Wormhole Deadlock Prediction

Miriam Di Ianni

Dipartimento di Scienze dell'Informazione, Università di Roma "La Sapienza", via Salaria 113, I-00198 Roma, Italy. E-mail: diianni@dsi.uniroma1.it

Abstract. Deadlock prevention is usually realized by imposing strong restrictions on packet transmissions in the network so that the resulting deadlock free routing algorithms are not optimal with respect to resources utilization. Optimality request can be satisfied by forbidding transmissions only when they would bring the network into a configuration that will necessarily evolve into a deadlock. Hence, optimal deadlock avoidance is closely related to deadlock prediction. In this paper it is shown that wormhole deadlock prediction is an hard problem. Such result is proved with respect to both static and dynamic routing.

1 Introduction

Large-scale multiprocessors are usually organized as ensembles of nodes, each having its own processor, local memory, and other supporting devices. Since they do not physically share memory, nodes must communicate by passing messages through an interconnection network. For efficient and fair use of network resources, a message is often divided into *packets* prior to transmission: a packet is the smallest unit of communication that contains routing information. Neighboring nodes may send packets to one another directly, while nodes that are not directly connected must rely on other nodes in the network to relay packets from source to destination. This is accomplished by a routing function that selects, for each pair of nodes u and v, the set of edges incident on u that can be used to forward messages to v. It is possible to choose *all* the channels a packet will use to reach its destination before the transmission is started (*static routing*) or, conversely, one link at a time during the transmission (*dynamic routing*). A dynamic routing algorithm is called *acyclic* if it forces packets to use acyclic routes. It is called *minimal* if packets are always transmitted along shortest paths.

Deadlock is a dramatic consequence of dynamic resource (node and channel capacity) sharing: no packet can be delivered because of a cyclic wait for resources to be released by other packets. Two approaches have been taken in the literature to cope with deadlocks, namely deadlock detection and resolution, in which the routing algorithm does not take care of deadlocks that are solved by a flow control procedure whenever they occur, and deadlock prevention [2, 3], in which the routing function is properly designed in order to avoid the occurrence of deadlocks. Usually, deadlocks are avoided by imposing strong restrictions on packet transmissions in the network. Thus, the resulting deadlock free routing algorithms are not optimal with respect both to resource utilization and to the number of network configurations allowed [4]. A deadlock avoidance algorithm is *optimal* if it forbids packet transmissions only when they would bring the network into a configuration from which it is impossible for at least one packet to reach its destination. Thus, the existence of a polynomial-time algorithm predicting if a deadlock will necessarily occur implies the existence of a polynomial-time optimal deadlock avoidance algorithm. The store and forward deadlock prediction problem has already received some attention in the literature. In [1] it has been shown that the problem is *Co*-NP complete if static or acyclic dynamic routing is used for packet transmissions. Conversely, in [5] the polynomial-time decidability of the problem has been proved in the case of unrestricted dynamic routing, that is, when packets are allowed to use the same buffer an arbitrary number of times.

Objective of this paper is studying the deadlock prediction problem with respect to wormhole routing. ***Wormhole routing*** [6, 9, 8, 10] was proposed for enjoying the benefits of store and forward (highly dynamic resource sharing) while discarding its disadvantages (large network latency). A packet is divided into a number of ***flits*** (flow control digits) for transmission. The ***header flit*** of a packet, or ***worm***, governs the route. As the header advances along the chosen route, the remaining flits follow in a pipeline fashion. If the header encounters a channel already in use, it is blocked until the channel becomes available. Rather than buffering the remaining flits by removing them from the network channels (as in virtual cut through, for instance) the flow control within the network blocks the trailing flits and they remain in flit buffers along the established route. Once a channel has been acquired by a worm it is reserved for that worm. The channel is released when the last flit has been transmitted on the channel. Since blocked worms holding channels remain in the network, wormhole routing is particularly susceptible to deadlock.

In section 2 it is proved that the deadlock prediction problem is *Co*-NP complete in case of both unrestricted and minimal dynamic routing. Because of the polynomial-time decidability of the store and forward problem with respect to unrestricted dynamic routing [5], the latter result implies that wormhole deadlock prediction is definitely more difficult (modulo P$\neq$NP) than store and forward deadlock prediction. Notice that the results concerning acyclic dynamic and static routing in [1] cannot be trivially extended (by generalization) to wormhole routing, since the two models are inherently different. Furthermore, the acyclic dynamic routing considered in that paper is not minimal. In section 3 the *Co*-NP completeness is proved for static routing. Finally, in section 4 some conclusions are briefly discussed.

1.1 Preliminary definitions

Formally, a network is modeled as a pair $N = \langle G, W \rangle$ in which $G = (V(G), E(G))$ denotes the ***support graph*** of N and W is the set of worms residing in the network. A worm in the network is subject to a ***transmission*** when a vertex u transfers its header to an adjacent node v through a free channel (u, v) according to the routing function and the other flits are pipelined behind it. As already remarked, the routing strategy can be ***static*** or ***dynamic*** according to when the output channel to forward a worm to its destination is chosen. The ***network configuration*** at a given time S specifies the channel occupied by each flit at that time and, in case of static routing, the channel requested by each header flit. Any channel is assigned to at most one worm (flit) at each time step.

A network is in the ***final configuration*** if each flit has reached its own destination and thus has been removed by consumption. A ***transition*** from S to some other configuration S' is performed every time at least one worm is subject to a transmission.

A network configuration S is said ***safe*** if there exists a sequence of transitions reaching the final configuration. An unsafe configuration is called ***bound to deadlock*** and may happen because either a deadlock or a livelock occurrence. The DEADLOCK PREDICTION problem consists in deciding if a given network configuration is bound to deadlock. Depending on whether static, (unrestricted) dynamic or minimal routing is used, the corresponding deadlock prediction problem will be denoted, respectively, as SR-WDP, DR-WDP or MDR-WDP.

2 Dynamic routing

In this section we prove the hardness of predicting deadlocks if worms can choose their routes dynamically, node by node at transmission time. We start by considering unre-

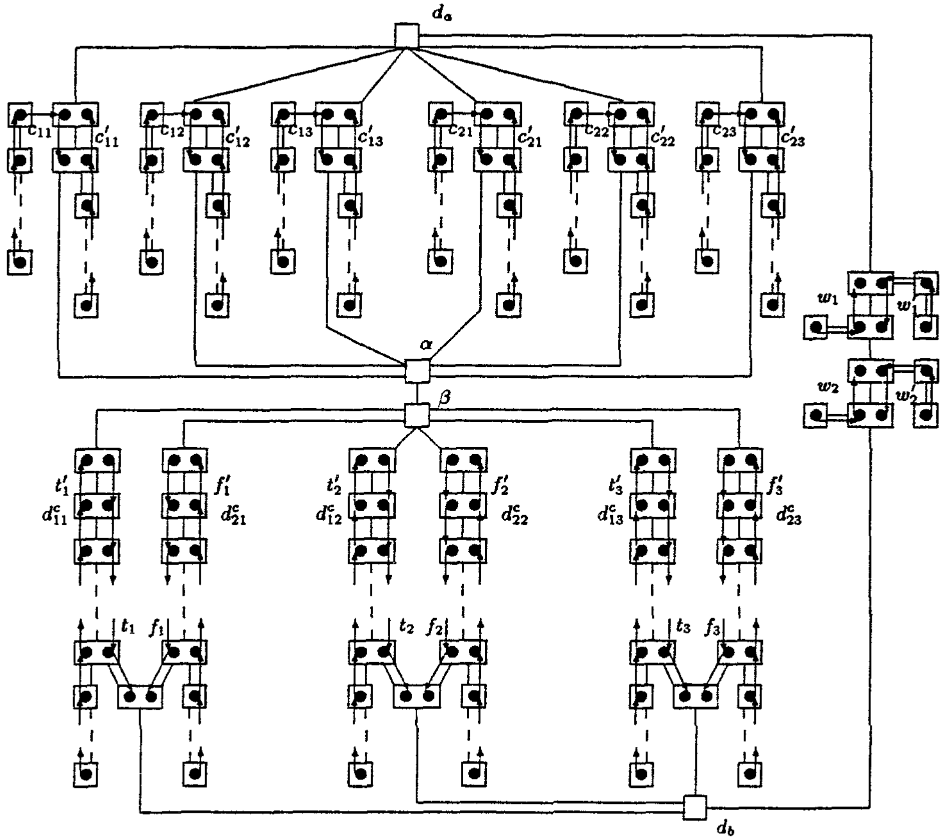

Fig. 1. Network corresponding to the boolean formula $(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$. Rectangles represent nodes, and worms are depicted as black dots chained by arrows.

stricted dynamic routing. In this case, worms are allowed to pass an arbitrary number of times through the same node, and this characteristic can be used to temporarily remove some worm from a congested zone of the network in order to allow to different worms to reach their destinations.

Theorem 1. *The* DR-WDP *problem is Co*-NP *complete also for underlying planar graphs.*

Proof. We prove that deciding if a network is in a safe configuration, is NP-complete. The membership to NP is trivial. In order to prove its completeness, we use a polynomial-time reduction from the NP-complete 3-SATISFIABILITY (in short, 3-SAT) problem [7]: given a boolean formula F in conjunctive normal form with size 3 clauses onto the set $X = \{x_1, x_2, \ldots, x_n\}$ of boolean variables, decide if there exists a truth assignment for X that makes the formula true. Let $F = c_1 \wedge c_2 \wedge \ldots \wedge c_m$ and $c_j = l_{j1} \vee l_{j2} \vee l_{j3}$, with $l_{jh} \in \{x_1, \ldots, x_n\} \cup \{\neg x_1, \ldots, \neg x_n\}$, $h = 1, 2, 3$. The corresponding planar network N_F consists of three main subnetworks: N_c, representing to the set of clauses of F, N_X, used to model a truth assignment for X, and N_w, introduced to test whether a chosen truth assignment satisfies F.

N_c is the "parallel" composition of $3m$ branches, each corresponding to a literal in a clause. Branch j_h, $j = 1, \ldots, m$ and $h = 1, 2, 3$, includes a pair of nodes containing the first two flits of worms c_{jh} and c'_{jh} which move in opposite directions and consist of $2m + 5$ flits each. Node containing the header of c_{jh} is connected to a free node α, while node containing the header of c'_{jh} is connected to a free node d_a. In turn, N_X is the "parallel" composition of n subnetworks, each corresponding to a boolean variable. Subnetwork $N(x_i)$ consists of two branches, one corresponding to the truth assignment **true** to variable x_i and the other to **false**: the first branch includes a chain of $2m + 3$ nodes containing worm t_i and the first flits of a $2m + 10$ flits long worm t'_i. Of course, the two worms move in opposite directions. Similarly, the second branch contains the two worms f_i and f'_i. The headers of t_i and f_i are contained in the same node. Nodes containing the headers of t'_i and f'_i are connected to a node β that, in turn, is connected to α, while nodes containing the headers of t_i and f_i are connected to a node d_b. Finally, N_w is a chain of $2m$ nodes. Nodes $2j - 1$ and $2j$, $j = 1, \ldots, m$, of the chain contain the first two flits of worms w_j, consisting of 3 flits, and the first two flits of worms w'_j, consisting of 4 flits. As usual, they move in opposite directions. Node containing the header of w_1 is connected to d_a, while node containing the header of w'_m is connected to d_b. Worms w_j and c_{jh}, $j = 1, \ldots, m$ and $h = 1, 2, 3$, have their destination in node d_b. Worms w_j, $j = 1, \ldots, m$, t_i, t'_i, f_i and f'_i, $i = 1, \ldots, n$, have their destination in node d_a. Finally, if $l_{jh} = x_i$ (respectively, $\neg x_i$) then the destination d^c_{jh} of worm c_{jh} is the second node of the chain containing t_i (f_i) of subnetwork $N(x_i)$. See figure 1 for an example of the reduction.

Suppose first F is satisfiable. Let $x_{i_1}, \ldots x_{i_k}$ be the variables set to **true** by a truth assignment satisfying F. If the headers of worms $t_{i_1}, \ldots t_{i_k}$ and $f_{i_{k+1}}, \ldots f_{i_n}$ are movedto d_b then at least one worm out of c_{j1}, c_{j2} and c_{j3} is able to reach its destination, for every $j = 1, \ldots, m$. Thus, all the $w_1, w_2, \ldots, w_m$ can be moved in the nodes on N_c left free after the previous movements making free the path to the destination d_a of the t_is and f_is. Next, the w_js are forwarded to their destination. Afterwards, all the t'_i and f'_i may arrive at d_a using a path through β. Hence the network is in a safe configuration.

Conversely, if F is not satisfiable then every truth assignment for X is unable to satisfy at least one clause. Thus, for every choice of $t_{i_1}, \ldots t_{i_k}$ and $f_{i_{k+1}}, \ldots f_{i_n}$ to move forward there exists at least one $j \leq m$ such that c_{j1}, c_{j2} and c_{j3} cannot reach their destinations. This implies that $w_j, w_{j+1}, \ldots, w_m$ cannot free the chain in N_w and, hence, the t_is and f_is cannot reach their destinations. Notice that the c_{jh} still remaining in N_c cannot be temporarily "parked" in some $N(x_i)$ in order to make room for w_j, since the number of free nodes is not large enough. Thus, the final configuration cannot be reached by this sequence of moves and it can be easily verified that this is true for any sequence of moves. Hence the network configuration is not safe.

Next theorem extends the previous result to minimal routing. Notice that it does not follow by generalization, since, in general, a given network configuration can be a deadlock one if only shortest paths must be used, but it can safe if worms are allowed to use *arbitrary* paths to their destinations.

Theorem 2. *The* MDR-WDP *problem is Co-*NP *complete.*

Proof. A reduction from 3-SAT is still used to prove the completeness of deciding if a network is in a safe configuration. Let $F = c_1 \wedge c_2 \wedge \ldots \wedge c_m$ and $c_j = l_{j1} \vee l_{j2} \vee l_{j3}$, with

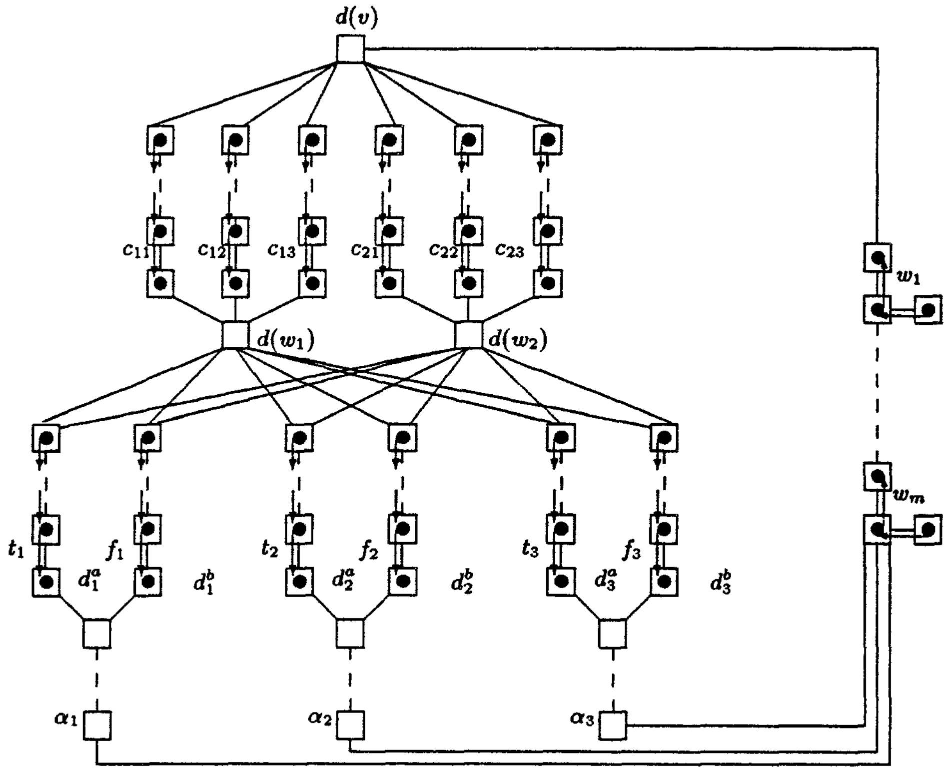

Fig. 2. Network N corresponding to a boolean formula $f(x_1, x_2, x_3)$ with two clauses.

$l_{jh} \in \{x_1, \ldots, x_n\} \cup \{\neg x_1, \ldots, \neg x_n\}$, $h = 1, 2, 3$. Clause c_j is mapped to a subnetwork $N(c_j)$. "parallel" composition of three branches, each corresponding to a literal in the clause. Branch h, $h = 1, 2, 3$, includes a chain containing a $(6m + 1)$ flits long worm c_{jh}. Node containing the header of c_{jh} is connected to a free node $d(w_j)$, while node containing its tail is connected to a free node $d(v)$; these last nodes are common to all the three branches. Variable x_i, $i = 1, \ldots, n$, corresponds to a subnetwork $N(x_i)$. As in the previous theorem, it includes the parallel composition of two branches each containing a $(2m + 1)$ flits long worm, respectively, t_i and f_i. Nodes containing the headers of t_i and f_i, respectively d_i^a and d_i^b, are connected to the first node of a $(2m+1)$ nodes long free chain ending with node α_i, while nodes containing the tail flits are connected to all nodes $d(w_j)$, $j = 1, \ldots, m$. Finally, a chain N_w of $2m$ nodes including the first two flits of m worms $w_1, \ldots w_m$, each long 3 flits, completes the network. Node containing the header of w_1 is connected to $d(v)$, while node containing the second flit of w_m is connected to all the α_i. The destination of worm w_j is $d(w_j)$, the destination of all the t_i and f_i, $i = 1, \ldots, n$, is $d(v)$ and, finally, if $l_{jh} = x_i$ (respectively, $\neg x_i$) then the destination of worm c_{jh} is node d_i^a (d_i^b). In figure 2 an example of the reduction is shown. Notice that there is only one shortest path connecting worms t_i and f_i to $d(v)$: it passes through the chain ending with α_i and then through $N(w)$. Similarly, the unique shortest path connecting worm c_{jh} to d_i^a passes through $d(w_j)$ and then through the opportune $N(x_i)$. Three shortest paths connect worm w_j to $d(w_j)$, one for every branch of $N(c_j)$, all passing through $d(v)$.

If F is satisfiable, it is easy to derive from a truth assignment satisfying F a sequence of worm movements reaching the final configuration (similarly to the proof of theorem 1). Conversely, if F is not satisfiable then every truth assignment for X is unable to satisfy at least one clause. Thus, for every choice of $t_{i_1}, \ldots t_{i_k}$ and $f_{i_{k+1}}, \ldots f_{i_n}$ to move forward there exists at least one $j \leq m$ such that c_{j1}, c_{j2} and c_{j3} cannot reach their destinations. This implies that $w_j, w_{j+1}, \ldots, w_m$ cannot reach their destinations and, hence, not even the t_is (f_is), whatever sequence of moves is performed. Indeed, let $l_{jh} = \neg x_{i_1}$ and t_{i_1} be the worm occupying the chain in $N(x_{i_1})$ that ends with α_{i_1}. In order to move f_{i_1} in the same chain and permit to c_{jh} to free the path to be used by w_j, t_{i_1} should be completely moved in $N(w)$. But t_{i_1} is $(2m+1)$ flits long and $N(w)$ includes a chain of $2m$ nodes, thus t_{i_1} cannot free the chain ending with α_{i_1} unless all the worms $w_1, \ldots w_m$ have already left $N(w)$. This means that the final configuration cannot be reached whatever sequence of moves is performed, that is, the network configuration is not safe.

3 Static routing

This section shows that knowing in advance the routes to be followed by worms does not help in predicting wormhole deadlocks, as stated in the next theorem.

Theorem 3. *The* SR-WDP *problem is Co-*NP *complete.*

Proof. Again, the NP-completeness of the complementary problem is proved by a reduction from 3-SAT. Network N_F corresponding to $F = c_1 \vee c_2 \ldots \vee c_m$, $c_j = l_{j1} \vee l_{j2} \vee l_{j3}$, $l_{jh} \in \{x_1, \ldots, x_n\} \cup \{\neg x_1, \ldots, \neg x_n\}$, $h = 1, 2, 3$, will be described (see figure 3).

Clause c_j corresponds to a subnetwork including a main cycle $\alpha_0^j, \alpha_1^j, \alpha_2^j, \alpha_3^j, \alpha_4^j, \alpha_5^j$ and a path $\beta_1^j, \beta_2^j, \beta_3^j, \beta_4^j, \beta_5^j, \beta_6^j$: the triple of nodes $\beta_{2h-1}^j, \beta_{2h}^j, \alpha_{2h-2}^j$ is occupied by worm c_{jh} corresponding to literal l_{jh}, $h = 1, 2, 3$, with the header contained in node α_{2h-2}^j, and each β_6^j is connected to β_1^{j+1}, $j = 1, \ldots, m-1$. Variable x_i, $i = 1, \ldots, n$, corresponds to a subnetwork composed by a chain $\psi_1^i, \psi_2^i, \ldots, \psi_{6m+6}^i$ with two branches connected to ψ_{6m+6}^i: the first branch $\tau^i, \mu_1^i, \mu_2^i, \ldots, \mu_{6m+6}^i$ corresponds to variable x_i and contains worm v_i, and the second one $\neg\tau^i, \neg\mu_1^i, \neg\mu_2^i, \ldots, \neg\mu_{6m+6}^i$ corresponds to variable $\neg x_i$ and contains worm $\neg v_i$. The headers of v_i and $\neg v_i$ are in nodes τ^i and $\neg\tau^i$. A chain of $6m+6$ nodes including a $6m+6$ flits long worm w completes the network. Node $d(v)$ containing the header of w is connected to β_1^1, while node ρ containing the tail flit of w is connected to all the ψ_1^i, $i = 1, \ldots, n$. Worm w must be

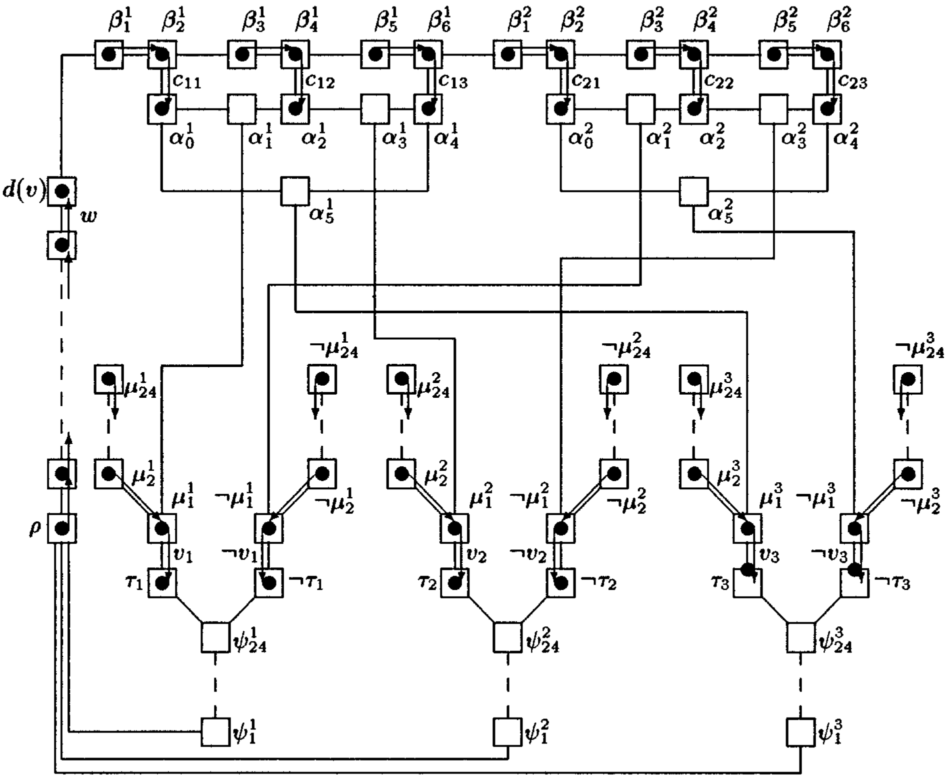

Fig. 3. Network N_F corresponding to $F = (x_2 \vee x_3 \vee x_1) \wedge (\neg x_2 \vee \neg x_3 \vee \neg x_1)$.

routed through $\beta_1^1, \beta_2^1, \ldots, \beta_6^1, \ldots \beta_1^m, \beta_2^m, \ldots, \beta_6^m$, this last node being its destination. Worms v_i and $\neg v_i$ must be routed through $\psi_{6m+6}^i, \ldots, \psi_1^i$, ρ and then through all the chain $N_F(w)$ till $d(v)$ that is their destination. Worm c_{jh} must be routed through $\alpha_{2h-1}^j, \alpha_{(2h) \bmod 6}^j, \alpha_{(2h+1) \bmod 6}^j$ and then, if the hth literal of clause c_j is x_i (respectively, $\neg x_i$), through nodes μ_1^i, τ^i $(\neg\mu_1^i, \neg\tau^i)$, this last node being its destination.

Similarly to theorem 1, if F is satisfiable it is easy to derive the sequence of movements corresponding to a truth assignment satisfying F.

Before proving the converse, notice that if v_i has occupied $\psi_1^i, \ldots, \psi_{6m+6}^i$, then the only way for a worm c_{jh} to reach its destination $\neg\tau_i$ is that v_i reaches $d(v)$. This statement is true because of the lengths of worms w, v_i and $\neg v_i$ and of the chain $\psi_1^i, \ldots, \psi_{6m+6}^i$ (i.e., worms w, v_i and $\neg v_i$ cannot be temporarily "parked" anywhere to permit to other worms to reach their destinations). Suppose now that F is not satisfiable. Then every truth assignment for X is unable to satisfy at least one clause. Thus, for every choice of $v_{i_1}, \ldots v_{i_k}$ and $\neg v_{i_{k+1}}, \ldots \neg v_{i_n}$ to move forward, there exists at least one $j \leq m$ such that c_{j1}, c_{j2} and c_{j3} cannot reach their destinations. Because of what previously noticed, the only possibility for reaching the final configuration is thus to let w arrive at its destination, in order to free the routes to $t(v)$ and thus to permit to c_{j1}, c_{j2} and c_{j3} to arrive at their destinations. To do this, c_{j1}, c_{j2} and c_{j3} must all be advanced: of one or two nodes it is easy to verify that the previous configurations are all bound to deadlock. This means that the final configuration cannot ever be reached whatever sequence of moves is performed, that is, the network configuration is not safe.

4 Conclusions

In this paper the problem of predicting wormhole deadlocks has been considered. Such problem is closely related to the one of optimally avoiding deadlocks with respect to channel utilization. Unfortunately, it turns out that predicting wormhole deadlocks is always an hard problem, both for static and for dynamic routing. Because of the results in [1] and especially of those in [5] about deadlock prediction in store and forward networks, this means that wormhole deadlock prediction definitely more difficult (modulo P$\neq$NP) than store and forward deadlock prediction.

References

1. C. Arbib, G. Italiano, A. Panconesi, "Predicting deadlock in Store-and-Forward networks", *Networks*, 20, 861-882, 1990.
2. B. Awerbuch, S. Kutten, D. Peleg, "Efficient deadlock-free routing", *Proc. of the Tenth Annual ACM Symposium on principles of Distributed Computing*, Montreal, Canada, 177-188, 1991.
3. F. Belik, "An efficient deadlock avoidance technique", *IEEE Trans. on Computers*, 39, 882-888, 1990.
4. J. Blazewicz, J. Brzezinski, G. Gambosi, "Optimization aspects of deadlock prevention in packet-switching networks", *European Journal of Operational Research*, 57, 1-12, 1992.
5. D.P. Bovet, P. Crescenzi, M. Di Ianni, "Deadlock prediction in the case of dynamic routing", *Int. Journal of Foundations of Computer Science*, 1, 185-199, 1990.
6. W.J. Dally, C.L. Seitz, "The torus routing chip", *Journal of Distributed Systems*, 1, 187-196, 1986.
7. M.R. Garey, D.S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*, Freeman, San Francisco, 1979.
8. X. Lin, P.K. McKinley, L.M. Ni, "The message flow model for routing in wormhole-routed networks", *Proc. Int. Conf. Parallel Processing*, 1993.
9. D.H. Linder, J.C. Harden, "An adaptive and fault-tolerant wormhole routing strategy for k-ary n-cubes", *IEEE Trans. Computers*, 40, 2-12, 1991.
10. P.K. McKinley, L.M. Ni, "A survey of wormhole routing techniques in direct networks", *IEEE Computer*, 62-77, 1993.