An Optimal Broadcasting Algorithm for de Bruijn Network dBG(d,k)

Ngoc Chi Nguyen and Sungyoung Lee

Computer Engineering Department, Kyung Hee Univerity
1, Seocheon, Giheung, Yongin, Gyeonggi 449-701 KOREA
{ncngoc, sylee}@oslab.khu.ac.kr

Abstract. Recent works have classified de Bruijn graph (dBG) based broadcasting algorithms into local broadcasting and arc-disjoint spanning trees based broadcasting. However, those algorithms can only work in binary dBG. In this paper, we investigate broadcasting in bidirectional dBG for a degree greater than or equal to two. A distributed broadcast algorithm for one-to-all broadcasting in the all port communication is proposed for dBG(d,k)[1].

1 Introduction

Broadcasting is one of the fundamental communication problems of interconnection networks. Some typical broadcasting applications are synchronizing different processors in a distributed computing network, and reconfiguring multiprocessor architecture. Recently, broadcasting problems on dBG have been investigated as local broadcasting[3] and arc-disjoint spanning trees based broadcasting[4][5].

However, the above can only work in a dBG(2,k) networks. Considering this limitation we intend to investigate broadcasting in bidirectional de Bruijn graph with a degree greater than or equal to two. A distributed broadcast algorithm is proposed for dBG(d,k). Our study shows that the maximum time steps to finish broadcast procedure is k regardless of the broadcast originator, time complexity at each node is 0(3d/2), and no overhead happens in the broadcast message.

This paper is organized as follows: background is discussed in section 2, section 3 explains the algorithm and the paper is concluded in section 4.

2 Background

The dBG graph denoted as dBG(d,k)[1] has N=d^k nodes with diameter k and degree 2d. If we represent a node by $d_0d_1...d_{k-2}d_{k-1}$, where $d_j \in 0, 1, ..., (d-1)$, 0≤j≤(k-1), then its neighbor are represented by $d_1...d_{k-2}d_{k-1}p$(L neighbors, by shifting left or L path) and $pd_0d_1...d_{k-2}$(R neighbors, by shifting right or R path), where $p = 0, 1, ..., (d-1)$. Shift string of a node A is a binary string (0 for left shift and 1 for right shift) which represents path from originator to A.

[1] This research was partially supported by ITRC project of Sunmoon University

For simplest broadcasting mechanism, the originator initiates the process by making a "call" to other neighboring vertices in the graph informing them of the message. Subsequently, the informed vertices call their neighboring vertices and the process continues until all vertices in the graph are informed. Basically, this mechanism is like flooding phenomenon. Note that the interval during which a call takes place will be referred to as a time step or simply step. In flooding broadcasting (FB), level of a node A is the number of steps by which a message from originator reaches A (or shortest path length between A and originator).

3 Broadcasting algorithm in dBG(d,k)

By applying FB, we can easily obtain k as the maximum number of steps to finish broadcasting. However, message overhead is very high in FB. Thus, how to reduce message overhead (or letting each informed vertices call its uninformed neighbors only) in FB states the motivation for our algorithm. We assume that each packet sent to the other node must contain originator address, sender's level, a shift string of receiver and all calls take the same amount of time.

There are two cases of message overhead when an informed node A wants to inform node X. Case 1, node X has been informed already. Thus, X must have lower or equal level to A. Case 2, uninformed node X can be informed by nodes B,C,D, which have the same level as A, at the same time. For case 1, we need to compare the shortest-path length between X and A to originator. And X is informed by A if X level is higher than A's level and case 2 not happen. For case 2, we have to define some conditions, based on these conditions only A or B or C or... inform X. The following theorems are proposed for calculating path length.

Theorem 1: *given p is shortest-path length between node a and b, the minimum length of matched strings between a and b is k-p (dBG(d,k)).*

Proof: as shown in [1], there are 3 types for determining shortest path (R,L; RL,LR; R_1LR_2, L_1RL_2). The minimum matched string[2] can be obtained in type R,L among them. And length for this minimum matched string is k-p.

Theorem 2: *path length between node s and d is* $min(2s_j + s_i + d_i, 2s_i + s_j + d_j)$*, where* s_i *and* d_i *are the left indices, and* s_j *and* d_j *are the right indices of matched string in s and d respectively.*

Proof: path length $2s_j + s_i + d_i$, $2s_i + s_j + d_j$ are for case $R_{s_j}L_{s_j+s_i}R_{d_i}$ and $L_{s_i}R_{s_i+s_j}L_{d_j}$ respectively . These cases are the general cases for 3 types presented in [1](ex. if $s_i, s_j, d_i, d_j \neq 0$ then they become type R_1LR_2 and L_1RL_2).

To solve the above two cases of message overhead, a Boolean valued function SPL is proposed. SPL has inputs: originator S, current node P, neighboring node X, current level n (level of P), shift string Q ($q_0q_1q_2...q_{z-1}$, length z≤k) (from S to X through P). Fig. 1a shows SPL algorithm. Step 1,2,3 solve message overhead of case 1. Step 1 is a result of theorem 1. Step 4,5,6 solve case 2 message overhead. In case 2, we have several shortest paths from S to X. One shortest path must be chosen based on the following conditions:

• The shortest path corresponds with the shortest matched string of S and X (step 5).

```
Step 1   Find all matched strings between X and S which have
         length higher or equal k-n-1
Step 2   For each matched string from step 1, calculate path
         length follow theorem 2
Step 3   Find shortest path length from step 2
         IF (shortest path length ≤ n) THEN
            RETURN false
            goto End
         ENDIF
Step 4   Find all matched strings (from step 1) which make
         shortest path length (from step 3)
         IF (only one matched string is found) THEN
            RETURN true
            goto End
         ENDIF
Step 5   Find minimum size matched strings from step 4
         IF (2 matched strings are found) THEN
            b=true
            ELSE b=false
         ENDIF
Step 6   CALL Typeiden for matched string in step 5 and
         save value to a
Step 7   IF (b=true) THEN
            FOR i=0 TO z-1 DO
               IF q_i ≠ q_i+1 THEN a=a-1
               ENDIF
            ENDFOR
            IF (a=0) and (q_0=1) THEN RETURN true
               ELSE RETURN false
            ENDIF
         ELSE
            FOR i=0 TO z-1 DO
               IF q_i ≠ q_i+1 THEN a=a-1
               ENDIF
            ENDFOR
            IF (a=0) THEN RETURN true
               ELSE RETURN false
            ENDIF
         ENDIF
End.
```

a)

```
Input s_i, s_j, d_i, d_j: left index, right index of matched
string in s and d respectively

IF (s_i,s_j,d_i,d_j ≠ 0) THEN   RETURN 2
IF (s_i=0) and (s_j,d_i,d_j ≠ 0) THEN   RETURN 1
IF (s_j=0) and (s_i,d_i,d_j ≠ 0) THEN   RETURN 1
IF (d_i=0) and (s_i,s_j,d_j ≠ 0) THEN   RETURN 1
IF (d_j=0) and (s_i,s_j,d_i ≠ 0) THEN   RETURN 1
IF (s_i=d_j=0) and (s_j,d_i ≠0) THEN   RETURN 0
IF (s_j=d_i=0) and (s_i,d_j ≠0) THEN   RETURN 0
```

b)

```
Input S: originator, P: current node, n: previous level, Q:
current shift string
n=n+1
i=0
WHILE (i<2d)
   M ← Q
   i=i+1
   X ← neighbor of P
   IF (X is right neighbor of P) THEN
     Append 1 to M
     IF (M is one of cases of theorem 3, 4) THEN
        CONTINUE WHILE Loop
     ENDIF
     IF (SPL(S,P,X,n,M)=true) THEN
        Send message to X
     ENDIF
   ELSE
     Append 0 to M
     IF (M is one of cases of theorem 3, 4) THEN
        CONTINUE WHILE Loop
     ENDIF
     IF (SPL(S,P,X,n,M)=true) THEN
        Send message to X
     ENDIF
   ENDIF
ENDWHILE
```

c)

Fig. 1. a)SPL function algorithm; b)Typeiden function algorithm; c)Broadcasting algorithm for dBG(d,k).

• In the case, there exist 2 shortest path from the first condition. Then, shortest path which begin with shifting right is chosen. (step 6)

Step 7 compares shift string Q to the condition gotten from step 5 and 6 to determine whether X should be informed or not.

Example 1: in dBG(3,10), given input S: 0012111001, P: 0111012110, n=7, X: 1110121100, Q=01111100. By applying SPL, we have

Step 1: find all matched strings[2] which have length higher or equal 10-7-1=2. These strings are 11, 111, 1110, 01, 012, 0121, 01211, 110, 1100.

Step 2: path lengths for strings in step 1 are 12, 10, 8, 14, 12, 10, 8, 13, 11.

Step 3: shortest path length is 8.

Step 4: matched string, which make shortest path length 8, are 1110, 01211.

Step 5: minimum size string from step 4 is 1110, b=false.

Step 6:Typeiden(input $s_i = 0, s_j = 6, d_i = 4, d_j = 2$)→returned value: 1,a=1.

Step 7: there are 2 places in Q in which two adjacent bits are different → a=-1 $\neq$0. Consequently, X is an uninformed node (step 3,8>n), but it isn't informed by P (message overhead case 2) due to our priority given in step 5 and 6.

If we apply SPL for all 2d neighbors of one node, then it cost 0(2d) for running our algorithm. The following theorems reduce from 0(2d) to 0(1.5d). Following are some notations used, where T is the previous shifting string.

R↔T: total number of right shift in T > total number of left shift in T

L↔T: total number of left shift in T > total number of right shift in T

Theorem 3: *by shifting RLR/LRL, results are duplicate with shifting R/L.*

Proof: given a node $a_0a_1...a_{n-1}$. By shifting RLR in dBG(d,k), we have $a_0a_1...a_{n-1}\rightarrow \alpha a_0a_1...a_{n-2}\rightarrow a_0a_1...a_{n-2}\beta\rightarrow\gamma a_0a_1...a_{n-2}$, $0\leq\alpha,\beta,\gamma<$d.

Substitute α for $\gamma \rightarrow \gamma a_0a_1...a_{n-2}\equiv\alpha a_0a_1...a_{n-2}$.

By proving similarly for case LRL, theorem 3 is proved.

Theorem 4: *if R↔T/L↔T, results provided by next shift LR/RL are duplicate.*

Proof: assume the beginning node is $a_0a_1...a_{n-1}$. For case R↔T, we have the following cases:

•$T = R_uL_vR_w, T = L_uR_vRw, T = L_uR_v$. By shifting LR, we have shift string $R_1L_1R_2L_2$ or $L_1R_1L_2R_2$, which are not existed for shortest path (as shown in Lemma 1 of [1]).

•$T = R_uL_v$ (u>v). By shifting R u times and L v times respectively, we have

$a_0a_1...a_{n-1}\rightarrow\beta_{u-1}...\beta_1\beta_0a_0a_1...a_{n-u-1} \rightarrow\beta_{u-v-1}...\beta_1\beta_0a_0a_1...a_{n-u-1}\delta_0\delta_1...\delta_{v-1}$

where $0\leq\beta_i,\delta_j <$d, 0≤i<u, 0≤j<v. By shifting LR we have,

$\beta_{u-v-1}...\beta_0a_0...a_{n-u-1}\delta_0...\delta v-1 \rightarrow \beta_{u-v-2}...\beta_0a_0...a_{n-u-1}\delta_0...\delta v \rightarrow$

$\gamma\beta_{u-v-2}...\beta_0a_0...a_{n-u-1}\delta_0...\delta v-1$ (K)

Substitute $\gamma(0\leq \gamma<$d) for $\beta_{u-v-1} \rightarrow$ K is duplicate.

• R=R_u. Shift string R_uLR makes duplicate as shown in theorem 3.

By proving similarly to case L↔T, we prove theorem 4.

As a result, broadcasting algorithm is proposed as shown in fig. 1c.

Theorem 5: *in the worst case, time complexity for our broadcasting algorithm is 0(1.5d).*

Proof: probability for theorem 3 happening is 25%, and for theorem 4 is less than 25%. Therefore, the probability for CONTINUE command (line 10, 18 fig. 1c) happening is 25%. So, theorem 5 is proved.

4 Conclusion

We have presented a distributed broadcasting algorithm for dBG(d,k), which requires k steps to finish broadcasting process, time complexity at each node is 0(3d/2) and no message overhead during broadcasting. Therefore, the algorithm can be considered feasible for broadcasting in the real interconnection network which is built based on de Bruijn graph dBG(d,k).

References

1. Zhen Liu, Ting-Yi Sung, "Routing and Transmitting Problem in de Bruijn Networks" IEEE Trans. on Comp., Vol. 45, Issue 9, Sept. 1996, pp 1056 - 1062.
2. Alfred V. Aho, Margaret J. Corasick, "Efficient String Matching: An Aid to Bibliographic Search", Comm. of the ACM, Vol. 18 Issue 6, June 1975.
3. A.H.Esfahanian, G. Zimmerman, "A Distributed Broadcast Algorithm for Binary De Bruijn networks", IEEE Conf. on Comp. and Comm., March 1988.
4. E.Ganesan, D.K.Pradhan, "Optimal Broadcasting in Binary de Bruijn Networks and Hyper-deBruijn Networks", IEEE Symposium on Parallel Processing, April 1993.
5. S.R.Ohring, D.H.Hondel, "Optimal Fault-Tolerant Communication Algorithms on Product Networks using Spanning Trees", IEEE Symp. on Parallel Processing, 1994.