ELSEVIER Simulation Practice and Theory 2 (1995) 237–255

SIMULATION PRACTICE AND THEORY

On modelling of an adaptive routing algorithm for massively parallel processing systems[☆]

Bruno Ciciani [a,*], Salvatore Tucci [b]

[a] *Systems and Computer Engineering Department, University of Rome "La Sapienza", Via Salaria 113, 00198 Rome, Italy*
[b] *University of Rome "Tor Vergata", Rome, Italy*

Received 1 July 1994; revised 11 January 1995

Abstract

In this work, the message latency of an adaptive routing algorithm for a hypercube circuit switching interconnection network is analyzed. This routing algorithm is based on back-track-to-the-origin-and-retry approach. Intermediate links are chosen among the ones that form the shortest paths from a source to the destination, after excluding all the allocated links, the algorithm selects one of the remaining free links at random. The subsequent link in the path is established through the selected link and the procedure is repeated until the destination is reached, or when no free links can be found. If no free links are to be found at an intermediate node in the path, then a break is returned to the origin, the partial path is dissolved, and after a back-off period the path set up phase is repeated.

The average message latency for such a protocol is here determined using an iterative approximated analytical model; peculiarity of the model is that it takes into account the feedback effect between the probability of conflict to obtain a link and the message latency itself. An accurate simulation analysis has been carried out to validate the proposed model.

Finally the adaptive protocol is compared with the e-cube based protocol currently used by commercial products. The derived results show a better behaviour of the previous ones independent of the message length distribution.

Keywords: Parallel processing systems; Interconnection networks; Circuit-switching techniques; Performance analysis; Approximated analytical model; Simulation

1. Introduction

Massively parallel computing systems are a reality today. There is a variety of different proposals due to the difficulty in optimizing performance indices (such as

[☆] This work was partially supported by CNR – PF "Calcolo Parallelo" under contract 89.00032.69 and by MURST 60% funds.
[*] Corresponding author. Email: ciciani@dis.uniroma1.it.

the speed-up, efficiency, memory requirements, etc.) given the class of applications to parallelize. Indeed, MIMD based systems can be programmed by two programming models: *shared memory* and *message passing*. MIMD systems that use the first paradigm are known as *multiprocessor* and the other ones as *multicomputer* [37]. In these kinds of systems the interconnection network can be the performance bottleneck, i.e. without adequate communication availability the processing elements are forced to wait for process interactions and a system performance degradation will occur [16].

According to Seitz [36], two types of interconnection networks are possible: the so-called *indirect* and the *direct* networks. In the former, the interconnection nodes are disjointed from the processing element, while in the latter, the two elements belong to the same logical entity. The interconnection structures can also be classified with respect to three basic operating characteristics: *timing philosophy* (synhronous, asynchronous), *control strategy* (centralized, distributed) and *switching methodology* (packet-switching, message-switching, wormhole, circuit-switching and virtual-cut-through) [4,15]. The wormhole, the virtual-cut-through and the circuit-switching are also known as *point to point* interprocessor communication techniques [28].

Both programming paradigms can be supported by massively parallel systems having indirect or direct interconnection networks [3]. In this paper, we are interested in direct interconnection structures for multicomputer systems characterized by decentralized control system, asynchronous timing approach and circuit switching methodology.

We are focusing on these kind of systems because, at the present, in order to reduce message latency the second generation of *multicomputers* has abandoned the message store and forward switching techniques (typical of the first generation, e.g. NCUBE/1 [7], iPSC/1 [18]) for point to point interprocessor communication techniques. For example iPSC/2 [29], NCUBE/2 [27] and a Jet Propulsion Laboratory experimental MIMD [33] use circuit switching techniques, while Symult 2010 and iWARP [7] use wormhole routing techniques. The point to point interprocessor communication networks are also the choice for the next multicomputer generation. An example is given by the announced Inmos IMS C104 chip that adopts a wormhole routing technique [17]. Moreover, point to point direct interconnection structures are also useful for *multiprocessor* systems because they permit exploitation of the locality in the communication patterns of parallel applications [3].

In using circuit switching technique, a path has to be established across the interconnection network before data can be transferred from the source node to the destination ones. In establishing the path, some intermediate segments (links) may already be being used by another path (previously established or in path set up phase). When this occurs, we say that a *conflict* has arisen. Actually, all the circuit-switching based commercial multicomputer use the so-called fixed “hold” strategy; i.e., the route between two nodes is statically determined and in case of conflict the partial path already established is held until the link becomes free. This strategy is easy to implement but it has drawbacks from the point of view of performance and reliability. Indeed, if one path is crowded with message traffic, it is not possible to try another path; moreover, if a link is faulty none of the paths using that link can be established, thus reducing the possibility of message exchange between nodes.

To reduce link utilization, when there is a link conflict the partial path can be released ("drop" strategy), or another route can be tried ("adaptive" strategy); meanwhile, if one path has a faulty node, another path may be taken to preserve communication ("adaptive" strategy) [6,14,25,33,38].

Our goal is to investigate the benefits in using an *adaptive* over *e-cube based hold* strategy [29] for direct hypercube multicomputers. To this purpose an approximate analytical model is introduced. In this paper we analyze the binary hypercube topology because it has been proven to be useful for general purpose processing and because it has been used in several significant supercomputers like Cosmic Cube, the Connection Machine, Ncube, iPSC/1, iPSC/2, iPSC/860 [7,18,27,29,31]. However the proposed modelling method can be used for all asynchronous direct or indirect point to point communication networks characterized by symmetric topologies (torus-mesh, k-ary n-cube, etc.). This method cannot be used for asymmetric topologies (such as mesh) because it assumes that the load is the same in all the links of the interconnection network [20].

The modelling of indirect or direct multicomputer networks using message store and forward switching communication paradigm [1,34,35] and of indirect synchronous point to point circuit switching communication networks [22,30,38] is well investigated. On the other hand, there are few papers which model asynchronous direct or indirect point to point communication networks [8,12,15]. Furthermore, these papers model only the hold strategy and the majority of them make the simplifying assumption that the link acquisition and releasing is instantaneous [2,15,21]. To the best of our knowledge there are no analytical models for asynchronous direct or indirect point to point communication networks using adaptive strategies; however, some performance considerations have been obtained by simulations [6,13,14,19,23,33].

Asynchronous direct or indirect point to point communication networks may be modelled using passive resources, but queueing networks that include them have no product-form solution [15]. In this paper we introduce an iterative modelling approach [9,10] that can be used to evaluate the message latency of asynchronous direct hypercube circuit switching interconnection networks based on an "adaptive" strategy. The model takes also into account the link acquisition and releasing time. The main peculiarity of the model (due to its iterativity) is that it permits the capturing of the feedback effect between the probability of conflict for a link and the message latency. Some of the assumptions used to build our model are generally accepted in the current literature and other ones are made to obtain a suitable mathematical model. A simulation model has also been developed in which the latter assumptions have been removed.

In this paper we analyze the adaptive routing strategy only, because the e-cube hold strategy has been modelled in a previous paper [11].

In Section 2 the operational characteristics of the interconnection network are presented. In Section 3 the model is developed. Section 4 is devoted to an analysis of the results and to a series of comparisons with the simulation models. Section 5 presents the concluding remarks.

2. Adaptive strategy

A d-dimensional binary hypercube is a graph of 2^d nodes and $d \cdot 2^{d-1}$ links. Each node is connected to d neighbor nodes and each node address is expressed with a string of d bit. The *Hamming distance* between two nodes with address $s = s_d s_{d-1} \dots s_1$ and $r = r_d r_{d-1} \dots r_1$ in an hypercube of d dimension is defined as

$$H(s, r) = \sum_{i=1}^{d} h(s_i, r_i), \qquad h(s_i, r_i) = \begin{cases} 1, & \text{if } s_i \neq r_i, \\ 0, & \text{if } s_i = r_i. \end{cases}$$

A path between two nodes in a hypercube is represented by a sequence of nodes in which every two consecutive nodes are physically adjacent to each other in the hypercube [5]. The number of links in the path is called the length of the path. If the length of the path is equal to the Hamming distance then the path is called *optimal*.

Each node consists of two components: a standard processing element (PE) and a communication router (CR). The router operates as a $(d+1) \times (d+1)$ crossbar switching network. A maximum of d simultaneous communications are possible for each router, having a DMA controller per each dimension.

The routers form a circuit switched network that dynamically establishes paths connecting source nodes to destination nodes. Paths are established by a step-by-step process, and for each step, the additional link assignment is determined by an arbitration process. The adaptive routing algorithm builds optimal paths. To this purpose intermediate links are chosen among the ones that form the shortest paths from a source to a destination. Since several paths may be forming in parallel, some of these links may already be allocated to some other paths. After excluding all the allocated links, the procedure randomly selects one of the remaining free links. The subsequent link in the path is established through the selected link and the procedure is repeated until the destination is reached, or when no free links could be found. If no free links are to be found at an intermediate node in the path, then a break is returned to the origin, the partial path is dissolved, and the path set-up phase is repeated. When a break occurs, in order to reduce the probabililty of subequent aborts, the source router waits for a "back-off" period before the path establish process is repeated.

When the destination node is reached, the local CR sends an acknowledgment signal to the source-CR along the established path. At the arrival of the acknowledgment signal, the source-CR can start the transmission of the message body to the destination-CR. At the end of data transmission, the source-CR sends an end-transmission signal to the destination-CR. This signal, propagated along the path, will cause a step-by-step link release process.

This adaptive strategy is, of course, deadlock free, since one of the four necessary conditions for deadlock, the hold and wait condition [32], is not met.

3. Message latency analysis

An iterative approximated analytical model to evaluate the message latency of "adaptive" link conflict resolution strategy is developed. Main goal of the model is the evaluation of the probability of link conflict. The difficulty of evaluating this probability is due to the dependence of the link utilization on the duration of the communication phases (composed of the path set-up phase, data transmission phase and path release phase). In turn, the duration of the path set-up phase depends on the probability of link confict. The iterative approach is used to capture this feedback [9,10].

To build a suitable mathematical model for the adaptive strategy we assume:

(a) communication requests arising from the Processing Elements (PEs) are independent and identically distributed according to a Poisson process with rate λ per each PE;
(b) the capacity of the buffers present at each PE is considered as unlimited;
(c) the link arbitration, link connection and link releasing are pure delays;
(d) when a conflict occurs there are no other communication requests waiting for the same link;
(e) the probability of link conflict is independent of the number of the "acquired" links;
(f) the probability of link conflict is independent of the number of aborts.

Some of the previous assumptions are generally accepted in the current literature ((a), (b) and (c)) and the other ones are specific for our modelling to obtain a suitable analytical model. A simulation model, in which the latter assumptions have been removed, has also been developed to validate the analytical model.

It should be noted that, according to the current literature [34], there are no assumptions about message routing distribution. In fact, due to the network symmetry and to the similar node message generation behaviour, a similar message routing behaviour can be assumed. Thus, each node and each link are equally likely to be visited independently of the type of message routing distribution assumed (uniform, sphere of locality, decreasing probability, etc.).

Assumption (a) permits an evaluation of the message latency, considering the progress of a single communication request in a network in which the link request is generated by a Poisson source with rate $\lambda \cdot N_{nodes}$, where N_{nodes} is the number of PEs (i.e. the superposition of Poisson processes is still a Poisson process). Taking assumption (a) and the network symmetry together, we deduce that the average number of links per path (M) is the same for all PEs, independently of their positions. The message average latency may therefore be determined simply by considering a communication request for the average path. In the Appendix, an evaluation of the average number of links per path (M), when there is uniformly distributed message routing, is presented. A similar approach can be used to evaluate M for different message routing distribution (sphere of locality, decreasing probability, etc.).

Assumption (b) permits neglect of the blockage of communication requests due to the lack of memory space for saving the arriving or aborted communication requests.

Assumption (c) permits neglect of the feedback effect between hardware utilization and "logical" link utilization (i.e. higher hardware utilization implies longer message latency and therefore a higher conflict probability, which in turn produces a higher hardware utilization, and so forth).

Assumption (d) permits the consideration of the probability of conflict only in terms of link utilization.

Assumption (e) permits the consideration of the probability of conflict for the ith link request independent of the "i" value.

Assumption (f) permits the consideration of the probability of conflict with the same communications that caused the previous aborts equal to be zero. In reality this probability can be controlled by the value of the back-off time (the bigger the back-off time, the lesser this probability).

Finally, from the interconnection network topology and the adopted assumptions, it is evident that the link requests are more concentrated in the neighboring links. This locality holds for each node; therefore the communication request rate arising from each router may be considered as identical. The average link utilization may thus be supposed the same throughout the network.

The communication request evolves through three phases (as shown in Fig. 1). The first is the path set-up phase. During this phase, the communication request can be in a requesting state, in a connecting state or in a releasing state. In Fig. 1, thes situations are represented by states $A_{i,j}$, C_i and R_i. In state $A_{i,j}$ the communication request has obtained $i-1$ link and is demanding for the jth alternative for ith link of the path. If the ith link is idle, then the communication request takes a transition to state C_i and connects the new link to the previous ones. Otherwise, if there are

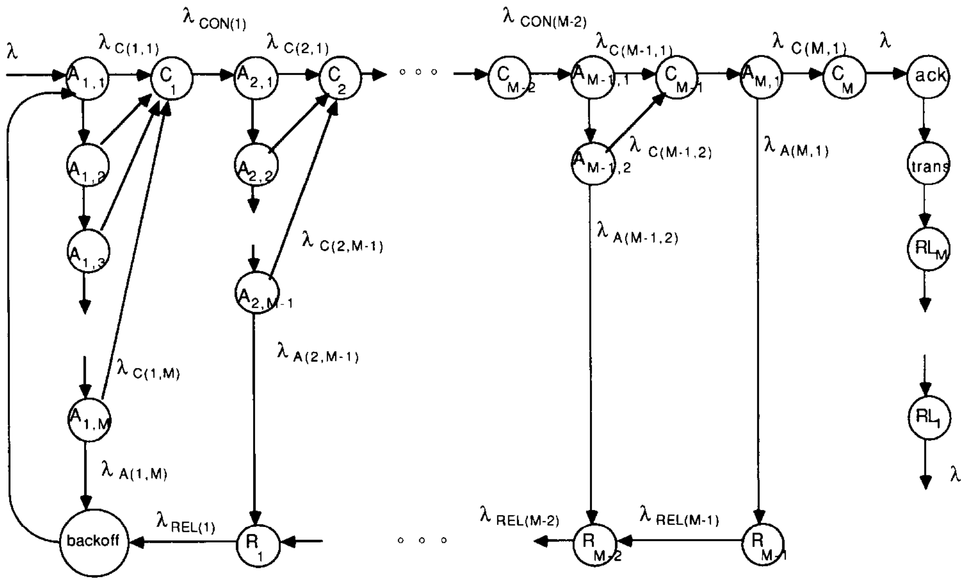

Fig. 1. Communication transfer request state diagram for the "adaptive" strategy.

other links it makes a transition to state $A_{i,j+1}$, otherwise to state R_{i-1}. From here the link releasing process takes place. Note that number of possible choices is $M+1-i$ (for $i=1,\ldots,M$).

The time spent in states $A_{i,j}$, C_i and R_i are constant (see assumption (c)) and they are denoted by T_{arb}, T_{con} and T_{rel}, respectively. After the conclusion of the releasing phase due to an abort, the communication request waits for a back-off period (T_{back_off}) before trying again to establish a path.

When the communication request has obtained the last link of the path, an acknowledgment signal is transmitted by the destination node to the source (state "ack"), and the time spent here is a constant and equal to T_{ack}. This signal enables data transmission (state "trans") to occur, the duration of which is equal to the average data transmission time (T_{data}). At the end of the communication transfer, the releasing of the links takes place (index i of state "RL_i" represents the residual link of the path not yet released), and the time spent here is a constant and equal to T_{rel}.

The message latency can be expressed by three terms

$$T_{msg\_latency} = T_{path\_set\_up} + T_{data\_trans} + T_{path\_release}, \tag{1}$$

where

$T_{path\_set\_up}$	is the average time necessary to set-up the path,
T_{data_trans}	is the average data transmission time,
$T_{path\_release}$	is the average time necessary to release the links of the path, and holds MT_{rel}.

The evaluation of the $T_{path\_set\_up}$ demands an additional explanation. This quantity is given by

$$T_{path\_set\_up} = N_{arb}T_{arb} + N_{con}T_{con} + N_{rel} + N_{aborts}T_{back\_off} + T_{ack}, \tag{2}$$

where

- N_{arb} is the number of visit to states $A_{i,j}$, therefore $N_{arb}T_{arb}$ is the total link arbitration time,
- N_{con} is the number of visit to states C_i, therefore $N_{con}T_{con}$ is the total time for link physical connection,
- N_{rel} is the number of visit to states R_i, therefore $N_{rel}T_{rel}$ is the total time for link release due to contention with other communication requests,
- N_{aborts} is the number of arborts, therefore $N_{aborts}T_{back\_off}$ is the total back-off time per communication request,
- T_{ack} is the average time required to send the signal "path is set" from the destination node to the sender node.

Now we have to evaluate N_{arb}, N_{con}, N_{rel} and N_{aborts}. These quantities are functions of the number of aborts per communication request before to complete a path, that at its turn depends on the probability of link conflict.

The probability that a link request conflicts with another communication request is equal to the probability that the link is held by another communication request, i.e. the link utilization factor [20]

$$P_{link\_conflict} = \frac{number\ of\ held\ links}{number\ of\ available\ links} = \frac{N_{links\_held}}{N_{links\_available}}, \tag{3}$$

where the number of available links is equal to

$$N_{links\_available} = d2^{d-1} \tag{4}$$

and the number of links held by all communication requests is equal to

$$N_{links\_held} = N^{set\_up}_{links\_held} + N^{data\_trans}_{links\_held} + N^{release}_{links\_held}. \tag{5}$$

The evaluation of the terms of expression (5) is straightforward. Indeed, the total number of links held in one of the possible states of Fig. 1 is given by the number of links held in the state, multiplied by the incoming rate and the average residence time in that state.

In particular the average number of links held by communications in data transmission phase (at each instant) is equal to

$$N^{data\_trans}_{links\_held} = N_{nodes} = \lambda M T_{data\_trans}, \tag{6}$$

where

$N_{nodes}\lambda$	is the incoming rate in data transmission state,
M	is the average number of links held by communications in data transmission state (M is the average number of links per path),
T_{data_trans}	is the average residence time in data transmission state (i.e. the average data transmission time per message).

Whereas the average number of links held by communications in path_release phase is equal to the sum of the average number of links held by communications in all the link release states (state RL_M–RL_1 in Fig. 1). At its turn the average number of links held by communications in the ith link release state is equal to

$$N^{RL_i}_{links\_held} = N_{nodes}\lambda i T_{rel} \tag{7}$$

where

$N_{nodes}\lambda$	is the incoming rate in the ith link release state,
i	is the average number of links held by communications in the ith link release state,
T_{rel}	is the average residence time in the ith link release state.

Therefore the average number of links held by communications in path release phase is equal to

$$N^{release}_{links\_held} = \sum_{i=1}^{M} N_{nodes}\lambda i T_{rel} = N_{nodes}\lambda T_{rel} \sum_{i=1}^{M} i = N_{nodes}\lambda T_{rel} \frac{M(M+1)}{2}. \tag{8}$$

Instead, the average number of links held by communications in path_set_up phase is equal to the sum of the average number of links held by the communications in all the states of this phase. These states (see Fig. 1) can be classified in four classes:

arbitration states, connection states, link-releasing states due a conflict and acknowledge state. Therefore,

$$N^{set\_up}_{links\_held} = N^{arb}_{links\_held} + N^{con}_{links\_held} + N^{rel}_{links\_held} + N^{ack}_{links\_held}, \tag{9}$$

where, with the same approach used to derive expressions (6) and (8), it is possible to write

$$N^{ack}_{links\_held} = N_{nodes}\lambda M T_{ack}, \tag{10}$$

$$N^{rel}_{links\_held} = N_{nodes} \sum_{i=1}^{M-1} \lambda_{REL(i)} i T_{rel}, \tag{11}$$

$$N^{con}_{links\_held} = N_{nodes} \sum_{i=1}^{M} \lambda_{CON(i)} i T_{con}, \tag{12}$$

$$N^{arb}_{links\_held} = N_{nodes} \sum_{i=1}^{M} \sum_{j=1}^{M+1-i} \lambda_{A(i,j)}(i-1)T_{arb}. \tag{13}$$

Now, the terms $\lambda_{REL}(i)$, $\lambda_{CON}(i)$ and $\lambda_A(i,j)$ in expressions (11), (12) and (13) have to be evaluated. To this purpose Fig. 1 is used to help in finding the communication flows among the states.

Denoting with λ_{IN} the incoming flow of state $A_{1,1}$, i.e.

$$\lambda_{IN} = \lambda(1 + N_{aborts}), \tag{14}$$

then the outcoming flow of state $A_{1,1}$ due a link conflict is equal to (note that this is the incoming flow to state $A_{1,2}$):

$$\lambda_{A(1,1)} = \lambda_{IN} P_{link\_conflict} \tag{15}$$

and the outcoming flows of the other states $A_{i,j}$ due a link conflict is equal to

$$\lambda_{A(i,1)} = \lambda_{CON(i-1)} P_{link\_conflict} \quad (i = 2, \ldots, M), \tag{16}$$

$$\lambda_{A(i,j)} = \lambda_{A(i,j-1)} P_{link\_conflict} \quad (i = 1, \ldots, M,\ j = 2, \ldots, M+1-i), \tag{17}$$

where the incoming flows to the connection states ($\lambda_{CON}(i)$) is equal to

$$\lambda_{CON(i)} = \sum_{j=1}^{M+1-i} \lambda_{C(i,j)} \quad (i = 1, \ldots, M) \tag{18}$$

in which

$$\lambda_{C(1,1)} = \lambda_{IN}(1 - P_{link\_conflict}), \tag{19}$$

$$\lambda_{C(i,1)} = \lambda_{CON(i-1)}(1 - P_{link\_conflict}) \quad (i = 2, \ldots, M), \tag{20}$$

$$\lambda_{C(i,j)} = \lambda_{A(i,j-1)}(1 - P_{link\_conflict}) \quad (i = 1, \ldots, M,\ j = 2, \ldots, M+1-i). \tag{21}$$

Finally, the incoming flows to link-releasing states due a conflict ($\lambda_{REL}(i)$) is equ to

$$\lambda_{REL(M-1)} = \lambda_{A(M,1)}, \qquad (2$$

$$\lambda_{REL(i)} = \lambda_{REL(i+1)} + \lambda_{A(i,M+1-i)} \quad (i = 1, \ldots, M-2). \qquad (2$$

The last unknown term in expression (14) is N_{aborts} (i.e. the number of abort This quantity can be evaluated easily by the flow balance given the stationa assumption: the incoming and the outcoming flow of the path_set_up phase a equal; i.e. (see Fig. 1)

$$\lambda = \lambda_{C(M,1)} = \lambda_{CON(M)} \qquad (2$$

from which it is possible to obtain the N_{aborts}. Indeed, from expression (24) it possible to derive that

$$\lambda = \lambda_{IN}(1 - P_{link\_conflict})^M \prod_{j=1}^{M-1} \sum_{i=0}^{j} P^i_{link\_conflict}$$
$$= \lambda(1 + N_{aborts})(1 - P_{link\_conflict})^M \prod_{j=1}^{M-1} \sum_{i=0}^{j} P^i_{link\_conflict} \qquad (2$$

and therefore

$$N_{aborts} = \frac{1 - (1 - P_{link\_conflict})^M \prod_{j=1}^{M-1} \sum_{i=0}^{j} P^i_{link\_conflict}}{(1 - P_{link\_conflict})^M \prod_{j=1}^{M-1} \sum_{i=0}^{j} P^i_{link\_conflict}}. \qquad (2$$

Knowing N_{aborts}, it is also possible now to evaluate N_{arb}, N_{con}, and N_{rel} in (2 Indeed, denoting with $NA(i,j)$ the number of visits to the state $A_{i,j}$ with $NC(i)$ t number of visits to the state C_i and with $NR(i)$ the number of visits to state R_i (s Fig. 1), then we can write

$$N_{arb} = \sum_{i=1}^{M} \sum_{j=1}^{M+1-i} NA(i,j), \qquad (2$$

$$N_{con} = \sum_{i=1}^{M} NC(i), \qquad (2$$

$$N_{rel} = \sum_{i=1}^{M-1} NR(i), \qquad (2$$

where for the stationary assumption

$$NA(i,j) = \frac{\lambda_{A(i,j)} + \lambda_{C(i,j)}}{\lambda} \quad (i = 1, \ldots, M,\ j = 1, \ldots, M+1-i), \qquad (30$$

$$NC(i) = \frac{\lambda_{CON(i)}}{\lambda} \qquad (i = 1, \ldots, M), \qquad (31$$

$$NR(i) = \frac{\lambda_{REL(i)}}{\lambda} \qquad (i = 1, \ldots, M-1). \tag{32}$$

Note that the T_{back_off} value is present only in Eq. (2). This is due for two reasons. First, the communications in back-off state (due an abort) have no links, therefore they cannot contribute in expressions related to Eq. (3) (see expressions (6)–(13) in which the number of links held by communications is multiplied by the incoming rate and the average residence time). Second, given assumption (f), the probability of conflict with the communications that caused previous aborts is equal to zero. This could not be true in the reality, but the bigger the back-off time, the lesser this probability. In the simulation experiments (see the following section) we see that this probability can be neglected if the back-off time value is equal or bigger than the average data transmission time.

4. Results and discussions

To validate our modelling, in this section we compare the results of the analysis for the "adaptive" strategy with data obtained by simulations. We then discuss the perfomance behaviour of the "adaptive" and "e-cube hold" strategies (the analytical modelling of the e-cube hold strategy has been carried out in another paper [11]). We have no measurements of a real hypercube, therefore we have no realistic communication parameter values. For this reason, the only way to see the performance behaviour of the two routing strategies is to verify their sensitivity to the variation of the communication parameter values (note that these ones are given in unit_times, denoted as u_times in the following figures). In particular the performance behaviours are analyzed changing the data transfer time distribution, the back-off time value, the dimension of the hypercube and the fraction of the path set-up and releasing phases with respect to the average data transmission time.

The "adaptive" strategy has been simulated by the "Independent Replications Method" [24], using models written in SIMULA '67 language. The confidence interval is based on the Jackknife estimator [26].

The model explicitly simulates the activity of link arbitration, link connection, link releasing, and data transmission, from which occur the effects of the link contentions and aborts. The simulation model is developed only with (a), (b) and (c) assumptions (see Section 3); assumptions (d), (e) and (f) have been instead removed. In case of conflict, the arriving communication request is aborted and after the release of the obtained links, is put in a waiting state for a time equal to a back-off period.

The simulations run on an IBM mainframe, and to obtain a single estimation, the simulator runs from 500 s CPU time to 50 000 s of CPU time, depending on the intensity of the network traffic modelled. Instead, the algorithm based on the analytical model runs only a few milliseconds to obtain a single estimation.

Fig. 2 shows the sensitivity of the "adaptive" strategy to the variation of the back-off time T_{back_off}. In particular it shows the average number of aborts per communica-

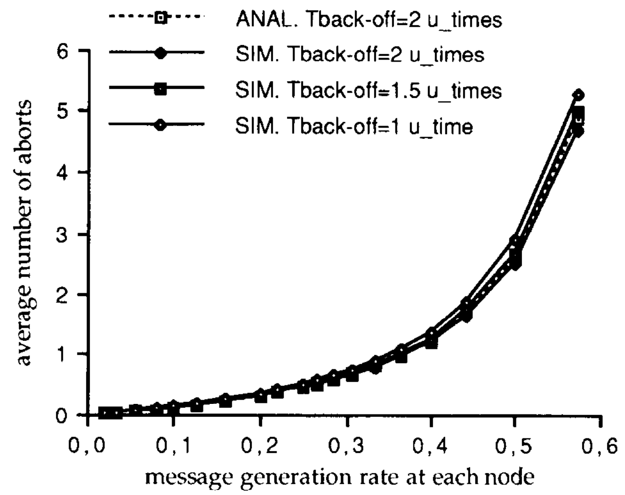

Fig. 2. Simulation and analytical average number of aborts per communication versus communication rate for a hypercube with $d = 10$, adopting the "adaptive" strategy and three different back-off times.

tion versus the message generation rate at each node. The graphs are related to the data obtained by the approximate analysis and by the simulation. We have presented only the mean value because the representation of the confidence interval (although the data is at a 95% level of confidence) is very narrow and may cause confusion among the graphs. The graphs are related to a hypercube with 1024 nodes ($d = 10$), the communication requests are a Poisson process, the message routing is uniformly distributed, moreover, data transfer time is distributed with negative exponential distribution (the average length is equal to $T_{data} = 1$ unit_time).

There are four graphs, they are obtained with $T_{arb} = T_{con} = T_{ack} = T_{rel} =$ 0.001 unit_times, one of these is related to the analytical model (in which T_{back_off} is fixed to 2 unit_times; however, given the assumption (f), in the analytical model there is no effect of the T_{back_off} value on the number of aborts) and the other three to simulation runs with a T_{back_off} equal to 2 unit_times, 1.5 unit_times and 1 unit_time, respectively.

When comparing the data obtained by the simulation and the analytical models of Fig. 2, it is possible to achieve a moderately better approximation of the analytical model increasing the back-off time of the simulation experiments. Indeed, the average number of aborts per communication estimates derived by the simulation and the analytical model are very close. The difference between the analysis and the average value of the simulation is within $1 \div 3\%$ for a wide range of message arrival rates; it is however less than 5% near the interconnection network saturation. Moreover, the difference between the data obtained by the analytical model and by the simulation with a back-off time equal to 1 unit_time is bigger than the difference between data obtained by the other two simulations and the analytical model. This phenomenon is due to the fact that the analytical model does not take into account the probability that the communication requests which caused the first abort are still in progress after the back-off time, and, therefore, in the simulations with reduced back-off time, the contention phenomenon with communication requests still in progress after the back-off time becomes to be manifest.

Fig. 3 shows the sensitivity of the "adaptive" strategy to the variation of the data transfer time distribution. The graphs are related to a hypercube with 1024 nodes, $T_{back\_off} = 1.5$ unit_times, $T_{arb} = T_{con} = T_{ack} = T_{rel} = 0.001$ unit_times, the communication requests are a Poisson process and the message routing is uniformly distributed. The first graph is related to a data transfer time distributed with negative exponential distribution (the average length is equal to $T_{data} = 1$ unit_time); the second is related to a data transfer time distributed with uniform distribution (the interval is between 0.1 and 1.9 unit_times, therefore $T_{data} = 1$ unit_time); the third graph is related to a fixed data transfer time ($T_{data} = 1$ unit_time).

From Fig. 3 it is possible to see that the representativeness of the analytical model is independent of the message length distribution. The same agreements between analytical model and simulation experiments are verifiable for the "e-cube hold" strategy. For the sake of room, we do not present them [11].

At this point we present some performance projections deriving from the approximated analysis for the two different strategies. In the following, the graphs have been obtained for a data transfer time distributed with negative exponential distribution. The analytical model is useful only if the probability that the communication requests which caused the first abort are still in progress after the back-off time is negligible; therefore, the following graphs are obtained for experiments in which this probability need not be taken into account.

Fig. 4 shows the effect on the message latency versus communication transfer request rate for a hypercube with dimension $d = 10$. The communication parameters

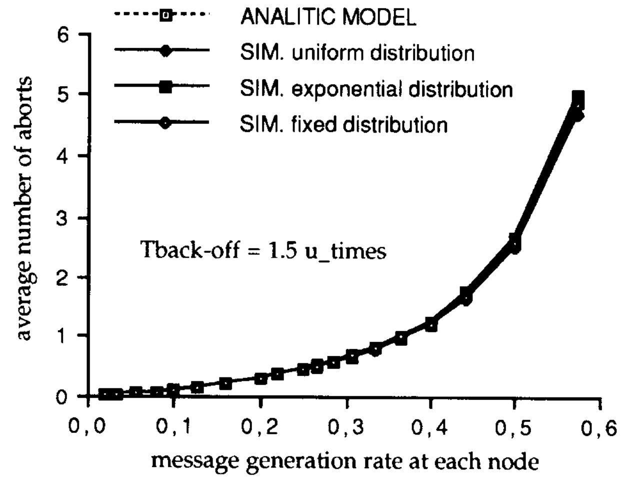

Fig. 3. Simulation and analytical average number of aborts per communication versus communication rate for a hypercube with $d = 10$, adopting the "adaptive" strategy and back-off time = 1.5 unit_times, for three different data transfer data distributions.

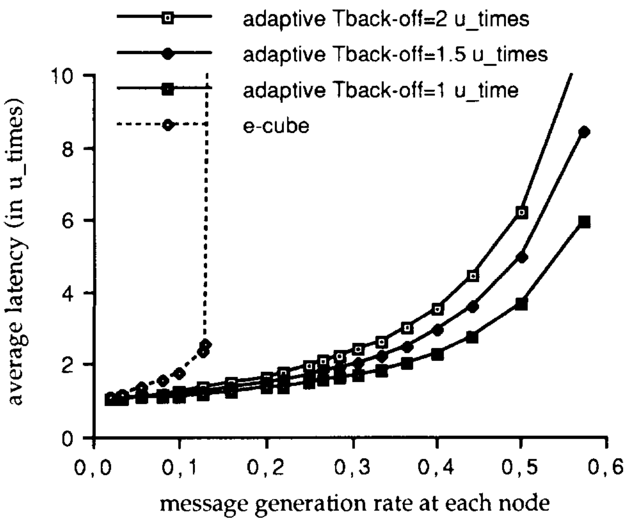

Fig. 4. Message latency versus communication rate for hypercube with $d = 10$ adopting the "e-cube hold" or the "adaptive" strategy.

considered are: $T_{data} = 1$ unit_time; $T_{arb} = T_{con} = T_{ack} = T_{rel} = 0.001$ unit_times. While for the "e-cube hold" strategy there is only one graph, for the "adaptive" strategy we derived three graphs: the first is obtained for a $T_{back\_off} = 2$ unit_times (i.e. the double of the average data transfer time), the second for a $T_{back\_off} = 1.5$ unit_times, the third for a $T_{back\_off} = 1$ unit_time (i.e. equal to the average data transfer time). Fig. 4 shows that for both link conflict resolution strategies there is a saturation present for the message latency. From the graphs, it may be seen that for the "adaptive" strategy, saturation will generally arise with higher values of message generation rate compared to "e-cube" ones.

Fig. 5 shows the message latency versus the message transfer request rate for three different hypercube dimensions (namely $d = 8$, $d = 10$ and $d = 12$, respectively). The graphs are obtained using the same communication parameters used for Fig. 4 and with a $T_{back\_off} = 2$ unit_times. From the graphs, the saturation phenomenon of the message latency for the three different hypercube dimensions may be seen. With "e-cube hold" strategy this phenomenon is due to the fact that with an increase in the diameter, there is a corresponding increase in the average number of links per path (M). Consequently, the link waiting time probability during the path set up will rise. Whereas in "adaptive" strategy, the bigger the average number of links per path, the greater the probability of abort.

Finally, Fig. 6 shows the sensitivity of the two routing strategies to the variation of the fraction of the path set-up and releasing phase parameters with respect to the average data transmission time. In particular Fig. 6 shows the average message latency versus the message generation rate for three different sets of path set-up and releasing phase parameters for a hypercube of 1024 nodes, $T_{back\_off} = 2$ unit_times and $T_{data} = 1$ unit_time. The other parameters of the first set are fixed as $T_{arb} = T_{con} =$

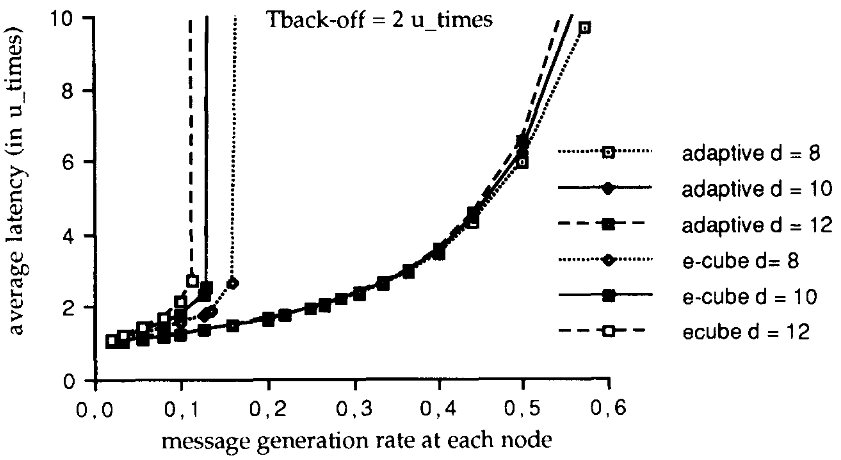

Fig. 5. Message latency versus communication rate for hypercube adopting the "e-cube hold" or "adaptive" strategy and for three different hypercube dimensions.

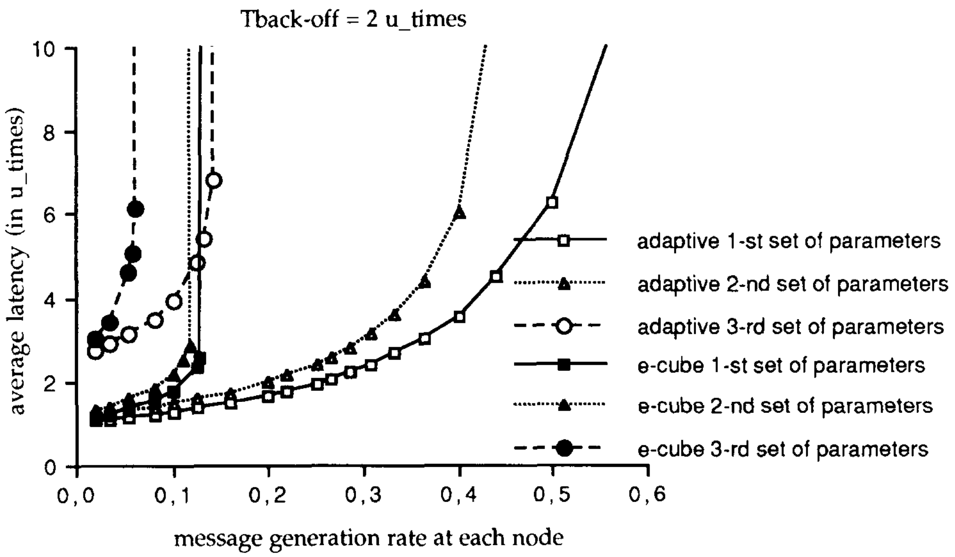

Fig. 6. Message latency versus communication rate for hypercube adopting the "e-cube hold" or "adaptive" strategy and for three different sets of communication parameters.

$T_{ack} = T_{rel} = 0.001$ unit_times; the second as $T_{arb} = T_{con} = T_{ack} = T_{rel} = 0.01$ unit_times and the third as $T_{arb} = T_{con} = T_{ack} = T_{rel} = 0.1$ unit_times. From the graphs it is possible to see that the adaptive strategy always performs better than the e-cube one; however, the sensitivity of the e-cube to the variation of the path set-up and releasing phases parameters is lesser than the former.

Concluding, from the evidence of the graphs, the better behaviour of the "adaptive" strategy with respect to the "e-cube hold" strategy is verifiable (the interconnection network saturation phenomena arises when the message arrival rate has aproximately quadruplicated). However, this better behaviour is obtained by increasing the complexity of the router module. Indeed, in this case, it is necessary to have a more complex logic for path construction (see the chip router proposal in [13]) and a message buffer in order to manage the aborted communication requests during the back-off period. The dimension of this buffer depends on the traffic congestion and on the back-off period value.

5. Concluding remarks

In this paper we introduced an iterative modelling approach that can be used to evaluate the average message latency of asynchronous direct hypercube circuit switching interconnection networks based on an "adaptive" strategy. The model also takes into account the link acquisition and releasing time. The main peculiarity of the model (due to its iterativity) is that it permits the capturing of the feedback effect between the probability of conflict for a link and the message latency. Indeed, longer message latency leads to a greater probability of link conflict (and consequently greater probability of abort), which in turn increases the message latency. Some of the assumptions used to build our model are generally accepted in the current literature and other ones are done to obtain a suitable mathematical model. To validate our approach a simulation model has also been developed in which the latter assumptions have been removed.

We have also shown that the methodology may be applied for a wide range of message traffic load, independent of the message length distribution, but the analytical model is useful only if the probability that the communication requests which caused the first abort are still in progress after the back-off time is negligible (from simulation experiments we found that this probability can be neglected if the back-off time value is equal or bigger than the average data transmission time).

We demonstrate the better behaviour of the "adaptive" strategy with respect to the "e-cube hold" strategy, currently used by commercial multicomputers (the modelling of the e-cube hold strategy has been carried out in another paper [11]). This better behaviour is however obtained by increasing the router module complexity.

Finally, we remark that although our model is developed for asynchronous direct circuit switching networks, the same approach can be extended to study all asynchronous direct or indirect point to point communication networks characterized by symmetric topologies (torus-mesh, k-ary, n-cube, etc.) and in general to all the "environments" in which there is contention on "passive" homogeneous resources. This approach cannot be used for asymmetric interconnection networks (like mesh) because it assumes that the load is the same in all the links of the network.

Further researches are focused on extending this methodology to study point-to-point interconnection networks based on the wormhole and virtual-cut-through routing strategies.

Acknowledgments

We gratefully acknowledge the contributions of G.F. Calavaro in implementing the simulation programs. Authors are also grateful to three anonymous referees for their constructive suggestions that greatly improved the presentation of the paper.

Appendix. Evaluation of the average number of links per path

In case of uniformly distributed message routing, denoting by $M(i)$ the number of nodes reachable in i steps from a generic node, the average number of links per path is equal to

$$M = \sum_{i=1}^{d} \frac{iM(i)}{2^d - 1}$$

in which $2^d - 1$ is the number of nodes of the d-dimensional hypercube reachable by the generic sender.

Given a node address coded in binary form, the number of node addresses that differ by exactly i bits is

$$M(i) = \binom{d}{i}.$$

Indeed, given a node address coded in binary on d bits, the number of different node addresses having a Hamming distance equal to i, is equal to the number of possible combinations of class i on d positions. Therefore,

$$M = \frac{d2^{d-1}}{2^d - 1}.$$

References

[1] S. Abraham and K. Padmanabhan, Performance of direct binary n-cube network for multiprocessors, *IEEE Trans. Comput.* **38**(7) (1989) 1000–1011.

[2] V.S. Adve and M.K. Vernon, Performance analysis of mesh interconnection networks with deterministic routing, *IEEE Trans. Parallel Distributed Systems* **5**(3) (1994) 225–246.

[3] A. Agarwal, Limits on interconnection network performance, *IEEE Trans. Parallel Distributed Systems* **2**(4) (1991) 398–412.

[4] L.N. Bhujan, Q. Yang and D.P. Agrawal, Performance of multiprocessor interconnection networks, *IEEE Comput.* (2) (1989) 25–37.

[5] M.S. Chen and K.G. Shin, Adaptive fault-tolerant routing in hypercube multicomputers, *IEEE Trans. Comput.* **39**(12) (1990) 1406–1416.

[6] A.A. Chien and J.H. Kim, Planar-adaptive routing: Low-cost adaptive networks for multiprocessors, in: *Proceedings of the 19th ACM International Symposium on Computer Architecture*, Gold Coast, Australia (1992) 268–277.

[7] S. Chittor and R. Enbody, Performance evaluation of mesh-connected wormhole-routed networks

for interprocessor communication in multicomputers, in: *Proceedings of ACM Supercomputing*, New York, USA (1990) 647–656.

[8] I. Chlamtac, A. Ganz and M. Kienzle, A performance model of a connection based hypercube interconnection system, in: *Proceedings of the 5th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, Torino, Italy (1991) 94–103.

[9] B. Ciciani, D.M. Dias and P.S. Yu, Analysis of replication in distributed database system, *IEEE Trans. Knowledge and Data Engineering* **2**(2) (1990) 247–261.

[10] B. Ciciani, D.M. Dias and P.S. Yu, Analysis of concurrency-coherency control protocols for transaction processing systems with regional locality, *IEEE Trans. Software Engineering* **18**(10) (1992) 899–914.

[11] B. Ciciani and S. Tucci, On modeling link conflict resolution strategies for circuit switching hypercubes, in: *Proceedings of IEEE CompEuro*, Bologna, Italy (1991) 662–666.

[12] W.J. Dally, Performance analysis of *k*-ary *n*-cube interconnection networks, *IEEE Trans. Comput.* **39**(6) (1990) 775–785.

[13] N.J. Dimopoulos, D. Radvan and K.F. Li, Performance evaluation of the backtracking-to-the-origin-and-retry routing for hypercycle-based interconnection networks, in: *Proceedings of IEEE Distributed Computing Systems Conference*, Paris, France (1990) 278–284.

[14] C.J. Glass and L.M. Ni, The turn model for adaptive routing, in: *Proceedings of the 19th ACM International Symposium on Computer Architecture*, Gold Coast, Australia (1992) 278–287.

[15] P.G. Harrison and N.M. Patel, The representation of multistage interconnection networks in queueing models of parallel systems, *J. ACM* **37**(4) (1990) 863–898.

[16] A.J.G. Hey, Supercomputing with transputers – Past, present and future, in: *Proceedings of ACM Supercomputing Conference*, New York, USA (1990) 222–227.

[17] IMS C104 packet routing switch, Preliminary Information, (SGS-THOMSON, 1991).

[18] *Intel iPSC System Overview* (Intel Scientific Computers, 1986).

[19] D.D. Kandlur and K.G. Shin, Traffic routing for multicomputer networks with virtual cut-through capability, *IEEE Trans. Comput.* **41**(10) (1992) 1257–1270.

[20] P. Kermani and L. Kleinrock, Virtual cut-through: A new computer communication switching technique, *Computer Networks* **3** (1979) 267–286.

[21] J. Kim and C.R. Das, Hypercube communication delay with wormhole routing, *IEEE Trans. Comput.* **43**(7) (1994) 806–814.

[22] I. Koren and Z. Koren, Discrete and continuous models for the performance of reconfigurable multistage systems, *IEEE Trans. Comput.* **40**(9) (1991) 1024–1033.

[23] C.A. Lamanna and W.H. Shaw, A performance study of the hypercube parallel processor architecture, *Simulation* **53**(3) (1991) 185–196.

[24] S.S. Lavenberg, *Computer Performance Modeling Handbook* (Academic Press, New York, 1983).

[25] D. Linder and J.C. Harden, An adaptive and fault tolerant wormhole routing strategy for *k*-ary *n*-cubes, *IEEE Trans. Comput.* **40**(1) (1991) 2–12.

[26] R.J. Miller, The Jackknife – A review, *Biometrica* **61**(1) (1974) 1–15.

[27] nCUBE 2 Supercomputer System, Technical Overview (1990).

[28] L.M. Ni and P.K. McKinley, A survey of wormhole routing techniques in direct networks, *IEEE Comput.* (2) (1993) 62–76.

[29] S.F. Nugen, *The iPSC/2 Direct-Connect Communications Technology* (Intel Scientific Computers, 1988).

[30] K. Padmanabham and D.H. Lawrie, Performance analysis of redundant-path networks for multiprocessor systems, *ACM Trans. Comput. Systems* **3**(2) (1985) 117–144.

[31] K. Padmanabham, Cube structures for multiprocessor, *Comm. ACM* **33**(1) (1990) 43–52.

[32] J.L. Peterson and A. Silberschatz, *Operating System Concepts* (Addison-Wesley, Reading, MA, 1985).

[33] J. Peterson, E. Chow and H. Madan, A high-speed message-driven communication architecture, in: *Proceedings of the ACM International Conference on Supercomputing*, St. Malo, France (1988).

[34] D.A. Reed and R.M. Fujimoto, *Multicomputer Networks: Message-Based Parallel Processing*, Scientific Computation Series (MIT Press, Cambridge, MA, 1987).

[35] D.A. Reed and D.C. Grunwald, The performance of multicomputer interconnection network, *IEEE Comput.* (6) (1987) 63–73.
[36] C.L. Seitz, The cosmic cube, *Comm. ACM* **28**(1) (1985) 22–33.
[37] G.V. Wilson, A glossary of parallel computing terminology, *IEEE Parallel Distributed Technology* (2) (1994) 52–64.
[38] C.L. Wu and M. Lee, Performance analysis of multistage interconnection network configurations and operations, *IEEE Trans. Comput.* **41**(1) (1992) 18–26.