ELSEVIER

Performance Evaluation 34 (1998) 1–26

Performance analysis of circuit-switching interconnection networks with deterministic and adaptive routing

M. Colajanni [a], B. Ciciani [b], S. Tucci [a,*]

[a] *Dipartimento di Informatica, Sistemi e Produzione, Università di Roma – Tor Vergata, Roma, 00133 Italy*
[b] *Dipartimento di Informatica e Sistemistica, Università di Roma – La Sapienza, Roma, 00198 Italy*

Received 24 October 1995; received in revised form 25 March 1998

Abstract

This paper compares three link conflict resolution strategies applied to multicomputers with symmetric topologies and circuit-switching interconnection networks. Several performance parameters are evaluated through an approximate analytical model based on the flow analysis. The main peculiarity of this method with respect to previous studies is the capacity to take into account actual network delays and all feedback effects among probability of link conflict, routing controller overhead, and message latency. An extensive simulation analysis has been carried out to validate the analytical models. The results show that our approach is quite accurate for a wide range of message traffic loads, independently of the link conflict resolution strategy and message length distribution.

Keywords: Approximate analytical model; Circuit-switching routing; Multicomputers; Message passing

1. Introduction

The growing demand for high speed interconnection networks requires a proper evaluation of their performance indices by means of models that take into account all actual delays. Several studies have paid attention to the performance, efficiency and reliability analysis of interconnection networks of distributed-memory parallel systems. The reason is that communications typically represent the bottleneck of computations running on multicomputers.

In this paper, we focus on direct interconnection structures characterised by asynchronous and decentralised control techniques [14,20]. Several switching techniques have been used in direct networks. The first generation multicomputers, such as Ncube-1 and iPSC/1, were characterised by *store and forward* message switching policies. To decrease high message latency and avoid large node buffers, the next generation multicomputers abandoned these strategies in favour of *circuit-switching* (adopted, for example, by Intel iPSC/2, iPSC/860, and an experimental Jet Propulsion Laboratory machine [21]) and *wormhole*

* Corresponding author. Tel.: +39 672 597385; fax: +33 672 597460; e-mail: tucci@uniroma2.it.

0166-5316/98/$ – see front matter
PII: S0166-5316(98)00023-6

routing techniques (used by Intel Paragon, MIT J-machine, Inmos Transputer IMS T9000). Message latency of these routing policies can be significantly affected by link contention, hence it is important to compare various link conflict resolution strategies. In particular, circuit-switching consists of three separated phases. In the *path-set-up phase*, a communication path from the source to the destination node is established. In the *data-transmission* phase, the message is transmitted across the links of the path. In the *path-release phase*, all acquired links are released. Therefore, link contention may occur only in the path-set-up phase.

The most common policy is the deterministic *hold* strategy, in which the path is determined in a static way and, in case of link conflict, the communication request waits until that link becomes free. This strategy has an easy implementation, however it suffers from low reliability and low performance especially when the traffic is heavy. Indeed, a crowded or faulty link prevents any message exchange among the nodes having that link in their path. Alternative link conflict resolution strategies are the *drop* (or *loss mode*) policy that releases all links obtained before the conflict, and the *adaptive* policy that tries to establish another path. (This latter policy preserves communications even in the presence of some faulty paths [4,11,19,26].) The goal of this paper is to investigate the performance of hold, drop, and adaptive policies in the case of symmetric topologies, such as hypercube and torus, with different dimensions and various traffic conditions. An approximate analytical model is introduced to carry out this comparison.

It should be observed that circuit-switching policies are gaining a new interest, thanks to the advent of optical interconnection networks. In these systems, it is convenient to establish the entire path before transmitting a message to the purpose of avoiding data buffering that would require costly optical–electronic conversion processes [25]. Moreover, other simulation experiments showed that, for many scenarios, adaptive circuit-switching has performance comparable to that of wormhole techniques [8].

The paper is organised as follows. Section 2 discusses related work. Section 3 describes the operational features of circuit-switching routing in a hypercube. Section 4 analyses the models for the message latency time evaluation in the case of three link conflict resolution policies. Section 5 indicates how to extend the equations for the hypercube to a symmetric torus topology. Section 6 validates the analytical results against simulation and compares the performance of the link conflict resolution strategies. Section 7 concludes the paper with some final remarks. For the sake of readability, some equations are solved in the appendices and discussed with more details in [7].

2. Related work

Most performance results concerning asynchronous direct and indirect communication networks are obtained through simulation models [2,4,10,12,13,18,21,22]. The few analytical results for circuit-switching are restricted to the hold strategy and assume instantaneous link acquisition/releasing times [1,5,9,14]. To the best of our knowledge, there are no analytical solutions for circuit-switching networks that use adaptive or drop link conflict resolution strategies. Harrison and Patel [14] evaluate the performance of indirect circuit-switching networks by means of decomposition techniques based on a Markov process. Kim and Das propose a model for the analysis of a random switching technique that is not actually adaptive [16]. This routing algorithm requires a deadlock detection strategy because, at each step, a random link is selected. Dally [9], Adve and Vernon [1] use approximate Mean Value Analysis to analyse the performance of wormhole routing in k-ary n-cubes networks. Chlamtac et al. [5] evaluate the performance of circuit-switching routing in a hypercube through an approximate Markov model. This approach is fine for the analysis of the hold strategy, however it cannot be applied to the drop strategy that does not satisfy the memory-less property.

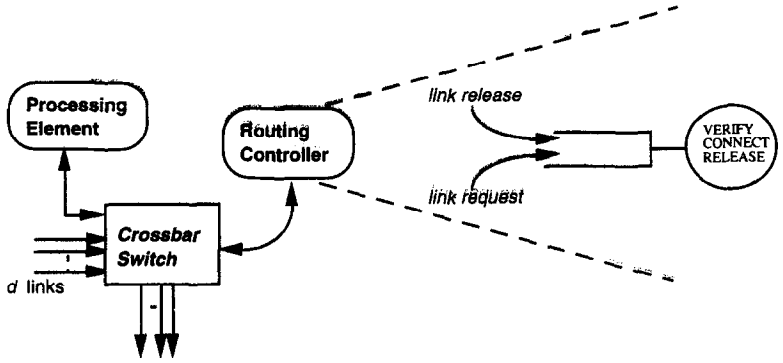


Fig. 1. The node architecture.

In this paper we propose an approximate modelling approach that can be used to evaluate the message latency in asynchronous circuit-switching interconnection networks where the link contention is solved through hold, drop or adaptive policy. The model also takes into account the overheads of the routing controllers such as link acquisition, link release, and queuing time of the requests (in the hold case). The novelty of the model is twofold. It represents a unifying approach for modelling all link conflict resolution strategies. It is able to capture the feedback effect of the link conflict probability on message latency and routing controller overhead. The same methodology can be extended to analyse other systems with contention on passive homogeneous resources [6] and indirect interconnection networks. Unlike direct networks in which each processor has a router, the routers of the indirect networks are separated from the processors. In principle, our approach can be applied to both kinds of systems because it models the communication path as a set of links and routers, independently of the number of processors. The main difference is that in direct networks we have to consider the mean distance between two nodes, while in indirect networks we have to evaluate the mean number of network stages.

3. The model

3.1. Network operation

To obtain a model analytically tractable we consider interconnection networks with symmetric topologies. For the sake of readability we focus on the binary hypercube topology that has been adopted in several multicomputers such as Cosmic Cube, Connection Machine, Ncube, iPSC/2, and iPSC/860. Nevertheless, the approach can be extended to other symmetric topologies such as the torus (see Section 5).

A d-dimensional binary hypercube is a graph of $p = 2^d$ processors and $2^{d-1}d$ links. A path between the source and destination is a sequence of nodes in which every two consecutive processors are physically adjacent to each other [3]. 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 said to be *optimal*. Each node of the interconnection network has three components: a *processing element* (called processor), a *routing controller*, and a $(d+1) \times (d+1)$ crossbar switching network (Fig. 1). A DMA controller (DMAC) per each dimension guarantees a maximum of d simultaneous communications for each router.

The path from the source to the destination node is completed by a step-by-step process, in which for each step an additional link is assigned to the path by the local routing controller only after verification of the availability of the required link. In the case of conflict, the links held by the communication request may be released or not, depending on the adopted link conflict resolution strategy. We analyse three policies that build optimal paths. In the case of hold strategy, the colliding communication request is en-queued at the local routing controller, and the request awaits the release of the link. In the case of drop strategy, all the acquired links are immediately released. To reduce the probability of subsequent aborts, the source router waits for a *back-off* period before attempting to re-establish the path. The *adaptive strategy* is a dynamic policy that, in the case of conflict, randomly selects one of the free links (if any) that forms the shortest path from the source to destination. If no link is free, the partial path is aborted and the new attempt comes after a *back-off* period, analogously to the drop strategy.

When the communication path is completed, the DMAC of the destination routing controller sends an ack signal to the DMAC source through the established path. At the arrival of this signal, the DMAC source is enabled to start the transmission of the message. At the end of the data transfer, an end-transmission signal propagated through the path enables the intermediate routing controllers to release the acquired links. It should be observed that our model takes into account all network operation features. In particular, the path is set up at the routing controller level, hence each link request experiments a certain service time and possible router queuing delays. On the other hand, message transmissions do not require any intervention of the routing controllers because they are carried out at the DMA level.

We assume that hold and drop strategy establishes the communication path through the e-cube routing algorithm which has been demonstrated to be deadlock-free. The drop and adaptive strategies are deadlock-free by definition since the hold and wait condition is not met.

3.2. Assumptions

Most parameters of interest to the model depend on the probability that a request for a link may enter into conflict against another active communication. The difficulty of evaluating the probability of link conflict for the three strategies is mainly due to a mutual dependence. The link utilisation depends on the duration of the communication which consists of the path-set-up, data-transmission, and path-release phase. In its turn, the duration of the path-set-up phase depends on the probability of contention/abort and overhead of the routing controller. An iterative solution of the model allows us to capture these feedback effects for all the link conflict resolution strategies. To build an analytically tractable model we make the following assumptions.

(a) The communication requests arising from each processor are independent and identically distributed in accordance with a Poisson process having rate λ for each processor.
(b) The memory capacity of each processor is unlimited. This assumption is quite accurate if one considers present available machines. It permits to neglect the loss of messages and consequent re-transmission due to the lack of memory space.
(c) For the *hold* strategy, a link conflict implies that no other requests are waiting for the same link. This allows the model to consider the probability of link conflict only in terms of link utilisation.
(d) The probability of link conflict is independent of the number of already obtained links. This makes the probability of conflict for the ith link request independent of the i value.
(e) For the *hold* strategy, the waiting times due to contention are independent variables with same mean. This permits the analytical evaluation of the mean waiting times to obtain a link.

(f) For the *adaptive* strategy, the probability of link conflict is independent of the number of occurred aborts. This allows to consider null the probability of new conflicts with the communication that caused the previous abort. This hypothesis is reasonable for the adaptive policy given the presence of a combinatorial number of paths from the source to destination, but it is unrealistic for the drop strategy that provides a single path.

In accordance with the current literature [23], there are no assumptions about the message routing distribution. Some assumptions, such as (a) and (b), are commonly accepted. The main consequence of the other assumptions is that the analytical results are lower bounds of the actual times because some potential conflicts are not taken into account. However, their impact do not affect the accuracy of the solutions, as demonstrated by the validation of the analytical results against simulation models not containing assumptions (c), (d), (e) and (f).

The assumption (a) and the network symmetry guarantee that the network is *balanced* [15] that is, each routing controller and link is equally likely to be visited independently of the message routing distribution.

The link request of a single communication is generated by a Poisson source with rate λp because the superposition of Poisson processes is still a Poisson process. Moreover, from assumption (a) and network symmetry we have that the mean number of links per path M is the same for all processors, independently of their positions. Therefore, the mean latency of each message may be simply evaluated by considering a communication request for the average path. For example, in the case of uniformly distributed message routing in a d-dimensional hypercube, the number of nodes reachable in i steps from a node corresponds to the number of node addresses that differ by exactly i bits, that is, $A(i) = \binom{d}{i}$. Hence, the mean number of links per path is:

$$M = \frac{\sum_{i=1}^{d} i A(i)}{2^d - 1} = \frac{d2^{d-1}}{2^d - 1}, \tag{1}$$

where $2^d - 1$ is the number of nodes reachable by a generic sender. A similar approach can be used to evaluate the mean number of links per path for different message routing distributions, such as uniform, sphere of locality, and decreasing probability.

In circuit-switching networks, the communication evolves through the path-set-up, data-transmission and path-release phases. Therefore, the mean message latency is the sum of the mean time to set up the path ($T_{\text{path\_set\_up}}$), the mean time to transfer the body message ($T_{\text{data\_trans}}$), and the mean time to release the links of the path ($T_{\text{path\_release}}$), that is,

$$T_{\text{latency}} = T_{\text{path\_set\_up}} + T_{\text{data\_trans}} + T_{\text{path\_release}}. \tag{2}$$

The last two terms in (2) are independent of the link conflict resolution strategy. In particular, $T_{\text{data\_trans}}$ is given by the mean message length divided by the network bandwidth; $T_{\text{path\_release}}$ is equal to MT_{rel}, where T_{rel} is the mean time to release one link and depends on the routing controller overhead. Each link conflict resolution policy is characterised by a different time to build the path. In the following sections we evaluate $T_{\text{path\_set\_up}}$ for the hold, drop and adaptive strategies, respectively.

Two remarks on the notation are in order: D and T denote deterministic and mean time values, respectively; all common terms, such as P_{conflict} and N_{abort}, are section dependent in the sense that they refer to the conflict resolution strategy where they are used.

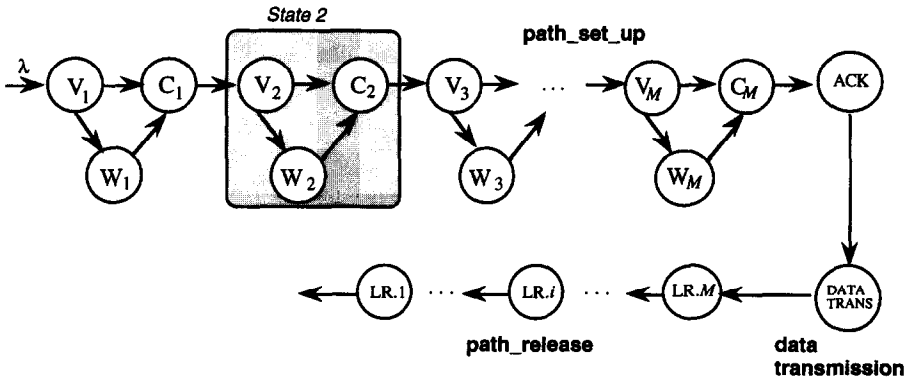


Fig. 2. State diagram for the *hold* strategy (V = Verification, W = Waiting, C = Connection).

4. Message latency analysis

4.1. Hold strategy

The state diagram that describes the operation of the hold strategy is represented in Fig. 2. In the path-set-up phase, a state i ($i \neq$ ACK) denotes a communication request that involves the ith link. This state is composed of three sub-states V_i, C_i, W_i, where V_i denotes the request for the ith link to the local routing controller. If this link is obtained, the transition to state C_i takes place. Otherwise, if there is contention, the waiting state W_i is reached. Once obtained all the M links, the DMAC receiver transmits to the DMAC sender an ack that signals the completion of the path-set-up phase (ACK state). In the data-transmission phase (DATA_TRANS state), the DMAC sender transmits the message body to the DMAC receiver. In the path-release phase, all links of the communication path are released. The index i of the LR.i state denotes the number of links that are not yet released. The mean time to acquire M links, that is, the duration of the path-set-up phase, is given by the sum of the following terms:

$$T^{\text{HOLD}}_{\text{path\_set\_up}} = M(T_{\text{verify}} + D_{\text{link\_conn}} + T_{\text{link\_wait}}) + D_{\text{ack}}, \tag{3}$$

where T_{verify} is the mean time to verify the availability of a link; $D_{\text{link\_conn}}$ and D_{ack} are the constant times to connect a link and transmit the ACK signal, respectively; $T_{\text{link\_wait}}$ is the mean time spent to wait for a link that is held by other communications.

Before the analysis of the $T_{\text{link\_wait}}$ term, let us focus on the three parameters that depend on the activities of the routing controller: T_{verify}, $D_{\text{link\_conn}}$, and T_{rel}. Previous evaluation studies on interconnection networks consider times for link verification, link connection, and link release as constant delays. This assumption does not allow the model to consider the activities and possible congestion of the routing controller. On the other hand, in our model T_{verify} and T_{rel} are considered mean response times given by a deterministic service time (D_{verify} and D_{rel}, respectively) plus a mean waiting time spent at the routing controller queue (Fig. 1). The time for a link connection $D_{\text{link\_conn}}$ is a pure delay because it immediately follows a link verification or a link release activity. The solution of the equations representing the activities of the routing controller, modelled as a M/G/1 queue, is postponed to Appendix A.

The time spent to wait for a link that is held by another communication depends on the link conflict probability and the mean residual time to release the requested link that is,

$$T_{\text{link\_wait}} = P_{\text{conflict}}^{\text{path\_set\_up}} W^{\text{path\_set\_up}} + P_{\text{conflict}}^{\text{data\_trans}} W^{\text{data\_trans}} + P_{\text{conflict}}^{\text{rel}} W^{\text{rel}}, \tag{4}$$

where $P_{\text{conflict}}^{\text{path\_set\_up}}$, $P_{\text{conflict}}^{\text{data\_trans}}$, $P_{\text{conflict}}^{\text{rel}}$ are the probabilities of conflicting with other communications that are in their path-set-up, data-transmission and path-release phase, and $W^{\text{path\_set\_up}}$, $W^{\text{data\_trans}}$, W^{rel} are the mean waiting times to obtain a link held by other communications in the corresponding phase. The evaluation of the probabilities of conflict and the waiting time is in Appendix B and Section 4.1.1, respectively.

4.1.1. Mean waiting time to obtain a busy link

The mean waiting time to obtain a busy link is equal to the mean residual time that the requested link is released by the communication holding it. By assuming that the conflict occurs while the other communication is in its x-phase, the mean waiting time W^x for a busy link is:

$$W^x = \sum_{k=1}^{M} P_k^x L_k^x R_k^x \Big/ \sum_{k=1}^{M} P_k^x L_k^x, \tag{5}$$

where P_k^x is the probability that the communication is in the state k of the x-phase, and L_k^x is the number of links held in the state k of the x-phase, and R_k^x is the residual holding time starting from the state k of the x-phase. The computation of R_k^x is in Section 4.1.2, while the evaluation of P_k^x and L_k^x is straightforward:

(i) For x = *path-set-up*, if one assumes that sub-states C_k, V_k, and W_k are aggregated into a single state (as example, see the shadow state in Fig. 2), then the probability of being in the state k is equal to

$$P_k^{\text{path\_set\_up}} = \frac{(T_{\text{path\_set\_up}}^{\text{HOLD}} - D_{\text{ack}})/M}{T_{\text{path\_set\_up}}^{\text{HOLD}}} \quad k = 1, \ldots, M$$

and in the ACK state is $P^{\text{ack}} = D_{\text{ack}}/T_{\text{path\_set\_up}}^{\text{HOLD}}$. The links held are approximately $L_k^{\text{path\_set\_up}} = k$, for $k = 1, \ldots, M$, and $L^{\text{ack}} = M$.

(ii) For x = *data-transmission*, we have $P^{\text{data\_trans}} = 1$ because the data-transmission phase consists of a unique state, and $L^{\text{data\_trans}} = M$.

(iii) For x = *path-release*, we have $P_k^{\text{rel}} = 1/M$ because the probability of residence in each of the M states is identical and equal to the mean time to release one link T_{rel}, and $T_k^{\text{rel}} = k$, for $k = 1, \ldots, M$.

4.1.2. Expected residual residence times

In this section we use assumption (c) stating that there are no other communications waiting for the same link requested by the blocked communication. Therefore, the expected residual time R_k^x to obtain the link is only due to the communication holding that link. This value depends on the state k of the x-phase in which the communication is.

Let $f_Z(z)$ be the density of the residence time in a state k, and let Y denote the residual lifetime in that state. If the communication that holds the requested link passes through just one state k, the mean residual

time to release this link can be evaluated by means of the density function $f_Y(y)$ of the residual residence time in the state k, i.e.

$$R_k = \int_0^\infty y f_Y(y)\,dy, \tag{6}$$

where, as shown in [24], $f_Y(y)$ can be obtained by the distribution function $F_Z(z)$ and the expected value $E[Z]$ of the residence time, that is,

$$f_Y(y) = \frac{1 - F_Z(y)}{E[Z]}. \tag{7}$$

This method requires the knowledge of the distribution function of the residence time in a state that is not easily obtainable. However, it is important to adopt this approach only for the phases, such as data-transmission, which consists of a single state with a long residence time. On the other hand, for the phases composed by several sub-states each with a short residence time, such as path-release, it is not a serious approximation to consider the residual residence time in a single sub-state equal to the residence time. Therefore, for those phases, we assume that the communication holding the link is at the beginning of a sub-state chosen with uniform probability among all sub-states. Let us now compute R_k^x for the following x phases: path-release (M states), data-transmission (1 state), and path-set-up phase ($M + 1$ states).

In the case of conflict with a communication being in a state k of the path-release phase, the contended link could be any of the k residual links. Hence, the expected residual time to release the requested link is

$$R_k^{\text{path\_release}} = \frac{1}{k}\sum_{i=1}^{k} i\,T_{\text{rel}} = T_{\text{rel}}\frac{k+1}{2} \quad \text{for } k = 1, \ldots, M. \tag{8a}$$

In the case of conflict with a communication in the data-transmission phase, the expected residual time to release the requested link is given by the mean residual time to complete the data-transmission plus the expected time to release a mean number of links (each of them could be the one requested by the other communication). Hence,

$$R^{\text{data\_trans}} = \int_0^\infty y F_Y(y)\,dy + R_M^{\text{path\_release}}, \tag{8b}$$

where $f_Y(y)$ is the density function of the residual residence time in the data transmission state. As shown in (7), this density can be obtained by means of the distribution function and the expected value of the data transmission time. In this paper, we do not assume any particular distribution for the data transmission time, and the accuracy of the model will be validated for various data transmission distributions.

In the case of conflict with a communication being in a generic state k of the path-set-up phase, the expected residual time to release the requested link is given by the mean time to acquire the remaining $M - k$ links of the path, the time to receive the ACK signal, the mean data-transmission time, and the expected residual time to release the requested link. Hence,

$$R_k^{\text{path\_set\_up}} = (M-k)(T_{\text{verify}} + D_{\text{link\_conn}} + T_{\text{link\_wait}}) + D_{\text{ack}} + T_{\text{data\_trans}} + R_M^{\text{path\_release}}. \tag{8c}$$

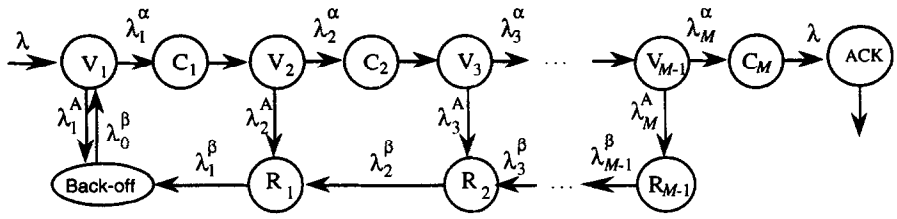


Fig. 3. Path-set-up phase for the *drop* strategy.

Eqs. (8c), (4) and (5) show the mutual dependency among $R_k^{\text{path\_set\_up}}$, $T_{\text{link\_wait}}$ and $W^{\text{path\_set\_up}}$. Hence, we evaluate the path-set-up phase through an iterative approach that captures this feedback effect. We have empirically verified that the convergence of this methods is always guaranteed after few steps.

4.2. Drop strategy

The drop strategy releases all the obtained links when a conflict occurs. Fig. 3 focuses on the path-set-up phase, because data-transmission and path-release phase are the same as in Fig. 2. In the path-set-up phase, the communication request may be in a *verification* (V_i), *link connection* (C_i) or *link release state* (R_i). The state V_i denotes that the communication has obtained $i-1$ links and asks for the ith link. If this link is idle, then the communication request reaches the state C_i. Otherwise, if another communication is using that link, the communication reaches the state R_{i-1} from which the link releasing process starts. To reduce the probability of further contentions, a back-off interval ($D_{\text{back\_off}}$) is introduced before the successive attempt of path-set-up. The times spent in the states V_i, C_i and R_i are denoted by T_{verify}, $D_{\text{link\_conn}}$ and T_{rel}, respectively.

When the communication request has obtained the last link of the path, the behaviour is the same as for the hold strategy. The mean time to build a complete path in accordance with the drop strategy is the sum of five terms:

$$T^{\text{DROP}}_{\text{path\_set\_up}} = N_{\text{verify}} T_{\text{verify}} + N_{\text{link\_conn}} D_{\text{link\_conn}} + N_{\text{rel}} T_{\text{rel}} + N_{\text{abort}} D_{\text{back\_off}} + D_{\text{ack}}, \tag{9}$$

where N_{verify}, $N_{\text{link\_conn}}$ and N_{rel} are the mean number of visits to the states V_i, C_i, and R_i, respectively. Therefore, $N_{\text{verify}} T_{\text{verify}}$ is the mean time to verify link availability. Analogously, $N_{\text{link\_conn}} D_{\text{link\_conn}}$ and $N_{\text{rel}} T_{\text{rel}}$ are the total time for link connection and for link release due to contentions. The link conflicts cause a mean number of aborts denoted by N_{abort}, hence the mean back-off time experimented by a communication request is $N_{\text{abort}} D_{\text{back\_off}}$. As for the hold strategy, T_{verify} and T_{rel} are mean times of the routing controller modelled as a M/G/1 queue (see Appendix A). All the crucial quantities of the path-set-up equation (that is, N_{verify}, $N_{\text{link\_conn}}$, N_{rel} and N_{abort}) depend on the probability (P_{conflict}) that a communication asks for a busy link.

4.2.1. The probability of link conflict

In this section we evaluate the probability that a link request causes a conflict. For the assumption (d), P_{conflict} is independent of the number of obtained links. However, two potential causes of contention have to be taken into account for the evaluation of P_{conflict}. The former is due to the first conflict with a generic communication. The latter is caused by the fact that the same communication still holds the required link. As an example, the second abort can be caused by two kinds of conflict: a conflict with a different

communication that could come up during the back-off period, or a second conflict with the previous communication that is still in progress after the back-off period. By assuming that the occurrences of these two events are mutually exclusive, we have

$$P_{\text{conflict}} = P_{\text{first\_conflict}} + P_{\text{further\_conflicts}}. \tag{10}$$

The distinction between first and further conflicts is a peculiarity of the drop strategy. For example, the adaptive policy has a negligible probability of conflicting twice or more with the same communication, thanks to the combinatorial number of feasible paths. The probability of conflicting for the first time with another communication is obtained by Eq. (B.3) as follows:

$$P_{\text{first\_conflict}} = \frac{N^{\text{V}}_{\text{link\_held}} + N^{\text{C}}_{\text{link\_held}} + N^{\text{R}}_{\text{link\_held}} + N^{\text{ack}}_{\text{link\_held}} + N^{\text{data\_trans}}_{\text{link\_held}} + N^{\text{rel}}_{\text{link\_held}}}{N_{\text{links}}}, \tag{11}$$

where the mean numbers of links held in each phase are in (B.5)–(B.7) and (B.8).

The probability that a communication, which has already caused an abort, induces further conflicts on the same link is equivalent to the probability that, this communication is still active after one or more back-off periods:

$$P_{\text{further\_conflicts}} = \sum_{i=1}^{\max} \text{Prob}\{\text{communication still in progress after } i\ D_{\text{back\_off}}\}, \tag{12}$$

where max denotes the maximum value over which the probability of further conflicts is negligible. Here we evaluate that probability for max = 1, however the results can be easily extended to the general case. We have

$$\begin{aligned}
&\Pr\{\text{communication still in progress after } D_{\text{back_off}}\} \\
&\quad = 1 - \Pr\{\text{contended link idle after } D_{\text{back_off}}\} \\
&\quad = 1 - (P^{\text{path_set_up}}_{\text{link_contention}} P^{\text{path_set_up}}_{\text{terminated}} + P^{\text{data_trans}}_{\text{link_contention}} P^{\text{data_trans}}_{\text{terminated}} + P^{\text{path_release}}_{\text{link_contention}} P^{\text{path_release}}_{\text{terminated}}),
\end{aligned} \tag{13}$$

where $P^{\text{path\_set\_up}}_{\text{link\_contention}}$, $P^{\text{data\_trans}}_{\text{link\_contention}}$, $P^{\text{path\_release}}_{\text{link\_contention}}$ are the probabilities that, given a link conflict, this has been caused by a communication in its path-set-up, data-transmission phase or path-release phase, respectively; $P^{\text{path\_set\_up}}_{\text{terminated}}$ is the probability that the communication in progress has terminated its path-set-up and data-transmission phases, and has released the contended link within $D_{\text{back\_off}}$. Analogously, for $P^{\text{data\_trans}}_{\text{terminated}}$ and $P^{\text{path\_release}}_{\text{terminated}}$.

The link contention terms in (13) are conditional probabilities that a first conflict occurs while the other communication is in its path-set-up, data-transmission or path-release phase. The evaluation of these terms can be easily obtained by Eqs. (B.3)–(B.6) and (B.11). For example,

$$P^{\text{path\_set\_up}}_{\text{link\_contention}} = (P^{\text{path\_set\_up}}_{\text{first\_conflict}} | P_{\text{first\_conflict}}) = \frac{N^{\text{V}}_{\text{link\_held}} + N^{\text{C}}_{\text{link\_held}} + N^{\text{R}}_{\text{link\_held}}}{N_{\text{link\_held}}}.$$

The other three probabilities $P^{\text{path\_set\_up}}_{\text{terminated}}$, $P^{\text{data\_trans}}_{\text{terminated}}$, $P^{\text{path\_release}}_{\text{terminated}}$ require a more complex analysis that takes into account the expected residual time of an event. Their evaluation is given in [7].

4.2.2. The other terms in the path-set-up equation

It is now possible to evaluate all the terms in the path-set-up equation (9) through the flow analysis. First of all, we estimate the mean number of aborts on which all other parameters depend. N_{abort} is equivalent to the mean number of visits to the back-off state in Fig. 3. Therefore, $N_{\text{abort}} = \lambda_0^\beta/\lambda$ from which it holds

$$\lambda_0^\beta = \lambda N_{\text{abort}}. \tag{14}$$

For the stationary assumption, the flow reaching a state is equal to the flow leaving that state. Therefore, we can write the following flow balance equation:

$$\lambda = \lambda_M^\alpha, \tag{15}$$

where λ_M^α is equivalent to the flow reaching the V_1 state (that is, $\lambda + \lambda_0^\beta$) times the probability of not conflicting in any of the M link requests. Hence, $\lambda_M^\alpha = (\lambda + \lambda_0^\beta)(1 - P_{\text{conflict}})^M$ that, through (14), becomes

$$\lambda_M^\alpha = \lambda(1 + N_{\text{abort}})(1 - P_{\text{conflict}})^M. \tag{16}$$

By combining (15) and (16), we obtain the mean number of aborts as

$$N_{\text{abort}} = 1/(1 - P_{\text{conflict}})^M - 1. \tag{17}$$

Evaluating the N_{arb}, $N_{\text{link\_conn}}$, and N_{rel} terms in (9) requires the estimation of the transaction rates shown in Fig. 3, that is, λ_i^α, λ_i^{A}, and λ_i^β, for $i = 1, \ldots, M$. As an example, for the V_1 state, we can write

$$\lambda_1^{\text{A}} = (\lambda + \lambda_0^\beta) P_{\text{conflict}}, \tag{18}$$

$$\lambda_1^\alpha = (\lambda + \lambda_0^\beta)(1 - P_{\text{conflict}}). \tag{19}$$

The other transaction rates are analogously evaluated,

$$\lambda_i^{\text{A}} = \lambda_{i-1}^\alpha P_{\text{conflict}}, \quad i = 2, \ldots, M, \tag{20}$$

$$\lambda_i^\alpha = \lambda_{i-1}^\alpha (1 - P_{\text{conflict}}), \quad i = 2, \ldots, M, \tag{21}$$

$$\lambda_{M-1}^\beta = \lambda_M^{\text{A}}, \tag{22}$$

$$\lambda_i^\beta = \lambda_{i+1}^\beta + \lambda_{i+1}^{\text{A}}, \quad i = m - 2, \ldots, 0. \tag{23}$$

Now we have all data necessary to evaluate the unknowns in (9), that is, N_{verify}, $N_{\text{link\_conn}}$, and N_{rel}. These values can be viewed as the mean number of visits to the verification, connection and releasing states, respectively. Hence,

$$N_{\text{verify}} = \sum_{i=1}^{M} \text{NV}_i = (\lambda_i^\alpha + \lambda_i^{\text{A}})/\lambda, \tag{24}$$

$$N_{\text{link\_conn}} = \sum_{i=1}^{M} \text{NC}_i = \lambda_i^\alpha/\lambda, \tag{25}$$

$$N_{\text{rel}} = \sum_{i=0}^{M-1} \text{NR}_i = \lambda_i^\beta/\lambda. \tag{26}$$

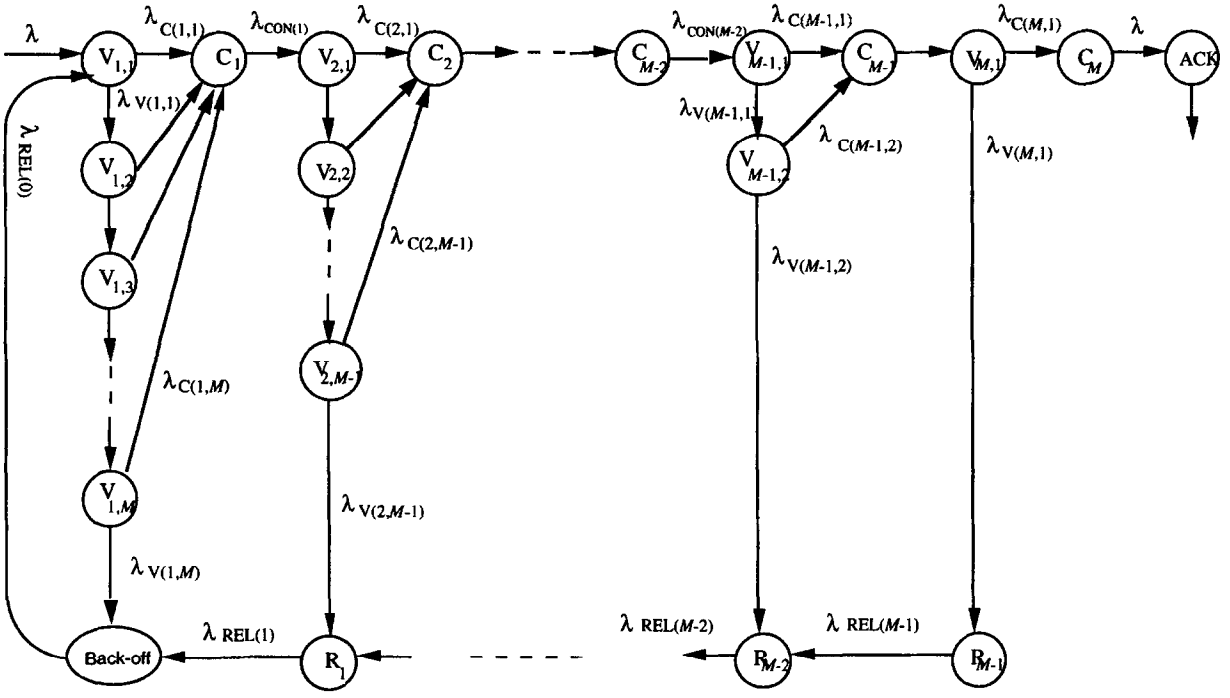


Fig. 4. Path-set-up phase for the *adaptive* strategy.

4.3. Adaptive strategy

The main difference between the adaptive strategy and the previous two policies is that the path choice depends on dynamic network conditions. The state diagram of the path-set-up phase in Fig. 4 shows that the communication request can be in a *verification* ($V_{i,j}$), *connecting* (C_i) or *releasing state* (R_i). The state $V_{i,j}$ denotes a communication request that has obtained $i-1$ links and demands, for the jth time, the ith link of the path. If this link is idle, then the communication request reaches the state C_i and connects the new link to the previous ones. Otherwise, if the request can choose other links, then it reaches the state $V_{i,j+1}$. If no link is available, then the communication request reaches the state R_{i-1} from which the link releasing process takes place. After the releasing phase due to an abort, the communication request waits for a back-off period ($D_{\text{back\_off}}$) before a new attempt.

The times spent in the states $V_{i,j}$, C_i and R_i are denoted by T_{verify}, $D_{\text{link\_conn}}$ and T_{rel}, respectively. These values depending on the activities of the routing controller are given in the Appendix A.

In this section we evaluate the mean time to build a path ($T^{\text{ADPT}}_{\text{path\_set\_up}}$) as the sum of six terms:

$$T^{\text{ADPT}}_{\text{path\_set\_up}} = N^{\text{I}}_{\text{verify}}(W^{\text{ADPT}} + D_{\text{verify}}) + N^{\text{II}}_{\text{verify}} D_{\text{verify}} + N_{\text{link\_conn}} D_{\text{link\_conn}} + N_{\text{rel}} T_{\text{rel}} + N_{\text{abort}} D_{\text{back\_off}} + D_{\text{ack}}, \tag{27}$$

where $N^{\text{I}}_{\text{verify}}$ and $N^{\text{II}}_{\text{verify}}$ are the number of visits to states $V_{i,1}$ and $V_{i,j}$ ($j > 1$), respectively; therefore $(W^{\text{ADPT}} + D_{\text{verify}}) + N^{\text{II}}_{\text{verify}} D_{\text{verify}}$ is the total time for link availability verification, and W^{ADPT} is the waiting time spent at the routing controller. We have distinct the first from the successive verification

because only the first request experiments a delay in the queue of the routing controller. If the link is not available, the router checks other possibilities without re-queuing the request.

$N_{\text{link\_conn}}$ is the number of visits to the states C_i, and $N_{\text{link\_conn}} D_{\text{link\_conn}}$ is the total time for link physical connection. N_{rel} is the number of visits to states R_i, and $N_{\text{rel}} T_{\text{rel}}$ is the total time for link release due to contention with other communication requests. N_{abort} is the number of aborts, and $N_{\text{abort}} D_{\text{back\_off}}$ is the total back-off time experimented by a communication request.

Eq. (27) is quite similar to Eq. (9) referring to the drop strategy. Therefore, an approach analogous to the one adopted to compute $T^{\text{DROP}}_{\text{path\_set\_up}}$ is followed to evaluate $T^{\text{ADPT}}_{\text{path\_set\_up}}$. The value of N_{abort} is computed in Section 4.3.1, and the other parameters in Section 4.3.2.

4.3.1. The number of aborts

To evaluate the mean number of aborted communications we use the same flow analysis adopted for the drop strategy. N_{abort} is equivalent to the mean number of visits to the back-off state in Fig. 4. Therefore, $\lambda_{\text{abort}} = \lambda_{\text{REL}(0)}/\lambda$ from which it holds

$$\lambda_{\text{REL}(0)} = \lambda N_{\text{abort}}. \tag{28}$$

For the stationary assumption, we can write the following flow balance equation:

$$\lambda = \lambda_{\text{C}(M,1)}, \tag{29}$$

where $\lambda_{\text{C}(M,1)}$ is equivalent to the flow reaching the $V_{1,1}$ state (that is, $\lambda + \lambda_{\text{REL}(0)}$) times the probability of obtaining all M links of the path. In the case of the adaptive strategy, an abort is a rather rare event because each link request has several possibilities. An abort arises only if the first link request conflicts M times or the second request conflicts $M-1$ times, and so on. Therefore, if we build all the feasible paths from $V_{1,1}$ to C_M, the probability of obtaining M links is given by, $(1 - P_{\text{conflict}})^M[(1 + P_{\text{conflict}})(1 + P_{\text{conflict}} + P^2_{\text{conflict}}) \cdots (1 + P_{\text{conflict}} + \cdots + P^{M-1}_{\text{conflict}})]$ and the flow rate $\lambda_{\text{C}(M,1)}$ can be written as

$$\lambda_{\text{C}(M,1)} = (\lambda_{\text{REL}(0)})(1 - P_{\text{conflict}})^M \prod_{j=1}^{M-1} \sum_{k=0}^{j} P^k_{\text{conflict}},$$

that, through (28) and (29), becomes

$$\lambda = \lambda(1 + N_{\text{abort}})(1 - P_{\text{conflict}})^M \prod_{j=1}^{M-1} \sum_{k=0}^{j} P^k_{\text{conflict}}. \tag{30}$$

From (30), we obtain the mean number of aborts as

$$N_{\text{abort}} = \frac{1}{(1 - P_{\text{conflict}})^M \prod_{j=1}^{M-1} \sum_{k=0}^{j} P^k_{\text{conflict}}} - 1, \tag{31}$$

where the probability of link conflict P_{conflict} is evaluated as in (B.3) and detailed in [7].

4.3.2. The other terms in the path-set-up equation

To evaluate the transaction rates, we consider Fig. 4 and the flow balance theorem that in the stationary condition guarantees the equality between incoming and outcoming flows. Let $\lambda_{\text{CON}(i)}$ and $\lambda_{\text{REL}(i)}$ denote the flow rate leaving the ith link connection state and the ith link releasing state, respectively.

With respect to the states $V_{i,j}$, we have

$$\lambda_{V_{1,1}} = (\lambda + \lambda_{REL(0)})P_{conflict}, \tag{32a}$$

$$\lambda_{V_{i,1}} = \lambda_{CON(i-1)}P_{conflict}, \quad i = 2, \ldots, M, \tag{32b}$$

$$\lambda_{V_{i,j}} = \lambda_{V_{i,j-1}}P_{conflict}, \quad i = 1, \ldots, M, \text{ and } j = 2, \ldots, M - i + 1, \tag{32c}$$

where the outcoming flows of the connection states C_i (for $i = 1, \ldots, M$) are

$$\lambda_{CON(i)} = \sum_{j=1}^{M-i+1} \lambda_{C_{i,j}}. \tag{33}$$

The $\lambda_{C_{i,j}}$ terms are given by the following equations:

$$\lambda_{C_{1,1}} = (\lambda + \lambda_{REL(0)})(1 - P_{conflict}), \tag{34a}$$

$$\lambda_{C_{i,1}} = \lambda_{CON(i-1)}(1 - P_{conflict}), \quad i = 2, \ldots, M, \tag{34b}$$

$$\lambda_{C_{i,j}} = \lambda_{V_{i,j-1}}(1 - P_{conflict}), \quad i = 1, \ldots, M, \text{ and } j = 2, \ldots, M - i + 1. \tag{34c}$$

Regarding the link release states, we have

$$\lambda_{REL(M-1)} = \lambda_{V_{M,1}}, \tag{35a}$$

$$\lambda_{REL(i)} = \lambda_{REL(i+1)} + \lambda_{V_{i+1,M-i}}, \quad i = M - 2, \ldots, 0. \tag{35b}$$

Once the transaction rates are estimated, it is possible to evaluate N_{verify}, N_{link_conn}, and N_{rel} in (27) through the number of visits to the respective states. Let $NV_{i,j}$, NC_i, and NR_i denote the mean number of visits to the verification $V_{i,j}$, connection C_i and releasing R_i states, respectively. We can write

$$N^{I}_{verify} = \sum_{i=1}^{M} NV_{i,1} = (\lambda_{V_{i,1}} + \lambda_{C_{i,1}})/\lambda, \tag{36}$$

$$N^{II}_{verify} = \sum_{i=1}^{M} \sum_{j=2}^{M-i+1} NV_{i,j} = (\lambda_{V_{i,j}} + \lambda_{C_{i,j}})\lambda, \tag{37}$$

$$N_{link\_conn} = \sum_{i=1}^{M} NC_i = \lambda_{CON(i)}/\lambda, \tag{38}$$

$$N_{rel} = \sum_{i=1}^{M-1} NR_i = \lambda_{REL(i)}/\lambda. \tag{39}$$

5. Extension of the model to the torus topology

In this section we demonstrate the flexibility of the proposed modelling approach by indicating how it can be extended to analyse the performance of other symmetric topologies, such as the torus that is a mesh with end-around connections. Due to space limitations, we outline only the main steps that are to be followed to model circuit-switching routing in a two-dimensional torus in the case of *hold* link conflict resolution strategy.

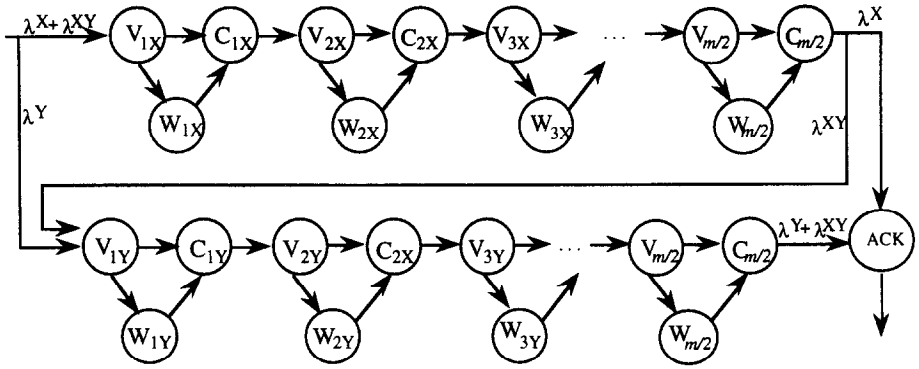


Fig. 5. Path-set-up phase for the *hold* strategy in a torus network.

Eq. (2) for the estimation of the mean latency time is still valid. Hence, we focus on the path-set-up phase. In this network, the most popular algorithm to establish a deterministic communication path is the so called XY routing that is, a dimension-ordering policy analogous to the e-cube algorithm in a hypercube.

In a torus interconnection network there are three classes of message flows generated by each node: the messages that reach the destination node using links of the dimension X only (λ^X), those using links of the dimension Y only (λ^Y), and those using links of both dimensions (λ^{XY}). Under the hypothesis of uniform traffic in an $m \times m$ torus with uni-directional links, the mean number of links per path is $M^X = M^Y = m/2$ and $M^{XY} = m$ for the flows λ^X, λ^Y and λ^{XY}, respectively. Fig. 5 shows the state diagram for the path-set-up phase of the hold strategy in a two-dimensional torus. The state labels have the same meaning as those in Fig. 2. Moreover, we have $\lambda^X = \lambda^Y = \lambda(m-1)/(m^2-1) = \lambda/(m+1)$, and $\lambda^{XY} = \lambda(1 - 2/(m+1))$.

The remaining steps are analogous to those discussed in Section 4.1. We have only to specialise some expressions to take into account the peculiarities of the torus architecture. For example, the mean time to complete the path-set-up phase is quite similar to Eq. (3), but for the first term that now is $M_{\text{torus}} = (\lambda^X M^X/\lambda + \lambda^Y M^Y/\lambda + \lambda^{XY} M^{XY}/\lambda)$. Hence,

$$T^{\text{HOLD}}_{\text{path\_set\_up}} = M_{\text{torus}}(T_{\text{verify}} + D_{\text{link\_conn}} + T_{\text{link\_wait}}) + D_{\text{ack}}.$$

Other equations, such as (4) and (B.1), having no explicit references to the architecture remain still valid. On the other hand, we have to modify all equations, such as (5) and (B.5)–(B.8), that contain terms referring to the topology (e.g., the mean number of links held in a phase). However, the few changes do not affect the spirit of the modelling approach. For example, we have only to replace M with M_{torus}, and distinguish the class of message flows in other equations.

6. Performance results

6.1. Validation of the model

The simulation results are obtained by means of the *Independent Replications Method* implemented in Simula language [20]. The model explicitly simulates the activity of link arbitration, link connection, link

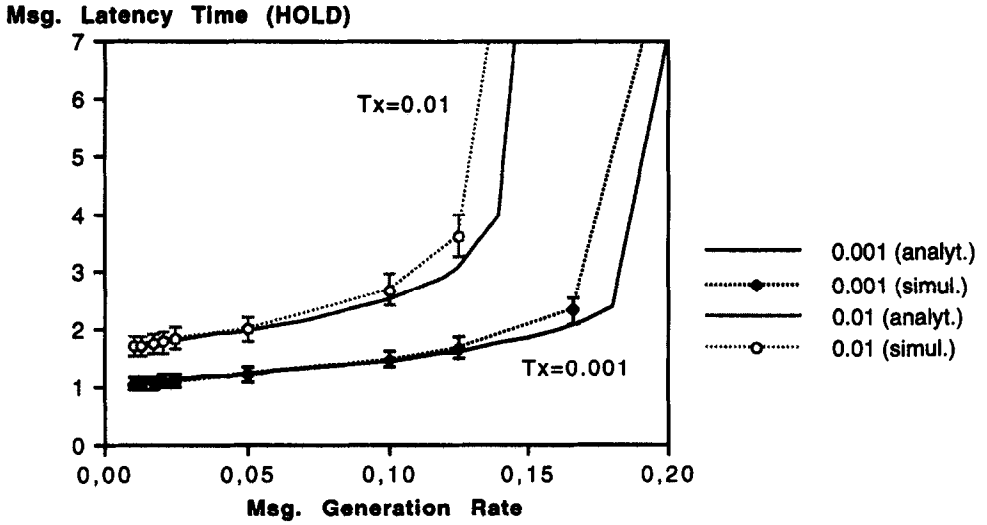


Fig. 6. Message latency time of *hold* strategy in a hypercube with dimension $d = 8$ for different message generation rates and two classes of network service times.

releasing, and data transmission, from which the effects of link contentions and aborts occur. The simulation model is developed under the assumptions (a)–(c), whereas the assumptions (d)–(f) are removed. The computation of the confidence intervals (at a 95% level of confidence) is based on the Jackknife estimator [20].

Fig. 6 compares the analytical and simulation results for the *hold* strategy in a hypercube with 256 nodes. Message generation is Poissonian, while the destinations are uniformly chosen among all nodes. Data transmission time is uniformly distributed in the interval [0.1, 1.9]. This figure presents two classes of curves. The former refers to an interconnection network having $T_{\text{verify}} = D_{\text{link\_conn}} = D_{\text{ack}} = T_{\text{rel}} = 0.001$, the latter to $T_{\text{verify}} = D_{\text{link\_conn}} = D_{\text{ack}} = T_{\text{rel}} = 0.01$. Moreover, this figure shows the sensitivity of the hold strategy to the variation of the message generation rate per each node. The difference between the analytical model and simulation is within 5% for a wide range of message arrival rates, and it does not overcome 10% when the interconnection network is close to saturation. The latter plot ($T_X = 0.01$) shows that the approximation is accurate even for high values of the path-set-up time, because the curve is close to the confidence interval.

Analogous correspondence between analytical and simulation results is in Fig. 7 concerning the *drop* strategy in a hypercube with 256 nodes, $T_{\text{verify}} = D_{\text{link\_conn}} = D_{\text{ack}} = T_{\text{rel}} = 0.001$, and $D_{\text{back\_off}} = 1.5$. This figure shows the sensitivity of the drop strategy to the variation of the data transfer time distribution. In particular, it reveals that the analytical model is independent of the message length distribution. The former class of curves refers to a constant data transfer time $D_{\text{data\_trans}} = 1$, the latter is related to a data transfer time exponentially distributed with mean equal to 1. For clarity reasons, we have not reported a third class of curves, referring to a data transfer time uniformly distributed in the interval [0.1, 1.9], that falls in the middle.

Fig. 8 shows the mean number of aborts per communication as a function of the message generation rate when the *adaptive* strategy is adopted. The curves are related to a hypercube with 256 nodes in which the message routing is uniformly distributed, $T_{\text{verify}} = D_{\text{link\_conn}} = D_{\text{ack}} = T_{\text{rel}} = 0.001$, the data transfer time

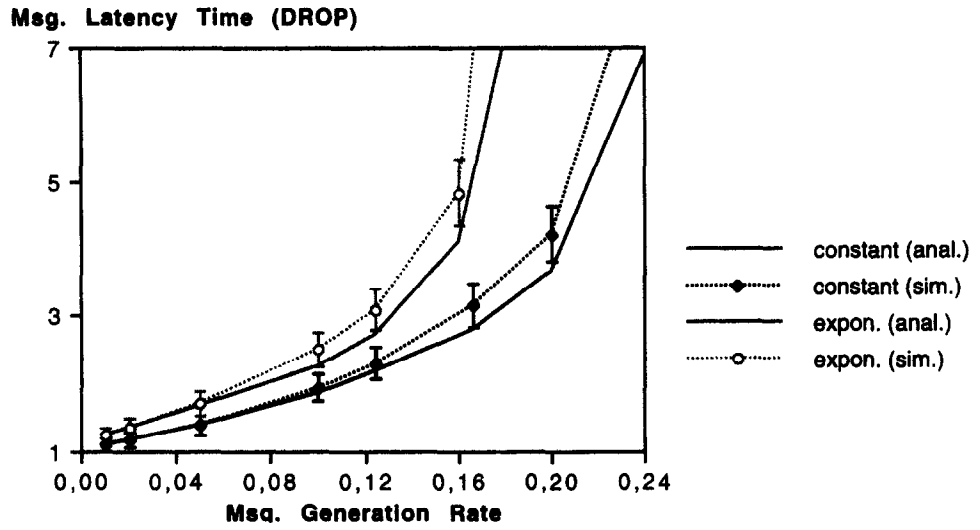


Fig. 7. Message latency time of drop strategy in a hypercube with dimension $d = 8$ for various message generation rates and data transfer distributions.

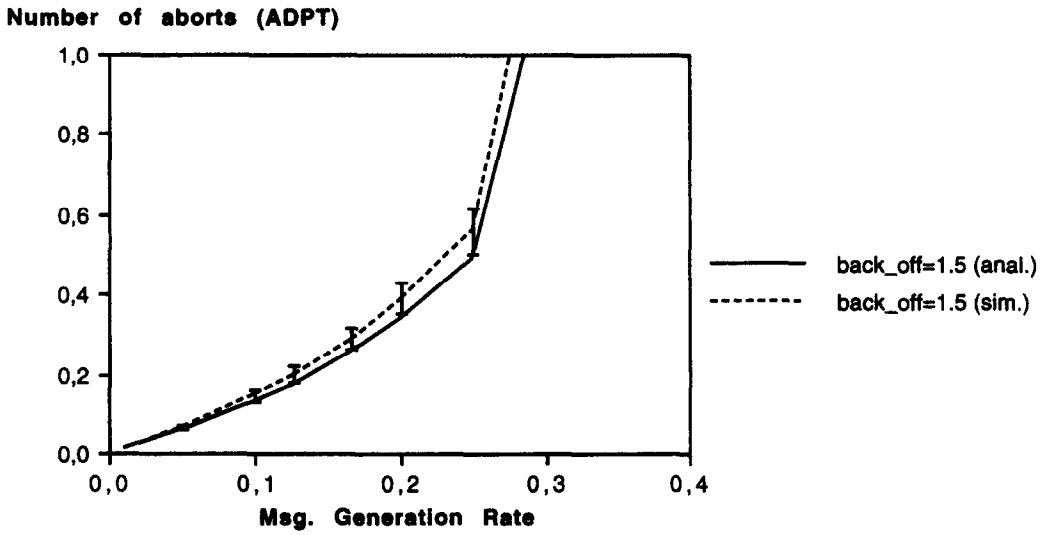


Fig. 8. Mean number of abort per communication of *adaptive* strategy in a hypercube with dimension $d = 8$ for different values of message generation rate and $D_{back\_off} = 1.5$.

is uniformly distributed, and $D_{back\_off} = 1.5$. The estimated mean number of aborts per communication is close to the simulation results. The difference between the analysis and the mean value of the simulation is within 5% for a wide range of message arrival rates and tends to be below the confidence interval only when the network tends to saturation. Analogous plots have been obtained to validate the analytical model for different values of the back-off time ($D_{back\_off} = 3.0, 2.0, 1.0, 0.5, 0.3$). We can conclude that when

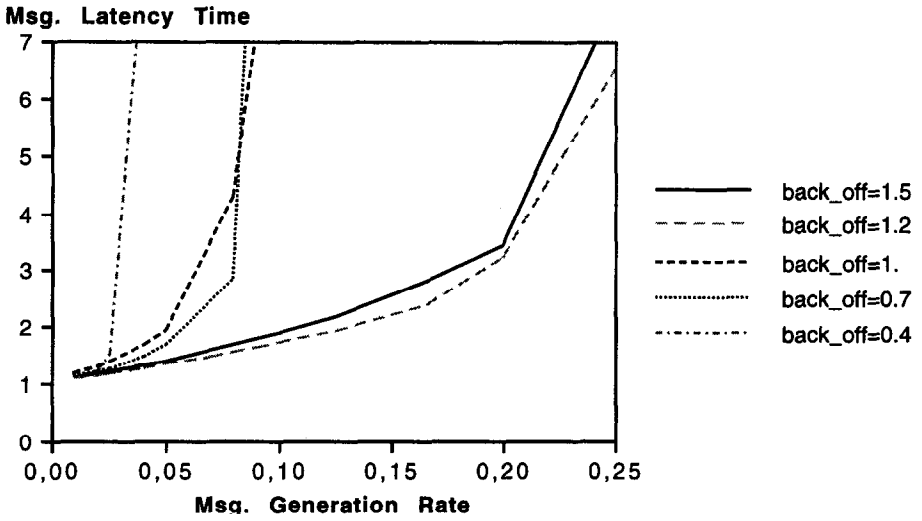


Fig. 9. Message latency time of *drop* strategy in a hypercube with dimension $d = 8$ for various message generation rates and five back-off times.

$D_{\text{back\_off}} \geq 1.5$ the graphs coincide, otherwise they slightly differ. In particular, until $D_{\text{back\_off}} = 0.3$ a limited reduction of the number of aborts can be observed.

The figures of this section demonstrate the quality of the analytical approach for all the link conflict resolution strategies. As expected, the analytical results are lower bounds of the actual times because the assumptions done to render tractable the model do not allow us to take into account some potential link conflicts.

6.2. Performance comparison

After the validation of the analytical models, we use them to obtain other interesting performance parameters. In particular, we intend

- to evaluate how the back-off time choice influences the latency time of the drop and adaptive strategies;
- to estimate the influence of the network dimension on the communication performances in case of low, medium, and high traffic conditions;
- to compare the performance of the three link conflict resolution strategies for various network dimensions, message dimensions, and router service times.

In the first experiment, we evaluate the influence of the back-off time on the message latency time of drop and adaptive strategies. For what concerns the adaptive policy, we can confirm results shown in Fig. 8, that is, the high number of alternative paths avoids network congestion even if the back-off time is chosen much lower than the mean latency time. Moreover, a proportional reduction of the latency time is experimented, thanks to the decrease of the $N_{\text{abort}} D_{\text{back\_off}}$ term in the path-set-up equation (27).

Fig. 9 shows the mean latency time as a function of the message generation rate for the drop strategy and various $D_{\text{back\_off}}$ times. The curves refer to a hypercube with 256 nodes in which the message routing is uniformly distributed, $T_{\text{verify}} = D_{\text{link\_conn}} = D_{\text{ack}} = T_{\text{rel}} = 0.001$ and data transfer time is uniformly

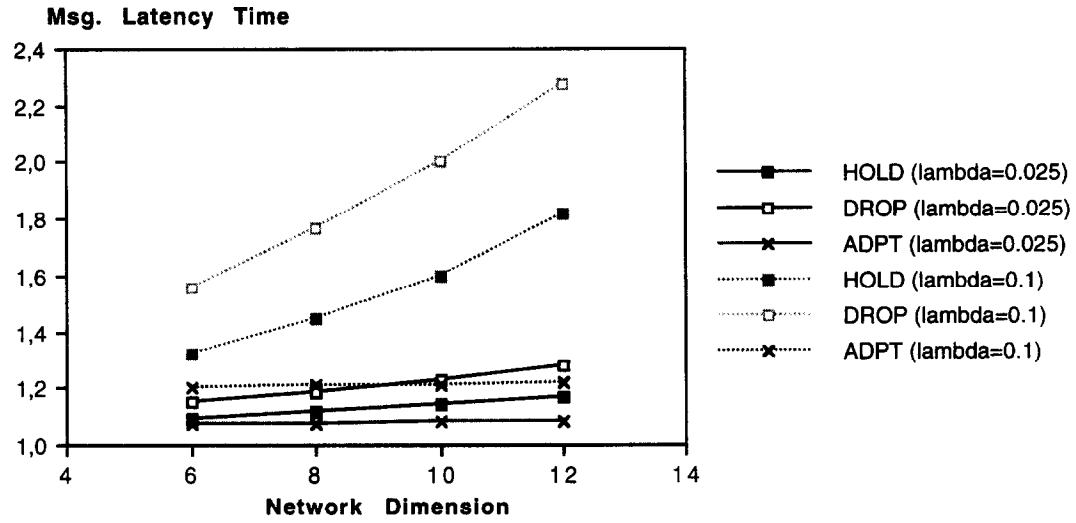


Fig. 10. Performance of link conflict resolution strategies for low and medium traffic in hypercubes of various dimensions.

distributed. The drop strategy is more interesting than the adaptive case because two opposite forces act on the $N_{\text{abort}} D_{\text{back\_off}}$ term of the path-set-up equation (9). Now a reduction of $D_{\text{back\_off}}$ causes an increment of N_{abort}. Fig. 9 gives a fine example of this situation. Starting from $D_{\text{back\_off}} = 1.5$, the drop strategy improves as the back-off time decreases until $D_{\text{back\_off}} = 1.2$ (that is, the number of aborts increases less than the 0.3 reduction of the back-off time). A further reduction of the back-off interval to $D_{\text{back\_off}} = 1$ causes a so high number of aborts that the latency time experiments a sharp growth. In other words, choosing the back-off interval much lower than the mean latency time causes that an aborted communication has high probability to further conflict with the previous communication (that is, $P_{\text{further\_conflicts}}$ augments). The increase in the number of attempts will increase load and link contention which, in their turn, impact system performance. A reduction of the back-off time to $D_{\text{back\_off}} = 0.7$ produces a slightly better performance than the previous one. Probably, the effect of the rough increase of N_{abort} is now less tangible, and a reduced $D_{\text{back\_off}}$ produces a temporary improvement on the mean path-set-up time. The short effect of this reduction is demonstrated by the fifth curve relative to $D_{\text{back\_off}} = 0.4$ that shows a congested network even for light traffic conditions.

We compare the three link conflict resolution strategies as a function of the network dimension. Four hypercube dimensions (namely $d = 6$, $d = 8$, $d = 10$ and $d = 12$) are considered, while the back-off time is set to 1.3. The other parameters are the same of the previous figure. Fig. 10 shows the mean latency time for light and medium traffic conditions (i.e., message generation rate equal to 0.025 and 0.1, respectively). This figure shows that the adaptive strategy is almost insensitive to the network dimension, and it performs best for any traffic condition. We verified that the relative performance of the three strategies for high traffic (i.e., message generation rate equal to 0.2) is similar to the medium traffic instance.

In the third experiment we have set the message generation rate to a medium traffic level (i.e. $\lambda = 0.125$) and let vary the message dimension (i.e., $T_{\text{data\_trans}}$) with the purpose of comparing the three link conflict resolution strategies as a function of the data transmission time. The network dimension is set to $d = 10$, and the parameters are those of Fig. 10. Fig. 11 confirms that the adaptive strategy performs always as the

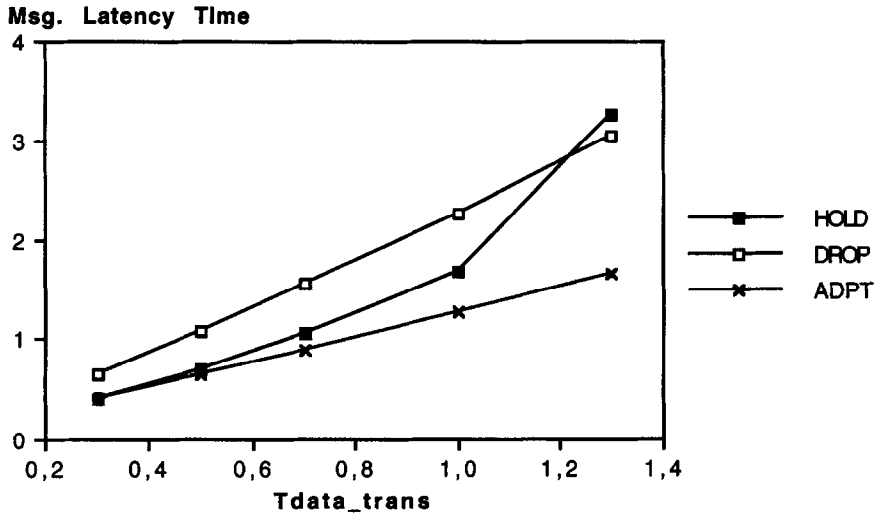


Fig. 11. Mean latency time of link conflict resolution strategies for different data transmission times.

best. For what concerns hold and drop strategies, Fig. 11 shows a better behaviour of the former when the data transmission time is small. Indeed, in these cases it seems more convenient to wait for an occupied link rather than to free up all the obtained links and reply the path-set-up phase from the beginning. On the other hand, when the message dimension increases, it is more profitable to release the acquired links in order to avoid network saturation.

In all the previous figures the router service times are supposed a very little fraction of other times, such back-off and data transmission. Typically, $T_{\text{verify}} = D_{\text{link\_conn}} = T_{\text{rel}} = 0.001$, while $T_{\text{data\_trans}}$ and $D_{\text{back\_off}}$ are close to 1. Other experiments aimed at demonstrating the router contribution in the path-set-up phase. To this purpose, we evaluated the mean path-set-up time as a function of increasing values of T_{verify}, $D_{\text{link\_conn}}$, and T_{rel} (from 0.001 to 0.1) for two message generation rates, that is, $\lambda = 0.1$ and $\lambda = 0.17$. All the strategies show a tendency to congestion when the router service times augment, even if less evident for the adaptive policy. In particular, the hold and drop strategies tend to saturate the network for time values higher than 0.01, independently of the back-off time (set to 1.2). These experiments are an example of the realism of the model. Many other results would not be possible with simpler analytical models that do not take into account aspects such as the role of the routing controller, the back-off time, the holding time, and the probability of further conflicts.

7. Conclusions

In this paper we propose a modelling approach to evaluate the message latency time of symmetric interconnection networks based on asynchronous direct circuit-switching policy in the case of *hold, drop* or *adaptive* link conflict resolution strategy. This model takes into account the time spent by the communication router in acquiring and releasing links, the back-off interval and other potential delays. In addition, it is able to capture the feedback effects among the probability of link conflict, the routing controller overhead

and the message latency. Indeed, a larger latency time leads to a greater probability of link conflict and to a higher link holding time (in the case of the *hold* strategy) or to a greater probability of conflict (in the case of *drop* and *adaptive* strategy). Higher values of these latter parameters, in their turn, increase the latency time, and so forth. Many previous studies ignore the mutual dependencies among those factors.

We verify that our approach is accurate for a wide range of traffic conditions, message length distributions, and network features. We demonstrate the best performance of the adaptive strategy with respect to deterministic policies such as the drop and the hold. However, this better behaviour is obtained at the price of an increase of the complexity of the router module that requires a more sophisticated logic for path construction (see the chip router proposal in [10]) and a message buffer to store the aborted communications during the back-off period.

The models proposed in this paper can be extended to evaluate the performance of different message routing strategies and network topologies. We are applying this methodology to study wormhole policies with deterministic and adaptive routing.

Appendix A. Performance of the routing controller

We evaluate the overhead due to the routing controller for the hold strategy and refer to [7] for additional details about drop and adaptive strategies. As shown in Fig. 1, the router serves two kinds of requests: link connection (path-set-up phase) and link releasing (path-set-up phase, for the hold strategy; path-set-up and path-release phases, for the drop and adaptive strategies).

In the case of the *hold* strategy, the router performs two activities for the path-set-up phase and two for the path-release phase. Each of them requires a deterministic service time that corresponds to:

(1) verify whether the requested link is free and, in that case, connect this link (the service time is equal to $D_1 = D_{\text{verify}} + D_{\text{link\_conn}}$);
(2) verify and en-queue link request, in the case of an already assigned link ($D_2 = D_{\text{verify}}$);
(3) release a link which has not been requested ($D_3 = D_{\text{rel}}$);
(4) release a link which has been requested, and connect this link ($D_4 = D_{\text{rel}} + D_{\text{link\_conn}}$).

By assuming that link connection and link release requests arrive with the same probability, these activities are carried out with the following probabilities: $p_1 = (1 - P_{\text{conflict}})/2$, $p_2 = P_{\text{conflict}}/2$, $p_3 = (1 - P_{\text{conflict}})/2$, $p_4 = P_{\text{conflict}}/2$.

The routing controller can be modelled as a M/G/1 queue where the arrivals are Poissonian with rate $2\lambda M$, and the service time is given by the weighted combination of the four activities executed with deterministic times. By using [17], we have that the mean service time for the routing controller is:

$$S^{\text{HOLD}} = p_1 D_1 + p_2 D_2 + p_3 D_3 + p_4 D_4. \tag{A.1}$$

To evaluate the mean waiting time of a M/G/1, we need the second moment of the service time as

$$S^{(2)} = p_1 D_1^2 + p_2 D_2^2 + p_3 D_3^2 + p_4 D_4^2. \tag{A.2}$$

Through (A.1) and (A.2), we can write that the mean waiting time of the routing controller is given by

$$W^{\text{HOLD}} = \lambda M S^{(2)} \frac{1}{(1-\rho)}, \tag{A.3}$$

where $\rho = 2\lambda M S^{\text{HOLD}}$.

The activity of the routing controller influences all the times that involve the router itself, and in particular the requests of connecting and releasing a link (T_{verify} and T_{rel}, respectively). Thanks to the previous equations we can consider all these parameters as mean times and not constant delays, as usually assumed in the literature:

$$T_{\text{verify}} = W^{\text{HOLD}} + D_{\text{verify}}, \tag{A.4}$$

$$T_{\text{rel}} = W^{\text{HOLD}} + D_{\text{rel}}. \tag{A.5}$$

On the other hand, the link connection ($D_{\text{link\_conn}}$) remains a deterministic time that does not experiment waiting times because it always follows a link request or a link release phase.

In the case of *drop* strategy, the routing controller carries out two activities for the path-set-up phase (connect or abort) and one for the path-release phase. Through a process analogous to that used for the hold strategy, we can write that the mean service time and the mean waiting time for the routing controller are

$$S^{\text{DROP}} = p_1 D_1 + p_2 D_2 + p_3 D_3, \tag{A.6}$$

$$W^{\text{DROP}} = \lambda M S^{(2)} \frac{1}{(1-\rho)}, \tag{A.7}$$

where $S^{(2)} = p_1 D_1^2 + p_2 D_2^2 + p_3 D_3^2$ and $r = 2\lambda M S^{\text{DROP}}$. Therefore, the mean times for link verification and link releasing for the drop strategy are $T_{\text{verify}} = W^{\text{DROP}} + D_{\text{verify}}$ and $T_{\text{rel}} = W^{\text{DROP}} + D_{\text{rel}}$, respectively.

For the *adaptive* strategy, if we assume that there are J available links at each node to build the path, the routing controller aborts the link request only after J checks. Therefore, we have (see [7]):

$$S^{\text{ADPT}} = \sum_{k=1}^{J} p_{1k} D_{1k} + p_2 D_2 + p_3 D_3, \tag{A.8}$$

$$W^{\text{ADPT}} = \lambda M S^{(2)} \frac{1}{(1-\rho)}, \tag{A.9}$$

where $S^{(2)} = \sum_{k=1}^{J} p_{1k} D_{1k}^2 + p_2 D_2^2 + p_3 D_3^2$ and $\rho = 2\lambda M S^{\text{ADPT}}$. The mean times for link connection and link release are $T_{\text{verify}} = W^{\text{ADPT}} + D_{\text{verify}} \sum_{k=1}^{J} k P_{\text{conflict}}^{k-1}$ and $T_{\text{rel}} = W^{\text{ADPT}} + D_{\text{rel}}$, respectively.

Appendix B. The probabilities of link conflict

The probability of link conflict is the key value for evaluating the performance parameters of hold, drop and adaptive policies. We propose a common approach that with slight differences can be adopted for any conflict resolution strategy.

Let us recall that we are considering a system with N_{links} links and p nodes, each generating a communication request as a Poisson process with rate λ. The probability that a link request causes a conflict is equal to the probability that this communication requires a link already assigned to another communication.

When the probability of further contention against the same communication can be neglected (*hold* and *adaptive* strategies), this probability is also equal to the utilisation factor of each link [15], hence

$$P_{\text{conflict}} = \frac{N_{\text{link\_held}}}{N_{\text{links}}}. \tag{B.1}$$

where $N_{\text{link\_held}}$ denotes the total number of links held by all the communications. When a link request causes a conflict, the other communication may be in any of the three phases that constitute a message transmission. Therefore,

$$N_{\text{link\_held}} = N_{\text{link\_held}}^{\text{path\_set\_up}} + N_{\text{link\_held}}^{\text{ack}} + N_{\text{link\_held}}^{\text{data\_trans}} + N_{\text{link\_held}}^{\text{rel}} \tag{B.2}$$

and

$$P_{\text{conflict}} = \frac{N_{\text{link\_held}}^{\text{path\_set\_up}} + N_{\text{link\_held}}^{\text{ack}} + N_{\text{link\_held}}^{\text{data\_trans}} + N_{\text{link\_held}}^{\text{rel}}}{N_{\text{links}}}. \tag{B.3}$$

The probabilities that a link conflict occurs while the other communication is in its path-set-up, ack, data-transmission and path-release phase are easily obtained from (B.3). For example,

$$P_{\text{conflict}}^{\text{path\_set\_up}} = \frac{N_{\text{link\_held}}^{\text{path\_set\_up}}}{N_{\text{links}}}. \tag{B.4}$$

Hence, the computation of the probabilities of link conflict always requires the evaluation of the number of links held in each state. This value is given by the number of links held in that state times the incoming rate and the mean residence time in that state. For the states different from the path-set-up phase, we have

$$N_{\text{link\_held}}^{\text{ack}} = p\lambda M D_{\text{ack}}, \tag{B.5}$$

$$N_{\text{link\_held}}^{\text{data\_trans}} = p\lambda M T_{\text{data\_trans}}, \tag{B.6}$$

$$N_{\text{link\_held}}^{\text{rel}} = p\lambda \sum_{i=1}^{M} i\, T_{\text{rel}}. \tag{B.7}$$

The number of links held in the path-set-up phase can be further detailed depending on the link conflict resolution strategy. For example, the path-set-up phase of the *hold* strategy has three kinds of states: *verification, waiting* and *link connection*. Nevertheless, the aggregation established in Section 4.1 allows us to consider a single class. Hence,

$$N_{\text{link\_held}}^{\text{path\_set\_up}} = p\lambda \sum_{i=1}^{M} i(T_{\text{verify}} + D_{\text{link\_conn}} + T_{\text{link\_wait}}). \tag{B.8}$$

For the *drop* strategy, we have to distinguish between probability of first conflict and probability of further conflicts. Here we refer to the former probability, while the latter is evaluated in [7]. (This implies that P_{conflict} of Eq. (B.3) should be read as $P_{\text{first\_conflict}}$.) The path-set-up phase of the drop strategy has three kinds of states: *verification, link connection*, and *link releasing*. Hence,

$$N_{\text{link\_held}}^{\text{path\_set\_up}} = p\left(\sum_{i=1}^{M-1} \lambda_i^{\alpha} i\, T_{\text{verify}} + \sum_{i=1}^{M} \lambda_i^{\alpha} i\, D_{\text{link\_conn}} + \sum_{i=1}^{M-1} \lambda_i^{\beta} i\, T_{\text{rel}}\right), \tag{B.9}$$

where the flow rates λ_i^{α} and λ_i^{β} are given in Eqs. (19)–(23).

The probability of link conflict for the *adaptive* strategy is directly obtained by Eq. (B.3). Unlike the drop strategy, the assumption (f) allows the model to exclude the probability of further conflicts with the same communication. The equation, analogous to (B.9), is given in [7].

References

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

[2] R.V. Boppana, S. Chalasani, A comparison of adaptive wormhole routing algorithms, in: Proc. 20th Annual Int. Symp. on Computer Architecture, San Diego, CA, May 1993, IEEE Computer Society Press, Sliver Spring, MD, pp. 351–360.

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

[4] A.A. Chien, J.H. Kim, Planar-adaptive routing: Low-cost adaptive networks for multiprocessors, J. ACM 42 (1) (1995) 91–123.

[5] I. Chlamtac, A. Ganz, M. Kienzle, A performance model of a connection based hypercube interconnection system, in: G. Balbo, G. Serazzi (Eds.), Computer Performance Evaluation, Modelling Techniques and Tools, Proc. 5th Int. Conf. on Modelling Tech. and Tools for Comp. Perf. Eval., Turin, Italy, Elsevier, Amsterdam, February 1991, pp. 215–228.

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

[7] M. Colajanni, B. Ciciani, S. Tucci, An analytical model for circuit-switching routing, Tech. Rep. RI-95.15, University of Rome – Tor Vergata, 1995 (http://www.ce.uniroma2.it/ ~colajan/routing/RI-95.15.ps).

[8] M. Colajanni, A. Dell'Arte, B. Ciciani, Communication switching techniques and link-conflict resolution strategies: A comparison analysis, J. Simulation Practice and Theory (1998).

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

[10] N.J. Dimopoulos, D. Radvan, K.F. Li, Performance evaluation of the backtracking-to-the-origin-and-retry routing for hypercycle-based interconnection networks, in: Proc. IEEE Distributed Computing Systems Conf., Paris, June 1990, IEEE Computer Society Press, Sliver Spring, MD, pp. 278–284.

[11] C.J. Glass, L.M. Ni, The turn model for adaptive routing, in: Proc. 19th Int. Symp. on Computer Architecture, Gold Cost, Australia, 1992, ACM, New York, pp. 278–287.

[12] L. Gravano, G.D. Pifarré, P.E. Berman, J.L.C. Sanz, Adaptive deadlock- and livelock-free routing with all minimal paths in torus networks, IEEE Trans. Parallel and Distributed Systems 5 (12) (1994) 1233–1251.

[13] D.C. Grunwald, D.A. Reed, Networks for parallel processors: Measurements and prognostications, in: Proc. 3rd Conf. on Hypercube Concurrent Computers and Applications, New York, 1988, pp. 610–619.

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

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

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

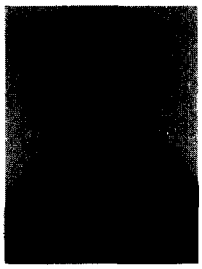
[17] L. Kleinrock, Queueing Systems, Wiley, New York, 1979.

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

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

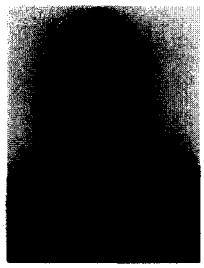
[20] I. Mitrani, Simulation Techniques for Discrete Event Systems, Cambridge University Press, Cambridge, 1982.

[21] J. Peterson, E. Chow, H. Madan, A high-speed message-driven communication architecture, in: Proc. Int. Conf. on Supercomputing, St. Malo, France, July 1988, ACM, New York.
[22] S. Ramany, D. Eager, The interaction between virtual channel flow control and adaptive routing in wormhole networks, in: Proc. Int. Conf. on Supercomputing, Manchester, UK, July 1994, ACM, New York, pp. 136–145.
[23] D.A. Reed, R.M. Fujimoto, Multicomputer Networks: Message-based Parallel Processing, Scientific Computation Series, MIT Press, Cambridge, MA, 1987.
[24] K.S. Trivedi, Probability and Statistics with Reliability, Queueing and Computer Science Applications, Prentice-Hall, Englewood cliffs, NJ, 1982.
[25] R.J. Vetter, D.H.C. Du, Distributed computing with high-speed optical networks, IEEE Comput. 26 (2) (1993) 8–18.
[26] C.-L. Wu, M. Lee, Performance analysis of multistage interconnection network configurations and operations, IEEE Trans. Comput. 41 (1) (1992) 18–26.



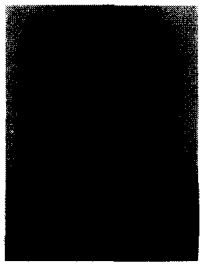
Michele Colajanni received the degree in computer science from the University of Pisa, in 1987, and the Ph.D. degree in computer engineering from the University of Roma "Tor Vergata", in 1991. In 1992, he joined the Department of Computer Engineering of the University of Roma Tor Vergata, where he is currently a Research Associate. In 1996, Dr. Colajanni was a visiting researcher at the IBM T.J. Watson Research Center in Yorktown Heights, NY.

His research interests include parallel and distributed systems, Web infrastructure, parallel computing, load balancing, and performance analysis. Dr. Colajanni is a member of the IEEE Computer Society and the ACM.



Bruno Ciciani is full professor of Computer Engineering at the University of Rome "La Sapienza" since 1994. His research interests include fault tolerance, computer architecture, distributed operating system, manufacturing yield prediction, performance and dependability evaluation of distributed and parallel computing systems. In these fields he has published more than 80 papers in referred conferences and journals.

He is the author of two books on digital circuit design and manufacturing yield evaluation published by McGraw-Hill and IEEE Computer Society Press, respectively. Prof. Ciciani spent more than one year as visiting researcher at the IBM T.J. Watson Research Center, Yorktown Heights, NY. He is a member of the IEEE Computer Society.



Salvatore Tucci received the doctorate degree in physics from the University of Pisa, Italy in 1972. From 1973 to 1975 he held a post-graduate fellowship in computer science at IEI-CNR, Pisa. In 1975, he joined the Department of Computer Science of the University of Pisa as an assistant professor. In 1987, he joined the Department of Electronics Engineering of the University of Roma "Tor Vergata" as full professor in "Calcolatori Elettronici". He held visiting positions in 1977 at INRIA, Paris, in the framework of the Sirius project, and in 1981/82 at the IBM T.J. Watson Research Center, Yorktown Heights, NY. His research interests include performance evaluation, high speed computer networks, parallel and distributed systems, multimedia applications in humanities and fine arts.

Since 1990 Prof. Tucci manages the computing centres of the University of Roma "Tor Vergata". From 1991 to 1997 he was the Dean of the Computer Engineering Curricula at the Faculty of Engineering. In 1994, he co-founded the "Dipartimento di Informatica, Sistemi e Produzione" of the University of Roma "Tor Vergata". Prof. Tucci is one of the proponents of the Center for Advanced Computing in Science and Technology (CAST) at the same university.

In addition to serving as program committee member on various conferences, he organised the summer school on Analytical Methods for Computer Performance Evaluation in Pisa in 1980, and served as the general chairman of 1989 Conference on Simulation Applied to Manufacturing, and the general chairman of Performance '93.

He co-founded the Italian Society for Computer Simulation, and is a member of IEEE Computer Society, ACM, AICA and SCS.