University of Castilla-La Mancha

A publication of the

Department of Computer Science

Evaluating the Effectiveness of Traffic Balancing Algorithms

by

J.E. Villalobos, J.L. Sánchez, J.A. Gámez, J.C. Sancho, A. Robles

Technical Report **#DIAB-04-06-1** May 2004

DEPARTAMENTO DE INFORMÁTICA
ESCUELA POLITÉCNICA SUPERIOR
UNIVERSIDAD DE CASTILLA-LA MANCHA
Campus Universitario s/n
Albacete - 02071 - Spain
Phone +34.967.599200, Fax +34.967.599224

A Methodology to Evaluate the Effectiveness of Traffic Balancing Algorithms [*]

J.E. Villalobos, J.L. Sánchez, J.A. Gámez

Dept. de Informática
Escuela Politécnica Superior
Universidad de Castilla-La Mancha
02071- Albacete, Spain
{jvillalobos, jsanchez, jgamez}@info-ab.uclm.es

J.C. Sancho, A. Robles

Dept. de Informática de
Sistemas y Computadores
Universidad Politécnica de Valencia
46071- Valencia, Spain
{jcsancho, arobles}@gap.upv.es

Abstract

Traffic balancing algorithms represent a cost-effective alternative to balance traffic in high performance interconnection networks. The importance of these algorithms is increasing since most of the current network technologies for clusters are either based on source routing or use deterministic routing. In source-routed networks, the host is responsible for selecting the suitable path among the set of paths provided by the routing algorithm. The selection of an optimal path that maximizes the channel utilization is not trivial because of the huge amount of combinations. Traffic balancing algorithms are based on heuristics in order to find an optimal solution. In this paper, we propose a new methodology based on the use of metaheuristic algorithms to evaluate the effectiveness of traffic balancing algorithms. Preliminary results show that the set of paths provided by current traffic balancing algorithms are still far from an optimized solution. Thus, it is worth continuing to design more efficient traffic balancing algorithms.

1 Introduction

Clusters of PCs represent a cost-effective alternative to parallel computers. The use of high performance switch-based interconnects, such as Myrinet [2], Gigabit Ethernet [18], and InfiniBand

[*]This work was partly supported by the Spanish CICYT under Grant TIC2003-08154-C06 and Junta de Comunidades de Castilla La-Mancha under Grants PBC-02-008 and PBC-02-002.

[1], provides the flexibility, low latency and high bandwidth required in these environments. Often, the interconnection pattern between switches is defined by the user, which may be irregular. To manage such an irregularity, generic routing algorithms can be used, such as *up*/down** [17], *smart* [3], *adaptive-trail* [13], and *minimal adaptive* [19]. Up*/down* is the most popular routing algorithm used in clusters. However, this algorithm imposes a large number of routing restrictions in order to remove cyclic channel dependencies in the network graph, which prevent most of the messages from being routed through minimal paths. Unfortunately, this fact has a negative impact on the traffic balance since it tends to concentrate the traffic in a few channels, which dramatically limits its performance.

In order to balance the traffic in the network, dynamic or static strategies can be applied. Dynamic strategies select the routing paths depending on the network status. This status can be known either locally (adaptive routing strategies [4],[5],[10]) or globally (source-based routing schemes [8]). However, current commercial interconnects do not support adaptive routing. In addition, acquiring global information about the network status can introduce high overhead in the network due to the generated control traffic. Therefore, dynamic strategies to balance traffic are not suitable for application to clusters of PCs.

On the other hand, unlike dynamic strategies, static strategies do not require any hardware support. Thus, they can be applied to any commercial interconnect. These strategies try to achieve an even routing path distribution among the network channels to maximize the network utilization. To this end, traffic balancing algorithms can be used. These algorithms focus on selecting only one path between every source-destination pair, without taking into account the network status. Traffic balancing algorithms select each routing path among the set of paths provided by the applied routing algorithm. The use of these algorithms is suitable in networks that either apply source routing [2] or where the routing tables at switches provide deterministic routing [1]. Moreover, unlike adaptive routing strategies, traffic balancing algorithms guarantee the message delivery in order, which is required by many parallel applications.

Traffic balancing algorithms are based on an iterative procedure by which the routing paths provided by the applied routing algorithm are progressively discarded until only one routing path remains between each source-destination pair. Different heuristics can be applied to discard routing paths. The simplest criterion is the random selection. Its main drawback is that it cannot guarantee an even distribution of routing paths over the network channels. More efficient algorithms have been proposed in the literature, such as Summatory of Crossing paths [2], Deviation of Crossing paths [7], and Maximum Crossing path [15].

Given that these algorithms are based on an iterative procedure, it is clear that their effectiveness will strongly depend on the order in which routing paths are discarded. In this sense, we wonder whether these algorithms are able to obtain an optimal solution that achieves the highest performance within the bounds imposed by the applied routing scheme. To answer this question, we should firstly obtain the optimal solution. To this end, it would be necessary to evaluate all the possible combinations among the routing paths provided by the applied routing algorithm. However, this approach is non-viable in terms of computational time due to the huge amount of possible combinations.

An alternative to the enumeration of all the possible combinations, is the use of a heuristic algorithm. In this way, we will be sure to obtain a *good* solution (possibly a *local optima*) to the problem in a reasonable time. Concretely, we propose to use *metaheuristic* algorithms [9] because they can be applied to a wide set of different optimization problems, requiring few effort to adapt them to a specific problem. Nowadays, metaheuristics are widely used to solve important practical combinatorial optimization problems, and so we think that this can be an attractive approach to tackle with the problem posed in this work.

In this paper, we propose a new methodology to evaluate the effectiveness of traffic balancing algorithms, which is based on the use of metaheuristic techniques. This evaluation methodology allows us to know at what extent the performance provided by current traffic balancing algorithms differs from that provided by the optimal solution.

The rest of this paper is organized as follows. In Section 2 we describe the traffic balancing algorithm selected in the evaluation of the new methodology. Section 3 briefly describes the metaheuristics used in this paper. Section 4 describes the methodology followed in the evaluation and Section 5 shows some preliminary results achieved. And finally, Section 6 provides some concluding remarks and indications on future research.

2 Computing Routing Tables

In this section we briefly describe the process to compute the routing tables in a interconnection network given its topology. We can identify three main stages in this process, as can be seen in the Figure 1:

1. ***Computing Routing Paths*** Given a topology of the network the routing algorithm computes all the shortest possible routing paths between every source-destination pair of nodes.

2. ***Path Selection*** Once the routing paths have been computed, the next stage is to select through the application of a balance routing algorithm only an unique routing path for every source-destination pair of nodes. The paths selected for every source-destination pair represents a possible combination of routing paths that tries to achieve the maximum network performance.

3. ***Building Routing Tables*** Routing tables are build based on the routing path selected for every source-destination pair of nodes in the previous stage. Depending on the network technology, routing tables can be source routing or switch-based routing. In source routing routing tables are allocated in the hosts, and for switch-based routing routing tables are allocated in every switch in the network.

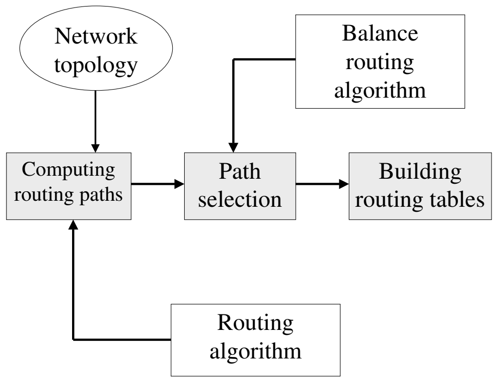


Figure 1: Stages of computing routing tables

Given that the Up*/down* routing algorithm is the most popular routing scheme currently used in commercial networks, we have selected this routing algorithm for computing the routing tables in our performance evaluation of the balance routing algorithms.

The Up*/down* routing algorithm avoids deadlocks by restricting routing in such a way that cyclic channel dependencies are avoided. In order to avoid deadlocks while still allowing all links to be used, Up*/down* builds an spanning tree and assigns a direction (“up” or “down”) to each output port based on the spanning tree. To compute the final paths it uses the following rule: a legal route must traverse zero or more links in the “up” direction followed by zero or more links in the “down” direction.

In order to compute the Up*/down* routing tables, different methodologies can be applied. These methodologies differ in the type of spanning tree to be built. The original methodology is based on BFS spanning trees, as it was proposed in Autonet [17], whereas an alternative methodology is based on DFS spanning trees, as it was proposed in [16].

Indeed, it has been demonstrated in several studies [16, 15, 14] that the DFS methodology substantially improves the network performance with respect to the BFS methodology. In fact, the DFS methodology imposes a less number of routing restrictions to avoid deadlocks providing more minimal paths and a better traffic balance than the BFS one.

On the other hand, we have selected in our evaluation the Maximum Crossing path (MaxCp) balance routing algorithm [14] which achieves the best performance at a lower computational cost than others routing algorithms [2, 7, 6], specially in large network size (64 switches). Moreover, the Up*/down* routing algorithm provides a better scenario to this balance routing algorithm because the Up*/down* routing devotes a large number of routing paths between every source-destination pair of nodes leading to an increases in the efficiency achieved by this balance routing algorithm.

In essence, the MaxCp algorithm is based on minimizing certain heuristic cost function, such as that associated to the routing paths between every pair of nodes according to their crossing path (channel utilization). Crossing path is defined as the number of routing paths crossing the channel.

This algorithm tries to achieve an uniform channel utilization, avoiding that a few channels become a bottleneck in the network. For this purpose, the algorithm associates a counter to every channel in the network. Each counter is initialized to the number of routing paths crossing the channel, that is, the crossing path. Additionally, a cost function is evaluated for each routing path. The cost associated to a routing path is computed by adding the crossing path of every channel in the routing path. The procedure defined below is applied repetitively to the channel with highest value of counter. In each step, a routing path crossing the channel is selected to be removed if there is more than one routing path between the source and the destination node of this routing path. In this way, we prevent the network to become disconnected. When a routing path is removed, the counters associated with every channel crossed by the path are updated. If there is more than one routing path able to be removed in a channel, the algorithm will first choose the routing path whose source and destination hosts have the highest number of routing paths between them. The algorithm finishes when the number of routing paths between every pair of nodes is reduced down to the unit.

3 Metaheuristics

As defined in the Metaheuristics Network[1] home page, "*a metaheuristic is a set of concepts that can be used to define heuristic methods that can be applied to a wide set of different problems*". That is, a metaheuristic can be viewed as a general algorithmic framework which allows us to apply a combinatorial optimization technique to a great variety of problems, with relatively few modifications when instancing the method to a particular problem. In this work, we have selected two metaheuristics which belongs to two different approaches: *genetic algorithms* (from evolutionary computation) and *simulated annealing* (from the neighborhood search approach).

3.1 Genetic Algorithms

Genetic algorithms (GAs) ([12], [9], chapter 3) are evolutionary algorithms based on the principles of natural evolution. GAs try to evolve a population of individuals (potential solutions to the problem) during the search process. Evolution is based on the application of *genetic*-based operators as *selection, crossover* and *mutation* to the individuals contained in the population. A GA works in three main steps:

1. Selection: some individuals are selected from the current population by considering their fitness (the goodness of the solution they codify).

2. Reproduction: the selected individuals are grouped (married) in pairs and some *offsprings* are obtained by crossover, that is, by interchanging genetic material between the parents. For example, from `aaaaaa` and `bbbbbb` we obtain `aabbbb` and `bbaaaa` when position two is selected as the crossover point. After crossover, some individuals are mutated (slightly changed) in order to introduce some diversity in the search. For example, following our example, `aabbab` corresponds to mutate position five of the first offspring.

3. Replacement: The new population/generation is obtained by selecting the best individuals from the old and new ones.

This process (steps 1 to 3) is repeated until a certain termination condition is achieved. In this work we use *Steady State* GAs, a model of GA in which only a pair of individuals is selected for reproduction in step 1. The good behavior of this model has been largely demonstrated in the specific literature. Figure 2 shows the pseudo-code corresponding to the particular *steady state*

[1]`http://www.metaheuristics.org`

```
Create the initial population
Repeat:
 Select two father p1 and p1
 {h1,h2}= crossing(p1,p2)
 Mutate {h1,h2} with probability pm
 Add h1 and h2 to the population
 Remove the two worst individuals in the population
Until the stopping condition is true
Return the best individual reached
   during the course of the execution
```

Figure 2: Pseudo-code for the steady state algorithm.

algorithm used in this work. Several parameters have to be defined prior, such as the formal representation of the solution, the operators used, the stopping condition, and so on. Section 3.1 describe in more details the parameters used in this work.

3.2 Simulated annealing

Simulated Annealing (SA) ([11], [9], chapter 10) is probably the oldest metaheuristic known and has its origin in statistical mechanics. In SA instead of maintaining a population of solutions, only a solution is considered at each state, being its neighborhood explored at each iteration.

SA behaves as a hill climbing algorithm, that is, given the current solution s, the algorithm moves to the best solution selected among the neighborhood ($N(s)$) of s (a neighbor s' is a solution obtained from s by making a small change or movement). The key idea in SA is to allow moves resulting in worse solutions than the current one in order to avoid being trapped in local optima (as usually happens in hill climbing). To implement this behavior, a parameter T called *temperature* is used. Concretely, the probability of accepting a neighbor s' of s is computed following the Boltzmann distribution: $exp(-\frac{f(s')-f(s)}{T})$. Therefore, when T is high, the probability of accepting a worse configuration is also high, but when the value of T decreases, the probability of accepting a worse solution also decreases. The value of T is decreased/cooled during the search process by following a cooling schedule. Figure 3 shows the pseudo-code of the cooling-down algorithm, where $V(x)$ is the set of neighbors of x, and $f(.)$ is the evaluation function.

Like the steady state algorithm described in Section 3.1, the value of several parameters have to be setup prior the execution. These values are significant because they have a great impact on the performance of the algorithm. These values are discussed in Section 5.

```
Set the parameters T and n
Selects the initial solution s0
x =s0
Repeat:
 For i=0 to n do
Select x′ from V(x)
If f(x′) > f(x) then x=x′
Otherwise, x=x′ with probability P(x,x′)
T = g(T)
      End For
Until the stopping condition is true
Return the best individual reached
 during the course of the execution
```

Figure 3: Pseudo-code for the cooling-down algorithm.

4 The Methodology of Evaluation

The methodology proposed to evaluate the effectiveness of the traffic balancing algorithms is based on the two metaheuristics previously described. As has been mentioned, the goal of these algorithms is to provide a high quality (the optimal one ideally) solution from a set of possibles configurations. In the problem of evaluating the effectiveness of traffic balancing algorithms, the optimal solution corresponds to the set of paths which achieve the highest performance in the network when considering only a unique route for every source-destination node pair. We will use the solution found by the metaheuristics to evaluate the effectiveness of the traffic balancing algorithm described in Section 2.

Below we will specify the parameter choices for both metaheuristic algorithms, but first let us to introduce two crucial points which are common to them: individual *representation* and *evaluation*.

- *Individual* **representation**. An individual will be a vector of integers, where each position corresponds to a pair of (source, destination) nodes, and its value is in the range $2..k$, being k the number of different paths from source to destination. Pairs having only a possible route between them are not included in the solution. So, the representation length is actually considerably smaller than the upper bound $n(n-1)$, with n the number of nodes in the network.

- *Individual* **evaluation**. Due to the large number of individuals to be evaluated during the search process, the use of a simulator, although very precise, is prohibitive in terms of CPU time. As an alternative we propose to use a performance metric, which can be easily computed from the routing paths codified by the individual. In our initial experiments we have tried with several routing metrics, as minimizing the maximum use of a channel, the deviation in channels use, and the sum of channels use. Among them, we have selected the minimization of the deviation in channel use, because it is the one which achieves the results closer to the simulator use.

4.1 Genetic Algorithm design

In order to completely specify the Steady State GA we have to consider the following parameters:

- Initial Population. Individuals in the initial population are generated at random.
- Selection. We use rank-based selection. That is, individuals are sorted according to its fitness and each individual receives a probability of selection proportional to its position in the ranking.
- Crossover operator. Two offsprings are generated by using the classical one-point crossover.
- Mutation operator. Each position of each individual in the population is selected for mutation with a small probability (0.01 or 0.02 in our case).
- Stopping Criterion. The algorithm is allowed to complete a fixed number of generations which depends on the network size.

4.2 Simulated annealing design

The main point here is to define the neighborhood of a given solution s. In our case, s' it is said a neighbor of s if they differ in exactly one position. That is, given s=`aaaaaa`, then `abaaaa` or `aaaaba` are in $N(s)$, but `abaaba` is not a neighbor of s. On the other hand, as in our GA design, the initial solution is randomly generated and a maximum number of iterations is considered as stopping criterion. The remaining parameters/decisions are:

- Iterations per Temperature. The number of iterations to be carried out before to modify the temperature is the minimum between 1000 and $\prod_{i=1}^{m} k_i$, being m the individual length and k_i the number of different paths between the pair (source-destination) associated with position i.

- Cooling schedule. The temperature decreases geometrically by using the expression: $T_{i+1} = \alpha T_i$, with $\alpha = 0.95$.

5 Performance Evaluation

In this section, we evaluate by simulation the performance of the optimal routing path configuration resulting from applying the metaheuristic algorithm proposed in Section 4 and compare it with that of the MaxCp traffic balancing algorithm.

We use the up*/down* routing algorithm based on the DFS methodology to compute the routing tables. To obtain realistic simulation results, we have used timing parameters for the switches taken from a commercial network. We have selected Myrinet because it is becoming increasingly popular due to its good performance/cost ratio. Myrinet uses source routing. Packet sizes of 32 and 1024 bytes are used. Also, we assume a uniform distribution of packet destinations.

5.1 Network Model

The network is composed of a set of switches and hosts, all of them interconnected by links. Irregular network topologies are considered in the evaluation. The topologies have been generated randomly. We have generated ten different topologies for each network size analyzed. Results in this paper correspond to the topologies that achieve the average behavior for each network size.

Each switch has 8 ports wherein 4 ports connect to hosts and the remainder connect to other switches. Network sizes of 8, 16, 32, and 64 switches have been considered in order to evaluate the influence of the network size on performance.

We assume short LAN cables to interconnect switches and hosts. These cables are 10 meters long, offer a bandwidth of 16 MB/s, and have a delay of 4.92 ns/m. Flits are one byte wide. Physical channel is also one flit wide. Transmission of data across channels is pipelined. Hence, a new flit can be injected into the physical channel every 6.25 ns and there can be a maximum of 8 flits on the link at a given time. A hardware *stop and go* flow control protocol [2] is used to prevent packet loss. The first flit latency through the switch is 150 ns.

5.2 Simulation Results

Figures 4, 5, 6, and 7 show the average latency and the throughput versus the injected traffic for the AOC and ADR schemes in networks sizes of 8, 16, 32, and 64 switches, respectively. As can be seen, the AOC scheme significantly achieves better performance than the ADR scheme. The improvements ranged from 10 percent to 15 percent. The explanation of these results is that the AOC scheme achieves a better traffic balance than the ADR scheme, though both algorithms are based on the same set of routing paths provided by the routing algorithm.

As the performance of the MaxCp algorithm is higher than that of other existing algorithms, the results obtained in this evaluation can be generalized to other low-performance algorithms. These results show that there is still enough margin to improve the performance of traffic balancing algorithms. Notice that the performance improvements resulting from applying traffic balancing algorithms are achieved without requiring any hardware support that would increase the system cost. Therefore, these algorithms represent a cost-effective approach to improve the performance of current commercial interconnects.

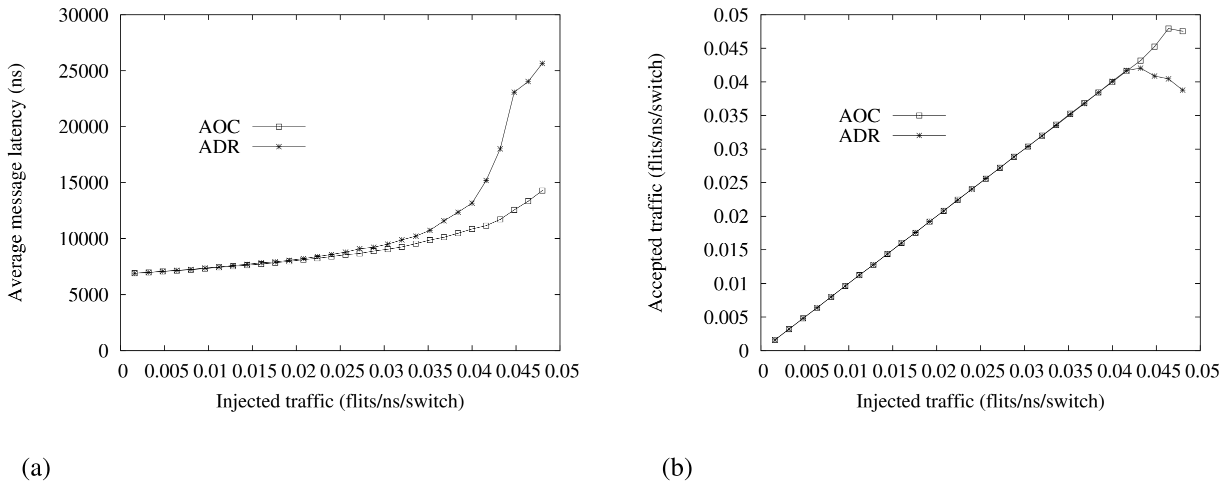


Figure 4: (a) Average message latency and (b) Throughput versus injected traffic for a network with 8 switches

6 Conclusions

In this paper, we proposed a new methodology to evaluate the effectiveness of traffic balancing algorithms. This methodology is based on comparing the performance achieved by the set of paths selected by applying a traffic balancing algorithm against that achieved by the set of routing paths selected by using a metaheuristic algorithm.

Preliminary results obtained for a representative traffic balancing algorithm have shown that its performance is significantly lower than that obtained by using a metaheuristic algorithm. In particular, a difference in throughput as high as 20 percent is obtained. However, the metaheuristic algorithms cannot guarantee that in all the configurations they will always provide an optimized

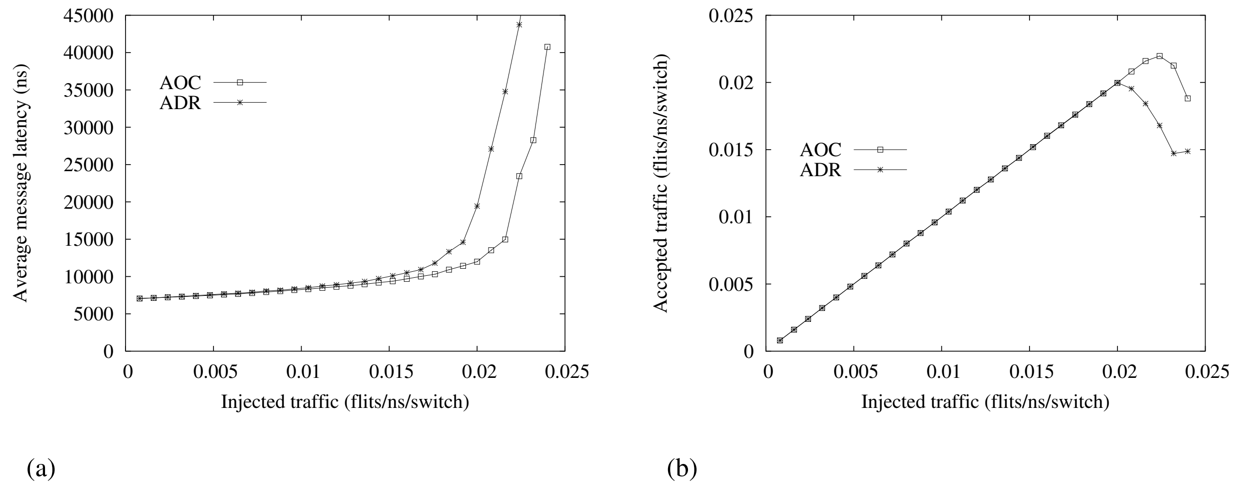


Figure 5: (a) Average message latency and (b) Throughput versus injected traffic for a network with 16 switches

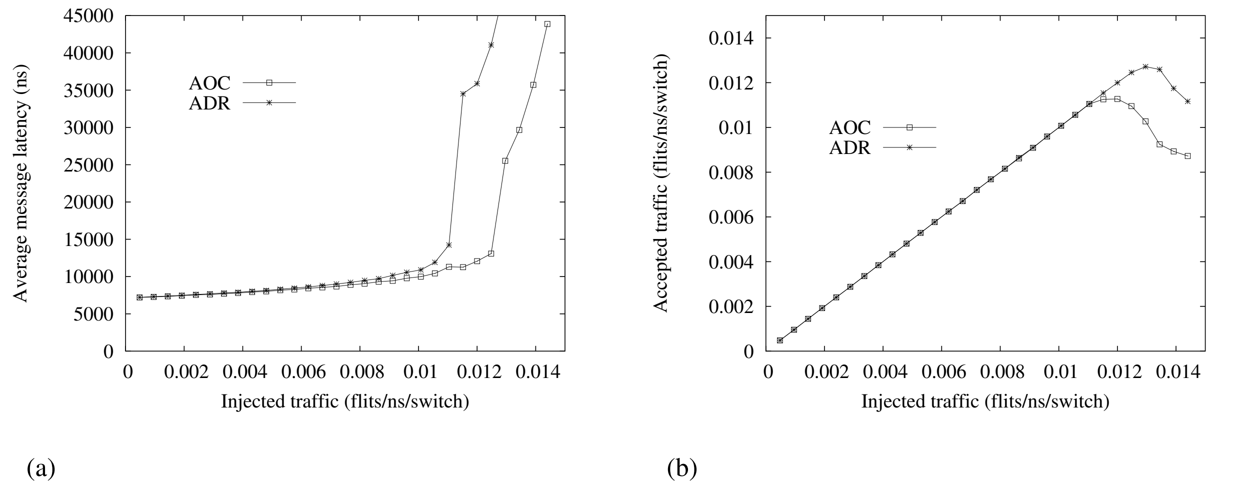


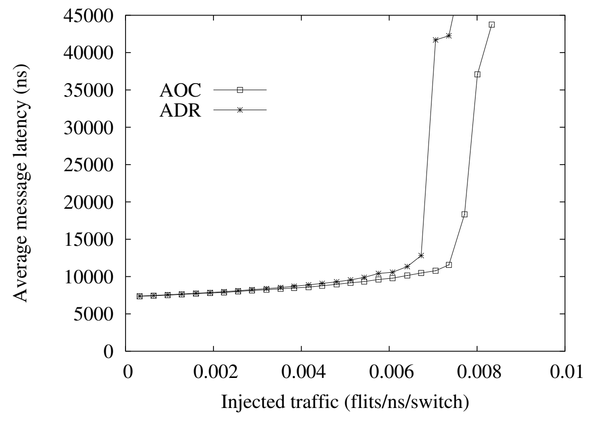
Figure 6: (a) Average message latency and (b) Throughput versus injected traffic for a network with 32 switches

set of routing paths that achieve the best performance, and can even be detrimental to performance. These results encourage us to continue to searching for more effective traffic balancing algorithms suitable to improve the performance of current commercial interconnects.

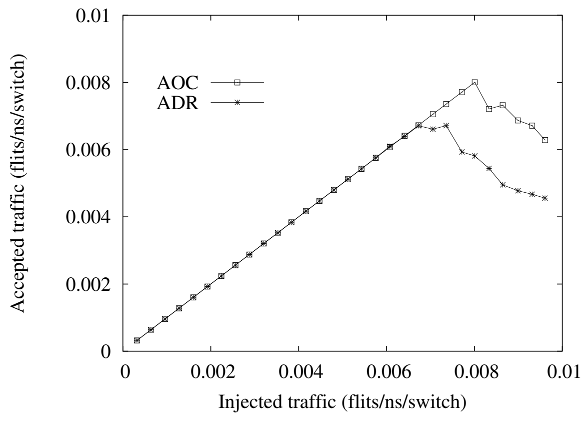
For future work, we plan to analyze the causes that limit the performance of current traffic balancing algorithms in order to find new criterions to select the final set of routing paths. The resulting algorithms must be viable in terms of computational time.

References

[1] I. T. Association. *InfiniBand Architecture Specification Volume 1. Release 1.0*, Oct. 2000.



(a)



(b)

Figure 7: (a) Average message latency and (b) Throughput versus injected traffic for a network with 64 switches

[2] N. Boden, D. Cohen, R. Felderman, A. Kulawik, C. Seitz, J. Seizovic, and W. Su. Myrinet - a gigabit per second local area network. *IEEE Micro*, pages 29–36, Feb. 1995.

[3] L. Cherkasova, V. Kotov, and T. Rockiki. Fibre channel fabrics: Evaluation and design. In *Proceedings of the 29th Hawaii International Conference on System Science*, Jan. 1995.

[4] W. Dally and H. Aoki. Deadlock-free adaptive routing in multicomputer networks using virtual channels. 4(4):466–475, Apr. 1993.

[5] J. Duato. A new theory of deadlock-free adaptive routing in wormhole networks. 4(12):1320–1331, Dec. 1993.

[6] J. Flich. *Mejora de las prestaciones de las redes de estaciones de trabajo con encaminamiento fuente*. PhD thesis, Universidad Politcnica de Valencia, 2001.

[7] J. Flich, M. Malumbres, P. Lpez, and J. Duato. Improving routing performance in myrinet networks. In *Proceedings of the International Parallel and Distributed Processing Symposium*, May 2000.

[8] D. Franco, I. Garcs, and E. Luque. A new method to make communication latency uniform. In *Procs. of ACM International Conference on Supercomputing*, pages 210–219, 1999.

[9] F. Glover and G. Kochenberger. *Handbook of Metaheuristics*. Kluwer Academic Publishers, 2003.

[10] S. Konstantinidou and L. Snyder. The Chaos router. *IEEE Trans. Comput.*, 43(12):1386–1397, Dec. 1994.

[11] P. V. Laarhoven and E. Aarts. *Simulated annealing*. Reidel Publishing Company, 1988.

[12] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1996.

[13] W. Qiao and L. Ni. Adaptive routing in irregular networks using cut-through switches. In *Proceedings of the 1996 International Conference on Parallel Processing*, Aug. 1996.

[14] J. Sancho. *Contribucin al diseo de algoritmos de encaminamiento en redes de estaciones de trabajo*. PhD thesis, Universidad Politcnica de Valencia, 2002.

[15] J. Sancho and A. Robles. Improving the up*/down* routing scheme for network of workstation. In *Proceedings of the European Conference on Parallel Computing*, Aug. 2000.

[16] J. Sancho, A. Robles, and J. Duato. A new methodology to compute deadlock-free routing tables for irregular topologies. In *Proceedings of the Fourth Workshop on Communication, Architecture and Applications for Network-based Parallel Computing*, Jan. 2000.

[17] M. Schroeder, A. Birrell, M. Burrows, H. Murray, R. Needham, T. Rodeheffer, E. Satterthwate, and C. Thacker. Autonet: a high-speed, self-configuring local area network using point-to-point links. *IEEE Journal on Selected Areas in Communications*, 9(8):1318–1334, Oct. 1991.

[18] R. Sheifert. *Gigabit Ethernet*. Addison-Wesley, Apr. 1998.

[19] F. Silla and J. Duato. High-performance routing in networks of workstations with irregular topology. *IEEE Transactions on Parallel and Distributed Systems*, 11(7), July 2000.