Journal of Universal Computer Science, vol. 18, no. 7 (2012), 917-936
submitted: 15/12/11, accepted: 27/3/12, appeared: 1/4/12 © J.UCS

ACO-based Algorithms for Search and Optimization of Routes in NoC Platform

Luneque Silva Junior
(Department of Electronics Engineering and Telecommunications
Engineering Faculty, State University of Rio de Janeiro
Rio de Janeiro, Brazil
luneque@hotmail.com)

Nadia Nedjah
(Department of Electronics Engineering and Telecommunications
Engineering Faculty, State University of Rio de Janeiro
Rio de Janeiro, Brazil
nadia@eng.uerj.br)

Luiza de Macedo Mourelle
(Department of Systems Engineering and Computation
Engineering Faculty, State University of Rio de Janeiro
Rio de Janeiro, Brazil
ldmm@eng.uerj.br)

Abstract: Network-on-Chip (NoC) have been used as an interesting option in design of communication infrastructures for embedded systems, providing a scalable structure and balancing the communication between cores. Because several data packets can be transmitted simultaneously through the network, an efficient routing strategy must be used in order to avoid congestion delays. In this paper, ant colony algorithms were used to find and optimize routes in a mesh-based NoC. The routing optimization is driven by the minimization of total latency in packets transmission. The simulation results show the effectiveness of the ant colony inspired routing by comparing it with general purpose algorithms for deadlock free routing under different traffic patterns.

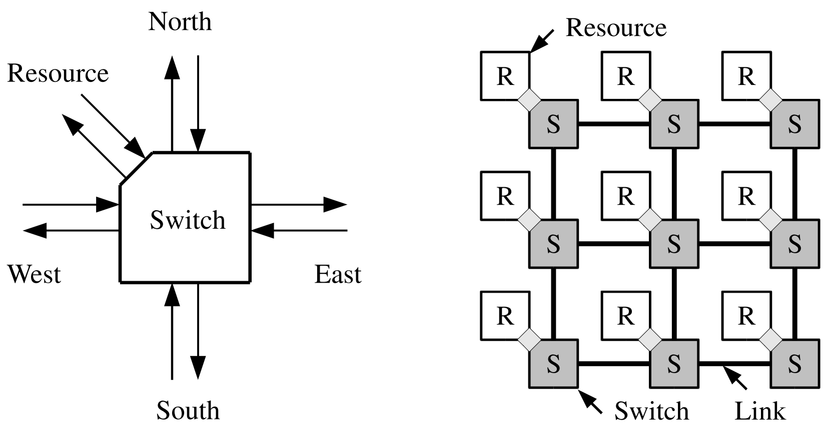
Key Words: network-on-chip, packet routing, ant colony optimization
Category: B.4.3, I.2.8, I.6.3

1 Introduction

A System-on-Chip (SoC) is an integrated circuit composed by a full computer system. SoCs contains, within the same package, processors, memory, input-output controllers and specific application devices. This block structure follows a design methodology based on intellectual property (IP) cores. Components designed for a specific project can be reused in other SoCs, reducing design time. Thus, under an extremely simplified view, to increase the number of tasks performed by the SoC, just add more IP cores with different features.

The increase of SoCs scale raises new design challenges. Among them is communication between IP cores. The blocks of a SoC are interconnected by a communication infrastructure, such as buses or point-to-point links. However, each of these models have their limitations. Shared buses can cause high delays if multiple blocks need to transmit data simultaneously. This does not happen in point-to-point architectures. In turn, the communication structure need to be redesigned for each new system. For many SoC designs, it is desirable to use a framework scalable as buses and fast as point-to-point links. An architecture that includes these two features are the NoCs, *Networks-on-Chip* [Benini and Micheli 2002].

In an NoC architecture, *switches* are interconnected by point-to-point links, thus describing a network topology. An example of network topology is the mesh shown in Fig. 1. The switches are also connected to the IP cores that constitute the system, also called *resources*. Switches exchange information in the form of messages and packages. The information generated by a resource is divided into smaller parts and sent over the network. These packages are organized in the destination switch and then delivered to resource. This operation is similar to that performed by computer networks. The structure formed by a switch and a resource is called a *network node*. NoCs can be used in the implementation of multi-processors systems-on-chip (MPSoCs) for running applications with high level of parallelism [Mourelle et al. 2010].



(a) Switch (b) Mesh topology

Figure 1: Network-on-chip architecture.

In the design of NoC-based systems, the communication infrastructure can be imported as a single configurable IP block. However, many are the ways to connect network and resources, in order to achieve the desired application. To assist the designer, computational tools for project assistance, or EDAs (Electronic Design Automation), are used [Jóźwiak et al. 2010]. The purpose of EDAs is to optimize intermediate stages of SoC and NoCs project, in order to obtain a more efficient design implementation [Edwards et al. 1997].

In general, NoCs are developed to perform a specific application. This application can be described initially as a software that must be embedded in hardware. The EDA tool must be able to use information about the desired application (at a high level of abstraction) and, through successive stages of optimization, implement a solution that meets the design specifications, which may include hardware area, power consumption and time of execution. This optimization may include several steps, such as *task allocation* [Da Silva et al. 2009], *IP mapping* [Nedjah et al. 2011] and *static routing*. The Fig. 2 shows in a simplified way the flowchart of a SoC design based on network-on-chip.

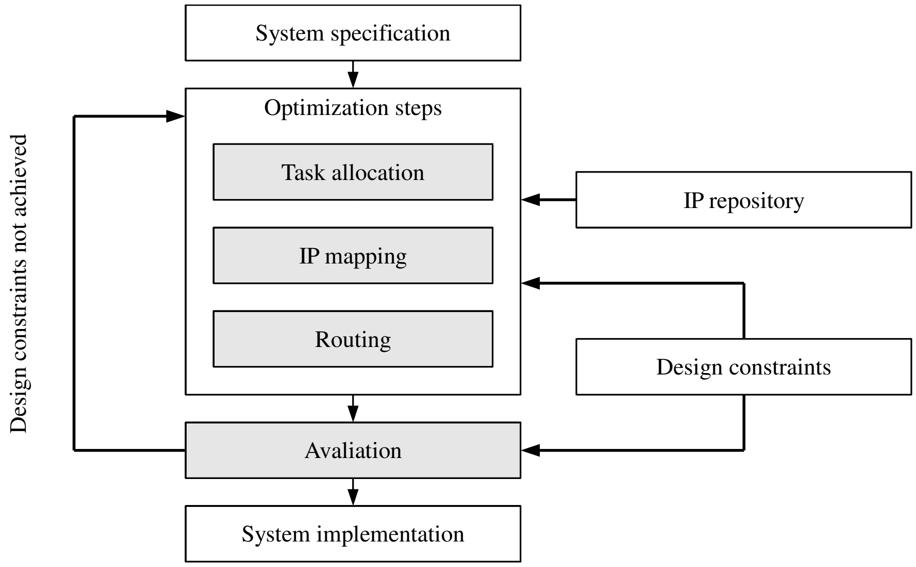
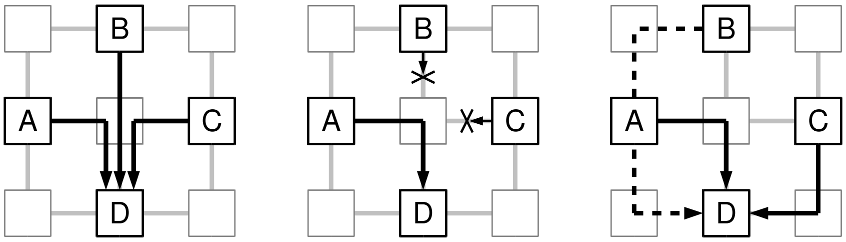


Figure 2: Typical embedded system design flow for NoC platform.

The process of IP allocation consists in associating each task (or set of tasks) to an appropriate IP block within a set of IPs or repository capable of performing such a task. The mapping of an application consists in associating the set of IPs resulting from the allocation to each node in communication infrastructure - in this case, the NoC. In other words, is spatially defined where each feature will be implemented, i.e., where in network each IP core is connected. Routing, in turn, defines which switches will be used for communication between cores.

Delays in communication may occur in congestion situations, when multiple packets could be transmitted using the same switch at the same time. If the routing algorithm adopted in the NoCs design is deterministic, the selection of the packet path from the source to the destination switch will not consider the load of intermediate switches - those between the source and destination switch. If these switches are under a heavy traffic, a given packet can only be transmitted after the end of congestion. This occurs even if other switches, not selected for routing, are free for transmission. On the other hand, adaptive routing algorithms can be used in order to avoid network congestion. These algorithms use not only the position of origin and destination nodes, but also the actual load condition of the network to calculate the route. When you find a region of network in use, the routing can set the package to follow another path. This congestion-free path may, however, be not minimal. These two situations are shown in Fig. 3.



(a) Ideal hypothetical situation. (b) Blocked packets in deterministic routing. (c) Non-minimal paths in adaptive routing.

Figure 3: Routing in a 3×3 mesh.

In order to overcome the congestion problem, this paper proposes a route optimization step in the design of NoCs, or more precisely, an adaptive and static routing. In this routing, a network model provides the communication patterns required for application execution. The calculation of routes is accomplished by an optimization algorithm to minimize the communication time. The search is always for a shortest path between origin and destination. If the intermediate switches of this path are in use, the algorithm should be able to find another route, so that the contention effects does not affect the transmission.

In this paper, the algorithm used in the search for routes is the ant colony optimization (ACO) [Dorigo et al. 1996]. This is an example of swarm intelligence, where a group of individuals work together to find a solution to a given problem. We compared the results of the network using the proposed routing algorithms and literature widely adopted routing algorithms.

The reminder of this paper is organized as follows. In Section 2 we review the related work in routing algorithms. The specification of the simulated network is shown in Section 3. In Section 4, we do an overview on ACO meta-heuristic. The proposed routing is presented in Section 5. Simulation results are presented in Section 6. The paper closes with a conclusion and the description of future work in Section 7.

2 Related work

There are several papers that study the efficient routing in parallel and distributed computing. For a broader reference, [Ni and McKinley 1993] presents a survey of routing techniques for direct networks.

Many of the techniques used for routing in NoCs, such as the XY algorithm, were originally developed for computer networks and multiprocessor systems. The XY algorithm is a routing technique widely used in 2D mesh networks with wormhole switching, such as the Intel Touchstone DELTA [Intel 1991], the Intel Paragon [Esser and Knecht 1993], the Symult 2010 [Seitz et al. 1988] and the Caltech MOASIC [Seitz et al. 1993]. It works by sending packets over the network first horizontally (X dimension), then vertically (Y dimension). This idea can be expanded to a larger number of dimensions, being known as such DOR (dimension order routing) [Ni and McKinley 1993]. In the context of NoCs, XY routing proves efficient due to its simplicity of implementation and because it is deadlock-free. Works that made use of this algorithm include the HERMES network [Moraes et al. 2004] and the SoCIN network [Zeferino and Susin 2003].

Glass and Ni have proposed the so-called Turn Model for adaptive, livelock and deadlock free algorithms [Glass and Ni 1992]. A turn is a change of 90° in the direction of packet transmission. The main idea of this model is to restrict the amount of turns that a package can perform in order to avoid the formation of cycles that cause deadlocks. Following this concept, three routing algorithms were proposed by Glass and Ni: the West-First, the North-last and Negative-First. A related approach is the Odd-Even turn model [Chiu 2000] for designing partially adaptive deadlock-free routing algorithms. Unlike the turn model, which relies on prohibiting certain turns in order to achieve freedom from deadlock, this model restricts the locations where some types of turns can be taken. As a result, the degree of routing adaptiveness provided is more even for different source-destination pairs.

The work of Jose Duato has addressed the mathematical foundations of routing algorithms. His main interests have been in the area of adaptive routing algorithms for multicomputer networks. Most of the concepts are directly applicable to NoC. In [Duato 1993], the theoretical foundation for deadlock-free adaptive routing in wormhole networks is given.

3 Network specification

The network model in this work uses switches with five communication ports. Four ports are responsible for communication with neighboring switches and one is for local communication with the resource. The switches are considered bufferless using no virtual channels. The network topology is a two dimension mesh, as shown in Fig 1. The switching technique adopted was the wormhole. In this method, packets are divided into smaller units called *flits* (flow-units). It is assumed that each communication channel has a width of a flit. The transmission of flits is performed in a pipeline way, as seen in Fig. 4.

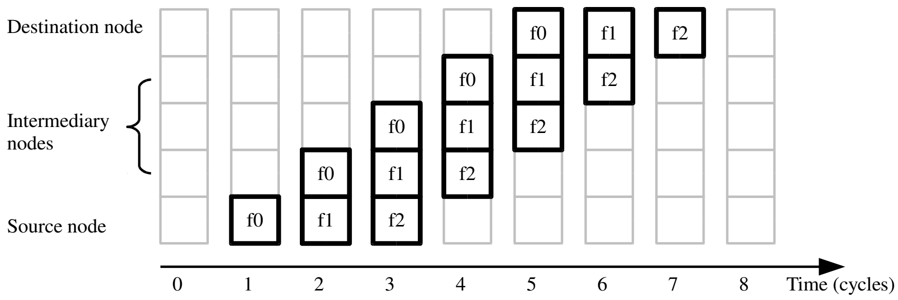


Figure 4: Transmission of 3 flits in wormhole switching.

The latency of a packet sent trough the network in wormhole switching is given by Equation 1, where t_{flit} is the transmission time of a flit in a channel, D is the number of switches in a path, L is the total length of a packet (in bits), W is the length of a channel, and L_{delay} is the number of bits that would have been transmitted in a period of congestion.

$$T_{packet} = t_{flit} \cdot \left(D + \left[\frac{L}{W} \right] + \left[\frac{L_{delay}}{W} \right] \right) \tag{1}$$

4 Ant Colony Optimization

Ant algorithms [Dorigo et al. 1996], also known as Ant Colony Optimization (ACO), are a class of heuristics search algorithms, that have been successfully applied to solving NP hard problems [Bonabeau et al. 1999]. Ant algorithms are biologically inspired in the behavior of colonies of ants, and in particular how they forage for food. One of the main ideas behind this approach is that the ants can communicate with one another through indirect means by making modifications to the concentration of highly volatile chemicals called *pheromones* in their neighbor environment. As it has been shown [Goss et al. 1989], indirect communication among ants via pheromone trails enables them to find shortest paths between their nest and food sources. The most emphatic and best known example of the use of pheromones by ants is the double bridge experiment. An ant nest is connected to a food source by two bridges with different lengths. This configuration is shown in Fig. 5. Initially, ants choose equally both ways. However, opting for shorter path are able to go back to the food supply before the ants that follow the long way. Thus, also the concentration of pheromone on the shortest path will be greater from the moment the ants complete the round trip. This capability of real ant colonies has inspired the definition of artificial ant colonies that can find approximate solutions to hard combinatorial optimization problems.

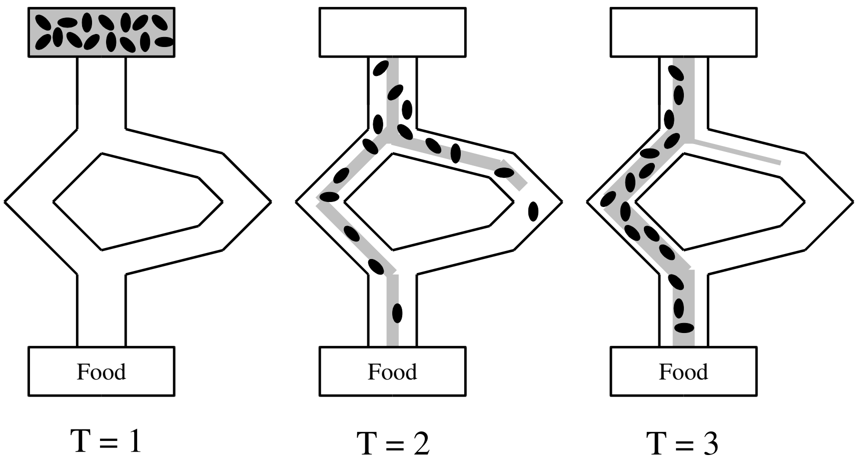


Figure 5: Pheromone concentration in the double bridge experiment.

The core ideas of ACO are (*i*) the use of repeated simulations carried out by a population of artificial agents called “ants” to generate new solutions to the problem, (*ii*) the use by the agents of stochastic local search to build the solutions in an incremental way, and (*iii*) the use of information collected during past simulations (artificial pheromones) to direct future search for better solutions. Several ant algorithms make use of the structure shown in the Algorithm 1 [Dorigo et al. 2006], the ACO meta-heuristics.

Algorithm 1 ACO meta-heuristic

initialize parameters and pheromone trails;
while termination condition not met **do**
    construct ant solutions;
    local search (optional);
    update pheromone trails;
end while;

In the artificial ant colony approach, each ant builds a solution by using two types of information locally accessible: problem-specific information, and information added by ants during previous iterations of the algorithm. In fact, while building a solution, each ant collects information on the problem characteristics and on its own performance, and uses this information to modify the representation of problem, as seen locally by the other ants. The representation of the problem is modified in such a way that information contained in past good solutions can be exploited to build new and hopefully better ones. This form of indirect communication mediated by the environment is called *stigmergy*, and is typical in social insects.

5 ACO based routing

The Ant Colony Optimization, with the ability to search for paths, emerging as a powerful solution for routing problems. Thus, this paper presents the use of the ACO meta-heuristic in the construction of routing algorithms. Two models of static routing for NoCs are proposed. These algorithms were called REAS (routing based on EAS [Dorigo et al. 1996]) and RACS (routing based on ACS [Dorigo and Gambardella 1997]). Both algorithms search paths in an *architecture characterization graph* that represents the network 2D mesh topology. These algorithms make use of multiple ant colonies, where each colony is responsible for searching the route of a package. In this approach, each colony has its own pheromone and ants. However, the colonies must exchange information in order to minimize the latency of their respective packages. Thus, the route found by an ant from a given colony is visible to the ants from other colonies, because these packets are being transmitted simultaneously and in the same network. In the proposed algorithms, ants in a network node knows only two things. The first is the pheromone concentration in the surrounding nodes. The second is the load on a node, the waiting time in each of the four possible transmission directions.

5.1 REAS algorithm

The *Elitist Ant System* is directly inspired by the *Ant System*, the first or ant algorithms [Dorigo et al. 1996]. The EAS is characterized mainly by the use of *elitism*, in order to differentiate the best ants. A simplified pseudo-code of REAS is shown in Algorithm 2.

Algorithm 2 REAS algorithm

```
Require: network parameters;
Require: EAS parameters;
Require: packets parameters;
while total of cycles do
  for k = 1 → number of ants do
    for g = 1 → number of packets do
      while node_actual ≠ node_destination do
        Ant_{k,g} select the node_next;
        calculates the load of Ant_{k,g} in node_actual;
        node_actual ← node_next
      end while
      calculates Ant_{k,g} pheromone;
    end for
    calculates the elitist pheromone;
    accumulate the pheromone of ants in k iteration;
  end for
  update the global pheromone;
end while
return best solution;
```

In the REAS algorithm, ants build paths through the network selecting the next node with base in Equation 2, where p_{ij}^k is the probability of the ant k go from the node i to the node j.

$$p_{ij}^k(t) = \begin{cases} \frac{\tau_j(t)^\alpha \cdot \eta_{ij}{}^\beta}{\sum\limits_{k \in allowed_k} \tau_k(t)^\alpha \cdot \eta_{ik}{}^\beta} & \text{if } j \in allowed_k \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The probability of selecting a particular direction is a function of pheromone concentration and network load in this direction. These two parameters are weighted by their importance constant α and β. The network load is used indirectly by η_{ij}, defined by:

$$\eta_{ij} = \frac{1}{C_{ij}} \tag{3}$$

where C_{ij} is the load in transmission from i to j.

At the end of each iterative cycle, the pheromone of all colonies is updated according to Equation 4. Part of the pheromone of the previous iteration is reduced by evaporation rate ρ, and then reinforced by the contribution of all m ants in the current cycle. The pheromone also receives the reinforcement of elitist ants: those that achieve the best solutions deposit their pheromone in every cycle, directing the search in subsequent cycles.

$$\tau_{t+1} = (1 - \rho) \cdot \tau_t + \sum_{k=1}^{m} \Delta\tau^k + \tau_{elite} \tag{4}$$

The pheromone in the path find by a single ant k is defined by:

$$\Delta\tau^k = \frac{Q}{L_k} \tag{5}$$

where Q is a constant and L_k represents the total latency of the solution. It is easy to see that the ants with the worst results provide a smaller amount of pheromone.

5.2 RACS algorithm

The second ant algorithm used in this work is described below. The RACS is very similar to REAS, with the same structure of *multiple colonies* being used. The algorithm on which the RACS was inspired, called *Ant Colony System* [Dorigo and Gambardella 1997], differs from others ant algorithms by:

- the selection method of next nodes in solutions building, and
- the use of a different pheromone update.

Because these two mechanisms, ACS improves over AS by increasing the importance of exploitation of information collected by previous ants with respect to exploration of the search space. The pseudo-code of RACS algorithm is shown in Algorithm 3.

Thus, the RACS uses the so-called *pseudo-random proportional* rule.

$$j = \begin{cases} argmax_{j\in[1,4]} \left\{\tau_j \cdot \eta_{ij}^{\beta}\right\} & \text{if } q \leq q_0, \\ S & \text{otherwise} \end{cases} \tag{6}$$

As shown in Equation 6, the probability for an ant to move from node i to node j depends on a random variable q, uniformly distributed over $[0, 1]$,

Algorithm 3 RACS algorithm

Require: network parameters;
Require: ACS parameters;
Require: packets parameters;

while total of cycles **do**
  for $k = 1 \rightarrow number\ of\ ants$ **do**
    for $g = 1 \rightarrow number\ of\ packets$ **do**
      while $node_{actual} \neq node_{destination}$ **do**
        $Ant_{k,g}$ select the $node_{next}$;
        calculates the load of $Ant_{k,g}$ in $node_{actual}$;
        update the local pheromone in $node_{actual}$;
        $node_{actual} \leftarrow node_{next}$
      end while
      calculates $Ant_{k,g}$ pheromone;
    end for
    if solution of ants in k iteration is the best **then**
      $\tau_{best} \leftarrow$ pheromone of ants in k iteration;
    end if
  end for
  update the global pheromone with τ_{best};
end while
return best solution;

and a parameter q_0. If $q \leq q_0$, then the next node is directly selected by $argmax_{j\in[1,4]}\{\tau_j \cdot \eta_{ij}^{\beta}\}$, i.e., the direction with the largest value of $\tau_j \cdot \eta_{ij}^{\beta}$. Otherwise, the next node is defined by S, that uses a selection method similar to that employed by EAS (Equation 2).

The RACS algorithm also uses a double pheromone update. The *offline update* is applied at the end of each iteration only by the *best-so-far* ant.

$$\tau_{t+1}^{j} = \begin{cases} (1-\rho)\cdot\tau_t^j + \rho\cdot\Delta\tau_j & \text{if } j \text{ belongs to best path} \\ \tau_t^j & \text{otherwise} \end{cases} \tag{7}$$

The offline update is given by Equation 7, where $\Delta\tau_j$ is the reinforcement of the best ant pheromone. As said, the offline update perform a strong elitist strategy. The best ant can be the iteration-best ant, that is, the best in the current iteration, or the global-best ant, that is, the ant that made the best tour from the start of the trial.

The *local update* is performed by all ants in each step of construction of a solution.

$$\tau_{t+1} = (1-\rho)\cdot\tau_t + \rho\cdot\tau_0 \tag{8}$$

This local update is defined by Equation 8, where ρ is the evaporation constant, and τ_0 is the initial pheromone at each node. In practice ACS ants "eat" some of the pheromone trail on the edges they visit. This has the effect of decreasing the probability that a same path is used by all the ants (i.e., it favors exploration, counterbalancing this way the above-mentioned modifications that strongly favor exploitation of the collected knowledge about the problem).

6 Evaluation experiments and results

A cycle-accurate network simulator was implemented in Matlab, supporting 2D mesh networks with wormhole switching. To estimate the performance of the proposed methods, networks were simulated with four different routing algorithms: REAS, RACS, XY and Odd-Even (OE). The efficiency of each type of routing is evaluated through $latency/packet \times injection\ rate$ curves. The time unit adopted is the *simulator cycle*, where one cycle is the transmission time of one flit. The performed tests uses synthetic patterns of packet generation, which vary the number of packets, the packet injection rate and the spatial arrangement of the nodes of origin and destination.

All algorithms were executed with Matlab Version 7.7.0.471 (R008b). The simulations were performed on PCs with Intel Core i7 950 3GHz, 8Gb RAM and Microsoft Windows 7 Home Premium operating system.

6.1 Tests with synthetic traffic patterns

The network was simulated with size of 5×5, a square of 25 nodes. The set of simulation tests were performed varying the network routing algorithm, the pattern of traffic generation, the rate of injection and the number of packets. These parameters are shown in Table 1.

Table 1: Simulation parameters.

Routing algorithms	REAS, RACS, XY, OE
Traffic pattern	Uniform, Hots-pot, Local, Complement, Trans. 1, Trans. 2
Injection rate	10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, 100%
Number of packets	10, 20, 30, 40, 50, 60, 70, 80, 90, 100

According to [Duato et al. 2002], the evaluation of interconnection networks requires the definition of representative workload models. This is a difficult task because the behavior of the network may differ considerably from one architecture to another and from one application to another. Moreover, in general, performance is more heavily affected by traffic conditions than by design parameters. Up to now, there has been no agreement on a set of standard traces that could be used for network evaluation. Most performance analysis used synthetic workloads with different characteristics. These models can be used in the absence of more detailed information about the applications. Workload models are basically defined by three parameters: *distribution of destinations*, *injection rate*, and *message length*.

6.1.1 Packet distribution

The source-destination pairs are generated following six different distribution patterns, as shown in Fig. 6. These patterns are based on models widely used in the evaluation of communication in multiprocessor and distributed systems [Duato et al. 2002]. The *uniform*, *hot-spot* and *local* were called random patterns, because both the source and destination nodes are chosen in a randomly way. In the uniform pattern, all nodes have the same probability of being selected. The hot-spot pattern is similar to uniform. However, for the destination nodes, a particular node has a higher probability of selection. In local pattern, only nodes around the source node can be selected as a destination.

The *complement*, *matrix transpose 1* and *matrix transpose 2* were called deterministic patterns. Although the selection of source nodes is random (following the uniform distribution), the destination nodes are selected according to the position of the source nodes. In the complement pattern, for a source node in the position (x, y), the destination node is in the position $(size-x+1, size-y+1)$, where $size$ is the number of nodes in a column or row of the mesh. For patterns matrix transpose 1 and 2, the destination nodes are respectively in the positions $(size-y+1, size-x+1)$ and (y, x).

6.1.2 Injection rate

The packet *injection rate* relates the transmission time of flits (from resource to switch) and idle time between the end of the transmission of a packet and the beginning of the transmission of next packet. The injection rate is a fraction of the network channel total bandwidth.

Two different injection rates are shown in Fig. 7. It is assumed that each flit is transmitted in one cycle, and that the period between the start of transmission of two consecutive packets is fixed. At the rate of 50%, a 5 flit size packet is injected into the network; only after a idle time of 5 cycles, the next packet starts to be injected. In this situation, only half of the total transmission capacity is used. At the rate of 100%, there is no idle time between packets. The use of injection rates lower than 100% is interesting in situations of network congestion, since the late flits can be sent during idle time between packets.

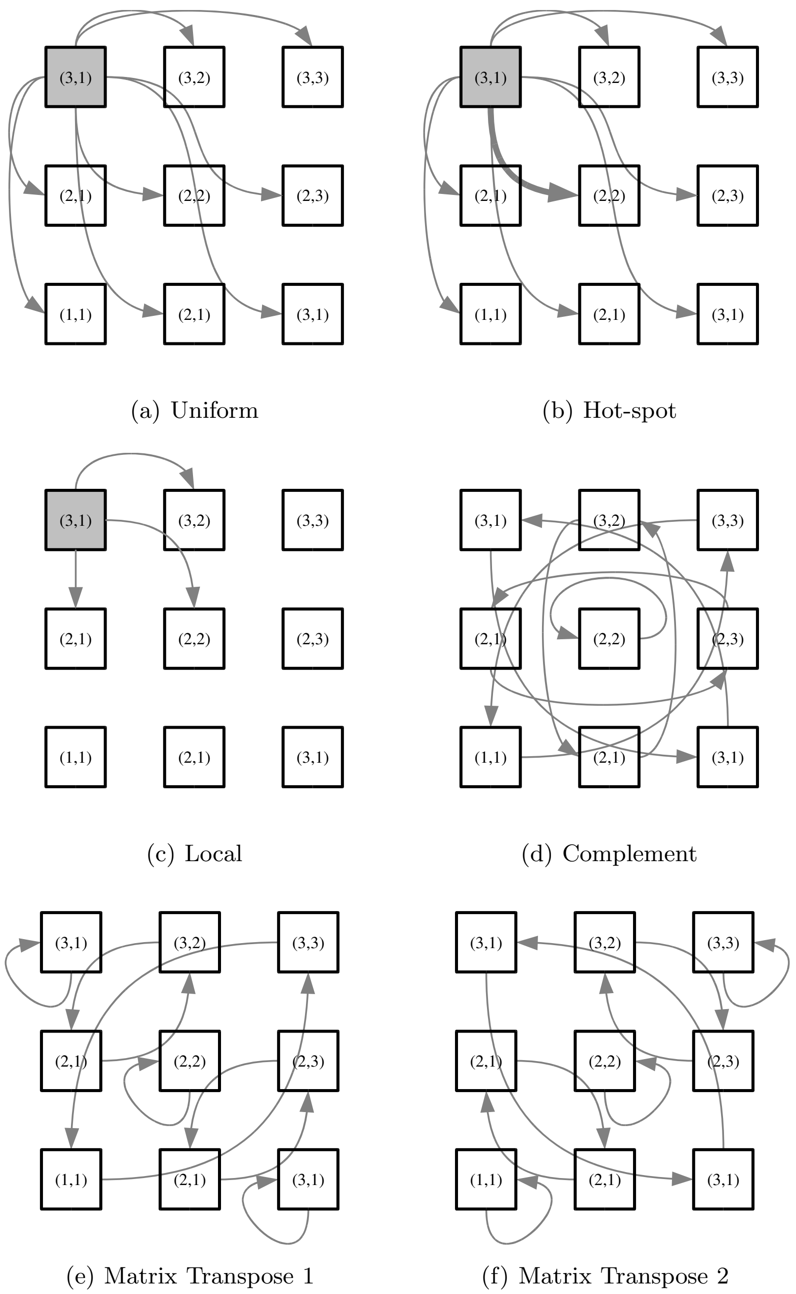


Figure 6: Possible communication pairs in a 3×3 mesh.

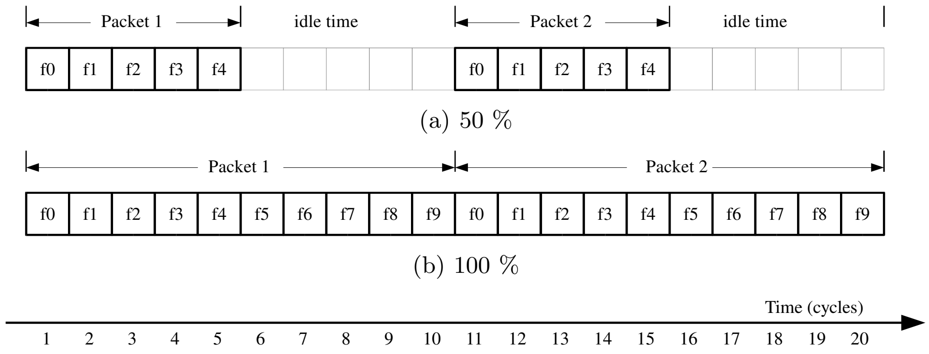


Figure 7: Two different injection rates.

6.1.3 Packet size

The packet size can also be shaped in various ways. Two values must be distinguished: the size of a packet and its amount of flits. The size L of a package is the value of its total length in bits. In turn, the amount of flits defined by Equation 9 is the largest integer value obtained by dividing the packet size by W, the size of a *phit* (physical unit, width of channel bits).

$$\#flits = \left\lceil \frac{L}{W} \right\rceil \tag{9}$$

Generally, packet length is defined as a constant in simulations. Alternatively, the length may be made variable in simulations, when studying the effects of different packet sizes on network [Duato et al. 2002]. In this situation, the size can be chosen at random according to a specific probability distribution, such as the spatial distribution of packets.

In this work, the amount of flits is associated with the injection rate. The number of cycles from the start of transmission of a packet and the start of the next is defined as the fixed value of 20 cycles. Thus, the amount of flits varies according to desired injection rate as seen in Table 2.

Table 2: Amount of flits in each injection rate.

Injection rate	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Flits	02	04	06	08	10	12	14	16	18	20
Idle cycles	18	16	14	12	10	08	06	04	02	00

6.2 Simulation results

For each simulation, we obtained the total latency and the average latency per packet. The total latency is the sum of the individual latency of all packets being transmitted on the network. The individual latency is the amount of simulation cycles that have elapsed since the injection of the first flit of a packet until the beginning of injection of the next packet of the same message. The average latency is the latency value obtained divided by the total number of packets.

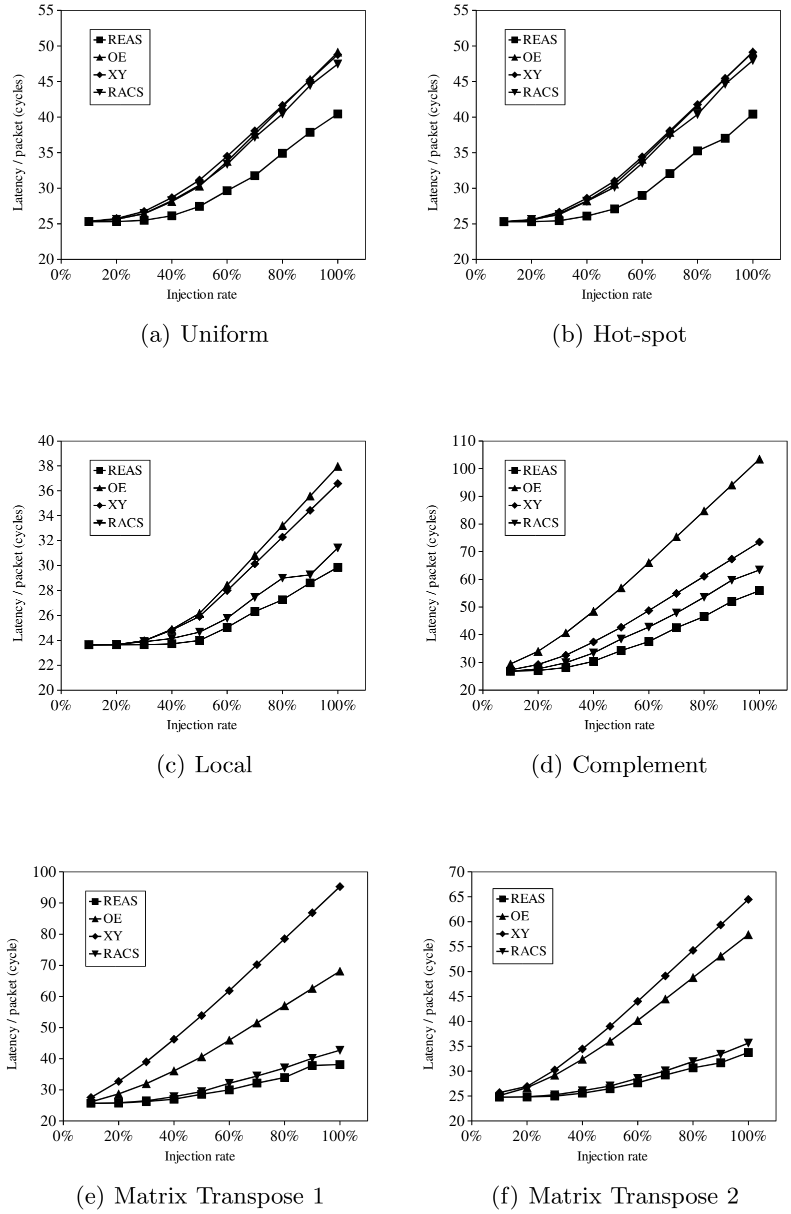
Results are shown in the Fig. 8. The general purpose of these tests is to verify the variation of latency under different injection rates. The curves of $latency/packet \times injection\ rate$ are, in fact, a mean of the values obtained for different quantity of packets. Each graph illustrates these curves for the four routing algorithms adopted.

The latency values (obtained in simulations) can be also arranged as function of number of packets. In Fig. 9, for each traffic pattern, $latency/packet \times number\ of\ packets$ curves are shown. For these curves, the value $latency/packet$ is the average (of obtained values for a same number of packets), for different injection rates.

For all traffic patterns, the latency curve of REAS is located below the curves of the other methods, indicating its ability to search for routes that provide a shorter transmission time. This performance is slightly better than the others at low injection rates, becoming more evident in rates above 50%. The RACS has a latency curve similar to the obtained by XY and OE algorithms for the uniform and hot-spot traffic patterns. For Local and Complement patterns these curves differ, with RACS getting lower latency values compared to the XY and OE. In matrix transpose patterns, the RACS achieved similar results to those obtained by REAS.

7 Conclusions

Static routing is an efficient solution in NoCs designed to run always the same set of applications. This is because the communication paths need only be defined one time. In this paper we propose the use of ACO-based algorithms in the optimization of paths in the static routing step in NoC design. The REAS and RACS algorithms proved to be effective in search of routes. Results of simulations with the network under different synthetic traffic patterns show the ant algorithms being able to find routes with latency less than that obtained with XY and OE algorithms. Future work may study the behavior of ant algorithms in packet routing for applications mapped on the network, showing its use in examples closer to real world NoCs.



(a) Uniform

(b) Hot-spot

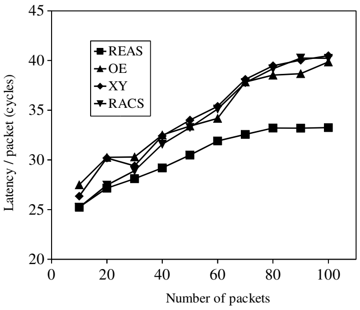
(c) Local

(d) Complement

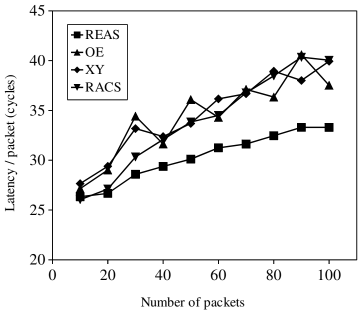
(e) Matrix Transpose 1

(f) Matrix Transpose 2

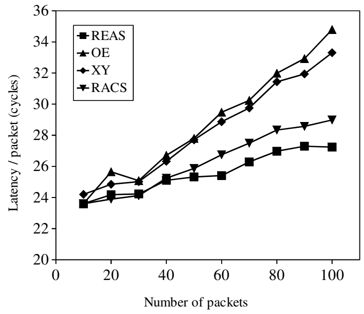
Figure 8: Results for the network under six different traffic patterns.



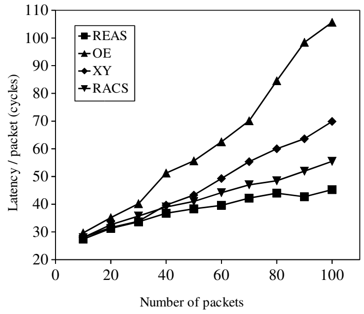
(a) Uniform



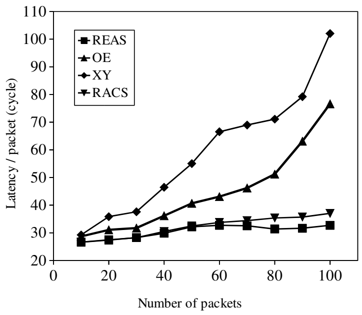
(b) Hot-spot



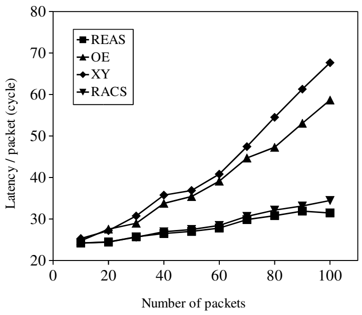
(c) Local



(d) Complement



(e) Matrix Transpose 1



(f) Matrix Transpose 2

Figure 9: Results for the network under six different traffic patterns.

Acknowledgment

We are grateful to FAPERJ (*Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro,* `www.faperj.br`), CNPq (*Conselho Nacional de Desenvolvimento Científico e Tecnológico,* `www.cnpq.br`) and CAPES (*Coordenação de Aperfeiçoamento de Pessoal de Ensino Superior,* `www.capes.gov.br`) for their continuous financial support.

References

[Benini and Micheli 2002] Benini, L., Micheli, G. D.: "Networks on chips: A new soc paradigm"; COMPUTER, Published by the IEEE Computer Society, p. 7078, 2002.

[Bonabeau et al. 1999] Bonabeau, E., Dorigo, M., and Theraulaz, G.: "Swarm Intelligence From Natural to Artificial Systems"; Oxford University Press, New York NY, 1999.

[Chiu 2000] Chiu, G.: "The odd-even turn model for adaptive routing"; Parallel and Distributed Systems, IEEE Transactions on, IEEE, v. 11, n. 7, p. 729738, 2000.

[Da Silva et al. 2009] Da Silva, M.V.C., Nedjah, N., and Mourelle, L.M.: "Optimal ip assignment for efficient noc-based system implementation using nsga-ii and microga"; IJCIS, 2(2):115–123, 2009.

[Dorigo et al. 1996] Dorigo, M., Maniezzo, V., Colorni, A.: "Ant system: optimization by a colony of cooperating agents"; Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, v. 26, n. 1, p. 29 41, 1996.

[Dorigo and Gambardella 1997] Dorigo, M., Gambardella, L.: "Ant colony system: A cooperative learning approach to the traveling salesman problem"; Evolutionary Computation, IEEE Transactions on, IEEE, v. 1, n. 1, p. 5366, 1997.

[Dorigo et al. 2006] Dorigo, M., Birattari, M., Sttzle, T.: "Ant colony optimization - Artificial ants as a computational intelligence technique"; Computational Intelligence Magazine, IEEE, v. 1, n. 4, p. 2839, 2006.

[Duato 1993] Duato, J.: "A new theory of deadlock-free adaptive routing in wormhole networks"; IEEE Trans. Parall. Distrib. Syst. 4, 12 (Dec.) 13201331.

[Duato et al. 2002] Duato, J., Yalamanchili, S., Lionel, N.: "Interconnection Networks: An Engineering Approach"; San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002.

[Edwards et al. 1997] Edwards, S., Lavagno, L., Lee, E., Sangiovanni-Vincentelli, A.: "Design of embedded systems: Formal models, validation, and synthesis"; Proceedings of the IEEE, IEEE, v. 85, n. 3, p. 366390, 1997.

[Esser and Knecht 1993] Esser, R., Knecht, R.: "Intel paragon xp/s - architecture and software environment"; Springer-Verlag, Berlin, p. 121–141, 1993.

[Glass and Ni 1992] Glass, C.J., Ni, L.M.: "The turn model for adaptive routing"; In: Proceedings of the 19th annual international symposium on Computer architecture. New York, NY, USA: ACM, 1992. (ISCA 92), p. 278287.

[Goss et al. 1989] Goss, S., Aron, S., Deneubourg, J., Pasteels, J.: "Self-organized shortcuts in the argentine ant"; Naturwissenschaften, 76:579–581, 1989.

[Intel 1991] Intel Corporation: "A Touchstone delta system description"; Supercomputer Systems Division, Intel Corporation, Beaverton, OR, v. 97006, 1991.

[Jóźwiak et al. 2010] Jóźwiak, L., Nedjah, N., Figueroa, M.: "Modern development methods and tools for embedded reconfigurable systems: A survey"; Integration, the VLSI Journal, 43(1):1–33, 2010.

[Moraes et al. 2004] Moraes, F., Calazans, N., Mello, A., Moller, L., Ost, L.: "Hermes: an infrastructure for low area overhead packet-switching networks on chip"; Integration, the VLSI Journal, Elsevier, v. 38, n. 1, p. 6993, 2004.

[Mourelle et al. 2010] Mourelle, L.M., Ferreira, R.E., Nedjah, N.: "Migration selection of strategies for parallel genetic algorithms: implementation on networks on chips"; International Journal of Electronics, 97(10):1227–1240, 2010.

[Nedjah et al. 2011] Nedjah, N., Da Silva, M.V.C., Mourelle,L.M.: "Customized computer-aided application mapping on noc infrastructure using multi-objective optimization"; Journal of Systems Architecture: the EUROMICRO Journal, 57(1):79–94, 2011.

[Ni and McKinley 1993] Ni, L.M. and McKinley, P.K.: "A survey of wormhole routing techniques in direct networks"; IEEE Tran. on Computers, 26:62-76, 1993.

[Seitz et al. 1988] Seitz, C.L., Athas, W.C., Flaig, C.M., Martin, A.J., Seizovic, J., Stelle, C.S., Su, W.K.: "The architecture and programming of the ametek series 2010 multicomputer"; In: Proceedings of the third conference on Hypercube concurrent computers and applications: Architecture, software, computer systems, and general issues - Volume 1. New York, NY, USA: ACM, 1988. (C3P), p. 33–37.

[Seitz et al. 1993] Seitz, C.L., Boden, N.J., Seizovic, J., Su, W.K.: "The design of the caltech mosaic c multicomputer"; In: Proceedings of the 1993 symposium on Research on integrated systems. Cambridge, MA, USA: MIT Press, 1993. p. 1–22.

[Zeferino and Susin 2003] Zeferino, C.A., Susin, A.A.: "Socin: A parametric and scalable network-on-chip"; In: Proceedings of the 16th symposium on Integrated circuits and systems design. Washington, DC, USA: IEEE Computer Society, 2003. (SBCCI 03), p. 169174.