On necessary and sufficient conditions for deadlock-free routing in wormhole networks

Freek Verbeek and Julien Schmaltz
Institute for Computing and Information Sciences, Radboud University Nijmegen
School of Computer Science, Open University of The Netherlands
E-mail: f.verbeek@cs.ru.nl, julien.schmaltz@ou.nl

Supplementary file

Abstract—This paper contains supplementary material to the IEEE TPDS paper entitled "On necessary and sufficient conditions for deadlock-free routing in wormhole networks". In Section 1, we prove that deciding deadlock freedom of wormhole networks is co-NP-complete. In Section 2, we provide a counter example to a polynomial algorithm for this decision problem.

1 Complexity of deciding deadlock freedom

We prove that deciding deadlock freedom in wormhole networks (DF) is co-NP-complete by proving that the complementary problem to DF is NP-Complete. We reduce the set packing problem [1] to the problem of deciding deadlock freedom of adaptive routing functions in wormhole networks. The set packing problem is known to be NP-complete [1].

Definition 1: The *Wormhole Switching Deadlock Decision Problem* (`WHS-DL`) is defined as follows:
Given an interconnection network $I = (N, C)$ and a routing function $\mathbf{R} : N \times N \mapsto \mathcal{P}(C)$, does there exist a deadlock configuration?

Definition 2: The *Set Packing Problem* (`SP`) is defined as follows:
Given a universal set $U = \{0, \ldots, n\}$, a list of subsets of $S = s_0 s_1 \ldots, s_i$ where $s_j \subseteq U$ $(\forall_{0 \leq j \leq i})$, and an integer $k \leq i$, does S contain k pairwise disjoint sets?

The proof that `WHS-DL` is NP-complete is a standard reduction proof. Assuming a machine that solves `WHS-DL` in polynomial time, we create a machine that solves `SP` in polynomial time. This is done by a polynomial transformation τ from $L_{\texttt{WHS-DL}}$ to $L_{\texttt{SP}}$.

The idea behind the transformation is to create a network such that it contains deadlocks, but these deadlocks can only occur in one specific area in the network. This area contains a channel for each integer in U. We will refer to these channels as *U-channels*. The network is created in such a way that any deadlock requires exactly k worms. Furthermore, these worms have to cross the area containing the U-channels. The routing function creates a route through these channels for each set in S. That is, for each set s_i the network contains a corresponding node s_i. This node is used as destination to route through the U-channels corresponding to set s_i. If there is a deadlock, then there are k pairwise disjoint worms holding channels of paths created by the routing function for k destinations s_i, corresponding to k pairwise disjoint sets in S.

This research is supported by NWO/EW project Formal Validation of Deadlock Avoidance Mechanisms (FVDAM) under grant no. 612.064.811.

1.1 Transformation example

We will first present our transformation τ with an example. Consider the following instance of `SP`: $U = \{0, 1, 2, 3, 4, 5\}$, $S = \{s_0, s_1, s_2\} = \{\{0, 1, 2\}, \{4, 5\}, \{2, 4\}\}$ and $k = 2$. There is a set of k pairwise disjoint sets, namely s_0 and s_1. We transform this instance to a network and routing function such that there is a deadlock configuration if and only if there are k pairwise disjoint sets in S.

Consider the network in Figure 1. There is a set of six U-channels corresponding to the six integers in U. Two nodes n_0 and n_1 are located at respectively the start and the end of this area of U-channels. Each set s_i has a corresponding node, also called s_i. As each set s_i has its own node – which is a destination for some messages – each set s_i can be associated with its own routing. More specifically, each set s_i has its own route to get from n_0 to n_1, through the U-channels of its set. For example, set $s_0 = \{0, 1, 2\}$ routes from n_0 to n_1 through channels 0, 1 and 2.

Furthermore, there is a node n_2, two channels A and B and a node x. These elements are needed to create a cycle in the network, thereby making deadlocks possible. Node x serves only as a destination for a worm holding channels A and B.

For sake of clarity, nodes x and s_i are not drawn in Figure 1. These nodes are used only as destinations and not part of any deadlock that might occur.

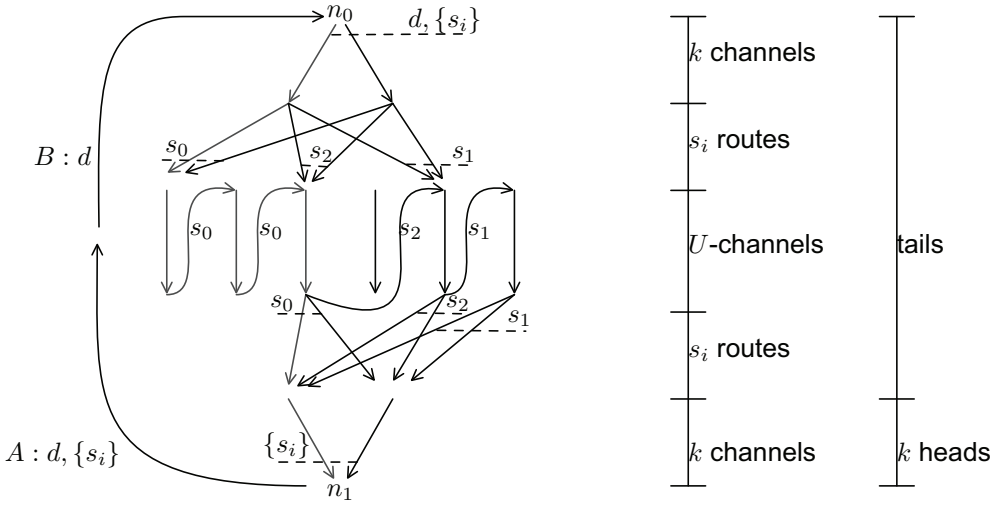

Figure 1. Example of transformation τ

A worm is drawn in Figure 1 in thick lines. It is destined for node s_0. The corresponding set is $\{0, 1, 2\}$. At node n_0 it could have taken any of the two (k) channels going to layer 1. From the nodes at layer 1, only one channel can be taken by a message destined for s_0. This channel leads to layer 2, then to the U-channel corresponding to integer 0, as 0 is the least element in s_0. Then a path from U-channel 0 to U-channel 1 to U-channel 2 is taken. After all U-channels corresponding to the set $\{0, 1, 2\}$ have been taken – in increasing order – the worm proceeds to any of the nodes at layer 4. Finally, its head is in the channel leading from layer 4 to node n_1.

Each worm that spans channels from n_0 to n_1 holds the U-channels corresponding to a set $s_i \in S$. The drawn worm holds U-channels 0, 1, and 2. A worm from n_0 to n_1 destined for s_2 holds U-channels 2 and 4. A legal configuration cannot have these two worms simultaneously, as both worms hold U-channel 2. Analogously, sets s_0 and s_2 are not pairwise disjoint.

Not all channels of the network are drawn. For all pairs of processing nodes (p_0, p_1) such that $p_0 \neq p_1$ and $p_0 \notin \{n_0, n_1\}$, there is a dedicated channel leading from p_0 to p_1. This dedicated channel is used only for messages destined for p_1. Thus, for all processing nodes other than n_0 and n_1 any message with any destination can be routed directly to its destination.

Any deadlock that may occur, occurs in the drawn channels. In a deadlock configuration all dedicated channels are empty, as otherwise a message arrives at its destination. A header of a message can only be permanently blocked in channels leading to either n_0 or n_1.

Consider the configuration where the two left channels A and B are filled with a worm destined for x. As n_0 has no outgoing dedicated channels, this worm needs to acquire one of the k channels going out of processing node n_0. If all these k channels are permanently blocked, then a deadlock configuration has occurred. However, if one of these channels contain a header, then this header will be able to use a dedicated channel to arrive at its destination. Thus all these k channels must contain tail flits only. This holds for all channels leading from n_0 to the k channels going into n_1. In a deadlock configuration, all of these channels must contain tail flits only.

Besides channel B, the only channels that can contain a header flit that can be permanently blocked are the k channels going into n_1. As processing node n_1 has no outgoing dedicated channels, the headers in these channels must acquire channel A, which is held by a worm. Thus, any deadlock configuration in this network contains exactly k+1 worms: one worm holding channels A and B and k worms holding channels from n_0 to n_1.

In this example, there were $k = 2$ pairwise disjoint sets in S, namely $s_0 = \{0, 1, 2\}$ and $s_1 = \{4, 5\}$. Thus the translated network must contain a deadlock configuration. The channels of the worm concerning s_0 have been drawn in thick lines. There is also a worm for s_1 going through channels 4 and 5. These two worms, together with a worm holding channels A and B constitute a deadlock. The reason there is a deadlock is that the area of U-channels can be crossed by 2 worms that do not intersect.

In general, if there is a deadlock configuration, then apparently there is a way to construct k worms crossing the area of the U-channels. These worms are pairwise disjoint. Each worm holds the channels belonging to one of the sets s_i. Thus k pairwise disjoint sets in S have been found. Assuming there is an algorithm that decides `WHS-DL` in polynomial time, it is possible to create an algorithm deciding `SP` in polynomial time. Deciding deadlock freedom of adaptive routing functions in wormhole networks is co-NP-complete.

1.2 Proof

Theorem 1: `WHS-DL` is NP-complete. That is:

$$L_{\texttt{SP}} \propto L_{\texttt{WHS-DL}}$$

Proof: We define a polynomial transformation τ and prove that:

$$x \in L_{\mathtt{SP}} \iff \tau(x) \in L_{\mathtt{WHS-DL}}$$

This proof follows the exact intuition in Figure 1. First, we formally define the transformation τ. Secondly we show that, given k pairwise disjoint sets, we can create a deadlock configuration in the interconnection network. All seven properties of Definition 7 hold for this configuration. Lastly, we show that any deadlock configuration necessarily has k+1 worms, where k worms cross the area of U-channels. Since these worms are – by Definition 7 – pairwise disjoint, k pairwise disjoint paths through the U-channels exist. Each path corresponds to a set in S. Thus there are k pairwise disjoint subsets in S.

Definition of τ

Given a universal set $U = \{0, \ldots, u'\}$, a set of subsets $S = \{s_0, \ldots, s_{i'}\}$ and an integer k', an interconnection network I_τ is constructed: $I_\tau = G(N_\tau, C_\tau)$.

The set of processing nodes N_τ is defined as follows:

$$N_\tau = \{n_0, n_1, n_2\} \cup$$
$$\{n_0^1, \ldots, n_{k'-1}^1, n_0^2, \ldots, n_{u'}^2, n_0^3, \ldots, n_{u'}^3, n_0^4, \ldots, n_{k'-1}^4, x, s_0, \ldots, s_{i'}\}$$

We explain this set by Figure 1. Nodes n_0 and n_1 are given. Node n_2 is the node between channels A and B. Nodes n_k^1 and n_k^4 $(0 \leq k < k')$ are the k' nodes at respectively the first and the fourth layer. Nodes n_u^2 and n_u^3 $(0 \leq u \leq u')$ are the $u' + 1$ nodes at respectively the second and third layer. There is a node x, which serves as separate destination for the worm in channels A and B. Lastly, each node s_i $(0 \leq i \leq i')$ serves as destination for the worms crossing the area of U-channels.

The set of channels C_τ is defined as follows. There is a channel from n_0 to each node n_k^1. There is a channel from each node n_k^1 to each node n_u^2. There is channel for each pair (n_u^2, n_u^3). For all $u < v \leq u'$, there is a channel from node n_u^3 to n_v^2 (note that not necessarily all these channels are ever supplied by the routing algorithm). There is a channel from each node n_u^3 to each node n_k^4. There is a channel from each node n_k^4 to n_1. Lastly, there are two channels A and B, respectively from n_1 to n_2 and from n_2 to n_0. All channels specified are identified by their processing nodes. E.g. channel B is identified as (n_2, n_0).

There is also a set of dedicated channels. For each set of processing nodes (p_0, p_1), where $p_0 \notin \{n_0, n_1\}$, there is a dedicated channel identified as $\mathcal{D}_{p_0p_1}$. Nodes n_0 and n_1 have outgoing dedicated channels as well, but not leading to any of the nodes s_i or x. This concludes the definition of the interconnection network.

Routing function $\mathbf{R}_\tau$ is defined as follows: given a current node s and a destination d, always supply the dedicated channel $\mathcal{D}_{sd}$. Furthermore, adaptively supply the following extra channels for the following pairs $< s, d >$ as specified in Table 1.

$< s, d >$	Channels
$< n_0, s_i >$	$\{(n_0, n_k^1)\}$
$< n_0, x >$	$\{(n_0, n_k^1)\}$
$< n_k^1, s_i >$	(n_k^1, n_u^2) if and only if u is the least element of set s_i
$< n_u^2, s_i >$	(n_u^2, n_u^3) if and only if $u \in s_i$
$< n_u^3, s_i >$	(n_u^3, n_v^2) if and only if v is the next element in s_i after u
$< n_u^3, s_i >$	$\{(n_u^3, n_k^4)\}$ if and only if u is greatest element of s_i
$< n_{k'}^4, s_i >$	$(n_{k'}^4, n_1)$
$< n_1, s_i >$	A
$< n_1, x >$	A
$< n_2, x >$	B

Table 1

This routing function only requires the current node and the destination node to compute the set of next hops. Its type is therefore $N \times N \mapsto \mathcal{P}(C)$ and it satisfies our assumption that routing is memoryless. A worm spanning channels from n_0 to n_1 crosses the U-channels in increasing order. There are no cyclic dependencies in the area of U-channels.

$(\Longrightarrow)$

Assume a universal set $U = \{0, \ldots, u'\}$, a set of subsets $S = \{s_0, \ldots, s_{i'}\}$ and an integer k'. Also, assume that there exists k' pairwise disjoint sets in S. Let P be a list of k' indices such that for all indices i and j in P $(i \neq j)$, sets s_i and s_j have an empty intersection.

We show that the transformation has a deadlock. This deadlock is obtained by creating $k' + 1$ worms. One worm, destined for destination x, holds channels A and B. Each worm w_k $(0 \leq k < k')$ is destined for $s_{P[k]}$. The tail holds the following channels: (n_0, n_k^1) and the intermediate path of (non-dedicated) channels supplied by the routing function from n_k^1 to n_k^4 for destination $s_{P[k]}$. The head of the worm holds channel (n_k^4, n_1). Each channel c of a worm is filled with exactly cap(c) flits.

To show that this is a deadlock, the seven properties of a deadlock configuration have to be discharged. Properties 1, 4, 5 and 7 clearly hold. These respectively state that buffer capacities are not exceeded, worms form valid paths, no header flit has arrived at its destination and that tail flits cannot proceed as the channels of the worms are full. Property 2 holds, as for each worm w_k both the starting channel (n_0, n_k^1), the last channel (n_k^4, n_1) and the channels of the intermediate path are supplied for destination $s_{P[k]}$. Property 3 states that each channel has flits belonging to one message only. Channels (n_0, n_k^1) and (n_k^4, n_1) are filled with flits belonging to worm w_k only. The channels of the intermediate paths are filled with flits belonging to one worm, as all sets $s_{P[k]}$ are pairwise disjoint. As for property 6, the worm holding channels A and B has no available next hops as node n_0 has no dedicated channel for destination x and all channels (n_0, n_k^1) are filled with tails of the worms w_k. As for the worms w_k, as node n_1 has no dedicated channels for any of the destinations s_i, all these worms require channel A. This channel is unavailable. The created deadlock is a legal configuration which satisfies all properties. It is also reachable, as for memoryless routing functions, i.e.,

routing functions of type $\mathbf{R} : N \times N \mapsto \mathcal{P}(C)$, any legal configuration is reachable (Lemma 1).

($\Longleftarrow$)
Assume an interconnection network I_τ and a routing function $\mathbf{R}_\tau$, result of the transformation $\tau(x)$, where $x = (U, S, k')$. Also, assume a deadlock configuration σ. We show that there exists k' pairwise disjoint subsets in S.

In I_τ, any deadlock configuration satisfies the following constraints. Any deadlock configuration has k' worms, whose tails start in the channels (n_0, n_k^1), and whose heads are in the channels (n_k^4, n_1). Lastly, any deadlock configuration has a worm holding channels A and B.

This is shown by analysis of the possible locations of the headers of the worms in the deadlock. In a deadlock, a header cannot be in a dedicated channel, as this contradicts property 4 of the deadlock definition. If the dedicated channel contains tail flits only, then a header has arrived at is destination and all these flits will eventually evacuate the network. In a deadlock configuration all dedicated channels are empty.

This implies that any channel leading to a dedicated channel for all destinations cannot contain a header flit. Otherwise property 5 is contradicted. Since all channels lead to dedicated channels for all destinations, except for channel B and the channels (n_k^4, n_1), the header flits of a deadlock must be in these channels.

There are no dependency cycles in the area of U-channels, as the routing function supplies all channels (n_u^2, n_u^3) in increasing order. Thus between nodes n_0 and n_1 there are no dependency cycles. The only other channels are A and B and the only other dependency is from A to B. Thus any cycle in the dependency graph of I_τ contains the dependency from A to B. As any deadlock necessarily contains a dependency cycle [2], channels A and B must be filled. Channel A cannot contain a header, implying that channels A and B contain a worm. This worm is destined for destination x, as channel B contains the header of this worm and can only contain messages destined for x. The worm cannot hold any other channels than A and B: all channels (n_k^4, n_1) are not supplied by $\mathbf{R}_\tau$ for destination x.

This worm has k' possible next hops: the channels (n_0, n_k^1). Thus all these channels must be filled with tail flits. This requires k' different worms, as it is impossible to route - for one destination - from channel (n_0, n_k^1) to another channel (n_0, n_l^1). This holds because channel B is only supplied for messages destined for x and channels (n_k^4, n_1) are only supplied for destinations s_i.

Thus σ has $k'+1$ worms. Each worm w_k starts in node n_0 and ends at node n_1. Each worm w_k has as destination s_i for some i, as its header is in some channel (n_k^4, n_1) and this channel can only be used by messages destined for some s_i. Furthermore, there are no two worms with the same destination s_i, as this implies they would intersect at the channels (n_u^2, n_u^3) with $u \in s_i$. Thus each worm w_k corresponds – one to one – to a destination s_i.

Each channel (n_u^2, n_u^3) corresponds to element $u \in U$. Each destination s_i routes a worm through channel (n_u^2, n_u^3) if and only if $u \in s_i$. Since 1.) the k' worms are destined for different s_i, 2.) each worm holds a path from n_0 to n_1, 3.) each path from n_0 to n_1 for destination s_i holds channels corresponding to the set of s_i, and 4.) all worms are pairwise disjoint, there are k' sets in S that are pairwise disjoint. □

This results complements Di Ianni's one about deadlock prediction [3]. The problems concern two related but inherently different questions. The prediction problem answers whether all traces result in a deadlock configuration whereas the decision problem answers whether there exists a reachable deadlock configuration.

2 Counter-example to existing algorithm

Taktak et al. defined an algorithm deciding deadlock-freedom of wormhole switched networks [4]. More specifically, their algorithm checks a condition for deadlock-freedom of which they prove it is necessary and sufficient. Their algorithm has a running time complexity of $O(N^4)$. We show that their algorithm decides a condition that is sufficient for deadlock-freedom, but not necessary. We show that their algorithm does not take into account that tails of worms cannot intersect. We shortly address their condition and provide a counterexample. For formal definitions and proofs, we refer to [4].

The algorithm of Taktak et al. divides the network into strongly connected components (SCC) and then decides deadlock-freedom for each SCC separately. Taktak et al. define a *tagging condition*. Each channel has a set of labels corresponding to the messages that can use the channel. Their algorithm attempts to tag all labels according to their tagging condition. If all labels can be tagged, then the network is deadlock-free.

2.1 Tagging condition

We supply some informal definitions:

Definition 3: Channel c will be labeled with label d if and only if c can be used by a message destined for d.

Definition 4: Given a labeled channel dependency graph, a label l of a node c is *tagged* if and only if for all successors of c which owns l as label, l is tagged, and there is at least one successor of c having l as label and having all its labels tagged.

Theorem 2: An SCC C has a deadlock configuration if and only if there is at least one label l of one node $c \in C$ that cannot be tagged according to the tagging condition.

Taktak et al. state that a deadlock configuration can be obtained by filling channels which have labels that cannot be tagged. They do not provide any methodology for this filling, e.g., they do not specify which channels are filled with headers and which channels are filled with tail flits.

2.2 Counterexample

We provide an example of a deadlock-free network of which Taktaks algorithm would state that it contains a deadlock.

Consider the interconnection network in Figure 2. It has processing nodes n_i ($0 \leq i \leq 5$), d_0 and d_1. For sake of clarity we consider the labels of messages destined for d_0 and d_1 only. The routing function is specified by Table 2 for destinations d_0 and d_1.

We make sure that if a deadlock is possible, it concerns only messages destined for d_0 and d_1. This is done by using dedicated channels. A dedicated channel is a channel that leads directly to its destination and can be used by messages destined for that destination only. For all pairs of processing nodes (n_i, n_j) $(i \neq j)$, we add a dedicated channel $\mathcal{D}_{ij}$ from n_i to n_j. Also, there are dedicated channels from d_0 to all other processing nodes. The same holds for d_1. The dedicated channels ensure that both any message destined for any node n_i and any message injected at nodes d_0 and d_1 can never be permanently blocked. Any possible deadlock-configuration consists of messages generated at nodes n_i and destined for nodes d_i. For these messages we will show the network to be deadlock-free.

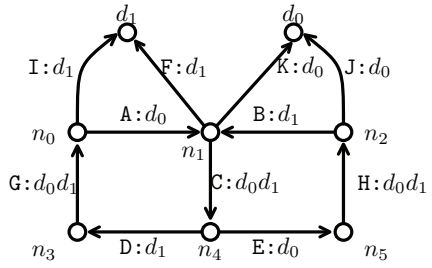

Figure 2. The interconnection network

R	d_0	d_1
n_0	A	I
n_1	{C, K}	{C, F}
n_2	J	B
n_3	G	G
n_4	E	D
n_5	H	H

Table 2
The routing function

Assume channels I, F, K and J are empty. Otherwise a message can arrive at its destination. If there is a head in either channel A or B, then this head can escape to channel F or K and arrive at its destination. If channels A and B both contain tail flits only, then at least one of the channels F or K contains a header flit, implying a message arrives at its destination. If only one of the channels A and B is filled with tail flits only and the other is empty, then there is always a header flit that can move. Say channel A has tail flits only and channel B is empty. There is a header flit in either C, E or H that can move forward. Say channel B has tail flits only and channel A is empty. There is a header flit in either C, D or G that can move forward. No deadlock is possible.

Although the network is deadlock-free, the network could be in deadlock if worms could intersect. Consider the – illegal – configuration in Figure 3. Channels G and H are filled with messages destined for d_0 and d_1 respectively. There are two intersecting worms, one occupying channels A, C, and E (destined for d_0) and one occupying B, C and D (destined for d_1). None of these messages can move. However, channel C is occupied by flits of worm 1 and worm 2, making the configuration illegal. There is a deadlock possible in this network only if worms can intersect.

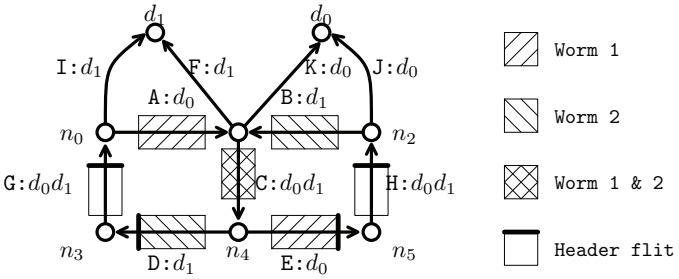

Figure 3. An illegal deadlock

Figure 4 shows the labeled channel dependency graph. All labels that can be tagged according to the tagging condition have been underlined. Taktak et al. state that a deadlock configuration can be obtained by filling channels which have labels that cannot be tagged with messages destined for untagged labels. The configuration obtained is however exactly the illegal deadlock-configuration where the tails of two worms intersect.

We show, informally, that the labels of channel C can never be tagged. We consider the main SCC consisting of channels A, B, C, D, E, G and H. The labels of the other channels are assumed to be tagged.

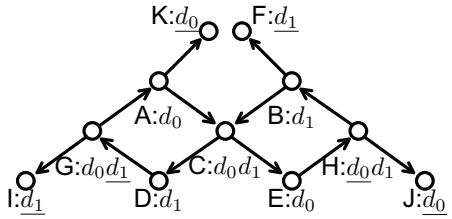

Figure 4. Labeled channel dependency graph. Labels that can be tagged are underlined.

Consider the tagging of label d_0 for channel C. In order to tag this label, all labels of channel E must be tagged, as this is the only successor with d_0 as label. For the tagging of the labels of E, all the labels of H must be tagged. Label d_0 can indeed be tagged for channel H since for all successors having d_0 as label (J only), d_0 is tagged and since there is one successor – J – having all its labels tagged. In order to tag d_1 for channel H, d_1 must be tagged for channel B. Channel B does have a

successor – F – which owns d_1 as label and has all its labels tagged. However, in order to tag d_1 for B, d_1 must be tagged for C as well.

In order to tag d_1 for C the exact same reasoning of the previous paragraph holds, with d_0 and d_1 swapped. Thus in order to tag label d_0 for channel C, label d_1 must be tagged for channel C and in order to tag label d_1 for channel C, label d_0 must be tagged for channel C. This leads into a cyclic reasoning, and thus the labels of destination C cannot be tagged.

As not all labels can be tagged, Theorem 2 states that the network has a deadlock. It is however deadlock-free.

References

[1] R. M. Karp, "Reducibility among combinatorial problems," pp. 83–105, 1972.

[2] W. Dally and C. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Transactions on Computers*, no. 36, 1987.

[3] M. D. Ianni, "Wormhole deadlock prediction," in *Euro-Par'97 Parallel Processing*, vol. 1300, 1997, pp. 188–195.

[4] S. Taktak, E. Encrenaz, and J.-L. Desbarbieux, "A polynomial algorithm to prove deadlock-freeness of wormhole networks," in *18th Euromicro International Conference on Parallel, Distributed and Network-Based Computing (PDP'10)*, February 2010.