A Statistical Model for the Average Number of Hops in Optical Networks

Claunir Pavan[1,2], Abel R. P. Correia[2], Armando Nolasco Pinto[1,2]

[1]Institute of Telecommunications, Campus Universitário de Santiago, 3810-193 Aveiro, Portugal
Phone: +351 234 377 900, Fax: +351 234 377 901.
[2]University of Aveiro, Department of Electronic, Telecommunications and Informatics, Campus Santiago, 3810-193 Aveiro, Portugal
Phone: +351 234 370 355, Fax: +351 234 381 128,
e-mails: pavan@av.it.pt, a12129@alunos.det.ua.pt, anp@det.ua.pt

***Abstract*—With the aim of approaching the optical networks dimensioning problem in a semi-analytical way, we present a statistical model for the average number of hops in the context of optical networks. The input parameters for the model are the number of nodes and links. From the number of nodes and links the probability function for the average number of hops is obtained. The model uses extensive computational resources to obtain the probability function. Therefore it is also performed an assessment of the problem in terms of computational cost.**

I. Introduction

The dimensioning of optical networks is frequently addressed using extensive numerical processing. We aim to provide semi-analytical tools for the dimensioning problem. We expect to obtain solutions with less time consumption and to gain a deep insight on the dimensioning problem. This approach was firstly, as far as we know, introduced in the context of optical networks by Korotky in [1]. This paper intends to be a contribution in that way.

A telecommunication network must be dimensioned in order to support a given traffic demand, or traffic demand expectation, minimizing the expenses with capital equipment (CAPEX), network operation (OPEX) and network management (MANEX). This is frequently a difficult task due to the size and complexity of the network. Furthermore, in the dimensioning problem frequently some decisions must be taken without a complete knowledge of all the variables involved. This work also addresses this last aspect, taking decisions with incomplete information. In our case we consider that we do not know the network topology but we only know the number of nodes N and links L. Therefore the average number of hops is an unknown variable. However, as we are going to show, a statistical model can be built for the average number of hops.

Given the number of nodes and links, the network topology can be arranged on several ways. For each topology the average number of hops can be computed. In this paper we present both a numerical solution for the problem and a computational time consumption model.

The complexity of an algorithm is measured in running time, storage, or whatever units are relevant[2]. In our case the relevant aspect is the running time, thus we expect to obtain an estimation for the running time as function of the number of nodes and links, our input parameters. We used an algorithms analysis based on an asymptotical approach.

This work was partially supported by the Portuguese Scientific Foundation FCT, through the "IP over WDM Networks" project (POSI/EEA-CPS/59566/2004), FEDER and POSI programs.

In this paper, we present a numerical solution to solve three problems. Given an N number of nodes and L links, we calculate the number of connected graphs, i.e. all possible networks with N nodes and L links where it is possible to define a path between each node pairs. For each connected graph we calculate the average number of hops required to interconnect all nodes. We also calculate the probability function for the average number of hops.

In section II, we present a model for an optical mesh network and clearly formulated our problem. In sections III and IV, we present a solution and a model for the computational consumption time. Finally we finish this work with a discussion of the main results, section V.

II. Optical Mesh Network

A network is formed by a set of nodes and a set of links, where the nodes are connected through the links. A network topology can be modeled as a graph $G(N, L)$, where N is the number of nodes and L is the number of links. A graph can be represented by an NxN adjacency matrix [g]. The matrix elements g_{ij} are either 1 or 0, respectively, when a pair of nodes is directly connected or not. The summation of all the values of [g] yields the number of one-way links, L_1, which is twice the number of two-ways links, $L = L_1/2$. In this work we assume full-duplex links, therefore the [g] matrix is always symmetrical. For a connected graph it is necessary at least $N - 1$ links and the maximum possible number of links is $L_{max} = (\frac{N^2-N}{2})$.

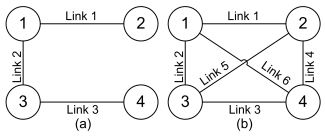

Fig. 1: A four nodes network is represented, in (a) with a minimum number of links and in (b) with a maximum number of links.

The number of hops between a pair of nodes is defined as the number of links traversed by a demand between the node pair, this number is dependent of the routing discipline. In this work we assume a minimum hops routing discipline. Dijkstra's algorithm allows us to determine the minimum number of hops. From each topology matrix [g] we obtain a [s] matrix. Each element of [s], s_{ij}, is the minimum number of links used to interconnect the nodes i and j. Having the [s] matrix, that represents the minimum number of hops between each node pairs (i, j) we can obtain the mean number of hops,

$$\langle s \rangle = \frac{1}{2L_{max}} \sum_{i=1}^{N} \sum_{j=1}^{N} s_{ij}. \quad (1)$$

Let's consider the example presented in figure 1(a). For this topology the following [s] matrix is obtained,

$$[s] = \begin{bmatrix} 0 & 1 & 1 & 2 \\ 1 & 0 & 2 & 3 \\ 1 & 2 & 0 & 1 \\ 2 & 3 & 1 & 0 \end{bmatrix},$$

and using (1) we obtain $\langle s \rangle = 1.67$.

For a given N and L and using the general formula for calculating combinations without repetition, the number of all possible graphs is given by

$$C = \frac{L_{max}!}{L!\,(L_{max} - L)!}. \quad (2)$$

For the case of $N = 4$ and $L = 3$ we have 20 possible topologies, but there are 4 invalid because they are disconnected graphs, see figure 2.

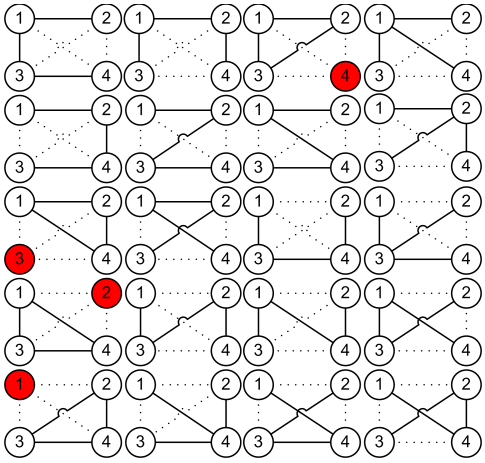

Fig. 2: All possible ways to arrange four nodes and three links, topologies 3, 9, 13 and 17 are not valid - disconnected graphs.

Performing (1) to all valid topologies, which are 16, we obtain only two values for $\langle s \rangle$, which is quite a surprising result, ($\langle s \rangle = 1.5$ and $\langle s \rangle = 1.67$), as we can see on figure (3). The probability of $\langle s \rangle$, $P_{\langle s \rangle}$, is shown in figure (4).

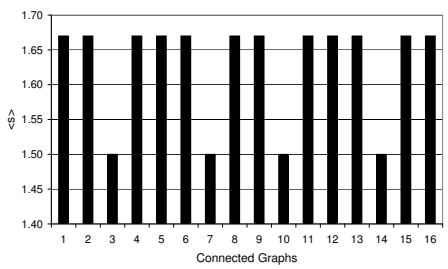

Fig. 3: In the xx´ axis are represented the 16 valid graph (see Fig. 2), counted row by row from the upper left corner to the bottom right corner, and in the yy´ axis are represented the average number of hops for each valid graph.

In conclusion our problem consists in the determination the probability function of $\langle s \rangle$, $P_{\langle s \rangle}$.

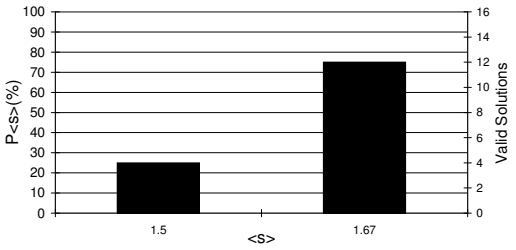

Fig. 4: Probability of $\langle s \rangle$, for the example considered in Fig. 2

III. Numerical Solution and Time Consuming Model

In order to solve numerically the proposed problem and to forecast how much computational time is required, we wrote a program in C++ and compiled it using the GNU C compiler. To estimate the processing time, we first estimate the number of cycles the algorithm performs.

The program has basically three main parts, the first is a combinatorial function which determines all the combinations with N nodes and L links, next is the validation of the combination process which consists on verifying whether or not all nodes are considered. The third part is a function that find out the shortest path - in number of hops - where we can also verify whether the graph is connected or not.

In the context of algorithms for computers, the symbol $\Omega(Omega)$ is used to express the thought that a certain calculation takes at least so-and-so long to do, and the symbol $O(Big - Oh)$ is used to describe an asymptotic upper bound for the magnitude of a function[2]. To generate all the combinations the program uses a bit vector with the STL (Standard Template Library)[3] iterators classes that runs in $\Omega(1)$ when only one bit is changed and $O(L)$ otherwise. A function starts with a vector of size L_{max} and each position is represented by an ordered pair i, j of nodes where a link can be activated or not. A bit vector is also defined with size L_{max} and its positions hold 1 or 0 consonant the link is activated or not, each combination of the bit vector represents a network and considering (2) the lower and upper bound complexities to all combinations are respectively $\Omega(C)$ and $O(C \times L)$.

Some networks are invalid because at least one node is not considered in the combination, see figure 2, therefore, the program also performs a binary search on each combination and the complexity of this part is $\Omega(L)$, when the nodes are in sequence and in the beginning of the combination vector, or $O(N \times log(L \times 2))$ in the worst case.

Note that the presence of all the nodes does not mean a connected graph, combinations that generate forests should also be discarded, see figure 5.

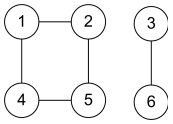

Fig. 5: An invalid topology - forest.

To detect the cases showed on figure 5 the program verifies whether all the nodes are reachable, for this, we use the Dijkstra's algorithm, which given a graph finds out the shortest path between the nodes. The use of Fibonacci heaps improve the asymptotic running time of Dijkstra's algorithm for computing shortest paths in a graph, F-Heap is a heap (priority queue) data structure similar to a binomial heap but with a better amortized running time [4]. It runs in $\Omega(\frac{N^2-N}{2})$ when the graph is complete and in $O(L + Nlog(N))$ in the worst case.

We have implemented the Dijkstra's algorithm with F-heaps and the function returns a boolean variable indicating whether the graph is connected or not. However, there are a function that runs previously to prepare the data for the Dijkstra's algorithm function, the complexity of this function is $O(L \times 2)$, then, using the costs presented previously, we can show the upper bound cost to the number of valid combinations as

$$Tc_{up} \approx C \times [L + Nlog(2L) + 2L + (L + N)log(N)], \quad (3)$$

which may also be written as

$$Tc_{up} \approx C \times [3L + Nlog(2L) + (L + N)log(N)]. \quad (4)$$

The lower cost can be written as

$$Tc_{lw} \approx C \times \left[L + (2L) + \left(\frac{N^2 - N}{2}\right)\right], \quad (5)$$

which may be rewritten as

$$Tc_{lw} \approx C \times [3L + L_{max}]. \quad (6)$$

In order to measure the algorithm efficiency, we frequently make use of the time the program takes to process something in function of the input data [2]. For our tests, we run a benchmark program to obtain the number of operations floating point per second of a single-CPU Pentium-IV, 3.2 GHz, with 1024 MB of physical memory and Linux operational system with kernel 2.6.8-2-386. The value obtained was 3149000 flops. We consider that the time required to perform each instruction of the algorithm is equal to the time required to perform a floating point operation. Therefore, to obtain the cost in seconds, we divided the number of operations by the number of floating point operations performed per second.

Table I: Results for $N = 9$ nodes and $L = 18$ links

$\langle s \rangle$	Connected graphs	$P_{\langle s \rangle}$
1.56	98.342.151	0.037722
1.58	991.693.929	0.380398
1.61	455.270.760	0.174634
1.64	976.117.744	0.374423
1.67	42.571.872	0.016330
1.69	30.602.880	0.011739
1.72	5.656.140	0.002170
1.75	2.031.120	0.000779
1.78	3.479.112	0.001335
1.81	45.360	0.000017
1.83	1.058.400	0.000406
1.86	7.560	0.000003
1.89	113.400	0.000043
1.94	3.024	0.000001

IV. Experimental Tests and Results

Initially we defined a set of three networks, with $N = \{7, 8, 9\}$ nodes respectively. For each network we considered all possible values for L, note that $N - 1 \leq L \leq L_{max}$, which for our samples yields 67 experiments. In order to obtain the time spent to process each experiment we used the function getrusage() of the sys/time.h library. Figures 6 and 7 show some of the results obtained. In table I we present the results obtained for the case of $N = 9$ nodes and $L = 18$ links.

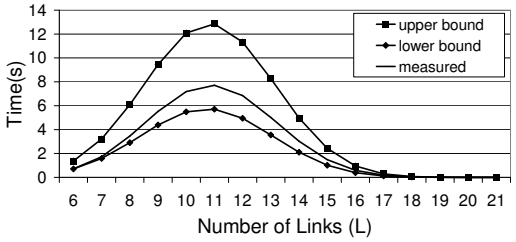

Fig. 6: Comparative lower bound, upper bound and measured time for all networks with $N = 7$ nodes.

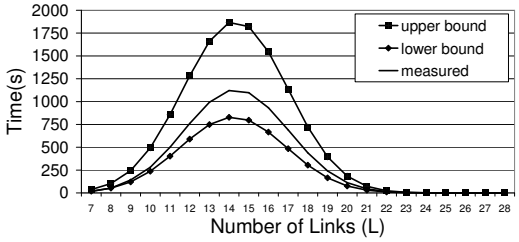

Fig. 7: Comparative lower bound, upper bound and measured time for all networks with $N = 8$ nodes.

V. Conclusions

In this paper, we presented a numerical solution for the problem of the average number of hops in the context of optical networks. It was also presented a model for the computational time required to solve the problem. We found for sparse and dense graphs the problem is easily computable, graphs with mean nodal degree in range of 3 and 4.5 increase the number of combinations strongly and demands a lot of computational time. In future works we intend to introduce constraints to the problem (to eliminate the isomorph graphs, to indicate a minimum nodal degree and/or maximum diameter) with the aim of decrease the number of graphs to be submitted to the functions and consequently the process time. However, here, the experimental results are between the bounds determined using the theoretical model, showing a good agreement between experimental results and theoretical predictions.

References

[1] S. K. Korotky, "Network Global Expectation Model: A Statistical Formalism for Quickly Quantifying Network Needs and Costs", *IEEE/OSA Journal of Lightwave Technology*, vol.22, NO. 3, March 2004.

[2] T. H. Cormen, C. E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms*, 2.ed. Cambridge: MIT Press and McGraw-Hill, 2001.

[3] R. O. Robson, *LinkUsing the STL: the C++ standard template library*. New York: Springer, 1998.

[4] M. L. Fredman, R. E. Tarjan, "Fibonacci Heaps and their uses in Improved Network Optimization Algorithms", *JACM Journal of the ACM* vol.34 NO.3, pp. 596-615, July, 1987.