JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 21, 239-257 (2005)

More Properties of Communication-Induced Checkpointing Protocols with Rollback-Dependency Trackability

JICHIANG TSAI*, SY-YEN KUO** AND YI-MIN WANG+
*Department of Electrical Engineering
National Chung Hsing University
Taichung, 407 Taiwan
E-mail: jctsai@ee.nchu.edu.tw
**Department of Electrical Engineering
National Taiwan University
Taipei, 106 Taiwan
E-mail: sykuo@cc.ee.ntu.edu.tw
+Microsoft Research, Microsoft Corporation
Redmond, Washington, U.S.A.
E-mail: ymwang@microsoft.com

Rollback-Dependency Trackability (RDT) is a property stating that all rollback dependencies between local checkpoints are on-line trackable using a transitive dependency vector. In this paper, we introduce some properties of communication-induced checkpointing protocols possessing the RDT property. First, we demonstrate that wherever an RDT protocol detects a PCM-path in the checkpoint and communication pattern associated with a distributed computation, it can also detect an EPSCM-path there. Moreover, if this detected PCM-path is non-visibly doubled, its corresponding EPSCM-path is also non-visibly doubled. Next, we go on to prove that if an RDT protocol breaks all EPSCM-cycles and non-visibly doubled EPSCM-paths, it breaks all visibly doubled EPSCM-paths as well. From these results, we find that some RDT protocols actually have the same behavior for all possible patterns. Furthermore, we also construct patterns to show that a few RDT protocols are incomparable in terms of the number of forced checkpoints. Last but not least, we discuss a simulation study to verify our previous theoretical results.

Keywords: distributed systems, fault tolerance, rollback-dependency trackability, communication-induced checkpointing protocols, rollback-recovery

1. INTRODUCTION

A distributed computation consists of a finite set of processes that communicate and synchronize with each other only by exchanging messages through a network. The set of messages and the set of local checkpoints form the *checkpoint and communication pattern* associated with the distributed computation. In a pattern, a message is called an *orphan* with regard to an ordered pair of local checkpoints if the receiving event of such a message happens before the latter checkpoint in the pair but its sending event occurs after the former one. An ordered pair of local checkpoints is said to be *consistent* if there is

Received April 5, 2004; revised August 3, 2004; accepted August 23, 2004.
Communicated by Chu-Sing Yang.

no orphan message with respect to this pair. A *global checkpoint* [1] is a set of local checkpoints, one from each process. A global checkpoint is consistent if and only if all the pairs of its constituent checkpoints are consistent [2]. All consistent global checkpoints in a distributed computation can be modeled as a partially ordered set, which possesses the lattice property [1]. The most recent consistent global checkpoint is called the *recovery line*. The problem of computing the recovery line is important for guaranteeing application consistency when a rollback-recovery results from a transient failure.

If local checkpoints are taken independently, there is a risk that no consistent global checkpoint can ever be formed from them. This is the well-known *domino effect* problem [3], in which unbounded, cascading rollback propagation can occur during the process of finding a consistent global checkpoint. Many protocols have been proposed that selectively employ local checkpoints in order to eliminate the possibility of the domino effect (see the survey paper in [4]). *Coordinated checkpointing* [2, 5] is one way to avoid the domino effect by synchronizing the checkpointing actions of all processes through explicit control messages. In contrast, *communication-induced checkpointing protocols* [6] achieve coordination by piggybacking control information on application messages. In addition to taking application-specific *basic* checkpoints, each process can also be asked by the protocol to take additional *forced* checkpoints, based on the piggybacked information as well as local control variables.

Communication-induced checkpointing protocols can also be used to achieve a stronger property called *Rollback-Dependency Trackability* (RDT), proposed by Wang [7]. In general, that two local checkpoints are not causally related is only a necessary, not a sufficient condition, for them to belong to the same consistent global checkpoint [1, 8]. They can have hidden, zigzag dependencies that make it impossible for them to belong to the same consistent global checkpoint. A checkpoint and communication pattern satisfies RDT if all such hidden dependencies are made on-line trackable by a simple transitive dependency vector. In addition to preventing the domino effect, RDT has two other noteworthy properties [7, 9]: (1) it ensures that any set of local checkpoints that are not pairwise causally related can be extended to form a consistent global checkpoint; (2) it offers efficient calculations of the minimum and the maximum consistent global checkpoints that contain a given set of local checkpoints. These properties allow RDT to have a wide range of applications, including software error recovery [10], deadlock recovery [11], nondeterministic computations [12], and so on.

Since the RDT property was first proposed, many studies have focused on this property [13-17]. In [14], Baldoni *et al*. investigated RDT at the message level and represented checkpoint dependencies by *Z-paths*. Moreover, some RDT characterizations that together comprise an important subset of Z-paths were addressed in this paper. The two most crucial characterizations, called *PCM-paths* and *EPSCM-paths*, are composed of a single message and a *prime path* with particular properties. Based on these two characterizations, the authors introduced a family of RDT communication-induced checkpointing protocols. This family contains not only existing protocols based on PCM-paths but also some new ones based on EPSCM-paths [14]. In our previous paper [17], we proposed several interesting properties of RDT protocols based on PCM-paths in this family, and we also theoretically compared these protocols in terms of the number of forced checkpoints. Because forcing additional checkpoints results in runtime overhead, it is desirable to force as few checkpoints as possible while still satisfying the RDT property.

Thus, throughout this paper, we compare *the performance of protocols* in terms of *the number of forced checkpoints*.

In this paper, we introduce more interesting properties of RDT protocols in the foregoing family, and we also give formal proofs to compare the performance of some protocols in it, which were not compared in [17]. First, we prove that wherever an RDT protocol detects a PCM-path in a checkpoint and communication pattern, it can also detect an EPSCM-path there. In addition, if this encountered PCM-path is *non-visibly doubled*, the associated EPSCM-path is non-visibly doubled as well. Then we show that a protocol that breaks all EPSCM-paths actually breaks all PCM-paths, and that a protocol that breaks all non-visibly doubled EPSCM-paths really breaks all non-visibly doubled PCM-paths. Next, we go on to show that if a protocol breaks all EPSCM-cycles and non-visibly doubled EPSCM-paths, it in fact breaks all visibly doubled EPSCM-paths, too. From these results, we conclude that some RDT protocols proposed in [14] have the same behavior for all possible patterns. Furthermore, we also build patterns to reveal the fact that no two RDT protocols in [14] are comparable, even though one protocol has a stronger checkpoint-inducing condition than the other. That is, for some patterns, the former outperforms the latter, while for other patterns, the latter is better than the former. Finally, we present some simulation experiments to verify our theoretical results in this context. The simulation was performed in the *point-to-point* environment.

This paper is structured as follows. Section 2 defines the computational model and introduces the definitions of two important RDT characterizations. In section 3, we investigate relationships between PCM-paths and EPSCM-paths, and a property of visibly doubling for EPSCM-paths. Some patterns showing that several RDT protocols are incomparable are given in section 4. In section 5, we present a simulation study and discuss the results of our experiments. Finally, we summarize our findings in section 6.

2. PRELIMINARIES

2.1 Checkpoint 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, reliable, directed logical channel with unpredictable but finite transmission delays; i.e., no message will be lost in the channel. Moreover, processes fail according to the fail-stop model.

A process can execute *internal*, *send*, and *receive* statements. An internal statement does not involve any communication. When P_i executes the statement "*send*(m) to P_j," it puts message m into the channel from P_i to P_j. When P_i executes the statement "*receive*(m)", it is blocked until at least one message directed to P_i has arrived, after which a message is delivered to P_i. Executions of internal, send, and receive statements are modeled by internal, sending, and receiving events, respectively.

The execution of each process produces a sequence of events, and all the events produced by a distributed computation can be modeled as a partially ordered set with the well-known Lamport's *happened-before* relation "$\underline{hb}$", defined as follows [18].

Definition 1 The relation "$\underline{hb}$" on the set of events satisfies the following conditions:

1. If a and b are events of the same process, and if a comes before b, then $a \ \underline{hb} \ b$.
2. If a is the event $send(m)$ and b is the event $receive(m)$, then $a \ \underline{hb} \ b$.
3. If $a \ \underline{hb} \ b$ and $b \ \underline{hb} \ c$, then $a \ \underline{hb} \ c$.

Given a distributed computation H, its associated checkpoint and communication pattern consists of the set of messages and the set of local checkpoints in H. Fig. 1 shows an example checkpoint and communication pattern. $C_{i,x}$ represents the x^{th} checkpoint of process P_i, where i is the *process id* and x the *checkpoint index*. The sequence of events occurring at P_i between $C_{i,x-1}$ and $C_{i,x}$ ($x > 0$) is called a *checkpoint interval* (or *interval* for short), denoted by $I_{i,x}$. Moreover, we assume that each process P_i starts to execute with an initial checkpoint $C_{i,0}$.

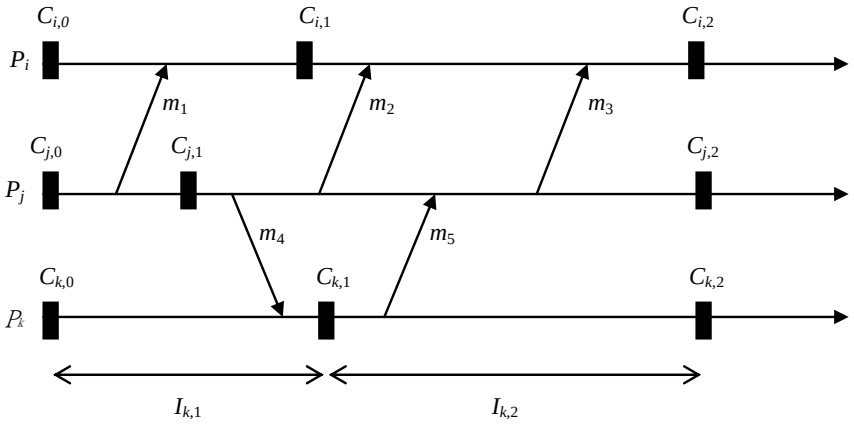


Fig. 1. A checkpoint and communication pattern.

2.2 Rollback-Dependency Trackability

A checkpoint and communication pattern satisfies Rollback-Dependency Trackability (RDT) if all rollback dependencies between local checkpoints are on-line trackable [7]. Specifically, if a checkpoint $C_{i,x}$ needs to be rolled back due to the rollback of checkpoint $C_{j,y}$, then $C_{i,x}$ must be able to detect this by using a transitive dependency vector. Equivalently, RDT can be stated based on the notion of *Z-paths* in [8] and the idea of *causal doubling* of Z-paths in [14].

First, a Z-path is defined as following [14].

Definition 2 A *Z-path* is a sequence of messages $[m_1, m_2, \ldots, m_q]$ ($q \geq 1$) such that, for each i, $1 \leq i \leq q - 1$, we have: $receive(m_i) \in I_{k,s} \wedge send(m_{i+1}) \in I_{k,t} \wedge s \leq t$.

That is, in a Z-path, every message is received in the same interval in which the succeeding message is sent or in an earlier interval. Furthermore, we say that a Z-path $[m_1,$

$m_2, \ldots, m_q$] extends from interval $I_{i,x}$ to interval $I_{j,y}$ if $send(m_1) \in I_{i,x}$ and $receive(m_q) \in I_{j,y}$. For example, in the pattern shown in Fig. 1, both message sequences [m_5, m_2] and [m_5, m_3] are Z-paths from $I_{k,2}$ to $I_{i,2}$. However, message sequence [m_5, m_1] is not a Z-path.

In addition, a Z-path is *causal* if the receiving event of each message (except for the last one) precedes the sending event of the next message in the sequence. A Z-path is *non-causal* if it is not causal. A Z-path with only one message is trivially causal. For the sake of neatness, a causal Z-path is also called a *causal path*. We present a definition of causally doubling in the following.

Definition 3 A Z-path from $I_{i,x}$ to $I_{j,y}$ is *causally doubled if* $i = j \wedge x \leq y$ or *if* there exists a causal path μ from $I_{i,x'}$ to $I_{j,y'}$, where $x \leq x'$ and $y' \leq y$ [14].

From the previous definition, every causal path is obviously causally doubled by itself. As an instance, Z-path [m_5, m_2] in the pattern shown in Fig. 1 is non-causal and is causally doubled by the causal path [m_5, m_3]. Next, the RDT property can be defined as follows.

Definition 4 A checkpoint and communication pattern satisfies RDT *if and only if* all Z-paths in it are causally doubled [14].

In the rest of this paper, we will use the following notation: the first (last) message of a Z-path ζ is denoted by $\zeta.first$ ($\zeta.last$). Given two Z-paths ζ and ζ', if their concatenation is also a Z-path, then we denote the concatenation as $\zeta \cdot \zeta'$.

2.3 RDT Characterizations

Given a checkpoint and communication pattern, it is not necessary to check that every non-causal Z-path is causally doubled to ensure that such a pattern satisfies RDT. Causally doubling a certain subset of non-causal Z-paths may suffice. Such a subset is called an *RDT characterization* in [14]. Before introducing some crucial RDT characterizations discussed in the present context, we will introduce the notion of *prime paths* first because every considered RDT characterization contains a prime path with a special property.

Definition 5 A causal path μ from $I_{i,x}$ to P_j is *prime if* every causal path ν from $I_{i,x'}$ to P_j with $x \leq x'$ satisfies that $receive(\mu.last) \underline{hb} receive(\nu.last)$ or $\mu.last$ and $\nu.last$ are the same message [14].

Intuitively, a prime path from $I_{i,x}$ to P_j is the first causal path that includes the interval $I_{i,x}$ in P_j's causal past. In Fig. 1, Z-path [m_5] is prime, but [m_3] is not prime. For simplicity, a prime causal path is called a prime path in the present context.

Next, an important RDT characterization, called a *PCM-path*, will be introduced in the following.

Definition 6 A PCM-path $\mu \cdot m$ is a Z-path that is the concatenation of a causal path μ and a single message m, where μ is prime and $send(m) \underline{hb} receive(\mu.last)$ [14].

For instance, Z-path $[m_5, m_2]$ in Fig. 1 is a PCM-path. The following theorem follows directly from the results given in [14].

Theorem 1 A checkpoint and communication pattern satisfies the RDT property *if and only if* all PCM-paths in it are causally doubled.

According to Theorem 1, in order to satisfy the RDT property, any PCM-path that is not causally doubled needs to be broken by a forced checkpoint. In particular, for the checkpointing decision of an on-line RDT protocol based only on the causal history, such information must be contained in the causal past of a process when it detects a PCM-path. This results in the notion of *visible doubling*[1]. Moreover, it has been shown that a visibly doubled PCM-path has the following property [14].

Theorem 2 A *PCM-path* $\mu \cdot m$ is *visibly doubled if and only if* it is causally doubled by a causal path μ' with *receive*($\mu'.last$) $\underline{hb}$ *send*($\mu.last$), as in the scenario shown in Fig. 2.

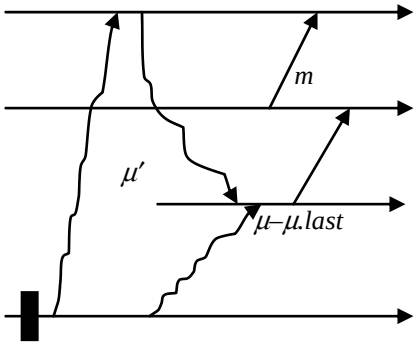


Fig. 2. Visibility of causal doubling.

Note that a causally doubled PCM-path is not necessarily visibly doubled, but that a non-causally doubled one must be non-visibly doubled. Based on the foregoing discussion and Theorem 1, we can deduce the following corollary for on-line protocols [14].

Corollary 1 A checkpoint and communication pattern produced by an on-line protocol satisfies the RDT property *if* all PCM-paths in it are visibly doubled.

In [14], the authors proposed another, more constrained RDT characterization, called an *EPSCM-path*. EPSCM-path are a subset of PCM-paths. First, we will formally define "elementary" and "simple" causal paths.

Definition 7 A causal path μ is *elementary if* its traversal sequence $P_i, P_{k_1}, \ldots, P_{k_\alpha}, P_j$, which is the sequence of processes traversed by μ, has no repetition [14].

[1] This notion was introduced in [13] for the first time and restated in [14].

That is, an elementary causal path only traverses a process *once*. For instance, in the pattern shown in Fig. 1, Z-path $[m_4, m_5, m_3]$ is not elementary because it traverses process P_j twice, while Z-path $[m_5, m_3]$ is elementary.

Definition 8 A causal path $\mu = [m_1, m_2, \ldots, m_q]$ is *simple if* the two events *receive*(m_i) and *send*(m_{i+1}) occur in the same interval, $\forall i$ $(1 \le i \le q - 1)$ [14].

Thus, a simple causal path does not include local checkpoints. As an example, Z-path $[m_4, m_5, m_3]$ in Fig. 1 is not simple since the local checkpoint $C_{k,1}$ is included. As for Z-path $[m_5, m_3]$, it is simple.

Next, an EPSCM-path is defined in the following.

Definition 9 An EPSCM-path is a PCM-path $\mu \cdot m$ such that μ is both elementary and simple [14].

Similarly, we have the following theorem and corollary for an EPSCM-path; consequently, it is also an RDT-characterization [14].

Theorem 3 A checkpoint and communication pattern satisfies the RDT property *if and only if* all EPSCM-paths in it are causally doubled.

Corollary 2 A checkpoint and communication pattern produced by an on-line protocol satisfies the RDT property *if* all EPSCM-paths in it are visibly doubled.

3. PROPERTIES OF EPSCM-PATHS

This section focuses on the fact that a few RDT protocols proposed in [14] actually have the same behavior for all possible checkpoint and communication patterns, according to the properties of EPSCM-paths given below.

3.1 Relationships between EPSCM-paths and PCM-paths

Here, we will demonstrate that although EPSCM-paths are a subset of PCM-paths, the existence of a PCM-path in fact implies the existence of an EPSCM-path. First, it is obvious that an elementary but not simple causal path μ can be written as $\mu_1 \cdot \mu_2 \ldots \mu_l$, where each component μ_i $(1 \le i \le l)$ is simple, divided by the checkpoints which μ includes. Now let us consider the following lemma.

Lemma 1 The last simple component of an elementary prime path is also prime.

Proof: As depicted in Fig. 3, there is an elementary prime path $\mu = \mu_1 \cdot \mu_2 \ldots \mu_l$, where μ_i is simple and absolutely elementary, for $1 \le i \le l$. To prove Lemma 1 by contradiction, suppose its last simple component μ_l, which starts after checkpoint C and reaches point y of process P_j, is not prime. Then, by Definition 5, there must exist a prime path ν which also starts after checkpoint C and reaches point y' of the same process such that point y'

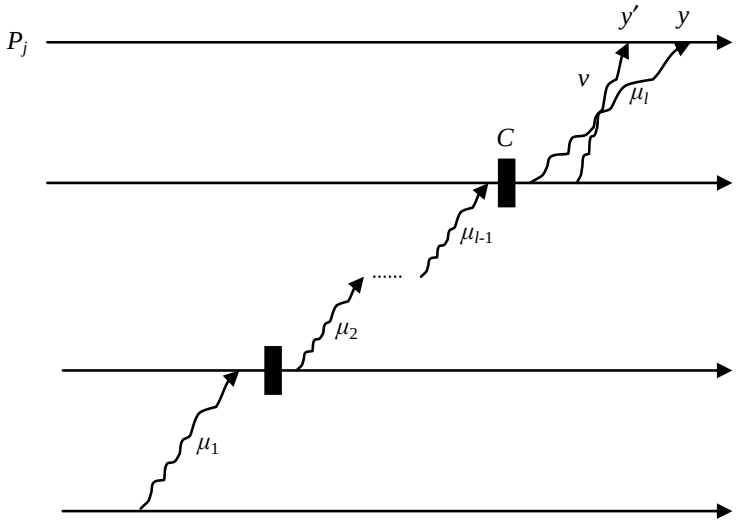


Fig. 3. The scenario of Lemma 1.

precedes point y. Obviously, $\mu_1 \cdot \mu_2 \ldots \mu_{l-1} \cdot \nu$ is a causal path, and its last message is received by P_j before the last message of μ is received. This violates that assumption μ is prime and, therefore, leads to a contradiction. ❑

Then, we have the following theorem.

Theorem 4 In a checkpoint and communication pattern, wherever an RDT protocol detects a PCM-path, it can also detect an EPSCM-path there.

Proof: First, if there is a causal path μ from $I_{i,x}$ to $I_{j,y}$, then we can trivially find an elementary causal path μ' from $I_{i,x'}$ to $I_{j,y'}$ with $x \le x'$ and $y' \le y$ by ignoring the causal cycles of μ such that the causal path composed of remainder messages only traverses a process once. Furthermore, if μ is also prime, then μ' has the same last message as μ; otherwise, the assumption that μ is prime would be violated. This means that μ' is also prime, according to Definition 5. Thus, applying Lemma 1, its last simple component, denoted as μ'_l, is prime and trivially has the same last message as μ. Hence, we know that wherever there is a PCM-path $\mu \cdot m$, there is also an EPSCM-path $\mu_l' \cdot m$. ❑

Furthermore, we can obtain the following corollary.

Corollary 3 In a checkpoint and communication pattern, wherever an RDT protocol detects a non-visibly doubled PCM-path, it can also detect a non-visibly doubled EPSCM-path there.

Proof: First, for a PCM-path $\mu \cdot m$, we can obviously find another PCM-path $\mu' \cdot m$, where μ' is elementary and extends from $I_{i,x'}$ to $I_{j,y'}$ with $x \le x'$ and $y' \le y$. According to Definition 3, if $\mu' \cdot m$ is causally doubled by a causal path ν, then we have that ν extends from $I_{i,x''}$ to $I_{j,y''}$ such that $x' \le x''$ and $y'' \le y'$. Hence, $\mu \cdot m$ is also causally doubled by ν

since $x \le x' \le x''$ and $y'' \le y' \le y$. Moreover, because μ and μ' have the same last message, if $\mu' \cdot m$ is visibly doubled by ν, then $\mu \cdot m$ is also visibly doubled by ν from Theorem 2.

Here, let $\mu' = \mu'_1 \cdot \mu'_2 \ldots \mu'_l$, where μ'_i is simple, for $1 \le i \le l$. For the PCM-path $\mu' \cdot m$, if its corresponding EPSCM-path $\mu'_l \cdot m$ is visibly doubled by a causal path ν', we trivially have that it is visibly doubled by the causal path $\mu'_1 \cdot \mu'_2 \ldots \mu'_{l-1} \cdot \nu'$. According to the foregoing discussion, we can see that wherever there is a non-visibly PCM-path, there is also a non-visibly EPSCM-path. ❑

As a result, protocol **No-EPSCM** [14], which breaks all EPSCM-paths, actually breaks all PCM-paths, not just a subset of PCM-paths. Thus, it in fact has the same behavior as protocol **FDAS** [7], which also breaks all PCM-paths. But **FDAS** requires less control information piggybacked on a message than **No-EPSCM** does because **FDAS** does not need to distinguish the "simple" condition. Moreover, it has been shown in [17] that protocols based on a stronger condition than **FDAS** outperform **FDAS**. Hence, we can also conclude that protocols based on a stronger condition than **No-EPSCM** do a better job than **No-EPSCM**. On the other hand, protocol **No-Non-Visibly-Doubled-EPSCM** (abbreviated as **NNVD-EPSCM**) [14], which breaks all non-visibly doubled EPSCM-paths, also breaks all non-visibly doubled PCM-paths, not just a subset of non-visibly doubled PCM-paths. Therefore, this protocol has exactly the same behavior as protocol **BHMR** [9], which breaks all non-visibly doubled PCM-paths.

3.2 The Property of Visibly Doubling for EPSCM-paths

First, we will introduce a few notations used in this section. Let V-paths denote a subset of Z-paths. If a V-path is from an interval $I_{i,x}$ to another interval $I_{i,x'}$ of the same process P_i, then we call this V-path a *V-cycle*. In the remainder of the paper, for the sake of clarity, only V-paths from one process to a different one are called V-paths. Those from one process to the same one are called V-cycles. Trivially, if $x' < x$, then a V-cycle cannot be causally doubled. Thus, we call this kind of V-cycle a *non-doubled* V-cycle. For example, the path $[m_5, m_4]$ in Fig. 1 is an EPSCM-cycle and is non-doubled. Another interesting result of this paper is a property of the relationship between protocol **No-EPSCM** and protocol **No-EPSCM-Cycle**, which breaks all EPSCM-cycles and non-visibly doubled EPSCM- paths [14]. Although the latter forces a checkpoint at a stronger condition than the former does, we can conclude that **No-EPSCM-Cycle** is, in fact, equivalent to **No-EPSCM** based on the following theorem. This result reveals that the extra piggybacked control information used by **No-EPSCM-Cycle** for detecting visibly doubled EPSCM-paths can be discarded.

Theorem 5 If an RDT protocol breaks all EPSCM-cycles and non-visibly doubled EPSCM-paths, then it also breaks all visibly doubled EPSCM-paths.

Proof: Suppose there exists an RDT protocol which breaks all EPSCM-cycles and non-visibly doubled EPSCM-paths but does not break some visibly doubled EPSCM-paths. Let $\mu \cdot m$ denote any such EPSCM-path. As shown in Fig. 4 (a), $\mu \cdot m$ must have a doubling causal path μ_1 and a causal path ν_1 that brings the knowledge about μ_1 to the sender of $\mu.last$ before $\mu.last$ is sent. Without loss of generality, we can assume that μ_1 is prime.

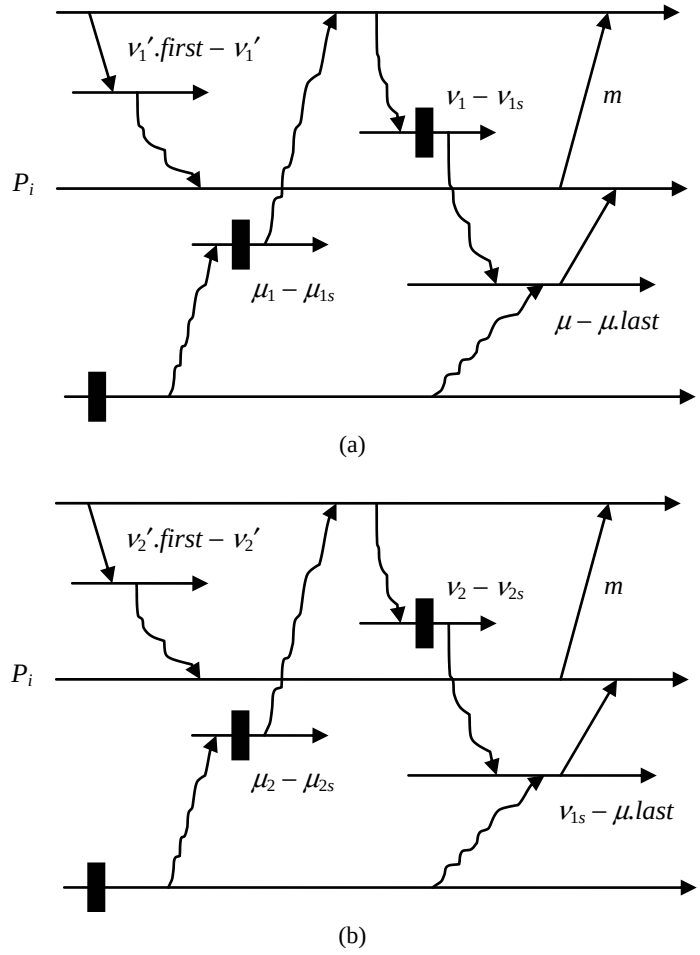


Fig. 4. The scenarios of Theorem 5.

We will show that the absence of a forced checkpoint to break $\mu \cdot m$ leads to a contradiction; thus, the protocol must break all visibly doubled EPSCM-paths. There are two cases to be considered.

First, assume that the causal path $\nu_1 \cdot \mu.last$ is prime. According to Lemma 1, we have that the last simple component of $\nu_1 \cdot \mu.last$, assumed to be $\nu_{1s} \cdot \mu.last$, is an elementary and simple prime path. Moreover, $\nu_{1s} \cdot \mu.last \cdot m$ cannot be an EPSCM-cycle; otherwise it would be an unbroken EPSCM-cycle. Thus, $\nu_{1s} \cdot \mu.last$ must be a visibly doubled EPSCM-path. Note that $\nu_{1s} \cdot \mu.last \cdot m$ cannot be μ; otherwise, the checkpoint between ν_{1s} and other part of ν_1 would be located between $send(\mu.first)$ and $send(\mu_1.first)$ such that μ could not be causally doubled by μ_1.

Next, assume that the causal path $\nu_1 \cdot \mu.last$ is not prime. Then, there must be another causal path ν'_1 that starts from the same process which ν_1 starts and reaches P_i be-

fore *receive*(μ.*last*). Furthermore, *send*(ν'_1.*first*) cannot happen after *receive*(μ_1.*last*) because that would violate the assumption that μ is prime. Without loss of generality, we pick ν'_1 as the one with *send*(ν'_1.*first*) closest to *receive*(μ_1.*last*). Likewise, since μ_1 is prime, its last simple component, assumed to be μ_{1s}, is also an elementary and simple prime path by Lemma 1. Clearly, the protocol cannot force a checkpoint between *send*(ν'_1.*first*) and *receive*(μ_1.*last*) because that would make $\nu_1 \cdot \mu$.*last* prime. Thus, we immediately have that $\mu_{1s} \cdot \nu'_1$.*first* must be a visibly doubled EPSCM-path.

So far, we have shown that, for the assumed protocol, an unbroken, visibly doubled EPSCM-path $\mu \cdot m$ implies the existence of another unbroken, visibly doubled EPSCM-path $\nu_{1s} \cdot \mu$.*last* $\cdot m$ (or $\mu_{1s} \cdot \nu'_1$.*first*). Note that *receive*(ν_{1s}.*last*) (or *receive*(μ_{1s}.*last*)) is in the causal past of *send*(μ.*last*); thus, $\nu_{1s} \cdot \mu$.*last* (or μ_{1s}) has at least one different message from μ. Similarly, the unbroken, visibly doubled EPSCM-path $\nu_{1s} \cdot \mu$.*last* $\cdot m$ also implies the existence of another unbroken, visibly doubled EPSCM-path $\nu_{2s} \cdot \mu$.*last* $\cdot m$ (or $\mu_{2s} \cdot \nu'_2$.*first*), as shown in Fig. 4 (b). Again, *receive*(ν_{2s}.*last*) (or *receive*(μ_{2s}.*last*)) is in the causal past of *send*(μ.*last*); thus $\nu_{2s} \cdot \mu$.*last* (or μ_{2s}) has at least one different message from $\nu_{1s} \cdot \mu$.*last*. Note that $\nu_{2s} \cdot \mu$.*last* cannot be μ, either; otherwise, the checkpoint between ν_{2s} and other part of ν_2 would also be located between *send*(μ.*first*) and *send*(μ_1.*first*) such that μ could not be causally doubled by μ_1. Thus, the path $\nu_{2s} \cdot \mu$.*last* also has at least one different message from μ. (And so does μ_{2s}.) Alternatively, the unbroken, visibly doubled EPSCM-path $\mu_{1s} \cdot \nu'_1$.*first* implies a similar scenario as well. Hence, with repeated application of the above argument, the existence of an unbroken $\mu \cdot m$ implies an infinite number of distinct unbroken visibly doubled EPSCM-paths in the causal past of μ.*last*. This contradicts the fact that the causal history must be finite. ❑

In [17], we proved that **FDAS** and protocol **No-PCM-Cycle**, which breaks all PCM-cycles and non-visibly doubled PCM-paths, have the same behavior. Since it has also been shown in the previous subsection that **FDAS** is equivalent to **No-EPSCM**, we can conclude that **No-PCM-Cycle** has the same behavior as **No-EPSCM-Cycle**.

4. INCOMPARABLE RDT PROTOCOLS

In this section, we will present performance comparisons of protocol **No-EPSCM-Path** (which breaks all EPSCM-paths and non-doubled EPSCM-cycles [14]) and two other protocols, **No-PCM-Path** and **NNVD-EPSCM**.

First, we will compare the performance of **No-EPSCM-Path** with that of protocol **No-PCM-Path**. The latter was also introduced in [14] and breaks all PCM-paths and any PCM-cycle $\mu \cdot m$ with *receive*(m) $\underline{hb}$ *send*(μ.*first*). Although **No-PCM-Path** clearly forces a checkpoint at a weaker condition than **No-EPSCM-Path**, we can construct the checkpoint and communication pattern depicted in Fig. 5, and apply both protocols to such a pattern to show that **No-EPSCM-Path** may not always outperform **No-PCM-Path**.

Figs. 5 (a) and (b) show the resulting patterns of **No-EPSCM-Path** and **No-PCM-Path**, respectively, where rectangular boxes represent basic checkpoints and diamond boxes represent forced checkpoints. The figure indicates that there exists a pattern for which **No-EPSCM-Path** must take two forced checkpoints to satisfy RDT, while **No-PCM-Path** requires only one. Therefore, we can conclude that these two protocols

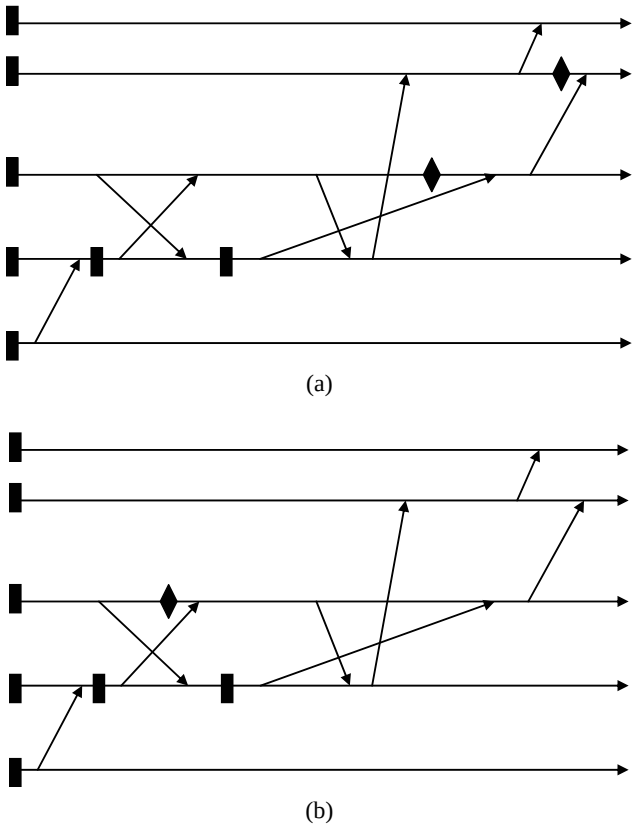
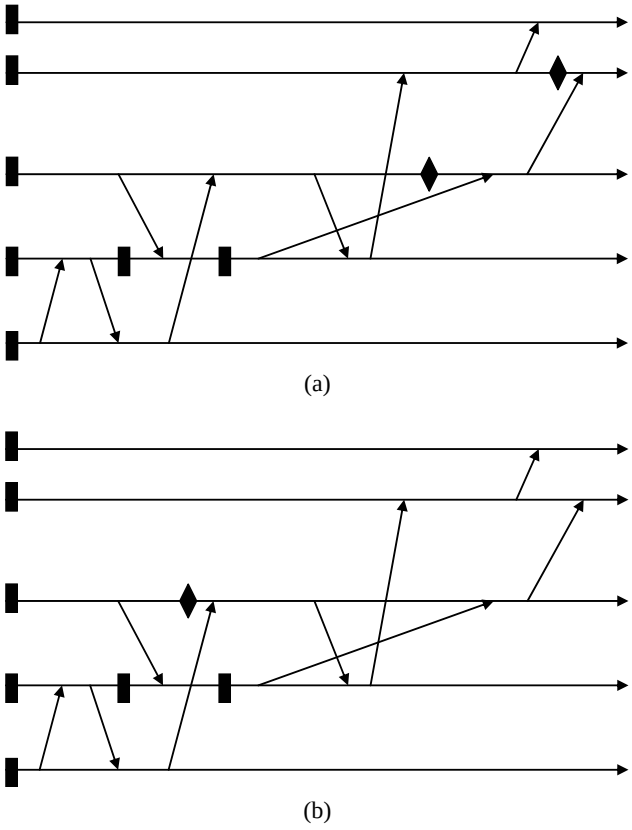


Fig. 5. (a) Applying **No-EPSCM-Path**; (b) applying **No-PCM-Path**.

are incomparable. But **No-EPSCM-Path** requires more information piggybacked on a message than **No-PCM-Path** does the former needs to distinguish whether a Z-path is simple and whether a Z-cycle is causally doubled or not.

Next, we will begin to compare **No-EPSCM-Path** with protocol **NNVD-EPSCM**. Obviously, **NNVD-EPSCM** is based on a stronger condition than **No-EPSCM-Path**, and since **NNVD-EPSCM** needs to decide if an EPSCM-path is visibly doubled or not, it requires more control information piggybacked on a message than **No-EPSCM-Path** does. We can also construct the pattern shown in Fig. 6 to demonstrate that **NNVD-EPSCM** may not always outperform **No-EPSCM-Path**. In Fig. 6 (a), it is shown that **NNVD-EPSCM** needs two checkpoints to satisfy RDT, whereas Fig. 6 (b) shows that **No-EPSCM-Path** needs only one. So these two protocols are also not comparable. Furthermore, it has been shown in [17] that **BHMR** and **No-PCM-Path** are incomparable. Because **NNVD-EPSCM** was demonstrated to be equivalent to **BHMR** in the foregoing section, we can also conclude that **NNVD-EPSCM** and **No-PCM-Path** are incomparable.



(a)

(b)

Fig. 6. (a) Applying **NNVD-EPSCM**; (b) applying **No-EPSCM-Path**.

5. A SIMULATION STUDY

We summarize our foregoing comparison results in the hierarchy graph depicted in Fig. 7, which shows how the RDT protocols discussed in the present context can be compared in terms of both the number of forced checkpoints and the amount of piggybacked information. Let #*f_ckpt*(**CP**) denote the number of forced checkpoints taken by the protocol **CP**. A solid arrow from a protocol **CP1** to another protocol **CP2** indicates that #*f_ckpt*(**CP1**) ≤ #*f_ckpt*(**CP2**), and a dotted arrow indicates that the amount of piggybacked information in **CP1** is less than that in **CP2**. A line with arrows at both ends means *equivalent*, and a line marked "*X*" means *incomparable*.

In the simulation study, each pair of adjacent processes were connected by a bidirectional communication channel for all environments. A process could execute *internal*, *send*, and *receive* operations. Unless explicitly mentioned otherwise, every process first checked if any message was waiting in the buffer to be received. If so, it managed this message; otherwise, it either executed a *send* operation with probability p_s or executed an *internal* operation with probability $1 - p_s$. The foregoing procedure proceeded continuously. The time needed to execute an operation in a process was exponentially distributed

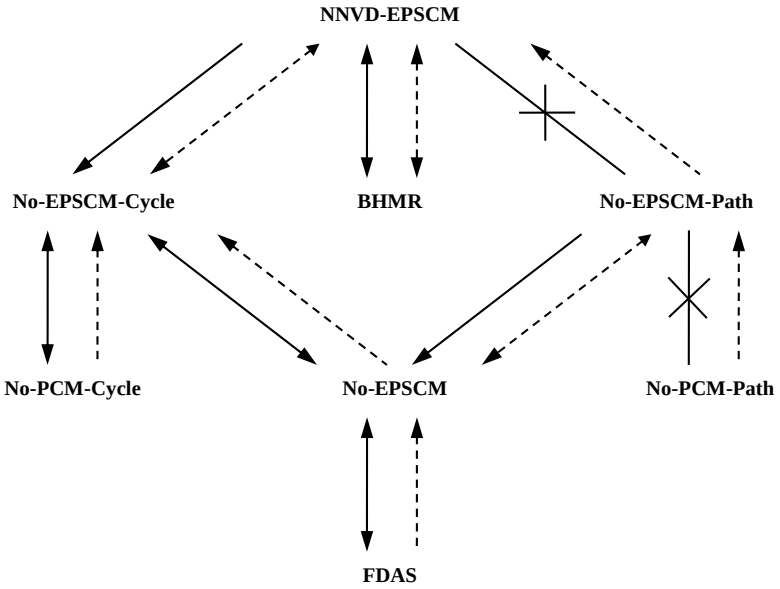


Fig. 7. Comparison of a family of RDT protocols.

with a mean value equal to 1 time unit, and the message propagation time was exponentially distributed with a mean value equal to 10 time units. Furthermore, basic checkpoints were taken periodically in a process.

The average of 10 measurements, running with different seeds, was taken for an experiment point. Every measurement was the execution time of all considered protocols under the same checkpoint and communication pattern. A simulation run contained 1000 message deliveries per process on average. Finally, we calculated the ratio between the number of forced checkpoints taken by a protocol and the number of messages of execution, for $4 \leq n \leq 16$, where n denotes the number of processes. This ratio could be regarded as the normalized value for the number of forced checkpoints taken by a process on average.

The simulation was performed in the *point-to-point* computational environment. For this environment, a complete network was assumed. The destination of each message was uniformly distributed. Furthermore, $p_s = 0.1$ and a basic checkpoint was taken every 50 operations for each process. As simulation ran, consistent with the theoretical results presented in section 3, the two protocols, **NNVD-EPSCM** and **BHMR**, really forced checkpoints at the same points, and the other two protocols, **No-EPSCM** and **FDAS**, had the same behavior. Fig. 8 depicts these results.

Furthermore, the results of another simulation experiment conducted in this environment are shown in Fig. 9. They demonstrate that protocols based on a stronger condition than **No-EPSCM** outperform **No-EPSCM**. We can also see that **No-EPSCM-Cycle** has exactly the same behavior as **No-EPSCM**. In addition, though any two protocols among **NNVD-EPSCM**, **No-EPSCM-Path**, and **No-PCM-Path** have been shown in section 4 to not be comparable, we can see that a protocol with a stronger condition is

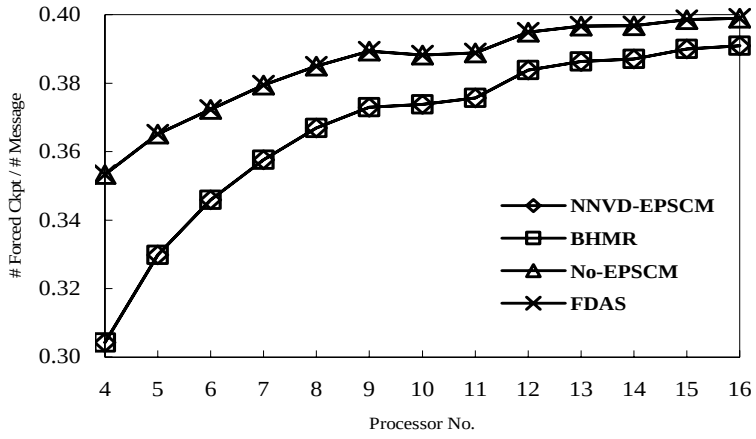


Fig. 8. Simulation results obtained in the point-to-point environment.

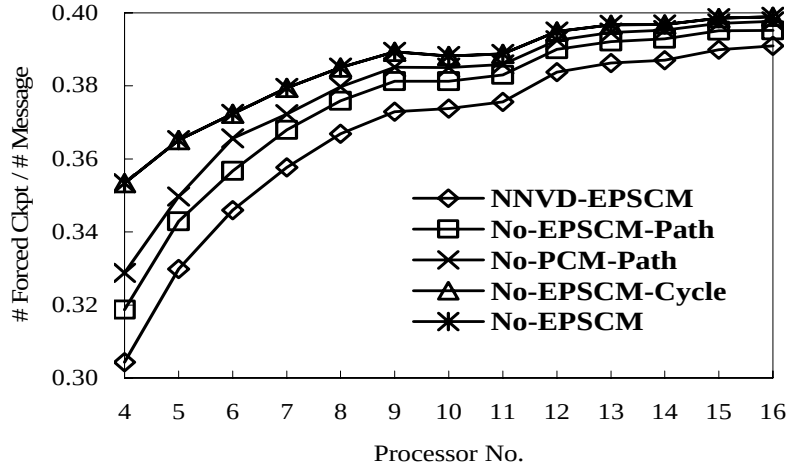


Fig. 9. Simulation results obtained in the point-to-point environment.

still better than a protocol with a weaker one. Thus, we know that those complex patterns constructed in section 4 seldom occur in a typical computational environment. Furthermore, the difference in the measured ratio between **NNVD-EPSCM** and **No-EPSCM-Path** reveals that some EPSCM- paths can be visibly doubled in this environment. This result also highlights the fact that **No-EPSCM-Cycle** is really equivalent to **No-EPSCM** since both have the same behavior although the former does not break a visibly doubled EPSCM-path.

In addition, we examined the influence of different basic checkpoint periods in the point-to-point environment. Here, we had 8 processes, and all of them had the same rates for taking a basic checkpoint. Moreover, the number of events needed to take a basic checkpoint in every process was varied from 20 to 140, in steps of 10 events. We com-

pared the performance of the four protocols, **NNVD-EPSCM**, **No-EPSCM-Path**, **No-PCM-Path** and **No-EPSCM**, and their results are depicted in Fig. 10. One can see that a protocol based on a stronger condition is still better than a protocol based on a weaker one. This means that the complex patterns built in section 4 are really hard to construct in such an environment even if the basic checkpoint period is prolonged. In addition, the longer the period for taking a basic checkpoint, the larger the number of forced checkpoints taken by every considered protocol. This is because more PCM-paths are formed in a longer basic checkpoint period.

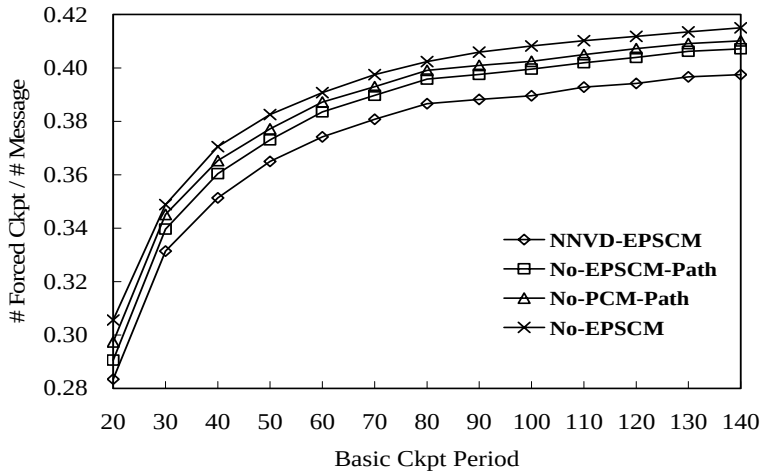


Fig. 10. Simulation results of the point-to-point environment.

6. CONCLUSIONS

Since the concept of Rollback-Dependency Trackability (RDT) was first introduced, designing more efficient RDT protocols has become an active research topic. In [14], the authors provided two crucial characterizations of the RDT property, *EPSCM-paths* and *PCM-paths*. They also derived a family of RDT protocols from these two characterizations that not only contain existing protocols based on PCM-paths but also some new ones based on EPSCM-paths. We compared the performance of protocols based on PCM-paths in [17]. In this study, we performed performance comparisons of protocols based on EPSCM-paths. We also addressed more interesting properties of RDT protocols in the previous family. From these results, we have found that some RDT protocols actually have the same behavior for all possible patterns. In addition, we have constructed patterns to show that any two among several RDT protocols are incomparable. Finally, we have carried out simulation experiments to verify the previously obtained theoretical results. Interestingly, the experimental results reveal that a protocol based on a stronger condition still requires a smaller number of checkpoints than a protocol based on a weaker one in a typical computational environment even though they have been shown to not be comparable.

ACKNOWLEDGMENTS

The authors wish to express their sincere thanks to the anonymous referees for their valuable comments. Moreover, Tsai's work was supported by the National Science Council, Taiwan, R.O.C., under Grant NSC 90-2213-E-002-113.

REFERENCES

1. Y. M. Wang, A. Lowry, and W. K. Fuchs, "Consistent global checkpoints based on direct dependency tracking," *Information Processing Letters*, Vol. 50, 1994, pp. 223-230.
2. K. M. Chandy and L. Lamport, "Distributed snapshots: determining global states of distributed systems," *ACM Transactions on Computing Systems*, Vol. 3, 1985, pp. 63-75.
3. B. Randell, "System structure for software fault-tolerant," *IEEE Transactions on Software Engineering*, Vol. 1, 1975, pp. 220-232.
4. E. N. Elnozahy, L. Alvisi, Y. M. Wang, and D. B. Johnson, "A survey of rollback-recovery protocols in message-passing systems," *ACM Computing Surveys*, Vol. 34, 2002, pp. 375-408.
5. R. Koo and S. Toueg, "Checkpointing and rollback-recovery for distributed systems," *IEEE Transactions on Software Engineering*, Vol. 13, 1987, pp. 23-31.
6. B. Janssens and W. K. Fuchs, "Experimental evaluation of multiprocessor cache-based error recovery," in *Proceedings of International Conference on Parallel Processing*, 1991, pp. 505-508.
7. Y. M. Wang, "Consistent global checkpoints that contain a given set of local checkpoints," *IEEE Transactions on Computers*, Vol. 46, 1997, pp. 456-468.
8. R. H. B. Netzer and J. Xu, "Necessary and sufficient conditions for consistent global snapshots," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 6, 1995, pp. 165-169.
9. R. Baldoni, J. M. Helary, A. Mostefaoui, and M. Raynal, "A communication-induced checkpointing protocol that ensures rollback-dependency trackability," in *Proceedings of IEEE Fault-Tolerant Computing Symposium*, 1997, pp. 68-77.
10. Y. M. Wang, "The maximum and minimum consistent global checkpoints and their applications," in *Proceedings of IEEE Symposium on Reliable Distributed Systems*, 1995, pp. 86-95.
11. Y. M. Wang, M. Merritt, and A. B. Romanovsky, "Guaranteed deadlock recovery: deadlock resolution with rollback propagation," in *Proceedings of Pacific Rim International Symposium on Fault-Tolerant Systems*, 1995, pp. 92-97.
12. E. Cohen, Y. M. Wang, and G. Suri, "When piecewise determinism is almost true," in *Proceedings of Pacific Rim International Symposium on Fault-Tolerant Systems*, 1995, pp. 66-71.
13. R. Baldoni, J. M. Helary, and M. Raynal, "Rollback-dependency trackability: visible characterizations," in *Proceedings of 18th ACM Symposium on Principles of Distributed Computing*, 1999, pp. 33-42.
14. R. Baldoni, J. M. Helary, and M. Raynal, "Rollback-dependency trackability: a

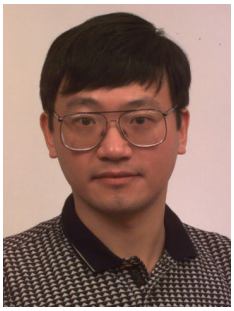
minimal characterization and its protocol," *Information and Computation*, Vol. 165, 2001, pp. 144-173.
15. R. Baldoni, J. M. Helary, and M. Raynal, "Impossibility of scalar clock-based communication-induced checkpointing protocols ensuring the RDT property," *Information Processing Letters*, Vol. 80, 2001, pp. 105-111.
16. D. Manivannan and M. Singhal, "Quasi-synchronous checkpointing: models, characterization, and classification," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 10, 1999, pp. 703-713.
17. J. Tsai, S. Y. Kuo, and Y. M. Wang, "Theoretical analysis for communication-induced checkpointing protocols with rollback-dependency trackability," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 9, 1998, pp. 963-971.
18. L. Lamport, "Time, clocks and the ordering of events in a distributed system," *Communications of the ACM*, Vol. 21, 1978, pp. 558-565.

Jichiang Tsai (蔡智強) received his B.S. degree in Electrical Engineering from National Taiwan University, Taipei, Taiwan, in 1991. Then he started his graduate study at the same university, and received the Ph.D. degree in Electrical Engineering in 1999. Dr. Tsai served as a postdoctoral research fellow in the Institute of Information Science, Academia Sinica, Taipei, Taiwan, from 1999 to 2001. Since 2002, he has been an Assistant Professor with the Department of Electrical Engineering, National Chung Hsing University, Taichung, Taiwan. His current research interests include fault tolerance, parallel and distributed systems, embedded systems, quantum computing, and computer networks.

Sy-Yen Kuo (郭斯彥) received the B.S. (1979) in Electrical Engineering from National Taiwan University, the M.S. (1982) in Electrical and Computer Engineering from the University of California at Santa Barbara, and the Ph.D. (1987) in Computer Science from the University of Illinois at Urbana-Champaign. Since 1991 he has been with National Taiwan University, where he is currently a Professor of the Department of Electrical Engineering, and served as the department head from 2001 to 2004. He spent his sabbatical years as a visiting researcher at AT&T Labs-Research, New Jersey from 1999 to 2000, and as a visiting professor at the Computer Science and Engineering Department of the Chinese University of Hong Kong. He was the Chairman of the Department of Computer Science and Information Engineering, National Dong Hwa University, Taiwan from 1995 to 1998, a faculty member in the Department of Electrical and Computer Engineering at the University of Arizona from 1988 to 1991, and an engineer at Fairchild Semiconductor and Silvar-Lisco, both in California, from 1982 to 1984. In 1989, he also worked as a sum-

mer faculty fellow at Jet Propulsion Laboratory of California Institute of Technology. His current research interests include software reliability engineering, mobile computing, dependable systems and networks, and optical WDM networks.

Professor Kuo is an IEEE Fellow. He has published more than 200 papers in journals and conferences. He received the distinguished research award (1997-2005) from the National Science Council, Taiwan. He was also a recipient of the Best Paper Award in the 1996 International Symposium on Software Reliability Engineering, the Best Paper Award in the simulation and test category at the 1986 IEEE/ACM Design Automation Conference (DAC), the National Science Foundation's Research Initiation Award in 1989, and the IEEE/ACM Design Automation Scholarship in 1990 and 1991.



Yi-Min Wang (王逸民) manages the Systems Management Research Group at Microsoft Research, Redmond. He received his Ph.D. in Electrical and Computer Engineering from University of Illinois at Urbana-Champaign in 1993, worked at AT&T Bell Labs from 1993 to 1997, and joined Microsoft in 1998. His research interests include systems and security management, fault tolerance, home networking, and distributed systems.