Brief Contributions

On the Complexity of Removing Z-Cycles from a Checkpoints and Communication Pattern

Luca Allulli, Roberto Baldoni,
Luigi Laura, and Sara Tucci Piergiovanni

Abstract—Communication-induced checkpointing protocols are mechanisms used to produce checkpoints and communication patterns which enjoy desirable properties, such as No-Z-Cycle (NZC). NZC guarantees that each checkpoint can be part of a global consistent checkpoint. It would be nice to define communication-induced checkpointing protocols that enforce NZC, adding a minimum number of checkpoints to remove all the Z-cycles from the distributed computation. In this paper, we prove that this is impossible by formulating the Minimum Z-Cycle Removal (MinZCR) problem and showing that there are no online competitive protocols for it. Moreover, we prove that the problem of enforcing NZC with an optimal number of checkpoints is difficult even if the whole input instance is known because its decision version is NP-complete. Finally, we also prove that MinZCR is difficult to approximate: it is APX-hard and this implies that no Polynomial Time Approximation Scheme exists for the problem.

Index Terms—Distributed computing, checkpointing, Z-cycles, progressive retry, online versus offline analysis, competitive analysis, NP-complete problem.

1 Introduction

A *local checkpoint* is a snapshot of a local state of a process, a *global checkpoint* is a set of local checkpoints, one from each process, and a *consistent global checkpoint* [5] is a global checkpoint in which no one thing *happens-before* another [12]. When processes independently take local checkpoints, there is a risk that no consistent global checkpoint can ever be formed. This is caused by the well-known *unbounded domino effect* [17]. Even if consistent global checkpoints can be formed, it is still possible that some local checkpoints may never be included in a consistent global checkpoint; such local checkpoints are called *useless* [14].

On-the-fly determination of consistent global checkpoints requires some coordination between processes when they take their local checkpoints [5]. This coordination, executed by *communication-induced checkpointing protocols* [6] (hereafter we also refer to such protocols as *online checkpointing protocols*), results in two kinds of local checkpoints, namely, *basic* and *forced* checkpoints. A *basic* checkpoint is a local checkpoint taken by a process on its own initiative. A *forced* checkpoint is a local checkpoint taken by a process due to the coordination. The execution of a distributed computation, extended with a checkpointing protocol, produces a *Checkpoints and Communication Pattern* (CCP) [6], [13], which consists of a set of local checkpoints plus a dependency relation on them. Checkpointing protocols have been designed in order that the CCPs produced by their executions will satisfy some desirable properties. The "minimal" property requires that the CCP has no useless checkpoint. At the conceptual level, this is equivalent to the absence of cycles in a dependency graph on local checkpoints. This is why this property is called *No Z-Cycle* (or NZC) [14]. If a CCP satisfies this property, then each checkpoint can be associated to a consistent global checkpoint. A stronger property, namely, Rollback-dependency Trackability (RDT), holds if each checkpoint can be associated on the fly to a consistent global checkpoint it belongs to [22].

There has been a lot of work first in finding online checkpointing protocols that guarantee the presence of such consistent global checkpoints with minimal effort (i.e., taking as few forced checkpoints as possible) [6] and second in comparing these protocols among each other. The comparison has been usually done through simulation (which shows the average behavior of an online checkpointing protocol with respect to the number of forced checkpoints taken to ensure some properties, for example, the NZC property) and through the analysis of the predicate associated to each checkpointing protocol [3], [16], [22]. In the latter case, each checkpointing protocol is associated to a checkpointing predicate which is evaluated by each process upon the receipt of a message. The evaluation takes into account information (more or less detailed) related to the causal past of that receipt [18]. If the evaluation of the predicate becomes true, a forced checkpoint is taken to avoid violating the desired property (either NZC or RDT) over the CCP [9]. Using such predicates, a protocol A is *stronger* than a protocol B iff, considering any CCP in the causal past of a receipt event, if A takes a forced checkpoint, then B takes it (the converse is not necessarily true). Tsai et al. in [20], [21] proved that the fact that a checkpointing protocol A is stronger than B does not imply that A takes fewer forced checkpoints than B in any possible run. This is due to the fact that, if a weaker protocol B takes a forced checkpoint when the stronger protocol does not not take it, this might luckily influence the future checkpoints and communication pattern produced by the weaker protocol. This means that predicate analysis cannot be a theoretical metric for measuring the efficiency of checkpointing protocols.

A natural way to address the issue of finding a theoretical measure to compare checkpointing protocols is to use competitive analysis [19]. In this paper, we show that, if we measure the efficiency of online checkpointing protocols against the optimal offline solution, all of the online checkpointing protocols perform arbitrarily badly. Furthermore, we show that the approach of using offline protocols (and their approximations) as subroutines of online protocols is impractical. More precisely, we focus on the weaker property, namely, NZC, and we prove that 1) it is impossible to enforce this property using an online strategy with a cost comparable to the optimum (i.e., such that the optimal number of forced checkpoints is a constant fraction of the number of checkpoints taken by the algorithm) and 2) it is computationally difficult to enforce NZC even in an offline setting, where an omniscient algorithm knows from the beginning the whole checkpoints and communication pattern. As a result, nowadays the only measure that can be used to compare checkpointing protocols is the one based on predicate analysis, which indeed turns out average checkpointing saving as confirmed in many simulation studies [2], [16], while the problem of defining a theoretical measure of the efficiency of online checkpointing protocols cannot be achieved through competitive analysis and is therefore still an open problem.

For reaching the mentioned results, we formulate the Z-Cycle Removal (ZCR) problem, which is as follows: *Let $\mathcal{C}$ be a checkpoints and communication pattern over a set of n processes $P = \{P_1, \ldots, P_n\}$ and let $k \in \mathbb{N}$. Is there a set F of* forced *checkpoints, with $|F| \leq k$, such that $\mathcal{C}'$, obtained by adding the forced checkpoints to $\mathcal{C}$, satisfies the NZC property?* We build a reduction from the Feedback Arc Set (FAS) problem [7] that we use to prove the hardness of ZCR and of its

• *The authors are with the Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza," Via Salaria, 113-00198 Roma, Italy. E-mail: {allulli, baldoni, laura, tucci}@dis.uniroma1.it.*

Manuscript received 5 May 2006; revised 6 Nov. 2006; accepted 2 Mar. 2007; published online 19 Mar. 2007.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-0173-0506.
Digital Object Identifier no. 10.1109/TC.2007.1044.

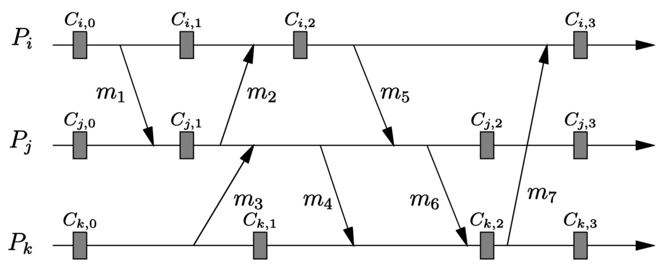

Fig. 1. A checkpoints and communication pattern.

minimization version, namely, Minimum Z-Cycle Removal (MinZCR). First, we show that ZCR itself is NP-complete. Second, we prove that MinZCR is difficult both from an offline and from an online point of view. From an offline point of view, MinZCR is APX-hard. When we switch to the online settings, we prove that there is no online competitive algorithm for the problem.

The rest of the paper is organized as follows: Section 2 presents the model of computation and the NZC property, together with a brief recall of necessary definitions. Section 3 contains basic definitions about competitive analysis and the approximation classes we are interested in. Section 4 states the ZCR problem and proves that it is NP-complete. Section 5 addresses the results concerning the Minimum Z-Cycle Removal problem, while conclusive remarks can be found in Section 6.

2 Checkpoints and Communication Patterns

A *distributed computation* consists of a finite set P of n processes $\{P_1, P_2, \ldots, P_n\}$ that communicate and synchronize only by exchanging messages. We assume that each ordered pair of processes is connected by an asynchronous directed logical channel whose transmission delays are unpredictable. Processes do not share a common memory and there is no bound on their relative speeds. A process executes internal, send, and deliver statements. The execution of an internal (respectively, send or deliver) statement is modeled by an internal (respectively, send or deliver) event. Each process P_i is *sequential*. In other words, P_i produces a (finite or infinite) *sequence* of events $e_{i,1} \cdots e_{i,s} \cdots$. Every process P_i has an initial local state denoted $\sigma_{i,0}$. The local state $\sigma_{i,s}$ $(s > 0)$ results from the execution of the sequence $e_{i,1} \ldots e_{i,s}$ applied to the initial state $\sigma_{i,0}$. Let H be the set of all the events produced by a distributed computation. This computation is represented by the partially ordered set $\widehat{H} = (H, \overset{hb}{\rightarrow})$, where $\overset{hb}{\rightarrow}$ denotes the well-known Lamport's *happened-before* relation [12].

2.1 Local and Global Checkpoints

A *local checkpoint* C is a recorded state (snapshot) of a process. Not every local state is necessarily recorded as a local checkpoint, so the set of local checkpoints is only a subset of the set of local states.

Definition 2.1. *A* checkpoints and communication pattern *(CCP in brief) is a pair $(\widehat{H}, \mathcal{C}_{\widehat{H}})$, where $\widehat{H}$ is a distributed computation and $\mathcal{C}_{\widehat{H}}$ is a set of local checkpoints defined on $\widehat{H}$.*

$C_{i,x}$ represents the xth local checkpoint of process P_i. The local checkpoint $C_{i,x}$ corresponds to some local state $\sigma_{i,s}$ with $x \leq s$. Fig. 1 shows an example of a checkpoints and communication pattern. (This figure uses the usual space-time diagram. Local checkpoints are indicated by shadowed rectangular boxes; the other local states are not explicitly indicated.) We assume that each process P_i takes an initial local checkpoint $C_{i,0}$ (corresponding to $\sigma_{i,0}$) and, after each event, a checkpoint will eventually be taken.

A message m sent by process P_i to process P_j is called an *orphan* with respect to the ordered pair of local checkpoints $(C_{i,x}, C_{j,y})$ if the delivery of m precedes $C_{j,y}$ in P_j but its sending event follows $C_{i,x}$ in P_i. An ordered pair of local checkpoints is *consistent* if there are no orphan messages with respect to this pair. For example, Fig. 1 shows that the pair $(C_{k,1}, C_{j,1})$ is consistent due to the absence of orphan messages, while the pair $(C_{i,2}, C_{j,2})$ is inconsistent (because of the orphan message m_5). A *global checkpoint* is a set of local checkpoints, one from each process. For example, $\{C_{i,1}, C_{j,1}, C_{k,1}\}$ and $\{C_{i,2}, C_{j,2}, C_{k,1}\}$ are two global checkpoints depicted in Fig. 1.

Definition 2.2. *A global checkpoint is* consistent *if all of its pairs of local checkpoints are consistent.*

For example, Fig. 1 shows that $\{C_{i,1}, C_{j,1}, C_{k,1}\}$ is a consistent global checkpoint and, due to the inconsistent pair $(C_{i,2}, C_{j,2})$, the global checkpoint $\{C_{i,2}, C_{j,2}, C_{k,1}\}$ is not consistent.

2.2 Checkpoint Dependencies and the NZC Property

Netzer and Xu [14] defined two kinds of dependencies in a checkpoints and communication pattern: *causal dependency* (C-dependency) and *zigzag dependency* (Z-dependency), denoted $\overset{C}{\rightarrow}$ and $\overset{Z}{\rightarrow}$, respectively.

Definition 2.3 (Causal Dependency). *$C_{i,x} \overset{C}{\rightarrow} C_{j,y}$ if the last event occurring on P_j before $C_{j,y}$ causally depends on the first event occurring on P_i after $C_{i,x}$.*

In other words, $C_{i,x} \overset{C}{\rightarrow} C_{j,y}$ if and only if $i = j$ and $x < y$ or there exists a sequence of messages $m_1, \ldots, m_k$ such that $\mathrm{send}(m_1)$ occurs on P_i after $C_{i,x}$, $\mathrm{send}(m_1) \overset{hb}{\rightarrow} \mathrm{deliver}(m_k)$, and $\mathrm{deliver}(m_k)$ occurs on P_j before $C_{j,y}$. We remark that $\overset{C}{\rightarrow}$ is a strict partial order on $\mathcal{C}_{\widehat{H}}$ and, thus, for any local checkpoint C, we have $\neg(C \overset{C}{\rightarrow} C)$. For example, in Fig. 1 we have $C_{i,0} \overset{C}{\rightarrow} C_{k,2}$ due to the sequence of messages m_1, m_4.

Definition 2.4 (Z-Path). *A sequence of local checkpoints $(C_{i,x}, C_{k_1,z_1}, C_{k_2,z_2} \ldots, C_{k_\alpha,z_\alpha}, C_{j,y})$, $\alpha \geq 0$, is a* Z-path *from $C_{i,x}$ to $C_{j,y}$ if $C_{i,x} \overset{C}{\rightarrow} C_{k_1,z_1} \wedge C_{k_1,z_1-1} \overset{C}{\rightarrow} C_{k_2,z_2} \wedge \ldots \wedge C_{k_\alpha,z_\alpha-1} \overset{C}{\rightarrow} C_{j,y}$.*

Definition 2.5 (Z-Dependency). *$C_{i,x} \overset{Z}{\rightarrow} C_{j,y}$ if there exists a Z-path $\mathcal{Z}$ from $C_{i,x}$ to $C_{j,y}$.*

We remark that $C_{i,x} \overset{C}{\rightarrow} C_{j,y} \Rightarrow C_{i,x} \overset{Z}{\rightarrow} C_{j,y}$. A Z-dependency which is not a C-dependency is called a "hidden" dependency, because it cannot be tracked online. For example, in Fig. 1, we have $C_{k,0} \overset{Z}{\rightarrow} C_{i,2}$ (because $C_{k,0} \overset{C}{\rightarrow} C_{j,2} \wedge C_{j,1} \overset{C}{\rightarrow} C_{i,2}$) but $\neg(C_{k,0} \overset{C}{\rightarrow} C_{i,2})$.

Definition 2.6 (Minimal Z-Path). *Z-path $\mathcal{Z}$ is said to be* minimal *if, by removing any checkpoint from $\mathcal{Z}$, we are left with a sequence that is not a Z-path.*

In other words, nonminimal Z-paths include some checkpoints that can be removed from the sequence without altering the Z-dependency. Z-path $(C_{i,0}, C_{j,1}, C_{k,2})$ of Fig. 1 is not minimal because $C_{j,1}$ can be removed from the sequence since $C_{i,0} \overset{C}{\rightarrow} C_{k,2}$. Every checkpoint in a minimal Z-path (except the first and the last one) captures a hidden dependency.

Definition 2.7 (Concatenation of Z-Paths). *If $\mathcal{Z}_1 = (C_{i,x}, C_{k_1,z_1}, \ldots, C_{k_\alpha,z_\alpha}, C_{j,y})$ and*

$$\mathcal{Z}_2 = (C_{j,y'}, C_{k_{\alpha+1},z_{\alpha+1}}, \ldots, C_{k_\beta,z_\beta}, C_{i',x'})$$

are Z-paths and $y' \geq y - 1$, then the sequence

$$(C_{i,x}, C_{k_1,z_1}, \dots, C_{k_\alpha,z_\alpha}, C_{j,y}, C_{k_{\alpha+1},z_{\alpha+1}}, \dots, C_{k_\beta,z_\beta}, C_{i',x'})$$

is a Z-path. We call it the concatenation $\mathcal{Z}_1\mathcal{Z}_2$ of $\mathcal{Z}_1$ and $\mathcal{Z}_2$.

Definition 2.8. *A* Z-cycle *is a Z-path from a local checkpoint $C_{i,x}$ to itself.*

We can observe that a Z-cycle from $C_{i,x}$ to itself necessarily involves at least another checkpoint $C_{j,y}$ with $i \neq j$, $C_{i,x} \xrightarrow{Z} C_{j,y}$, and $C_{j,y-1} \xrightarrow{Z} C_{i,x}$. For example, in Fig. 1, $C_{i,2} \xrightarrow{Z} C_{i,2}$ and $C_{k,2} \xrightarrow{Z} C_{k,2}$ are two Z-cycles: $(C_{i,2}, C_{j,2}, C_{i,2})$ and $(C_{k,2}, C_{i,3}, C_{k,2})$.

The following result by Netzer and Xu characterizes *useless* checkpoints, i.e., local checkpoints that cannot belong to a consistent global checkpoint, in terms of Z-cycles.

Theorem 2.9 [14]. *A local checkpoint $C_{i,x}$ is useless iff $C_{i,x} \xrightarrow{Z} C_{i,x}$.*

As a consequence, every local checkpoint of a CCP $(\widehat{H}, \mathcal{C}_{\widehat{H}})$ belongs to at least one consistent global checkpoint if and only if $(\widehat{H}, \mathcal{C}_{\widehat{H}})$ satisfies the following property:

Definition 2.10. (NZC Property). *$(\widehat{H}, \mathcal{C}_{\widehat{H}})$ satisfies the No Z-cycle (NZC) property if it has no useless checkpoints, i.e., $\forall C \in \mathcal{C}_{\widehat{H}} : \neg(C \xrightarrow{Z} C)$.*

Local checkpoints can be associated to consistent global checkpoints to which they belong either on-the-fly or during a rollback recovery phase [22]. To avoid the formation of Z-cycles, a communication-induced checkpointing protocol takes additional checkpoints. For example, in Fig. 1, the Z-cycle $C_{i,2} \xrightarrow{Z} C_{i,2}$ could be broken by adding a forced checkpoint immediately before the delivery of m_5 (see also [6]).

3 Competitive Analysis and Approximation Classes

To evaluate the performance of algorithms in the online setting, we refer to the widely used framework provided by the *competitive analysis* (CA). The CA was introduced by Sleator and Tarjan in [19], although it had been implicitly used in previous works since 1966 [8]. In the CA framework, the performance of online algorithms is evaluated against the optimal offline algorithm. A good introduction to online algorithms and CA can be found in the book by Borodin and El-Yaniv [4].

In CA, we say that an online algorithm A for a minimization problem P is ρ-competitive ($\rho \in \mathbb{R}^+$) if, for every input instance σ, $\mathrm{A}(\sigma) \leq \rho \cdot \mathrm{OPT}(\sigma)$; we denote by $\mathrm{A}(\sigma)$ and $\mathrm{OPT}(\sigma)$ the cost, on input σ, of the solution found by A and of the optimal solution, respectively. If there does not exist a $\rho \in \mathbb{R}^+$ for which A is ρ-competitive, we say that A is *not competitive*.

Here, we briefly recall some definitions related to approximation classes (see also [1]).

Class NPO. NPO is the class of all NP optimization problems.

ρ-approximate algorithm. Given a minimization problem $P \in$ NPO and an algorithm A for P, we say that A is a *ρ-approximate algorithm for P* if, for every input instance σ of P, we have $\mathrm{A}(\sigma) \leq \rho \cdot \mathrm{OPT}(\sigma)$; as before, $\mathrm{A}(\sigma)$ and $\mathrm{OPT}(\sigma)$ indicate, respectively, the cost of the solution found by A and the optimal cost on instance σ.

Class APX. APX is the class of all optimization problems $P \in$ NPO such that, for some $\rho > 1$, there exists a polynomial-time ρ-approximate algorithm for P.

Polynomial Time Approximation Scheme. Let $P \in$ NPO be an optimization problem. A *Polynomial-Time Approximation Scheme* for

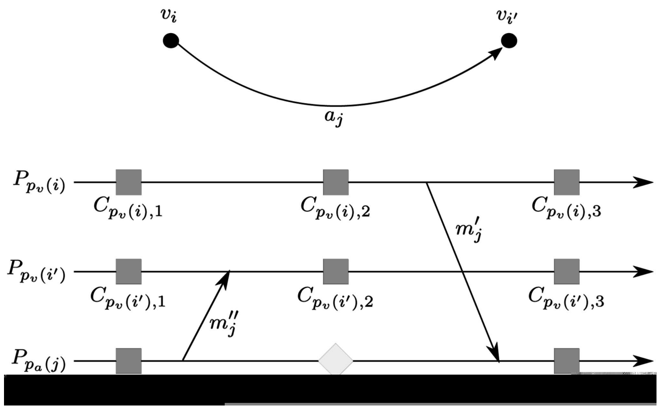

Fig. 2. Mapping an arc to a basic Z-path.

P is a parametric algorithm A_ρ, $\rho > 1$, such that, for every $\rho_0 > 1$, A_{ρ_0} is a polynomial-time ρ-approximate algorithm for P.

Class PTAS. PTAS is the class of NP optimization problems that admit a polynomial-time approximation scheme.

Notice that, if $\mathrm{P} \neq \mathrm{NP}$, it holds that $\mathrm{PTAS} \subset \mathrm{APX} \subset \mathrm{NPO}$.

4 The Z-Cycle Removal Problem

In this and in the following sections, we show several complexity results about the problem of removing all of the Z-cycles from a checkpoints and communication pattern. To this aim, we introduce a reduction from another problem, namely, the Feedback Arc Set problem. We begin by formally defining these problems.

Problem 4.1 (Z-Cycle Removal (ZCR)). ***Instance:*** *A pair $(\mathcal{C}, k)$, where $\mathcal{C} = (\widehat{H}, \mathcal{C}_{\widehat{H}})$ is a checkpoints and communication pattern over a set of n processes $P = \{P_1, \dots, P_n\}$, and $k \in \mathbb{N}$.* ***Question:*** *Is there a set F of* forced *checkpoints, with $|F| \leq k$, such that $\mathcal{C}' = (\widehat{H}, \mathcal{C}_{\widehat{H}} \cup F)$ satisfies the NZC property?*

Problem 4.2 (Feedback Arc Set (FAS)). ***Instance:*** *A pair (G, h), where $G = (V, A)$ is a directed graph, and $h \in \mathbb{N}$, $h \leq |A|$.* ***Question:*** *Is there a subset $A' \subseteq A$ with $|A'| \leq h$ such that A' contains at least one arc from every directed cycle in G?*

Let $\mathcal{I}_{\mathrm{FAS}}$ and $\mathcal{I}_{\mathrm{ZCR}}$ be, respectively, the sets of input instances of FAS and ZCR. We present a polynomial-time reduction $\mathcal{R} : \mathcal{I}_{\mathrm{FAS}} \rightarrow \mathcal{I}_{\mathrm{ZCR}}$ that maps instances of FAS to instances of ZCR. Let $(G, h) \in \mathcal{I}_{\mathrm{FAS}}$ be an input instance of FAS. $\mathcal{R}$ maps (G, h) to the instance $(\mathcal{R}_G(G), h) \in \mathcal{I}_{\mathrm{ZCR}}$ of ZCR, where $\mathcal{R}_G$ is a function mapping graphs to checkpoints and communications patterns that we describe in the following. We assume without loss of generality that $G = (V, A)$ with $V = \{v_1, \dots, v_n\}$ and $A = \{a_1, \dots, a_m\}$. $\mathcal{R}_G(G) = (\widehat{H}, \mathcal{C}_{\widehat{H}})$ is the CCP defined by the following properties:

- $\widehat{H}$ is a distributed computation of the set of processes $P = \{P_{p_v(1)}, \dots, P_{p_v(n)}, P_{p_a(1)}, \dots, P_{p_a(m)}\}$. P contains one *vertex* process $P_{p_v(i)}$ for each vertex $v_i \in V$ and one *arc* process $P_{p_a(j)}$ for each arc $a_j \in A$.
- Each vertex process $P_{p_v(i)}$, $1 \leq i \leq n$, includes three distinct checkpoints: $C_{p_v(i),1}$ at the beginning of the computation, $C_{p_v(i),2}$ in the middle, and $C_{p_v(i),3}$ at the end.
- Each arc process $P_{p_a(j)}$, $1 \leq j \leq m$, includes two checkpoints: $C_{p_a(j),1}$ at the beginning of the computation and $C_{p_a(j),2}$ at the end.
- For every arc $a_j = (v_i, v_{i'}) \in A$, $\widehat{H}$ includes two messages, m'_j and m''_j; there are no other messages in $\widehat{H}$ (see Fig. 2):

TABLE 1
Correspondence between Elements of G and of $\mathcal{R}_G(G)$

Element of G		**Element of $\mathcal{R}_G(G)$**
Arc $a_j = (v_i, v_{i'})$	$\longleftrightarrow$	basic Z-path $z(a_j) = (C_{p_v(i),2}, C_{p_a(j),2}, C_{p_v(i'),2})$
Path $\mathcal{P} = a_{j_1} a_{j_2} \dots a_{j_\alpha}$	$\longleftrightarrow$	Z-path $z(\mathcal{P}) = z(a_{j_1}) z(a_{j_2}) \dots z(a_{j_\alpha})$
Cycle $\mathcal{P}$	$\longleftrightarrow$	Z-cycle $z(\mathcal{P})$
Arc a_j in A' that covers cycle $\mathcal{P}$	$\longleftrightarrow$	Candidate checkpoint $C(a_j)$ in $C(A')$, that removes $z(\mathcal{P})$

- m'_j is sent[1] by $P_{p_v(i)}$ to $P_{p_a(j)}$; event $\text{send}(m'_j)$ follows checkpoint $C_{p_v(i),2}$ and event $\text{deliver}(m'_j)$ precedes checkpoint $C_{p_a(j),2}$.
- m''_j is sent by $P_{p_a(j)}$ to $P_{p_v(i')}$; event $\text{send}(m''_j)$ follows checkpoint $C_{p_a(j),1}$ and event $\text{deliver}(m''_j)$ precedes checkpoint $C_{p_v(i'),2}$.
- event $\text{send}(m''_j)$ precedes event $\text{deliver}(m'_j)$ on $P_{p_a(j)}$.

Observe that, for each arc $a_j = (v_i, v_{i'}) \in A$, we have created a Z-path $(C_{p_v(i),2}, C_{p_a(j),2}, C_{p_v(i'),2})$ from $C_{p_v(i),2}$ to $C_{p_v(i'),2}$. We indicate this Z-path with $z(a_j)$ and we call it a *basic Z-path* of $\mathcal{R}_G(G)$ (see Fig. 2). Basically, our construction maps paths and cycles of G to Z-paths and Z-cycles of $\mathcal{R}_G(G)$ such that selecting arcs in G is equivalent to adding forced checkpoints in $\mathcal{R}_G(G)$, as summarized in Table 1. Fig. 3 shows an example of reduced instance.

Consider any two consecutive arcs $a_j = (v_i, v_{i'})$ and $a_{j'} = (v_{i'}, v_{i''})$. $z(a_j)$ and $z(a_{j'})$ can be concatenated, yielding a Z-path from $C_{p_v(i),2}$ to $C_{p_v(i''),2}$. More generally, for every path $\mathcal{P} = a_{j_1} a_{j_2} \dots a_{j_r}$ of G with $a_{j_s} = (v_{i_s}, v_{i_{s+1}})$, the concatenation of all of the basic Z-paths $z(a_{j_1}) \cdots z(a_{j_r})$ is a Z-path from $C_{p_v(i_1),2}$ to $C_{p_v(i_{r+1}),2}$. We denote it with $z(\mathcal{P})$ and observe that $z(\mathcal{P})$ is a Z-cycle if and only if $i_1 = i_{r+1}$, that is, if and only if $\mathcal{P}$ is a cycle. The purpose of the next two technical lemmas is to formally state the converse: Every Z-cycle of $\mathcal{R}_G(G)$ is the concatenation of basic Z-paths corresponding to the arcs of a cycle $\mathcal{P}$ of G.

Lemma 4.3. *Let $P_{p_v(i)}$ and $P_{p_v(i')}$ be vertex processes such that $C_{p_v(i),2} \xrightarrow{Z} C_{p_v(i'),2}$. Every minimal Z-path $\mathcal{Z}$ from $C_{p_v(i),2}$ to $C_{p_v(i'),2}$ has the form $\mathcal{Z} = (C_{p_v(i),2}, C_{p_a(j_1),2}, C_{p_a(j_2),2}, \dots, C_{p_a(j_\alpha),2}, C_{p_v(i'),2})$, where $\mathcal{P} = a_{j_1} a_{j_2} \dots a_{j_\alpha}$ is a path of G from v_i to $v_{i'}$.*

Proof. Let $\mathcal{Z} = (C_{p_v(i),2}, C_{k_1,z_1}, \dots, C_{k_\alpha,z_\alpha}, C_{p_v(i'),2})$ be a minimal Z-path from $C_{p_v(i),2}$ to $C_{p_v(i'),2}$. When a vertex process sends a message m'_j to an arc process $P_{p_a(j)}$, $P_{p_a(j)}$ delivers m'_j, takes checkpoint $C_{p_a(j),2}$ and terminates. Since a vertex process only sends messages to arc processes, checkpoint $C_{p_v(i),2}$ must be followed in $\mathcal{Z}$ by a checkpoint $C_{p_a(j_1),2}$, for some $a_{j_1} = (v_i, v) \in A$, $v \in V$. Now, consider any checkpoint C_{k_x,z_x} of $\mathcal{Z}$: We prove that, if $C_{k_x,z_x} = C_{p_a(j_x),2}$, then $C_{k_{x+1},z_{x+1}}$ is either $C_{p_v(i'),2}$ or $C_{p_a(j_{x+1}),2}$, for some arc $a_{j_{x+1}} \in A$ such that $a_{j_x} = (u, v)$ and $a_{j_{x+1}} = (v, w)$, $u, v, w \in V$. $P_{p_a(j_x)}$ sends only a message m''_{j_x} to a vertex process, say $P_{p_v(i_x)}$; $\text{deliver}(m''_{j_x})$ precedes checkpoint $C_{p_v(i_x),2}$. If $i_x = i'$, the thesis holds because $C_{k_{x+1},z_{x+1}} = C_{p_v(i'),2}$. Otherwise, observe that $P_{p_v(i_x)}$ does not send any messages before $C_{p_v(i_x),2}$: Thus, there cannot be hidden dependencies involving $C_{p_v(i_x),2}$, which means that the checkpoint does not take part of the minimal Z-path $\mathcal{Z}$ (see Definition 2.6). $P_{p_v(i_x)}$ only sends messages after $C_{p_v(i_x),2}$ to arc processes: Then, there is an arc process $P_{p_a(j_{x+1})}$ such that $C_{p_a(j_x),1} \xrightarrow{C} C_{p_a(j_{x+1}),2}$ and, thus, $C_{k_{x+1},z_{x+1}} = C_{p_a(j_{x+1}),2}$. □

Corollary 4.4. *For every Z-cycle $\mathcal{Z}$ of $\mathcal{R}_G(G)$ from a checkpoint C to C, there exists a cycle $\mathcal{P}$ of G such that $\mathcal{Z} = z(\mathcal{P})$.*

Proof. It suffices to apply Lemma 4.3, observing that, for some $i \in \{1, \dots, n\}$, $C = C_{p_v(i),2}$. This is true because, by Definition 2.4, C must be preceded by a $\text{deliver}(\cdot)$ event, and followed by a $\text{send}(\cdot)$ event. □

Consider any event $e_{i,j}$ in $\widehat{H}$. We say that $e_{i,j}$ is a *candidate event* if, by taking a checkpoint immediately before $e_{i,j}$, at least one Z-path is removed from $\mathcal{R}_G(G)$. For example, in Fig. 3, event $\text{deliver}(m'_2)$ is a candidate event because, by taking the forced checkpoint $C(a_2)$, the Z-path $(C_{p_v(2),2}, C_{p_a(2),2}, C_{p_v(3),2})$ is removed.

Lemma 4.5. *For every arc $a_j \in A$, $\text{deliver}(m'_j)$ is a candidate event. There are no other candidate events in $\mathcal{R}_G(G)$.*

Proof. Adding to $\mathcal{C}_{\widehat{H}}$ a checkpoint immediately before $\text{deliver}(m'_j)$ removes at least the Z-path $z(a_j)$: Thus, $\text{deliver}(m'_j)$ is a candidate event. We show that there are no other candidate events. If we consider an arc process $P_{p_a(j)}$, the other event is $\text{send}(m''_j)$; before it there is already checkpoint $C_{p_a(j),1}$ and, thus, the event is not candidate. On the other hand, in any vertex

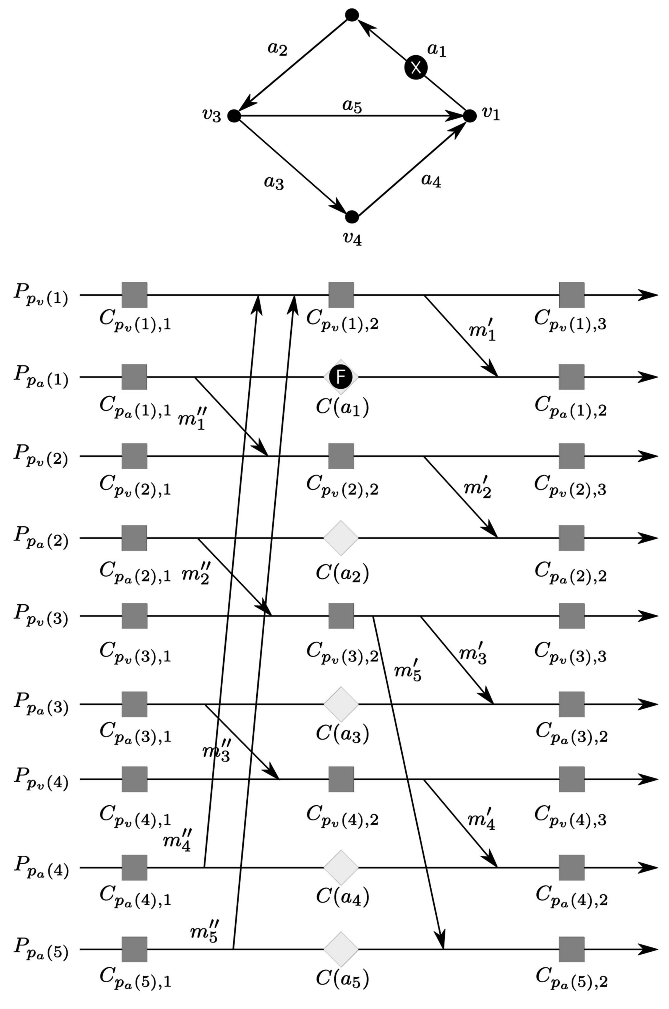

Fig. 3. A complete reduction from a graph G to $\mathcal{R}_G(G)$: solution $A' = \{a_1\}$ of the instance $(G, 1)$ on FAS corresponds to solution $F = \{C(a_1)\}$ of $(\mathcal{R}_G(G), 1)$ on ZCR.

1. A vertex process $P_{p_v(i)}$ may send multiple messages after $C_{p_v(i),2}$. The relative order of these $\text{send}(\cdot)$ events is irrelevant in our reduction as long as they all take place after the checkpoint. The same holds for all messages delivered before $C_{p_v(i),2}$: Their relative order is irrelevant.

process $P_{p_v(i)}$, there are no hidden dependencies because all messages are delivered before any message is sent: Thus, no checkpoint can remove a Z-path. □

Given an arc $a_j \in A$, we denote by $C(a_j)$ the checkpoint that, taken before the candidate event $\text{deliver}(m'_j)$, removes $z(a_j)$. We call $C(a_j)$ a *candidate checkpoint*. C is a bijective function between A and the set of candidate checkpoints of $\mathcal{R}_G(G)$. If $A' \subseteq A$ is a set of arcs, we denote by $C(A')$ the set of checkpoints $\cup_{a_j \in A'}\{C(a_j)\}$. We now prove that the introduction of candidate checkpoints does not create new Z-dependencies.

Lemma 4.6. *For every set of arcs $A' \subseteq A$, all Z-dependencies of $(\widehat{H}, \mathcal{C}_{\widehat{H}} \cup C(A'))$ are also Z-dependencies of $\mathcal{R}_G(G) = (\widehat{H}, \mathcal{C}_{\widehat{H}})$.*

Proof. We prove the thesis by induction on $|A'|$. If $|A'| = \emptyset$, the thesis trivially holds. So, assume $|A'| > 0$: Removing an arbitrary arc a_j from A', we are left with the CCP $(\widehat{H}, \mathcal{C}_{\widehat{H}} \cup C(A') - \{C(a_j)\})$ for which the inductive hypothesis holds. Now, we add $C(a_j)$ to this CCP: $C(a_j)$ does not create any new Z-dependency because, in order to be involved in a Z-dependency, by Definitions 2.3, 2.4, and 2.5, a checkpoint must either precede a $\text{send}(\cdot)$ event or follow a $\text{deliver}(\cdot)$ event, which is not the case of $C(a_j)$. □

The following lemma contains the main idea behind our reduction.

Lemma 4.7. *Given a graph $G = (V, A)$, let $\mathcal{R}_G(G) = (\widehat{H}, \mathcal{C}_{\widehat{H}})$. A set of arcs $A' \subseteq A$ contains at least one arc from every directed cycle in G if and only if the checkpoints and communication pattern $(\widehat{H}, \mathcal{C}_{\widehat{H}} \cup C(A'))$ contains no Z-cycles.*

Proof. Assume that $(\widehat{H}, \mathcal{C}_{\widehat{H}} \cup C(A'))$ contains no Z-cycles. Let $\mathcal{P}$ be a cycle of G: we claim that one arc of $\mathcal{P}$ is in A'. Since the Z-cycle $z(\mathcal{P})$ of $\mathcal{R}_G(G)$ no longer exists in $(\widehat{H}, \mathcal{C}_{\widehat{H}} \cup C(A'))$, at least one of the basic Z-dependencies $z(a_j)$ of $z(\mathcal{P})$ has been broken by checkpoint $C(a_j) \in C(A')$. Thus, the arc a_j, which belongs to $\mathcal{P}$, is in A'.

Conversely, suppose that every directed cycle of G contains an arc of A'. We claim that $(\widehat{H}, \mathcal{C}_{\widehat{H}} \cup C(A'))$ contains no Z-cycles. By Lemma 4.6, any Z-cycle of $(\widehat{H}, \mathcal{C}_{\widehat{H}} \cup C(A'))$ is also a Z-cycle of $\mathcal{R}_G(G)$. But, by Corollary 4.4, for every Z-cycle $\mathcal{Z}$ of $\mathcal{R}_G(G)$, we have $\mathcal{Z} = z(\mathcal{P})$, where $\mathcal{P}$ is a cycle of G. By hypothesis, A' contains an arc a_j of $\mathcal{P}$. Thus, $C(a_j)$ removes $\mathcal{Z}$, i.e., $\mathcal{Z}$ is not a Z-cycle of $(\widehat{H}, \mathcal{C}_{\widehat{H}} \cup C(A'))$. □

Theorem 4.8. *Z-Cycle Removal is NP-complete.*

Proof. ZCR $\in$ NP because a nondeterministic Turing Machine T can decide in polynomial time whether all of the Z-cycles of a CCP $(\widehat{H}, \mathcal{C}_{\widehat{H}})$ can be removed with $|F| \leq k$ forced checkpoints. Since any forced checkpoint of F shall be taken before the delivery of a message, T nondeterministically guesses whether or not to take a forced checkpoint before the delivery of every message. After that, T tests deterministically in polynomial time 1) if not more than k forced checkpoints have been taken, i.e., $|F| \leq k$, and 2) if all of the Z-cycles have been removed in $(\widehat{H}, \mathcal{C}_{\widehat{H}} \cup F)$.

In order to prove the completeness of ZCR in NP, we use the polynomial-time reduction $\mathcal{R}$. Given an instance $(G, h) \in \mathcal{I}_{\text{FAS}}$, we show that the answer of FAS is **YES** if and only if the answer of ZCR on $\mathcal{R}(G, h) = (\mathcal{R}_G(G), h) \in \mathcal{I}_{\text{ZCR}}$ is **YES**. If the answer of FAS on (G, h) is **YES**, there is an arc set $A' \subseteq A$ with $|A'| \leq h$ such that every directed cycle of G contains an arc of A'. Then, by Lemma 4.7, $C(A')$ is a set of not more than h candidate checkpoints that removes all Z-cycles from $\mathcal{R}_G(G)$: The answer to ZCR on $(\mathcal{R}_G(G), h)$ is **YES**. Conversely, if the answer to ZCR on $(\mathcal{R}_G(G), h)$ is **YES**, there exists a set F of candidate checkpoints that remove all Z-cycles, $|F| \leq h$. By Lemma 4.7, $A' = C^{-1}(F)$ is a set of not more than h arcs such that every directed cycle of G contains an arc of A': Then, the answer to FAS on (G, h) is **YES**. □

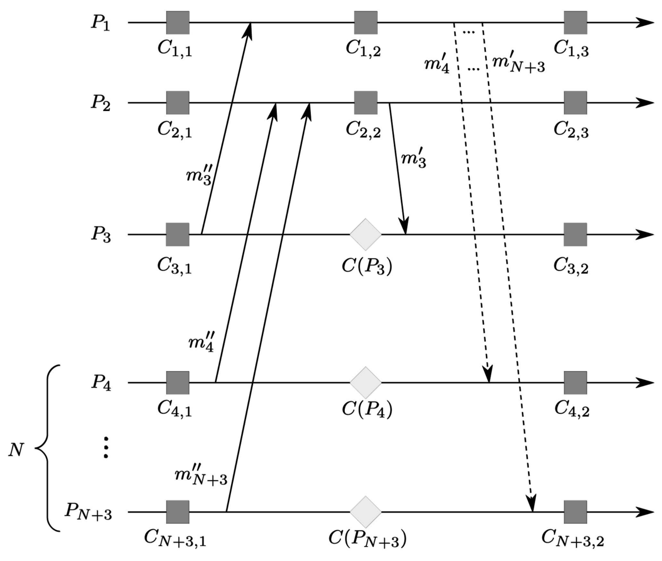

Fig. 4. Adversarial CCP to prove the noncompetitiveness of online protocols: Dashed messages are sent if and only if the checkpointing protocol takes the forced checkpoint $C(P_3)$.

5 The Minimum Z-Cycle Removal Problem

We now consider Minimum Z-Cycle Removal, the optimization version of Z-Cycle Removal.

Theorem 5.1. *There does not exist any competitive deterministic online protocol for the Minimum Z-Cycle Removal problem.*

Proof. Consider the CCP depicted in Fig. 4. There are $N + 3$ processes, $P_1, \dots, P_{N+3}$, where N is an integer that will be determined later. Process P_1 delivers a message m''_3, sent by P_3, and thereafter takes a forced checkpoint $C_{1,2}$. Process P_2 delivers N messages $m''_4, \dots, m''_{N+3}$, sent by processes $P_4, \dots, P_{N+3}$, respectively, and thereafter takes checkpoint $C_{2,2}$. Subsequently, process P_2 sends a message m'_3 to process P_3. The online checkpointing protocol has to decide whether to take a forced checkpoint $C(P_3)$ immediately before event $\text{deliver}(m'_3)$. If it does, no other message is sent: There is no Z-cycle in the CCP and, thus, the optimal cost is 0 (i.e., no forced checkpoint is necessary to enforce NZC), while the cost of the online protocol is at least 1, thus establishing its noncompetitiveness. If, on the other hand, $C(P_3)$ is not taken, then P_1 sends N messages $m'_4, \dots, m'_{N+3}$ to $P_4, \dots, P_{N+3}$. To break the N Z-cycles $(C_{1,2}, C_{i,2}, C_{3,2}, C_{1,2})$ with $i \in \{4, \dots, N+3\}$ and enforce NZC, N forced checkpoints must be taken, one before the delivery of each of these N messages (checkpoints $C(P_4), \dots, C(P_{N+3})$ in Fig. 4). Thus, the cost of the protocol is N, while the optimal cost is 1 because forced checkpoint $C(P_3)$ is enough to enforce NZC on this instance. The online protocol is hence not better than N-competitive, where N can be chosen arbitrarily large. □

Theorem 5.2. *Minimum Z-Cycle Removal is APX-hard.*

Proof. It is well known that Minimum Feedback Arc Set, the optimization version of Feedback Arc Set, is APX-hard [1], [10].[2] By Lemma 4.7, $\mathcal{R}_G$ is an approximation preserving reduction (see [1]) form Minimum Feedback Arc Set to Minimum Z-Cycle Removal: In fact, consider an instance G of Minimum Feedback Arc Set. The cost of an optimal, or approximate, solution of Minimum Z-Cycle Removal on $\mathcal{R}_G(G)$ (i.e., the number of forced checkpoints taken) is equal to the cost of the corresponding solution of Minimum Feedback Arc Set on G (i.e., the number of selected arcs). □

6 Conclusion

In this paper, we formulated the MinZCR problem over a checkpoints and communication pattern and showed that no online competitive protocol exists for this problem. This result, to our knowledge, had never been formally proven. It has a noteworthy consequence, that is, the impossibility of designing an optimal communication-induced checkpointing protocol that should ensure a minimum set of forced checkpoints. We also showed that, even reasoning offline, the problem of removing all of the Z-cycles from a given checkpoints and communication pattern using the minimum number of forced checkpoints is computationally difficult and difficult to approximate. Let us finally remark on the context and the importance of our results. Tsai et al. in [20], [21] showed that protocol hierarchies based on predicate analysis (as the ones presented in [3], [16], [22]) cannot be used as a theoretical metric for measuring the efficiency of communication-induced checkpointing protocols. Results contained in this paper show that a theoretical measure of efficiency cannot be found even using online competitive analysis. Therefore, predicate analysis is nowadays the only "practical" metric to compare online checkpointing protocols that shows an actual performance advantage (on average) with respect to checkpointing saving. As a consequence, the problem of finding a theoretical metric for measuring the efficiency of online checkpointing protocols is still open.

Acknowledgments

The authors wish to thank Viggo Kann for a discussion concerning the APX-hardness of Minimum Feedback Arc Set and the anonymous reviewers who helped them to structure and to improve the content of this paper. This work is partially supported by *Progetto FIRB Università di Roma La Sapienza-Technion RBIN047MH9*.

2. The APX-hardness of Minimum Feedback Arc Set can be easily established by observing that the Karp-reduction from Vertex Cover to Feedback Arc Set presented in [11] is an approximation preserving reduction (see [1]) and that Minimum Vertex Cover is APX-hard [15].

References

[1] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi, *Complexity and Approximation.* Springer, 1999.

[2] R. Baldoni, J.M. Helary, A. Mostéfaoui, and M. Raynal, "A Communication-Induced Checkpointing Protocol that Ensures Rollback-Dependency Trackability," *Proc. 27th Ann. Int'l Symp. Fault-Tolerant Computing (FTCS),* pp. 68-77, 1997.

[3] R. Baldoni, J.M. Helary, and M. Raynal, "Rollback-Dependency Trackability: A Minimal Characterization and Its Protocol," *Information and Computation,* vol. 165, no. 2, pp. 144-173, 2001.

[4] A. Borodin and R. El-Yaniv, *Online Computation and Competitive Analysis.* Cambridge Univ. Press, 1998.

[5] K.M. Chandy and L. Lamport, "Distributed Snapshots: Determining Global States of Distributed Systems," *ACM Trans. Computer Systems,* vol. 3, no. 1, pp. 63-75, 1985.

[6] E.N. Elnozahy, L. Alvisi, D.B. Johnson, and Y.M. Wang, "A Survey of Rollback-Recovery Protocols in Message-Passing Systems," *ACM Computing Surveys,* vol. 34, no. 3, pp. 375-408, 2002.

[7] M. Garey and D. Johnson, *Computers and Intractability. A Guide to the Theory of NP-Completeness.* Freeman, 1978.

[8] R.L. Graham, "Bounds for Certain Multiprocessor Anomalies," *Bell System Technical J.,* vol. 45, 1966.

[9] J.-M. Hélary, A. Mostefaoui, R.H.B. Netzer, and M. Raynal, "Communication-Based Prevention of Useless Checkpoints in Distributed Computation," *Distributed Computing,* vol. 13, no. 1, 1999.

[10] V. Kann, "On the Approximability of NP-Complete Optimization Problems," NADA report TRITA-NA-9206, PhD thesis, Dept. of Numerical Analysis and Computer Science, Royal Inst. of Technology, Stockholm, 1992.

[11] R. Karp, "Reducibility among Combinatorial Problems," *Proc. Symp. Complexity of Computer Computations,* pp. 85-103, 1972.

[12] L. Lamport, "Time, Clocks and the Ordering of Events in a Distributed System," *Comm. ACM,* vol. 21, no. 7, pp. 558-565, 1978.

[13] D. Manivannan and M. Singhal, "Quasi-Synchronous Checkpointing: Models, Characterization, and Classification," *IEEE Trans. Parallel and Distributed Systems,* vol. 10, no. 7, pp. 703-713, July 1999.

[14] R.H.B. Netzer and J. Xu, "Necessary and Sufficient Conditions for Consistent Global Snapshots," *IEEE Trans. Parallel and Distributed Systems,* vol. 6, no. 2, pp. 165-169, Feb. 1995.

[15] C.H. Papadimitriou and M. Yannakakis, "Optimization, Approximation, and Complexity Classes," *J. Computer and System Sciences,* vol. 43, no. 3, pp. 425-440, 1991.

[16] F. Quaglia, R. Baldoni, and B. Ciciani, "On the No-Z-Cycle Property in Distributed Executions," *J. Computer and System Sciences,* vol. 61, no. 3, pp. 400-427, 2000.

[17] B. Randell, "System Structure for Software Fault-Tolerance," *IEEE Trans. Software Eng.,* vol. 1, no. 2, pp. 220-232, 1975.

[18] R. Schwarz and F. Mattern, "Detecting Causal Relationships in Distributed Computations: In Search of the Holy Grail," *Distributed Computing,* vol. 7, no. 3, pp. 149-174, 1994.

[19] D.D. Sleator and R.E. Tarjan, "Amortized Efficiency of List Update and Paging Rules," *Comm. ACM,* vol. 28, no. 2, pp. 202-208, 1985.

[20] J. Tsai, S.Y. Kuo, and Y.M. Wang, "Theoretical Analysis for Communication-Induced Checkpointing Protocols with Rollback-Dependency Trackability," *IEEE Trans. Parallel and Distributed Systems,* vol. 9, no. 10, pp. 963-971, Oct. 1998.

[21] J. Tsai, Y.-M. Wang, and S.-Y. Kuo, "Evaluations of Domino-Free Communication-Induced Checkpointing Protocols," *Information Processing Letters,* vol. 69, pp. 31-37, 1999.

[22] Y.-M. Wang, "Consistent Global Checkpoints that Contain a Given Set of Local Checkpoints," *IEEE Trans. Computers,* vol. 46, no. 4, pp. 456-468, Apr. 1997.