Edge Loading Algorithms for De Bruijn and Modified De Bruijn Graphs and their Performance Analysis in Lightwave Networks

Bachala Sathyanarayana

ABSTRACT

This paper proposes edge-loading algorithms for the De Bruijn graphs [4,5,8,11] and Modified De Bruijn graphs [10], which are logical topologies for multihop light wave networks. For a given in-degree, out-degree and average number of hops between nodes in a network, the edge loading algorithms for Modified De Bruijn graphs has shown smaller average edge loading, better network utilization than that of De Bruijn graphs. Results for De Bruijn and Modified De Bruijn graphs are computed for different values of 'Δ' (degree) and 'd' (diameter) are presented.

Keywords : Edge loading, regular logical topologies, WDM optical networks, Multihop lightwave networks.

1. INTRODUCTION

This paper is about regular logical topologies in multihop WDM lightwave networks[1,2,3]. A logical topology can be superimposed on a physical topology, viz., a broadcast star, by using many channels at different wavelengths. This technique is called Wavelength Division Multiplexing(WDM) which significantly enhances the network capacity.

De Bruijn graphs are presented as regular logical topologies for multihop lightwave networks [4],[5]. Modified De Bruijn graphs are presented as regular logical topologies for multihop lightwave networks[10], which has shown that the average hop length for Modified De Bruijn graphs is minimum when compared to De Bruijn graphs. In this paper I propose the development of edge loading algorithms in De Bruijn graphs and Modified De Bruijn graphs. Using these algorithms, the various parameters related to edge loading namely, average edge loading, maximum edge loading, average network utilization, edge load distribution, are derived for different values of Δ (degree) and d (diameter) of De Bruijn graphs(G(Δ,d)) and Modified De Bruijn graphs(G*(Δ,d)).

In the following sections an overview of the competing topologies, their edge loading algorithms and their performance measures are presented.

2. LOGICAL TOPOLOGIES

In this section consider the definitions of the two logical topologies viz., De Bruijn graphs and Modified De Bruijn graphs.

2.1 De Bruijn graphs

A De Bruijn graph[4] $G(\Delta,d)$, $\Delta,d \geq 2$, is a directed graph with the set of nodes(0,1,2,... N-1), where $N = \Delta^d$. A De Bruijn garph G(2,3) is shown in fig.1.

The salient properties of De Bruijn graphs G(Δ,d) are given below:

i) There are Δ^d nodes in the graphs

ii) The diameter of the graph is D

iii) The degree of the graph is fixed and equal to Δ

* Associate Professor & HOD, Department of Computer Science & Technology, S.K.University, Anantapur, Andhra Pradesh. E-mail : bachalasatya@yahoo.com

iv) Even under uniform load, the link loading is not uniformly distributed over all the links/edges in the De Bruijn graph

2.2 Modified De Bruijn Graphs

The construction of Modified De Bruijn graphs G*(Δ,d) is mainly based on the De Bruijn graphs [3] and [4]. The network for Modified De Bruijn graph[10] is defined for 'N' nodes, where $N = \Delta^d$ for $\Delta \geq 2$ and $d \geq 2$. A typical representation of Modified De Bruijn graph is given in fig. 2.

On similar lines of the De Bruijn graph, a node in the Modified De Bruijn graph can be represented by a string of 'd' digits. An edge from node 'X ' to node 'Y' can be represented by a string of d+1 digits, the first 'd' digits representing the node 'X 'and the last 'd 'digits representing node' Y'. Similarly, any path in the graph of length 'k' hops can be represented by a string of d+k digits. In the case of self-edges, an edge from self-node 'X' and self-node 'Y' can be represented by a string of d+1 digits, all of which representing the node 'X,' and there exists a self-edge between self-node 'X' and self-edge ' Y 'if and only if and only if $b - a_i = 1$.

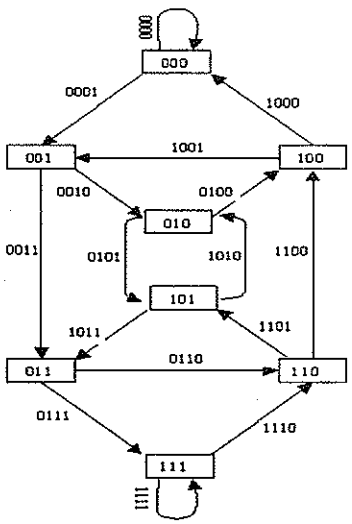

Figure 1. De Bruijn Graph G(2,3)

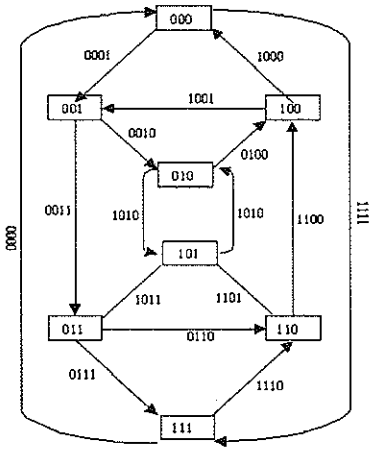

Figure 2. Modified De Bruijn Graph G* (2,3)

3. Algorithm for Edge Loading in De Bruijn graphs (Networks)

3.1. Notations and Definitions of both algorithms

Δ—degree of the network

d —string length of each node label, which denotes the maximum distance in Hops between any two nodes when static routing applied.

$N=\Delta^d$, which is the total number of Nodes in the Network.

Src, Dest — be any source and destination in the network and denoted as $A=(a_1,a_2..a_d)$ and $B=(b_1,b_2,...b_d)$ where a(i's) and b(i's) can take values

Self-node- in which all symbols in address are same

Spath - gives the path between source and destination as a sequence of node labels

Edge (M), M=1,..,ΔN - is an array of M edges in the network. Its label consists of the sequence of slen characters in the bottom node label plus the last character in the top edge label

IsSelf(Node) -> finds whether it is a Self node or not.

DistBothSelfload-> finds the shortest distance in number of hops between two elf-node nodes **PreSelfToDest->** refers to the preceding self node to the destination node.

AdjselfToSrc-> refers to adjacent self node to the source node

DistPreselfToDest-> returns the shortest distance between the PreselfToDest(self node) and destination.

DistOfAdjselfToSrc-> returns the shortest distance between the source and AdjselfToSrc.

DistSrcToAdjSelf -> returns the shortest distance between the source and the immediate adjacent self node

Shiftmatch(i,Src,Dest),0<=i<=slen —> An operation on the two strings Src and Dest to TRUE iff $(b_1,b_2,...,b_{d-i}) = (a_{i+1},a_{i+2},...,a_d)$ and FALSE otherwise.

Merge(i,Src,Dest),0<=i<=d —> is a string(or sequence) of length slen+I given by = $(a_1,a_2,...,a_{d,}\ b_{d-i+1,....},b_d)$

Edgeload(spath,edge) —> finds the edges on reading the **spath**, increments the occurrence each identified edge in the spath.

SelfLoad —> increments the counters of the corresponding edges on the path taken between self nodes.

Transmit next node —> refers to the adjacent node from source node to which the packet has to be transmitted.

FindNormalLength —> finds the distance between source node and the destination node

Input to the algorithm:

For a given Δ and d, the total no of edges = ΔN

Example: Δ=2,d=3

Then N = 8 total Edges=M =2*8=16. Since Δ=2, there 2 symbols namely '0'and '1' are used for labeling the network nodes. Hence the nodes of the network are '000','001','010','011','100', '101','110', '111'. Edges of the network are

Edge(M), M=0,1,..,15 is an array of 16(=2*8) edges in the network.

They form 16 counters corresponding to 16 edges used for determining the loading of each of the edges.

An Edge from two nodes say '000' to '001' is '0001'. So the 16 possible edges can be written as '0000', '0001', '0010', '0011', '0100', '0101', '0110', '0111', '1000', '1001', '1010', '1011', '1100', '1101', '1110', '1111'.

3.2 Algorithm for Edge Loading in G(Δ,d) De Bruijn Networks

```
Procedure DeBruijnEdgeLoad (Src,Dest,Δ,d)
 begin
  for (i=1 to d)
  begin
   if (ShiftMatch(i,Src,Dest) = TRUE)
    break;
  end;
  Length=i;
  Merge(Src,Dest,Length,spath);
  Edgeload(spath,edge);
  end;
 end;

Procedure EdgeLoad(spath,edge)
 begin
   pathlength=length(spath);
   while(TRUE)
```

```
begin
 if(pathLength = d) then
       break;
       ei=0;
       for(j=0;j<=d)
        ei=ei+spath[i-j] * Δ^j);
       edge[ei]=edge[ei]+1;
       pathlength = pathlength -1;
 end;
end;
```

4. Algorithm for Edge Loading in Modified De Bruijn Graphs (Networks)

4.1. Notations

Here the notations used in this algorithm are as presented in section 3.1

4.2 Algorithm for Modified De Bruijn networks (G*(Δ,d)

Procedure ModifiedDeBruijnEdgeLoad (Src, Dest, Δ,d)

```
begin
 if (Isself(Src=FALSE) AND Isself(Dest=FALSE))
then
    Length=BothNonselfLoad(Src,Dest);
       Else if(IsSelf(Src=TRUE) AND
IsSelf(Dest=TRUE))
        then     Length = BothSelfLoad(Src,Dest);
       Else if(IsSelf(Src=TRUE)AND
Isself(Dest=FALSE)) then
                 Length=SourceSelfLoad(Src,Dest);
        else if(Isself(Src=FALSE) AND
IsSelf(Dest=TRUE)) then
                 Length=DestSelfLoad(Src,Dest);
       next_node=FindAdjacent(type,tc);
       Transmit Next node;
       return Length;
end;
```

Procedure EdgeLoad(spath,edge)

```
begin
 pathlength=length(spath);
 while(TRUE)
 begin
  if(pathLength = d) then
       break;
       ei=0;
       for(j=0;j<=d)
       ei=ei+spath[i-j]*(ns^j);
       edge[ei]=edge[ei]+1;
       pathlength = pathlength -1;
  end;
 end.
```

5. Performance measures of Edge loading in G(Δ,D) and G*(Δ,D)

This section presents the definitions of various performance measures which could be computed using the edge loading algorithms (presented in sections (2.2) and (3.2).

5.1 Average Edge Loading ($\bar{L}$)

The loading on edge i is defined as the number of source-destination pairs that use that edge i for communication. The average edge load is defined as the sum of edge load over all edges averaged by total number of edges(M) in the network.

Let M be the number of edges in the network.. The average edge load of a network is denoted by $\bar{L}$ and is given as

$$\bar{L} = \frac{1}{M} \sum_{i=1}^{M} Li \quad \ldots\ldots\ldots\ldots 1$$

For the De Bruijn graph, G(Δ,d) Δ self-loops are present which are not counted as edges. Hence number of nodes

$$N = \Delta^d \quad \ldots\ldots\ldots\ldots 2$$

is the number of edges is

$$2N - \Delta \quad \ldots\ldots\ldots\ldots 3$$

From the eqns. Fig. (2) and (3) the average edge loading in G (Δ,d) is

$$\bar{L} = \frac{1}{2N-\Delta} \sum_{i=1}^{2N-\Delta} Li \quad \ldots\ldots\ldots\ldots 4$$

Now consider the average edge loading in Modified De Bruijn graph,G(Δ,d). Here for G*(Δ,d) the number of edges is 2N (as per the definition G*(Δ,d) in [2]) . On using eqn. (1) and (3) the average edge loading on G*(Δ,d) can be written as

$$\bar{L} = \frac{1}{2N} \sum_{i=1}^{2N} Li \quad \ldots\ldots\ldots\ldots 5$$

A comparison of the average edge loading values for G(Δ,d) and G*(Δ,d) using the edge loading algorithms presented in sections (3.2) and (3.3) respectively is shown in Table (1).

5.2 Maximum Edge Loading (L_{max})

The maximum edge load (L_{max}) is the load on the edge, which has the maximum load. This parameter is computed for G(Δ,d) and G*(Δ,d) using their edge loading algorithms (presented in sections (2.2) and (3.2)). A comparision of the L_{max} values for G(Δ,d) and G*(Δ,d) for different network sizes are presented in Table 1.

5.3 Edge Load Distribution

The edge load distribution is defined as the distribution of loading on all the edges in the network through which N(N-1) source-destination pairs that communicate. This is an important network parameter for determining the network throughput. For instance, comparison of the edge loading distribution as shown in fig (3) and fig Fig. (4) for G(4,5) and G*(4,5).

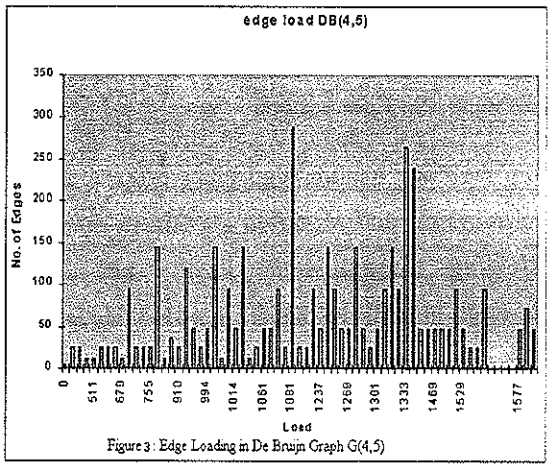

Figure 3 : Edge Loading in De Bruijn Graph G(4,5)

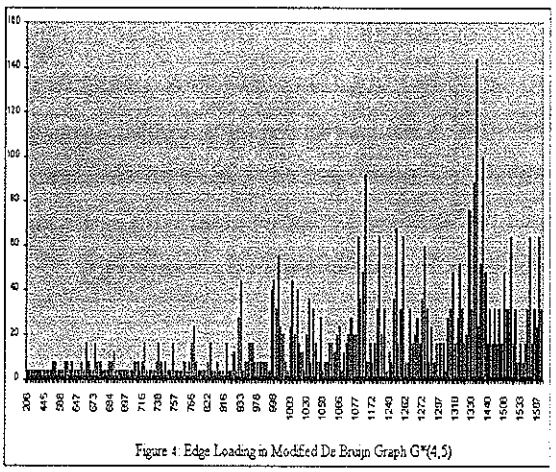
Figure 4 : Edge Loading in Modified De Bruijn Graph G*(4,5)

5.4 Average Network Utilization

The utilization of an edge "i" is defined as the proportion of the loading on edge "i" to an edge with maximum loading (L_{max}) and it can be written as

$$Ui = \frac{Li}{L_{max}} \quad \ldots\ldots\ldots\ldots 6$$

The average network utilization is defined as the edge utilization averaged over all the edges in the network, which is denoted by U_{avg} and is given by

$$U_{avg} = \frac{1}{M} \sum_{i=1}^{M} \frac{Li}{L_{max}} \quad \ldots\ldots\ldots\ldots 7$$

Where M = total number of edges in the graph.

A comparison of the average edge utilization for De Bruijn and Modified De Bruijn networks of different networks is shown in Table (2).

6. Results and Conclusions

From the Edge loading algorithms presented for De Bruijn graphs and Modified De Bruijn graphs, the following conclusions can be drawn: The results in Table-1 exhibited that Modified De Bruijn Graphs have smaller edge loading than De Bruijn graphs. Further it is observed from Table-2, that the average network utilization for Modified De Bruijn graphs is better than De Bruijn graphs. The edge load distribution for G*(4,5) and G(4,5) in Figures 3 & 4 describe more uniform edge load distribution for Modified De Bruijn graphs. Finally it is concluded that a physical topology based on Modified De Bruijn graphs are better logical topologies for multihop light wave networks.

References

[1] Acampora. A.S, "*A multichannel multihop local lightwave network*", GLOBECOM '87 Conf. Rec., pp 1459-1467, Nov.1987.

[2] Acampora.A.S and Karol.M.J,"*An Overview of Lightwave Packet Networks*", IEEE Network Mag., Vol.3, No.1, pp 29-41, Jan. 1989.

[3] Imase.M,Soneoka.T and Okada.K, "*Connectivity of regular directed graphs with small diameter*", IEEE Trans. Comput. C-34 ,267-274,1985.

[4] Kumar.N.Sivarajan and Rajiv Rama Swami, "*Multihop Lightwave Networks Based on De Bruijn Graphs*", IEEE, pp 1001-1011, April 1991.

[5] Kumar.N,Sivarajan,Member,IEEE and Rajiv Rama Swami, Member, IEEE, "*Lightwave Networks Based on De Bruijn Graphs*", IEEE/ACM Transactions on Networking, Vol.2, No.1, Feb.1994, pp 70-79.

[6] Maxemchuk.N.F, "*Routing in the Manhattan Street Network*",IEEE Trans. and Commun,Vol.COM-35, pp 503-512, May 1987.

[7] Mukherjee .B,"*WDM- based local lightwave networks Pat-II: Multihop systems*", IEEE, Network, 6: pp 20-32, July 1992.

[8] Mukherjee. B, "*WDM-Based local lightwave networks, Part I: Singlehop systems*", IEEE Network, 12-27, May 1992 .

[9] Sridhar.M.A," *On the connectivity of the de Bruijn graph*",Information Processing Letters, Vol.27, pp.315-318, May 1988.

[10] .Sathyanarayana.B and Narasimhan.G, "*De Bruijn Graphs as Regular Logical Topologies*", Journal of Computer Science, Karpagram Publication, PP 109-115, Sept-Oct. 2006.

[11] Tanenbaum.A.S,Computer Networks, Prentice-Hall Inc., 1981.

TABLE 1 Comparision of Edge Loading Values for De Bruijn & Modified De Bruijn Graphs as a function of Δ and d for N

NS	Slength	N	De Bruijn		Modified De Bruijn	
			L_{max}	L	L_{max}	L
2	3	8	11	8.426572	8	6.875
2	4	16	29	22.6666	26	20.5
2	5	32	81	56.3871	80	55.5625
2	6	64	208	145.0332	206	141.5156
2	7	128	503	349.4488	499	345.2188
2	8	256	1151	821.3726	1149	816.4141
2	9	512	2789	1892.274	2788	1886.578
2	10	1024	4289.117	4289	6307	4282.679
2	11	2048	14187	9593.829	14184	9586.645
3	2	9	7	5	7	4.333333
3	3	27	31	22.30769	31	21.22222
3	4	81	138	91.425	138	89.88889
3	5	243	535	351.8676	535	349.8642
3	6	729	1945	1294.247	1945	1291.768
3	7	2187	6997	4606.407	6997	4603.449
4	2	16	9	7	9	6.5
4	3	64	57	42.23603	57	41.4375
4	4	256	313	230.3059	313	229.1636
4	5	1024	1589	1173.619	1589	1172.168
5	2	25	11	9	11	8.6
5	3	125	86	68.19355	86	67.56
5	4	625	586	463.2372	586	462.3584
5	5	3125	3711	2937.478	3711	2936.35
6	2	36	13	11	8	10.66667
6	3	216	121	100.1628	121	99.63889
6	4	1296	985	814.1931	985	813.4714

TABLE 2 Comparision of Average Network Utilization Results of De Bruijn & Modified De Bruijn Graphs as a function of Δ and d for N

Δ	d	N	Debruijn	Modified Debruijn
			Uavg	Uavg
2	3	8	0.766234	0.859375
2	4	16	0.781609	0.788462
2	5	32	0.720829	0.694531
2	6	64	0.697268	0.686969
2	7	128	0.69473	0.691821
2	8	256	0.713615	0.710543
2	9	512	0.678477	0.676679
2	10	1024	0.679842	0.679036
2	11	2048	0.676239	0.675878
3	2	9	0.714286	0.619048
3	3	27	0.719603	0.684588
3	4	81	0.662499	0.651369
3	5	243	0.657697	0.653951
3	6	729	0.665424	0.664148
3	7	2187	0.658342	0.657918
4	2	16	0.777778	0.72222
4	3	64	0.74102	0.726974
4	4	256	0.731804	0.732217
4	5	1024	0.73858	0.737671
5	2	25	0.818182	0.781818
5	3	125	0.79295	0.785583
5	4	625	0.790515	0.789015
5	5	3125	0.791579	0.791283
6	2	36	0.846154	0.820513
6	3	216	0.827795	0.823466
6	4	1296	0.826629	0.825883

Author's Biography

Dr. B. Sathyanarayana, working as Associate Professor & Head of Department of Computer Science and Technology with 16 years of experience in Sri Krishnadevaraya University, Anantapur 515003 (AP). He obtained B.Sc with Statisttics as major from Madras Christian College of University of Madras in the year 1985, MCA degree from Thiagarajar College of Engineering of Madurai Kamaraj University in the year 1988. Later he obtained his Ph.D from Sri Krishnadevaraya University, Anantapur in the year 2000. He has presented many valuable papers in National/ International Journals and Conferences. He is supervising for 10 Doctoral theses in the areas of computer networks, mobile computing, Data warehouse management and image processing.