Deadlock in Packet Switching Networks

Anna Stramaglia, Jeroen Keiren, Hans Zantema

Deadlock in packet switching networks

Anna Stramaglia, Jeroen J.A. Keiren[0000−0002−5772−9527], and Hans Zantema

Eindhoven University of Technology, the Netherlands
{a.stramaglia, j.j.a.keiren, h.zantema}@tue.nl

Abstract. A deadlock in a packet switching network is a state in which one or more messages have not yet reached their target, yet cannot progress any further. We formalize three different notions of deadlock in the context of packet switching networks, to which we refer as global, local and weak deadlock. We establish the precise relations between these notions, and prove they characterize different sets of deadlocks. Moreover, we implement checking of deadlock freedom of packet switching networks using the symbolic model checker nuXmv. We show experimentally that the implementation is effective at finding subtle deadlock situations in packet switching networks.

Keywords: Packet switching network · Deadlock · Model checking.

1 Introduction

Deadlock is a historically well known bug pattern in computer systems where, in the most general sense, a system reaches a state in which no operation can progress any further. Deadlocks can occur in many different contexts, such as operating systems [6], databases [17], computer networks, and many others [18], provided one interprets the processes and resources involved appropriately. Regardless of the context, deadlock is a situation that we generally want to avoid.

A packet switching network consists of nodes, connected by (directed) channels. Packets are exchanged in a store-and-forward manner. This means that a node in the network first receives a packet in its entirety, and then decides along which output channel to forward the packet based on a routing function. The possible steps in the network are: sending a packet to some other node, processing the packet by first receiving and then forwarding it, and finally, receiving a packet when it reaches its destination node.

Packet switching networks have been around for decades, and the problem of deadlock in such networks was already described early on [10]. Basically a deadlock arises if packets compete for available channels. There are different ways to deal with deadlocks. First, in deadlock avoidance, extra information in the network is used to dynamically ensure deadlock freedom. Second, in deadlock prevention, deadlock freedom is ensured statically, e.g. based on the network topology and the routing function. Finally, networks with deadlock detection are less restrictive in their routing. Deadlocks that result from these relaxed routing schemes are detected and resolved using an online algorithm [4, 9].

Many packet switching networks have a dynamic topology, and therefore use deadlock avoidance or deadlock detection. However, from the early 2000s, Networks on Chip (NoCs) brought packet switching and deadlock prevention to the level of interconnect networks in integrated circuits [1, 7]. Since such NoCs have a static topology, they are amenable to deadlock prevention.

Deadlock prevention was studied, e.g., by Chen in 1974 [4], who referred to prevention as *"system designs with built-in constraints which guarantee freedom from deadlocks without imposing any constraints on real-time resource allocation"*. Later, in the 1980s, Toueg and Ullman addressed deadlock prevention using local controllers [13]. Duato [8] was the first one to propose necessary and sufficient conditions for deadlock-free routing in packet switching. In the context of NoCs, Verbeek [14] and Verbeek and Schmaltz [15, 16] formulated a necessary and sufficient condition for deadlock-free routing that is equivalent to that of Duato. In this paper, like [15], we consider networks with deterministic routing functions. However, [15] formalizes sufficient conditions on the network that guarantee deadlock freedom; whereas, even if those conditions are not satisfied, our approach is able to prove or disprove deadlock freedom of the network. The notion of *local deadlock* we introduce in Section 3.2 is equivalent to those of Duato and Verbeek. This paper is based on preliminary results in [12].

Contributions. In this paper, we focus on *deadlock prevention* in packet switching networks, with a particular interest in NoCs. With this we mean the static-checking for deadlock-freedom prior to the development of the packet switching network. We restrict ourselves to networks with deterministic, incremental and node-based routing functions. We formalize three different notions of deadlock, namely global, local and weak deadlock. The definition of global deadlock is the standard definition in which no message can make progress in the entire network. A weak deadlock is a state in which no steps other than send steps are possible. A state is a local deadlock if some filled channels are blocked, i.e., they contain a message that can never be forwarded by the target of the channel. We show that every global deadlock is a weak deadlock, and every weak deadlock is a local deadlock. Furthermore, not every local deadlock is a weak deadlock. However, from a weak deadlock a local deadlock in the same network can be constructed.

Finally, we show how a packet switching network and the deadlock properties can be formalized using nuXmv [2] and CTL [5]. Our experiments indicate that different types of deadlock are found, or their absence is proven, effectively in packet switching networks. However, verification times out due to the state space explosion when numbers of nodes and channels increase.

Structure of the paper. In Section 2 we define packet switching networks and their semantics. Subsequently, in Section 3 we introduce three notions of deadlock, that are compared in detail in Section 4. In Section 5 we describe a translation of packet switching networks and deadlocks into nuXmv and CTL, and describe an experiment with this setup. Conclusions are presented in Section 6. The paper includes proof sketches of the results. For full proofs, the reader is referred to [11].

2 Preliminaries

2.1 Packet switching network

A packet switching network consists of a set of nodes connected by (unidirectional) channels. A subset of the nodes is considered to be terminal. Any node in the network can receive a message from an incoming channel and forward it to an outgoing channel. Terminal nodes can, furthermore, send messages into the network and receive messages from the network. When forwarding a message or sending a message, this is always done in accordance with the routing function. In this paper we consider networks with a static, deterministic routing function. The framework we present could be generalized to a non-deterministic setting. Formally, a packet switching network is defined as follows [7].

Definition 1. *A packet switching network is a tuple* $\mathcal{N} = (N, M, C, rout)$ *where:*

- N *is a finite set of nodes,*
- $M \subseteq N$ *is the set of terminals, nodes that are able to send and receive messages, with* $|M| \geq 2$*,*
- $C \subseteq N \times N$ *is a finite set of channels, and*
- $rout \colon N \times M \to C$ *is a (deterministic) routing function.*

For channel $(n, m) \in C$ *we write* $source((n, m)) = n$ *and* $target((n, m)) = m$*. We require* $source(rout(n, m)) = n$ *for every* $n \in N$*,* $m \in M$*, with* $m \neq n$*. We write* $c = m$ *to denote that channel* c *contains a message with destination* m*, and write* $c = \bot$ *to denote channel* c *is empty. We write* $M_\bot$ *to denote* $M \cup \{\bot\}$*.*

Routing function *rout* decides the outgoing channel of node n to which messages with destination m should be forwarded.

For $m \in M$, $n \in N$ (with $m \neq n$), the next hop $next_m \colon N \to N$ is defined as $next_m(n) = target(rout(n, m))$.

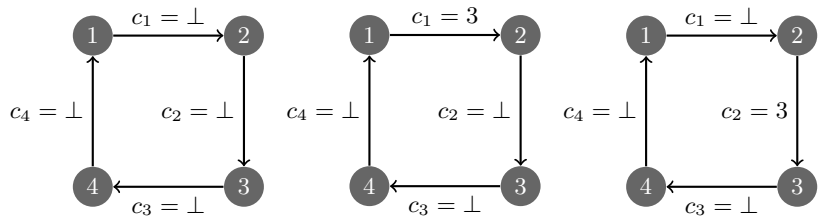
A packet switching network is *correct* if, whenever a message is in a channel, the routing function is such that the message can reach its destination in a bounded number of steps. In essence, this means the routing function does not cause any messages to cycle in the network.

Definition 2. *Let* $\mathcal{N} = (N, M, C, rout)$ *be a packet switching network. The network is* correct *if for every* $m \in M$*, and* $n \in N$ *there exists* $k \geq 0$ *such that*

$$next_m^k(n) = m,$$

where $next_n^0(n) = n$ *and* $next_m^{k+1}(n) = next_m^k(next_m(n))$*.*

In our examples, we typically choose the routing function such that k is minimal, i.e., the routing function always follows the shortest path to the destination. In any given state of the network a channel may be free, or it may be occupied by a message. In the latter case it blocks access to that channel for other messages. Processing in the network is *asynchronous*, which means that at any moment a step can be done without central control by a clock. The content of a channel is identified by the destination $m \in M$ of the corresponding message. More precisely, the following steps can be performed in a packet switching network:



(a) Initial state (b) send step (c) process step

Fig. 1: A packet switching network with send, process and receive steps

Send Terminal $m \in M$ can send a message to terminal $m' \in M$ by inserting a message in channel $rout(m, m')$, provided this channel is currently empty. After sending, this channel is occupied by m'.

Receive If channel $c \in C$ with $target(c) = m$ contains a message with destination m, the message can be received by terminal m and c becomes free.

Process If channel $c \in C$ contains a message with destination $m \in M$, and $target(c) = n \neq m$ for $n \in N$, then the message can be processed by node n by forwarding it to channel $c' = rout(n, m)$. This step can only be taken if channel c' is free. As a result, the message is removed from channel c (which now becomes free) and moved to channel c'.

We illustrate these steps in a packet switching network in Example 1.

Example 1. Consider the packet switching network in Figure 1. The network consists of four nodes, i.e., $N = \{1, 2, 3, 4\}$, all of which are terminals, so $M = N$, and four channels, $C = \{c_1, c_2, c_3, c_4\}$, shown as arrows from source to target. The routing function is $rout(n, m) = c_n$ for all $n \in N$ and $m \in M$. Initially, all channels are empty, this is shown in Figure 1a. From the initial state it is possible to perform a *send* step from any of the nodes. For example, since channel $c_1 = \bot$, a message can be sent from node 1 to node 3. The message is routed to c_1. The resulting state is shown in Figure 1b. Now, $c_1 = 3$ and $c_2 = \bot$, hence node 2 can perform a *process* step, and forward the message to c_2. The resulting situation is shown in Figure 1c. Finally, since $c_2 = 3$, and $target(c_2) = 3$, node 3 can execute a *receive* step, and consume the message from channel c_2. Consequently, all channels are empty and the system is back to the initial state shown in Figure 1a.

2.2 Semantics of packet switching networks

We formalize the semantics of packet switching networks using Kripke structures.

Definition 3. *Let AP be a set of atomic propositions. A Kripke structure over AP is a four-tuple $K = (S, I, \rightarrow, L)$, where:*

- *S is a (finite) set of states,*

- $I \subseteq S$ *is the set of initial states,*
- $\rightarrow \subseteq S \times S$ *is the transition relation, which is total, i.e., for all* $s \in S$ *there exists* $t \in S$ *such that* $s \rightarrow t$*, and*
- $L\colon S \rightarrow 2^{AP}$ *is a labelling function that assigns a set of atomic propositions to each state.*

In general, the set of states in a Kripke structure may be an overapproximation of the states that can be reached from an initial state. In this paper we sometimes only consider the *reachable* states of the system.

Definition 4. *Let* $K = (S, I, \rightarrow, L)$ *be a Kripke structure. The set of* reachable states *of* K *is defined as follows:*

$$R(K) = \{s' \in S \mid \exists s \in I\colon s \rightarrow^* s'\}$$

where $\rightarrow^*$ *denotes the reflexive transitive closure of* $\rightarrow$*.*

We now formalize the semantics of a packet switching network. This captures the intuitions described in Section 2.1.

Definition 5. *Given packet switching network* $\mathcal{N} = (N, M, C, rout)$*, its semantics is defined as the Kripke structure* $K_{\mathcal{N}} = (S, I, \rightarrow, L)$ *over* $AP = \{c = m \mid c \in C \wedge m \in M_\perp\}$*, defined as follows:*

- $S = M_\perp^{|C|}$*, i.e., the state of the network is the content of its channels. If* $C = \{c_1, \ldots, c_{|C|}\}$ *we write* $\pi_{c_i}(s) = v_i$ *if* $s = (v_1, \ldots, v_{|C|}) \in S$*,*
- $I = \{s \in S \mid \forall c \in C\colon \pi_c(s) = \perp\}$*, i.e., initially all channels are empty,*
- *transition relation* $\rightarrow \subseteq S \times S$ *is* $\rightarrow_s \cup \rightarrow_p \cup \rightarrow_r$*, where*
  - $\rightarrow_s$ *is the least relation satisfying*

$$\frac{m, m' \in M \quad m \neq m' \quad c = rout(m, m') \quad v_c = \perp}{(v_1, \ldots, v_c, \ldots, v_{|C|}) \rightarrow_s (v_1, \ldots, m', \ldots, v_{|C|})}$$

    characterising that terminal m *sends a message to terminal* m'*,*
  - $\rightarrow_p$ *is the least relation satisfying*

$$\frac{m \in M \quad v_c = m \quad target(c) = n \quad rout(n, m) = c' \quad v_{c'} = \perp}{(v_1, \ldots, v_c, \ldots, v_{c'}, \ldots, v_{|C|}) \rightarrow_p (v_1, \ldots, \perp, \ldots, m, \ldots, v_{|C|})}$$

    characterising that node n *forwards a message with destination* m *that comes in on channel* c *to channel* c'*, and*
  - $\rightarrow_r$ *is the least relation satisfying*

$$\frac{m \in M \quad v_c = m \quad target(c) = m}{(v_1, \ldots, v_c, \ldots, v_{|C|}) \rightarrow_r (v_1, \ldots, \perp, \ldots, v_{|C|})}$$

    characterising that terminal m *receives a message along its incoming channel* c*.*
- $L(s) = \bigcup_{c \in C}\{c = m \mid \pi_c(s) = m\}$*, for every* $s \in S$*.*

Note that it is straightforward to show that $\rightarrow_s$*,* $\rightarrow_p$ *and* $\rightarrow_r$ *are pairwise disjoint. We sometimes write, e.g.,* $\rightarrow_{pr}$ *instead of* $\rightarrow_p \cup \rightarrow_r$*. We write* $\not\rightarrow_X$ *if there is no* $s' \in S$ *such that* $s \rightarrow_X s'$ *for* $x \subseteq \{s, p, r\}$*. To ensure that the transition relation is total, we extend* $\rightarrow$ *with transitions* $s \rightarrow s$ *whenever* $s \not\rightarrow_{spr}$*.*

3 Deadlocks

The key question about packet switching we are interested in is whether a network is deadlock free. Intuitively, a network contains a deadlock if a message is stuck in a channel, and it will never be processed or received by the target of the channel. In practice, we can distinguish different notions of deadlock, each of which has a different interpretation of this informal requirement. We introduce three such notions, and study the relation between them.

3.1 Global deadlock

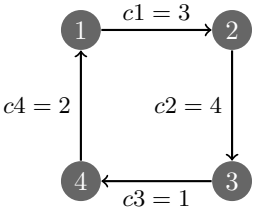
Typically a global deadlock is a state that has no outgoing transitions. However, since we are dealing with Kripke structures, which have a total transition relation, every state has an outgoing transition. A *global deadlock* is, therefore, a state that has no outgoing transitions to a state other than itself.

Definition 6. *Let $K = (S, I, \rightarrow, L)$ be a Kripke structure. The set of global deadlock states in K is defined as:*

$$G(K) = \{s \in S \mid \nexists s' \in S \colon s \neq s' \wedge s \rightarrow s'\}$$

When $s \in G(K)$, we say that s is a global deadlock.

Example 2. Recall the packet-switching network from Example 1. The situation in which all nodes have sent a message two hops away is shown on the right. All channels contain a value $m \in M$, but none of them can make progress because the next hop is blocked by another message. For instance, message 3 in c_1 has to reach node 3, but $rout(2, 3) = c_2$ is blocked by message 4. There is a cycle of blocked channels, where all of them are filled, hence the network is in a global deadlock.



The semantics of packet switching networks guarantees that there is no global deadlock among the initial states.

Lemma 1. *Let $\mathcal{N} = (N, M, C, rout)$ be a packet switching network with $K_{\mathcal{N}} = (S, I, \rightarrow, L)$ its semantics. Then $I \cap G(K_{\mathcal{N}}) = \emptyset$*

Proof. Since $|M| \geq 2$, and all channels are initially empty, there is a terminal node that can send a message into the network. □

3.2 Local deadlock

Even if not all of the channels in a packet switching network are blocked, it can happen that a subset of the channels is deadlocked. Such a situation is not covered by the global deadlock. We therefore introduce the *local deadlock*. Intuitively, a state is a local deadlock if it has a channel that indefinitely contains the same message.

Definition 7. *Let $\mathcal{N} = (N, M, C, rout)$ be a packet switching network, and $K_{\mathcal{N}} = (S, I, \rightarrow, L)$ its semantics. The set of local deadlock states in $K_{\mathcal{N}}$ in which channel $c \in C$ is deadlocked is defined as:*

$$L_c(K_{\mathcal{N}}) = \{s \in S \mid \forall s' \in S \colon s \rightarrow^* s' \implies \pi_c(s) \neq \bot \wedge \pi_c(s') = \pi_c(s)\}$$

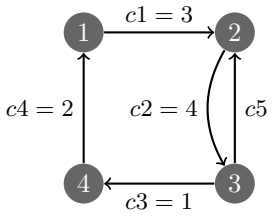
The set of local deadlock states is defined as:

$$L(K_{\mathcal{N}}) = \bigcup_{c \in C} L_c(K_{\mathcal{N}})$$

We illustrate the local deadlock in the following example.

Example 3. Consider the packet switching network with $N = M = \{1, 2, 3, 4\}$ and $C = \{c_1, c_2, c_3, c_4, c_5\}$ shown on the right. The routing function $rout(n, m) = c_5$ if $n = 3$ and $m = 2$, and c_n otherwise. None of the messages in channels c_1, c_2, c_3 and c_4 can make another step because the next hop is blocked. For instance, message 4 in c_2 has to reach node 4, but $rout(3, 4) = c_3$ is blocked by message 1.

Channel c_5, by definition of the routing function, is only used in case node 3 sends a message to node 2, $rout(3, 2) = c5$. Therefore, node 3 can still send such a message (which can be received by node 2 immediately afterwards). Thus, these two steps will always be possible, even if all of the other channels are deadlocked.



3.3 Weak deadlock

Local deadlock does not distinguish between sending a new message—which is always possible if the target channel is empty—, and processing or receiving a message. In this section, we introduce the notion of *weak deadlock*. A state is a weak deadlock if no *receive* or *process* step is possible in that state.

Before defining weak deadlock, we first observe that in the initial states of a Kripke structure representing a packet-switching network, trivially no *process* or *receive* step is possible.

Lemma 2. *Let $\mathcal{N} = (N, M, C, rout)$ be a packet switching network, and $K_{\mathcal{N}} = (S, I, \rightarrow, L)$ its semantics. Then $\forall s \in I \colon s \not\rightarrow_{pr}$*

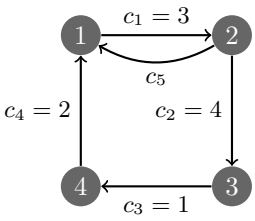
Proof. Initially all channels are empty. The result then follows immediately from the definitions of $\rightarrow_r$ and $\rightarrow_p$. □

Because of this observation, we explicitly exclude the initial states. The definition of weak deadlocks is as follows.

Definition 8. *Let $\mathcal{N} = (N, M, C, rout)$ be a packet switching network, and $K_{\mathcal{N}} = (S, I, \rightarrow, L)$ its semantics. The set of weak deadlocks is defined as:*

$$W(K_{\mathcal{N}}) = \{s \in S \setminus I \mid s \not\rightarrow_{pr}\}$$

Example 4. Consider the packet switching network with $N = M = \{1, 2, 3, 4\}$ and $C = \{c_1, c_2, c_3, c_4, c_5\}$ shown on the right. The routing function $rout(n, m) = c_5$ if $n = 2$ and $m = 1$, and c_n otherwise. None of the messages in c_1, c_2, c_3, c_4 can reach its destination because the next hop is blocked. For instance, message 2 in c_4 has to reach node 2, but $rout(1, 2) = c_1$ is blocked by message 3. Channel c_5, by definition of the routing function, is used only when node 2 sends a message to node 1, $rout(2, 1) = c_5$. This means c_5 can be filled with value 1, after which it can be received immediately by node 1. Thus, node 2, through channel c_5, will always be able to send messages to node 1, but in this particular configuration no *process* or *receive* step is possible. Hence, this situation is a weak deadlock.



4 Expressivity of different notions of deadlock

In this section we compare the different notions of deadlock introduced in the previous section. We first relate global deadlocks to local and weak deadlocks, and ultimately we investigate the relation between local and weak deadlocks.

4.1 Comparing global deadlocks to local and weak deadlocks

It is not hard to see that every global deadlock is both a local deadlock and a weak deadlock. Furthermore, neither local nor weak deadlocks necessarily constitute a global deadlock.

We first formalize this for local deadlocks in the following lemma.

Lemma 3. *Let $\mathcal{N} = (N, M, C, rout)$ be a packet switching network, and $K_{\mathcal{N}} = (S, I, \to, L)$ its semantics. Then we have:*

$$G(K_{\mathcal{N}}) \subseteq L(K_{\mathcal{N}})$$

Proof. From the Definitions 6 and 7 it follows immediately that for all $c \in C$, $G(K_{\mathcal{N}}) \subseteq L_c(K_{\mathcal{N}})$, hence $G(K_{\mathcal{N}}) \subseteq \bigcup_{c \in C} L_c(K_{\mathcal{N}}) = L(K_{\mathcal{N}})$. □

It is not generally the case that $L(K_{\mathcal{N}}) \subseteq G(K_{\mathcal{N}})$. This follows immediately from Example 3, which shows a packet-switching network with a local deadlock that is not a global deadlock. For weak deadlocks, similar results hold as formalized by the following lemma.

Lemma 4. *Let $\mathcal{N} = (N, M, C, rout)$ be a packet switching network, and $K_{\mathcal{N}} = (S, I, \to, L)$ its semantics. Then we have*

$$G(K_{\mathcal{N}}) \subseteq W(K_{\mathcal{N}})$$

Proof. Fix $s \in G(K_{\mathcal{N}})$. Note that $\nexists s' \in S \colon s \neq s' \wedge s \to s'$ according to Definition 6. Towards a contradiction, suppose $s \to_{pr} s'$ for some s'. It follows from the definition of $\to_{pr}$ that $s \neq s'$, and since $\to_{pr} \subseteq \to$, this is a contradiction. So, $s \not\to_{pr}$. Hence according to Definition 8, $s \in W(K_{\mathcal{N}})$. So, $G(K_{\mathcal{N}}) \subseteq W(K_{\mathcal{N}})$. □

Again, the converse does not necessarily hold. This follows immediately from Example 4, which shows a packet switching network with a weak deadlock that is not a global deadlock.

4.2 Comparing local deadlocks to weak deadlocks

Now that we have shown that local and weak deadlocks are not necessarily global deadlocks, the obvious question is how local and weak deadlocks are related. In particular, what we show in this section is that there is a local deadlock in a packet switching network if, and only if, there is a weak deadlock in the network.

Before we prove this main result, we first present several lemmata supporting the proof. First, in subsequent results we have to reason about the number of *process* and *receive* transitions that can be taken from a particular state, provided that no *send* transitions are taken. To this end, we first formalize the number of steps required to reach the destination for message m in channel c.

Definition 9. *Let $c \in C$ be a channel, and $m \in M_\perp$ the destination of the message carried by the channel.*

$$N(c,m) = \begin{cases} 0 & \text{if } m = \perp \\ 1 & \text{if } m = \mathit{target}(c) \\ 1 + N(\mathit{rout}(\mathit{target}(c), m), m) & \text{otherwise} \end{cases}$$

For correct packet switching networks, since the routing function is cycle free, we know that N is well-defined.

Lemma 5. *Let $\mathcal{N} = (N, M, C, \mathit{rout})$ be a correct packet switching network, then for all channels $c \in C$ and messages $m \in M_\perp$, there exists $l \in \mathbb{N}$ such that $N(c,m) = l$.*

Proof. Fix $c \in C$ and $m \in M_\perp$. Note that if $m = \perp$, then $N(c,m) = 0$, so the result follows immediately. Now, assume that $m \neq \perp$. Since the network is correct, there must be some $k \in \mathbb{N}$ such that $\mathit{next}_m^k(\mathit{target}(c)) = m$. Pick the smallest such k. It follows using an inductive argument that $N(c,m) = k+1$. □

We use this property to show that, from a given state in a packet switching network, if we only execute *process* or *receive* steps, the number of steps that can be taken is finite.

Lemma 6. *Let $\mathcal{N} = (N, M, C, \mathit{rout})$ be a correct packet switching network with $K_\mathcal{N} = (S, I, \rightarrow, L)$ its semantics, then*

$$\forall s \in S \colon \exists s' \in S \colon s \rightarrow_{pr}^* s' \wedge s' \not\rightarrow_{pr}$$

i.e., the number of possible steps of type $\rightarrow_{pr}$, from state s, is bounded.

Proof. First, we define the weight of a state $s \in S$ as $wt(s) = \sum_{c \in C} N(c, \pi_c(s))$. The weight captures the total number of steps required such that all the messages currently in the network can reach their destination. Note that N is well-defined according to Lemma 5, hence wt is well-defined.

We can show that for all $s, s' \in S$ that if $s \to_{pr} s'$, then $wt(s') < wt(s)$. So, the weight of the state decreases on every transition taken in $\to_{pr}$. It follows immediately from the definition of N that, if there is a $\to_{pr}$ transition from state s, then for some channel c and message $\pi_c(s)$, $N(c, \pi_c(s)) > 0$. Therefore, the number of $\to_{pr}$ steps is finite. Hence, for all states s in K, there is a state s' such that $s \to_{pr}^* s'$ such that $s' \not\to_{pr}$. □

At this point, we can finally formalize the correspondence between weak and local deadlocks. We first prove that a weak deadlock is also a local deadlock.

Theorem 1. *Let $\mathcal{N} = (N, M, C, rout)$ be a correct packet switching network and $K_{\mathcal{N}} = (S, I, \to, L)$ its semantics. Then we have*

$$W(K_{\mathcal{N}}) \subseteq L(K_{\mathcal{N}})$$

Proof. Fix arbitrary $s \in W(K_{\mathcal{N}})$. From the definition of $W(K_{\mathcal{N}})$, we observe that $s \notin I$ and $s' \not\to_{pr}$. Let $C' = \{c \in C \mid \pi_c(s) \neq \bot\}$ be the set of non-empty channels in state s. Since $s \notin I$, $C' \neq \emptyset$.

Observe that for all $c \in C'$, $\pi_c(s) \neq target(c)$, and $rout(target(c), \pi_c(s)) \in C'$ from the definitions of $\to_p$ and $\to_r$, since $s \not\to_{pr}$.

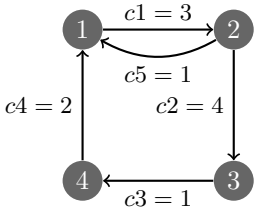
Next we show that for all $s' \in S$ such that $s \to^* s'$, for all $c \in C'$, $\pi_c(s') = \pi_c(s)$. We proceed by induction. If $s \to^0 s'$, then $s = s'$ and the result follows immediately. Now, assume there exists s'' such that $s \to^n s'' \to s'$. According to the induction hypothesis, for all $c \in C'$, $\pi_c(s'') = \pi_c(s)$. Fix arbitrary $c \in C'$. It follows from our observations that $\pi_c(s'') \neq target(c)$ and $rout(target(c), \pi_c(s'')) \in C'$, hence $rout(target(c), \pi_c(s'')) \neq \bot$. Therefore, the only possible transitions are a self-loop in which $s'' \to s'$ with $s'' = s'$, or a transition $\to_r$, in which case $\pi_c(s') = \pi_c(s'')$ according to the definition of $\to_r$.

Hence, it follows that $s \in L_c(K_{\mathcal{N}})$ for all $c \in C'$, and since c' is non-empty, $s \in L(K_{\mathcal{N}})$. So $W(K_{\mathcal{N}}) \subseteq L(K_{\mathcal{N}})$. □

The following example shows that generally not $L(K_{\mathcal{N}}) \subseteq W(K_{\mathcal{N}})$.

Example 5. Consider the packet switching network with $N = M = \{1, 2, 3, 4\}$ and $C = \{c_1, c_2, c_3, c_4, c_5\}$ shown on the right.[1] Note that this configuration is reachable by applying the following *send* steps in any order:

from node 1 to node 3, from node 2 to node 4, from node 2 to node 1, from node 3 to node 1, and from node 4 to node 2. This results in the configuration we show. This is a local deadlock for channels c_1 through c_4. Note, however, that node 1 can receive the message from channel c_5, so this is not a weak deadlock.



[1] This is the same network as in Example 4, but with $c_5 = 1$ instead of $c_5 = \bot$.

The essence of a local deadlock is a cycle of nodes, each of which is waiting for an outgoing channel to become free. This suggests that from a local deadlock, we can construct a weak deadlock by removing all messages that do not play a role in such a cycle. This is what we prove in the following theorem.

Theorem 2. *Let $\mathcal{N} = (N, M, C, rout)$ be a correct packet switching network and $K_{\mathcal{N}} = (S, I, \to, L)$ its semantics. Then we have*

$$L(K_{\mathcal{N}}) \neq \emptyset \implies W(K_{\mathcal{N}}) \neq \emptyset$$

Proof. Fix arbitrary $s \in L(K_{\mathcal{N}})$. We show that from s we can reach a state s' such that $s' \in W(K_{\mathcal{N}})$. Since $s \in L(K_{\mathcal{N}})$, there exists a channel $c \in C$ such that $s \in L_c(K_{\mathcal{N}})$, hence $\pi_c(s) \neq \bot$ and for all $s' \in S$ such that $s \to^* s'$, we have $\pi_c(s') = \pi_c(s)$. According to Lemma 6, there exists s' such that $s \to^*_{pr} s'$ and $s' \not\to_{pr}$. Pick such s'. Since $s \to^* s'$, we have $\pi_c(s') = \pi_c(s)$ and $\pi_c(s') \neq \bot$. Note that since $\pi_c(s') \neq \bot$, $s' \notin I$. Hence $s' \in W(K_{\mathcal{N}})$, so $W(K_{\mathcal{N}}) \neq \emptyset$. □

5 Proof of Concept Implementation

In this section we present a proof-of-concept implementation of the theory formalized in this paper and evaluate its results. We translate packet switching networks into SMV, and the notions of deadlock to CTL. We use nuXmv [2,3] to find deadlocks in the models or show their absence.

In the rest of this section, fix packet switching network $\mathcal{N} = (N, M, C, rout)$, with Kripke structure $K_{\mathcal{N}} = (S, I, \to, L)$. For channels $c \in C$ and nodes $m \in M$, $nextC_m(c) = rout(target(c), m)$ denotes the next channel for message m when it is currently in c.

5.1 An SMV model for packet switching networks

We sketch the translation of packet switching network $\mathcal{N}$ to the SMV format used by nuXmv. The SMV model consists of the following parts:

- DECLARATIONS: $c_i : 1 \dots |N|$. That is, the model has a variable c_i for every channel $c_i \in C$, whose value is in the range $0 \dots |N|$; $c_i = 0$ encodes $c_i = \bot$.
- INITIALIZATION: $\bigwedge_{i=1}^{|C|} c_i = 0$. That is, initially all channels are empty.
- TRANSITION RELATION: The transition relation is the disjunction over all send, process and receive transitions that are specified as follows. For each $c_i \in C$ such that $source(c_i) \in M$ (i.e. its source is terminal), and message $m \neq source(c_i)$ that c_i can insert into the network, we have a SEND transition:

$$\begin{array}{ll} \texttt{case } c_i = 0 & : \texttt{next}(c_i) = m \wedge \bigwedge_{j \neq i} \texttt{next}(c_j) = c_j; \\ \texttt{TRUE} & : \bigwedge_{i=1}^{|C|} \texttt{next}(c_i) = c_i; \qquad\qquad \texttt{esac} \end{array}$$

For all channels $c_i, c_j \in C$ and messages m, such that $nextC_m(c_i) = c_j$, we have the following PROCESS transition:

$$\begin{array}{l}\texttt{case}\\ \quad c_i = m \wedge c_j = 0 : \texttt{next}(c_i) = 0 \wedge \texttt{next}(c_j) = m \wedge \bigwedge_{k \notin \{i,j\}} \texttt{next}(c_k) = c_k;\\ \quad \texttt{TRUE} \qquad\qquad : \bigwedge_{i=1}^{|C|} \texttt{next}(c_i) = c_i;\\ \texttt{esac}\end{array}$$

For all channels $c_i \in C$ and messages m such that $target(c_i) = m$, we have the following RECEIVE transition:

$$\begin{array}{l}\texttt{case}\ c_i = m : \texttt{next}(c_i) = 0 \wedge \bigwedge_{j \neq i} \texttt{next}(c_j) = c_j;\\ \qquad \texttt{TRUE} \quad : \bigwedge_{i=1}^{|C|} \texttt{next}(c_i) = c_i; \qquad\qquad\qquad\qquad \texttt{esac}\end{array}$$

In this encoding `next` returns the value of its argument in the next state.

5.2 Deadlock formulas in CTL

To find deadlocks using nuXmv, we translate the properties to CTL. For the sake of readability, we give the CTL formulas as defined for the Kripke structures. These formulas are easily translated into the explicit syntax of nuXmv.

Definition 10. *The CTL formula for global deadlock is the following:*

$$\mathsf{EF}(\neg(\bigvee_{c \in C} v_c = \bot \vee \bigvee_{c \in C} \bigvee_{m \in M} (target(c) \neq m \wedge v_c = m \wedge v_{nextC_m(c)} = \bot) \vee \bigvee_{c \in C} \bigvee_{m \in M} (target(c) = m \wedge v_c = m))).$$

This formula expresses that a state can be reached, in which non of the conditions required to take a transition holds. The disjuncts are the conditions for send, process and receive transitions, respectively.

Definition 11. *Local deadlock is defined in CTL as follows:*

$$\bigvee_{c \in C} \bigvee_{m \in M} (\mathsf{EF}(\mathsf{AG}(v_c = m))).$$

This formula checks whether, a state can be reached in which, for some channel c and message m, c contains m, and m can never be removed from c.

Definition 12. *Weak deadlock is defined in CTL as follows:*

$$\mathsf{EF}(\bigvee_{c \in C} (v_c \neq \bot) \wedge \neg(\bigvee_{c \in C} \bigvee_{m \in M} (target(c) \neq m \wedge v_c = m \wedge v_{nextC_m(c)} = \bot) \vee \bigvee_{c \in C} \bigvee_{m \in M} (target(c) = m \wedge v_c = m))).$$

This formula expresses a non-initial state can be reached in which no *process* or *receive* transition is enabled. The conditions are the same as in Definition 10.

5.3 Experiments

We evaluate our proof-of-concept implementation on a network that consists of 17 nodes and 27 channels. The network is shown in Figure 2a. Nodes are numbered consecutively from 1 to 17. Double-ended arrows represent pairs of channels; one channel per direction. The routing function used is the shortest path, which is unique for every pair of nodes n and n'. We vary the set of terminals in this network, and determine for each of the notions of deadlock whether a deadlock exists. A timeout is set at 2 hours and 30 minutes.

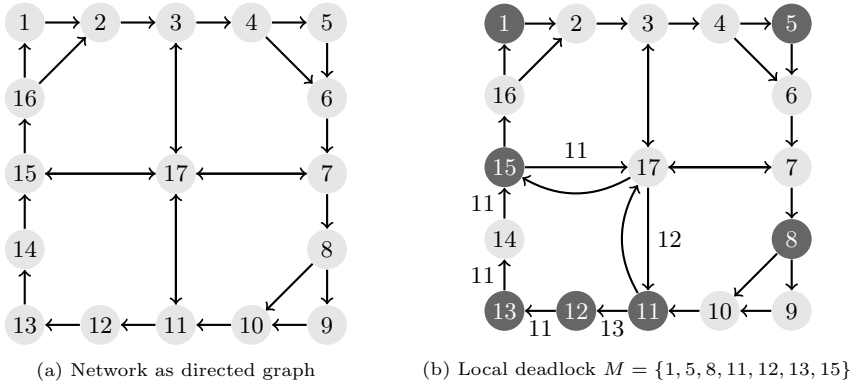
The experiments were done using nuXmv 1.1.1, on a system running Windows 10 Home, 64 bit Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz and 8GB of RAM.

Results Table 1 lists the results. For each set of terminals and notion of deadlock, we report whether a deadlock is found in column 'dl' ('d' means a deadlock was found) and the execution time (s) in column 'time'; 'n/a' indicates a timeout.

For the given network, finding global deadlocks is often fast, yet for larger instances it times out. Global deadlocks are often found faster than local deadlocks, which are generally found faster than weak deadlocks.

Discussion Table 1 shows there are sets of terminals M that are deadlock free for all types of deadlock; contain all different types of deadlock; and in which there is no global deadlock, but weak and local deadlocks are found. The results are consistent with theory: for all instances with a local deadlock, also a weak deadlock is reported (see Lemma 2). Also, there are examples with a local deadlock that do not contain a global deadlock. This is consistent with Lemma 3. Figure 2b shows the local deadlock found for $M = \{1, 5, 8, 11, 12, 13, 15\}$.

Note that execution times increase in particular for sets of terminals that require many channels in routing. This is consistent with our expectation: if more channels and more terminals are involved, the size of the reachable state space increases, which is also likely to increase the model checking time.



(a) Network as directed graph

(b) Local deadlock $M = \{1, 5, 8, 11, 12, 13, 15\}$

Fig. 2: 17 nodes and 27 channels packet switching network

Table 1: Deadlock results (column 'dl' contains 'd' iff a deadlock was found) and execution times (s, 'n/a' indicates timeout) using nuXmv for the network in Fig. 2a.

Set of terminals M	**Global**		**Local**		**Weak**	
	dl	**time**	**dl**	**time**	**dl**	**time**
{2,4,6}	d	0.74	d	0.71	d	0.71
{1,8,10}	d	0.95	d	0.83	d	0.96
{5,12,14}	d	0.92	d	1.04	d	0.93
{5,11,14}		0.39		0.42		0.42
{11,13,15}		0.32		0.40		0.42
{1,5,9,13}		0.69		0.74		0.70
{1,3,5,15}		0.48		0.50		0.58
{3,7,11,15}		0.51		0.46		0.50
{1,2,3,4,5}		0.61		0.67		0.66
{11,12,13,15}		0.50	d	0.76	d	1.10
{1,5,9,13,17}		0.57		0.74		0.61
{2,4,6,10,12}	d	37.55	d	21.20	d	124.40
{3,7,11,15,17}		0.56		0.57		0.61
{2,4,7,10,12,15,17}		0.83		1.50		1.10
{1,5,8,11,12,13,15}		0.97	d	32.58	d	7204.98
{1,5,9,11,12,13,15}		0.82	d	62.50	d	6132.20
{1,3,5,7,9,11,13,15,17}		1.13		2.10		1.32
{2,3,4,7,10,11,12,15,17}		1.04		1.89		1.21
{2,4,6,10,12,14}		n/a		n/a		n/a
{6,8,10,12,14,16}		n/a		n/a		n/a
{2,4,6,8,10,12,14,16}		n/a		n/a		n/a

6 Conclusions

We formalized three notions of deadlock in packet switching networks: global, local and weak deadlock. We proved that a global deadlock is a weak deadlock, and a weak deadlock is a local deadlock. A local deadlock is not necessarily a weak deadlock. However, a network has a local deadlock if and only if it has a weak deadlock. Presence of a local or weak deadlock does not imply the existence of a global deadlock. We compared the three notions on a packet switching network using nuXmv.

Future work. In this paper we considered networks with deterministic routing functions. The work should be generalized to non-deterministic routing functions. Furthermore, scalability of the approach in the verification of (on-chip) interconnect networks should be evaluated.

References

1. Benini, L., Micheli, G.D.: Networks on chips: A new SoC paradigm. Computer **35**(1), 70–78 (Jan 2002). https://doi.org/10.1109/2.976921

2. Cavada, R., Cimatti, A., Dorigatti, M., Griggio, A., Mariotti, A., Micheli, A., Mover, S., Roveri, M., Tonetta, S.: The nuXmv Symbolic Model Checker. In: Biere, A., Bloem, R. (eds.) Computer Aided Verification. pp. 334–342. LNCS, Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08867-9_22
3. Cavada, R., Cimatti, A., Dorigatti, M., Griggio, A., Mariotti, A., Micheli, A., Mover, S., Roveri, M., Tonetta, S.: nuXmv 1.1.1 user manual. Tech. rep. (2016)
4. Chen, R.C.: Deadlock prevention in message switched networks. In: Proceedings of the 1974 Annual Conference - Volume 1. p. 306–310. ACM, New York, NY, USA (1974), https://doi.org/10.1145/800182.810417
5. Clarke, E.M., Emerson, E.A.: Design and synthesis of synchronization skeletons using branching time temporal logic. In: Kozen, D. (ed.) Logics of Programs. pp. 52–71. LNCS, Springer, Berlin, Heidelberg (1982). https://doi.org/10.1007/BFb0025774
6. Coffman, E.G., Elphick, M.J., Shoshani, A.: Deadlock problems in computer system. In: Händler, W., Spies, P.P. (eds.) Rechnerstrukturen Und Betriebsprogrammierung. pp. 311–325. LNCS, Springer, Berlin, Heidelberg (1974). https://doi.org/10.1007/3-540-06815-5_147
7. Dally, W.J., Towles, B.P.: Principles and Practices of Interconnection Networks. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2004)
8. Duato, J.: A necessary and sufficient condition for deadlock-free routing in cut-through and store-and-forward networks. IEEE Transactions on Parallel and Distributed Systems **7**(8), 841–854 (Aug 1996). https://doi.org/10.1109/71.532115
9. López, P.: Routing (Including Deadlock Avoidance). Springer US, Boston, MA (2011). https://doi.org/10.1007/978-0-387-09766-4_314
10. Merlin, P., Schweitzer, P.: Deadlock avoidance in store-and-forward networks - i: Store-and-forward deadlock. IEEE Transactions on Communications **28**(3), 345–354 (1980). https://doi.org/10.1109/TCOM.1980.1094666
11. Stramaglia, A., Keiren, J., Zantema, H.: Deadlocks in packet switching networks arXiv:2101.06015 [cs.NI] (2021), https://arxiv.org/abs/2101.06015
12. Stramaglia, A.: Deadlock in Packet Switching Networks. Master's thesis, Università degli Studi di Trieste, Dipartimento di Ingegneria e Architettura, Trieste, Italy (2020)
13. Toueg, S., Ullman, J.D.: Deadlock-Free Packet Switching Networks. SIAM Journal on Computing **10**(3), 594–611 (Aug 1981). https://doi.org/10.1137/0210044
14. Verbeek, F.: Formal Verification of On-Chip Communication Fabrics. Ph.D. thesis, Radboud Universiteit Nijmegen (2013), https://hdl.handle.net/2066/103932
15. Verbeek, F., Schmaltz, J.: Formal specification of networks-on-chips: Deadlock and evacuation. In: 2010 Design, Automation Test in Europe Conference Exhibition (DATE 2010). pp. 1701–1706 (Mar 2010). https://doi.org/10.1109/DATE.2010.5457089
16. Verbeek, F., Schmaltz, J.: Formal validation of deadlock prevention in networks-on-chips. In: Proceedings of the Eighth International Workshop on the ACL2 Theorem Prover and Its Applications. pp. 128–138. ACL2 '09, ACM, New York, NY, USA (May 2009). https://doi.org/10.1145/1637837.1637858
17. Wolfson, O.: A new characterization of distributed deadlock in databases. In: Ausiello, G., Atzeni, P. (eds.) ICDT '86. pp. 436–444. LNCS, Springer, Berlin, Heidelberg (1986). https://doi.org/10.1007/3-540-17187-8_52
18. Zöbel, D.: The Deadlock problem: A classifying bibliography. ACM SIGOPS Operating Systems Review **17**(4), 6–15 (Oct 1983). https://doi.org/10.1145/850752.850753