list, if one exists, and transmits the DISCOVER message with the updated bit-array to the selected son node. To choose an UNVISITED neighbor, next node in the adjacency list of node i is taken and the state of the chosen node is checked in the received bit-array; if found VISITED, the process is repeated with other nodes in its adjacency list. If all the neighboring nodes are already VISITED, i.e., node i is a leaf node in DFS, the received DISCOVER message with updated bit-array is returned to the father node.

If the DISCOVER message is received from a son node, then only step iv. above is executed. The algorithm terminates when the root node receives the DISCOVER message with all nodes in its adjacency list exhausted in the search.

If the total number of nodes in the network is known *a priori*, then the root node creates a bit-array of size |V|-bits and no other node in the network need extend the bit-array.

4.Complexity Analysis and Proof of Correctness.

In the previous section, we presented the DDFS algorithm employing two messages with the above-said message format. In this section we prove that time and messsage complexities of the algorithm are both O(|V|). We also show that the algorithm is optimal in message complexity and further demonstrate that DDFS has message complexity lower bound of O(|V|). We also establish that the communication complexity of our algorithm is less than that reported in [3], in terms of total number of bits

3.3 Formal description of the algorithm

We present formal description of the proposed algorithm, executed at each node i. The algorithm is described in a Pascal-like language and record structure for the DISCOVER message is used for clarity. Following are the message format and variables used at each node i.

Message:

```
Type message = record
    begin
        HEAD : integer;  /* Identification of discover message  type */
        NLIST []: bit-array;  /* The bit-array apendage */
    end;
```

START: Message used to initiate the algorithm at any node chosen to be the root of DFS.

DISCOVER: Message sent to/ received from a son node.

Variables:

$Adjacent_i$ = Set of node adjacent to node i.

$Father_i$ = Father of node i in DFS.

Son_i = Set of sons of node i in DFS.

$Maxadj_i$ = Node in $Adjacent_i$ with largest i.d.

Source = initialized to -1. Used to mark the root node.

PROCEDURE *INIT*;

```
Begin
    On receiving the START message from external world,
        begin
            Source = i;  /* mark the root node */
            Create bit-array of size max ( Maxadj_i , i);
            Initialize the bit-array to UNVISITED state;
            Form the message record MSG;
            MSG.HEAD = DISCOVER;
```

```
            MSG.NLIST = the bit-array formed;
            MSG.NLIST [i] = VISITED; /* set the node state to visited */
            SEARCH;          /* Start dfs */
        end;

    On receiving the message record MSG  from node j,
    if  MSG.HEAD = DISCOVER then,
        begin
            if  j ∈ Son_i  then  /* return msg from son */
              SEARCH;        /* continue search with other neighboring nodes. */
            else

              begin
                Father_i  = j; /* msg from father node */
                MSG.NLIST [i] = VISITED ; /* mark the state */
                if ( size of MSG.NLIST[] ) < Maxadj_i  then
                  extend MSG.NLIST to the size Maxadj_i and
                  initialize the added bits to UNVISITED state.
                SEARCH;  /* start search for possible son */
              end;

        end;

end ;   /* End of procedure INIT */

PROCEDURE SEARCH;

    Begin
        if for some k,  MSG.NLIST[Adjacent_i  [k] ] = UNVISITED
        then
            begin
                Son_i  = Son_i  ∪ Adjacent_i [k] ;
                send MSG to node Son_i ;
            end
        else
            /* all neighboring nodes are in VISITED state */
```

```
begin
    if Source ≠ i then
        Send MSG to Father_i /* send return */
    else
        /* algorithm terminates. */
        STOP.
end;
end;  /* end of procedure SEARCH */
```

used for communication, despite increasing the message size in our algorithm.

Theorem 4.1.

DDFS has the message complexity lower bound of O(|V|) for |V| > 1.

Proof

This bound is easily seen. It is clear that atleast ONE message must be sent from any node i to any node j in the network, i,j ε {V}, since no memory is shared in the network. Hence atleast |V|-1 messages must be sent over the links so as to communicate to all the |V| nodes atleast once. Hence DDFS has an O(|V|) lower bound in message complexity. □

Theorem 4.2 The proposed algorithm is optimal in communication complexity and uses exactly 2|V| - 2 messages.

Proof

Every node in the network except the root node receives only one DISCOVER message from its father (forward path) and sends one DISCOVER message to its father (return path) . Also it is clear from the algorithm that DISCOVER message is not sent to an already VISITED node in the network. Thus, each of the |V| - 1 nodes (excluding the root) exchanges DISCOVER messages with its father exactly twice. Hence, the total number of messages used in the algorithm is exactly 2|V| - 2. Clearly, the message complexity of the algorithm is O(|V|) and is optimal within a constant. □

Theorem 4.3 The algorithm terminates after 2|V|-2 units of time, if all messages are delivered in one unit of time and is time optimal.

Proof

Total time needed for the algorithm to construct DDFS is the time required to transmit the messages over the links. From theorem 4.2 , total time needed to transmit a total of 2|V|-2 messages is 2|V|-2 units of time if all messages are delivered in atmost one unit of time.

It is shown in [5] that distributed algorithm for DFS has a worst case time complexity of 2|V| - 2 and hence the proposed algorithm is optimal in time. □

We now compare our algorithm with the algorithm in [3] for bit-wise message complexity. For convenience, we call our algorithm as A1 and that in [3] as A2. Let n be the total number of nodes in the network and m be the number of bits transmitted for each message of A2.

Lemma 4.4 The total number of bits used for communication in the algorithm A1 is less than that for the algorithm A2, for n > 1 and m > 4 bits, for a fully connected network.

Proof

In algorithm A1, the number of bits transmitted for the DISCOVER message is atmost (m+n) bits due to the possible n-bit appendage to the message of A2. Total number of messages that are used in A2 is < 3|E| where |E| ≤ n(n-1)/2. Hence, for a fully connected network, total number of bits used for communication is < 3n(n-1) * m/2. In A1, the total number of messages used is 2(n-1) and hence the total number of bits used for communication is atmost 2(n-1) (m+n). We derive the condition for m and n such that

$$2(n-1)(n+m) < 3n(n-1)m/2.$$

$$\text{i.e.} \quad 4(n+m) < 3nm$$

$$\text{i.e.} \quad 4m/(3m-4) < n.$$

For m > 4 , we get n > 1 for the inequality to be satisfied and hence the proof.□

In table 1 we summarise the performance of algorithms of the algorithms for DDFS. However, it should be noted that the unbounded message sizes are used for comparison in the table.

5.Discussion

We have presented a new DDFS algorithm that is optimal in message complexity. The algorithm is shown to use exactly 2|V| - 2 messages with the time complexity of 2|V|-2 units of time. We have shown that the extended message format reduces overall message complexity in terms of total number of bits used for communication. It may be noted from table 1 that although the message complexity of our algorithm is O(|V|), the number of bits per message in our algorithm is more than that in the other algorithms in the table. It is interesting to see that the proposed algorithm performs the same way for both synchronous and asynchronous networks since the 'center of activity' moves sequentially in a deterministic manner.

Table 1

Author	Year	Time Complexity	Communication Complexity
T.Cheung	1983	2|E|	2|E|
B.Awerbuch	1985	< 4|V|	4|E|
K.B.Lakshmanan, N.Meenakshi and K.Thulasiraman	1987	2|V| - 2	< 4|E| -(|V| - 1)
I.Cidon	1988	≤ 2|V|	< 3|E|
S.B.Mohan, M.K.Narasimha and S.S.Iyengar	1988	2|V| - 2	2|V| - 2

References.

[1]. B.Awerbuch, A new distributed depth-first-search algorithm, Inform. Process. Lett., 20, (3), 1985, pp. 147-150.

[2]. T.Cheung, Graph traversal techniques and the maximum flow problem in distributed computation, IEEE Trans. Software Engineering.,Vol. SE-9,(4), 1983, pp. 504-512.

[3]. I.Cidon, Yet another distributed depth-first-search algorithm, Inform. Process. Lett, 26,(6), Jan 1988, pp. 301-305.

[4]. E.Horowitz and S.Sahni, Fundamentals of Computer Algorithms, Computer Science Press Inc, 1984.

[5]. K.B.Lakshmanan, N.Meenakshi and K.Thulasiraman, A time optimal message-efficient distributed algorithm for depth-first-search, Inform. Process. Lett.,25, (2), May 1987, pp. 103-109.