On Properties of RDT Communication-Induced Checkpointing Protocols

Jichiang Tsai

Abstract—*Rollback-Dependency Trackability* (RDT) is a property stating that all rollback dependencies between local checkpoints are online trackable by using a transitive dependency vector. The most crucial RDT characterizations introduced in the literature can be represented as certain types of RDT-PXCM-paths. Here, let the U-path and V-path be any two types of RDT-PXCM-paths. In this paper, we investigate several properties of communication-induced checkpointing protocols that ensure the RDT property. First, we prove that if an online RDT protocol encounters a U-path at a point of a checkpoint and communication pattern associated with a distributed computation, it also encounters a V-path there. Moreover, if this encountered U-path is invisibly doubled, the corresponding encountered V-path is invisibly doubled as well. Therefore, we can conclude that breaking all invisibly doubled U-paths is equivalent to breaking all invisibly doubled V-paths for an online RDT protocol. Next, we continue to demonstrate that a visibly doubled U-path must contain a doubled U-cycle in the causal past. These results can further deduce that some different checkpointing protocols actually have the same behavior for all possible patterns. Finally, we present a commendatory systematic technique for comparing the performance of online RDT protocols.

Index Terms—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 connected by a communication network. Processes communicate and synchronize only by exchanging messages through the network. A local checkpoint is a snapshot of the local state of a process, saved on nonvolatile storage to survive process failures. It can be reloaded into volatile memory in case of a failure to reduce the amount of lost work. When a process records such a local state, we say that this process takes a (local) checkpoint. The set of messages and the set of local checkpoints form the *checkpoint and communication pattern* associated with the distributed computation. A global checkpoint [1] is a set of local checkpoints, one from each process. A global checkpoint M is *consistent* if no message is sent after a checkpoint of M and received before another checkpoint of M [2]. The problem of computing consistent global checkpoints is important for ensuring application consistency in the presence of failures.

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 problem of the *domino effect* [3], in which unbounded, cascading rollback propagation can occur during the process of finding a consistent global checkpoint. Many protocols have been proposed to selectively take local checkpoints to eliminate the possibility of the domino effect (see the survey paper [4]). Coordinated checkpointing [2], [5] is one way of avoiding 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, two local checkpoints not being causally related is only a necessary but not sufficient condition for them to belong to the same consistent global checkpoint [1], [8]. They can have hidden, zigzag dependencies that make them impossible to belong to the same consistent global checkpoint. A checkpoint and communication pattern satisfies RDT if all such hidden dependencies are made online trackable by a simple transitive dependency vector. In addition to ensuring domino freedom, 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 and 2) it enjoys 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, deadlock recovery, mobile computing, distributed debugging, nondeterministic computations, etc. [7].

Since the RDT property was proposed in [7], many papers have been proposed to study this property [10], [11], [12], [13], [14], [15], [16]. In [13], [15], the authors introduced

• *The author is with the Department of Electrical Engineering, Rm. 821, National Chung Hsing University, Taichung, Taiwan 402, Republic of China. E-mail: jctsai@ee.nchu.edu.tw.*

Manuscript received 20 Aug. 2001; revised 21 Nov. 2002; accepted 26 Mar. 2003.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number 114808.

several *characterizations* of RDT in terms of checkpoint and communication patterns. An RDT characterization is a special checkpoint and communication pattern that has to be detected by a checkpointing protocol to see if such a pattern possesses a certain property in order to guarantee RDT. In [13], Baldoni et al. investigated RDT at the message level and represented checkpoint dependencies by *Z-paths*. Some RDT characterizations were addressed by the authors and are all important subsets of Z-paths. The most crucial three among these characterizations, called *PCM-paths*, *EPCM-paths*, and *EPSCM-paths*, are composed of a single message and a *prime path* with particular properties. Recently, Garcia and Buzato also found another characterization of RDT, called *PMM-paths*, in [15]. A PMM-path is composed of a single message and a prime path that is also a single message. Now, let X be a special property defined on prime paths and a *PXCM-path* be a Z-path constituted by a single message and a prime path with the property X. Obviously, these crucial characterizations presented in the literature can be denoted as certain types of PXCM-paths. Moreover, if one type of PXCM-path is also an RDT characterization, we denote it as an *RDT-PXCM-path* in the context. So, PCM-paths, EPCM-paths, EPSCM-paths, and PMM-paths are all types of RDT-PXCM-paths.

In this paper, we will further explore some more properties of online RDT communication-induced checkpointing protocols. Here, consider two properties U and V defined on Z-paths and Z-paths satisfying the two properties are called *U-paths* and *V-paths*, respectively. Now, let the U-path and V-path also be two types of RDT-PXCM-paths. First, although there have been many types of RDT-PXCM-paths proposed in the literature, we prove that if an online RDT protocol meets a U-path at a point of a checkpoint and communication pattern, it also meets a V-path there. In addition, if this encountered U-path is *invisibly doubled*, the encountered V-path is invisibly doubled also. According to the previous results, we can obtain the following converse result since there is no constraint on U-paths and V-paths other than being two types of RDT-PXCM-paths: If an online RDT protocol meets an invisibly doubled V-path at a point of a pattern, it also meets an invisibly doubled U-path there. Hence, we can conclude that, for an online protocol, breaking all invisibly doubled U-paths is equivalent to breaking all invisibly doubled V-paths. Next, it is shown that a visibly doubled U-path must contain a doubled U-cycle in the causal past. From the foregoing results, we can deduce that some different checkpointing protocols in fact have the same behavior for all possible checkpoint and communication patterns. Thus, for a certain protocol, we may find an equivalent but more efficient protocol with less control information piggybacked on each message to detect the checkpoint-inducing condition. Finally, we present a commendatory systematic technique for comparing the performance of online RDT checkpointing protocols.

This paper is structured as follows: Section 2 defines the computational model and introduces the definitions of some important RDT characterizations. Section 3 investigates some interesting properties of RDT-PXCM-paths and an important property on visibly doubling will also be demonstrated in the next section. Section 5 gives a systematic technique to compare the performance of existing RDT protocols and some related works are described in Section 6. Finally, we summarize the paper in Section 7.

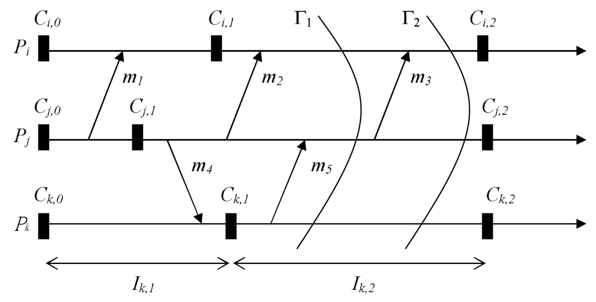

Fig. 1. A checkpoint and communication pattern.

2 Preliminaries

2.1 Checkpoint and Communication Patterns

A distributed computation consists of a finite set P of n processes $\{P_1, P_2, \dots, P_n\}$, which communicate and synchronize only by exchanging messages. We assume that each pair of processes is connected by a reliable, asynchronous channel with unpredictable but finite transmission delays. 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 the 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 "$\overset{hb}{\rightarrow}$," defined as follows [17]:

Definition 1. *The relation "$\overset{hb}{\rightarrow}$" on the set of events satisfies the following conditions:*

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

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 xth checkpoint of process P_i, where i is called the process id and x the *index* of this checkpoint. The sequence of events occurring at P_i between $C_{i,x-1}$ and $C_{i,x}$ $(x > 0)$ is called a *checkpoint interval* and is denoted by $I_{i,x}$. We will denote a checkpoint interval as an *interval* in the context for the sake of simplicity. Finally, each process P_i is assumed to start its execution with an initial checkpoint $C_{i,0}$.

2.2 Rollback-Dependency Trackability

A checkpoint and communication pattern satisfies Rollback-Dependency Trackability (RDT) if all rollback dependencies between local checkpoints are online 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 that by using a transitive dependency vector. Equivalently, RDT can be stated in terms of the notion of *Z-paths* by Netzer and Xu [8] and the idea of *causal doubling* of Z-paths by Baldoni et al. [13].

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:* $receive(m_i) \in I_{k,s} \wedge send(m_{i+1}) \in I_{k,t} \wedge s \leq t$ *[13].*

We say that a Z-path $[m_1, m_2, \ldots, m_q]$ is 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 the paths $[m_5, m_2]$ and $[m_5, m_3]$ are Z-paths from $I_{k,2}$ to $I_{i,2}$. However, the path $[m_5, m_1]$ is not a Z-path. For the rest of this paper, we 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 $\zeta \cdot \zeta'$.

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 *noncausal* if it is not causal. A Z-path with only one message is trivially causal. For simplicity, a causal Z-path is also called a *causal path*.

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 $y' \leq y$ [13].*

From the previous definition, every causal path is obviously causally doubled by itself. As an example, the Z-path $[m_5, m_2]$ in the pattern of Fig. 1 is noncausal and is causally doubled by the causal path $[m_5, m_3]$.

RDT can then be expressed in terms of causal doubling.

Definition 4. *A checkpoint and communication pattern satisfies RDT* if and only if *all Z-paths are causally doubled [13].*

Since the future of a distributed computation is not available for an online RDT protocol based only on causal history, the information of a Z-path being causally doubled must be online detectable upon the arrival of a message in order to make the checkpointing decision. This observation introduces the concept of *visibly doubling*, that is, the property of causally doubling can be online tested upon the arrival of a message. Such a concept was introduced in [12] for the first time and restated in [13]. Note that, from the definition, a visibly doubled Z-path is causally doubled, but not vice versa. Then, we have the following corollary for the RDT property from the practical point of view.

Corollary 1. *A checkpoint and communication pattern produced by an online protocol satisfies the RDT property* if *all Z-paths are visibly doubled.*

2.3 RDT Characterizations

Given a checkpoint and communication pattern, it is not necessary to check that every noncausal Z-path is causally doubled to ensure that such a pattern satisfies RDT.

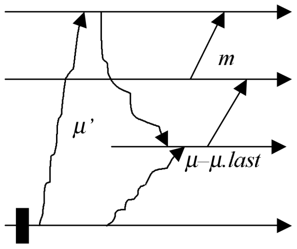

Fig. 2. Visibility of causal doubling.

Causally doubling a certain subset of noncausal Z-paths may suffice. Such a subset is called an RDT characterization in [13]. Particularly, an X-path being an RDT characterization is defined as: In a checkpoint and communication pattern, all X-paths are causally doubled if and only if all Z-paths are causally doubled (i.e., ensuring the RDT property). But, since only online protocols were proposed in the literature, we give a *different* definition for RDT characterizations in this paper.

Definition 5. *In a checkpoint and communication pattern, if all X-paths being* visibly doubled *result in that all Z-paths are causally doubled, the X-path is called an RDT characterization.*

Before we proceed to introduce some crucial RDT characterizations discussed in the context, we will introduce the notion of *prime paths* in advance because every considered RDT characterization contains a prime path with a special property.

Definition 6. *A causal path μ from $I_{i,x}$ to P_j is a* prime path *if every causal path ν from $I_{i,x'}$ to P_j with $x \leq x'$ satisfies that $receive(\mu.last) \overset{hb}{\rightarrow} receive(\nu.last)$.*

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 the pattern shown in Fig. 1, the path $[m_5]$ is prime, but $[m_3]$ is not prime.

Now, an important RDT characterization, called *PCM-paths*, is introduced below.

Definition 7. *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 $send(m) \overset{hb}{\rightarrow} receive(\mu.last)$ [13].*

For instance, the path $[m_5, m_2]$ in Fig. 1 is a PCM-path. The following theorem follows directly from the results in [13].

Theorem 1. *A checkpoint and communication pattern produced by an online protocol satisfies the RDT property if all PCM-paths are visibly doubled.*

Moreover, a visibly doubled PCM-path needs to possess the following property:

Theorem 2. *Suppose a PCM-path $\mu \cdot m$ is causally doubled by a causal path μ'. It is also visibly doubled by μ' if and only if $receive(\mu'.last) \overset{hb}{\rightarrow} send(\mu.last)$ [13].*

Fig. 2 shows an example for Theorem 2. According to the foregoing theorem, an online checkpointing protocol can break any invisibly doubled PCM-path $\mu \cdot m$ by taking a forced checkpoint before $receive(\mu.last)$ to satisfy the RDT property because $\mu \cdot m$ no longer forms a Z-path by doing so.

In [13], Baldoni et al. further proposed some more constrained RDT characterizations for designing protocols. The first one is called the *EPCM-paths* and is defined as follows:

Definition 8. *An EPCM-path is a PCM-path $\mu \cdot m$ such that μ is also elementary [13].*

The formal definition of the term "elementary" is given in the following:

Definition 9. *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 [13].*

Namely, an elementary causal path only traverses a process *once*. For instance, in the checkpoint and communication pattern shown in Fig. 1, the path $[m_4, m_5, m_3]$ is not elementary because it traverses process P_j twice, while the path $[m_5, m_3]$ is elementary. Since EPCM-paths are an RDT-characterization, we also have the theorem below [13].

Theorem 3. *A checkpoint and communication pattern produced by an online protocol satisfies the RDT property if all EPCM-paths are visibly doubled.*

Remark that the size of the necessary control information piggybacked on a message to detect an EPCM-path is the same as that required to detect a PCM-path [13]. Hence, it makes no difference for a protocol to detect an EPCM-path or to detect a PCM-path.

Another RDT characterization proposed in [13] is called *EPSCM-paths*. In advance, we formally describe the definition for the term "simple."

Definition 10. *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, $\forall i$ $(1 \le i \le q-1)$ [13].*

So, a simple causal path does not "*include*" local checkpoints. As an example, the path $[m_4, m_5, m_3]$ in Fig. 1 is not simple since the local checkpoint $C_{k,1}$ is included. As for the path $[m_5, m_3]$, it is simple, however. Then, an EPSCM-path is defined in the following:

Definition 11. *An EPSCM-path is a PCM-path $\mu \cdot m$ such that μ is elementary and simple as well [13].*

Again, we have the following theorem [13]:

Theorem 4. *A checkpoint and communication pattern produced by an online protocol satisfies the RDT property if all EPSCM-paths are visibly doubled.*

Recently, a new RDT characterization, called *PMM-paths*, was introduced by Garcia and Buzato and is defined below.

Definition 12. *A PMM-path $m_1 \cdot m_2$ is a Z-path that is the concatenation of two messages m_1 and m_2, where m_1 is prime and $send(m_2) \overset{hb}{\rightarrow} receive(m_1)$ [15].*

Such a path can be exemplified as the path $[m_5, m_2]$ shown in Fig. 1. Since a PMM-path is composed of only two messages, it is the minimal Z-path allowed in the computation model. Therefore, PMM-paths are a minimal RDT characterization, i.e., it cannot be implied by any other RDT characterization. Similarly, we can also have the following theorem [15]:

Theorem 5. *A checkpoint and communication pattern produced by an online protocol satisfies the RDT property if all PMM-paths are visibly doubled.*

Now, let X be a particular property defined on causal paths. A *PXC-path* is defined as a prime path satisfying the property X and, then, a *PXCM-path* is defined as a PCM-path $\mu \cdot m$ such that μ is a PXC-path. We have no doubt that each of these crucial characterizations mentioned in this section can be represented as a certain type of PXCM-path corresponding to a certain property X. For instance, an EPSCM-path is a type of PXCM-path such that X is defined as the *elementary* and *simple* properties. In addition, if one type of PXCM-path is also an RDT characterization, we denote it as an *RDT-PXCM-path* in the context. For example, PMM-paths are a type of RDT-PXCM-path. Obviously, all the types of RDT-PXCM-paths described previously impose some restrictions on the part of the prime path of a PCM-path to make the length (i.e., the number of messages) of the prime path shorter. Therefore, we conjecture that to find some restrictions to make the part of the prime path of a PCM-path shorter is the only way to find a new type of RDT-PXCM-path. Although it is still an open problem to prove such a conjecture, we only consider a type of RDT-PXCM-path that satisfies the foregoing conjecture in this paper. Thus, we trivially have the following theorem:

Theorem 6. *Let U-paths be one type of RDT-PXCM-path. Then, a U-path is a PCM-path, and a PMM-path is a U-path.*

2.4 Progressive Views

A *cut* of a distributed computation contains an initial prefix of the local history of every process. A *consistent cut* is left-closed under Lamport's happened-before relation and is defined as follows:

Definition 13. *A cut Γ is* consistent *if and only if $e \in \Gamma \wedge e' \overset{hb}{\rightarrow} e \Rightarrow e' \in \Gamma$ [15], where e and e' are two events.*

For instance, there are two consistent cuts, Γ_1 and Γ_2, depicted in the pattern of Fig. 1. And, we have that $\Gamma_1 \subset \Gamma_2$ since Γ_1 is in the left (past) of Γ_2.

A distributed computation is able to be viewed as a sequence of consistent cuts $\{\Gamma_1, \Gamma_2, \ldots, \Gamma_m\}$, called a *progressive view*, associated with the distributed computation, in which every consistent cut happens after the other. That is, we have $\Gamma_1 \subset \Gamma_2 \subset \ldots \subset \Gamma_m$. Note that a distributed computation may have many progressive views. Obviously, an online RDT checkpointing protocol must ensure RDT in every consistent cut of the computation. This leads to the concept of *left-doubling* defined in the following [15]:

Definition 14. *A causal path μ belongs to a consistent cut Γ if $receive(\mu.last) \in \Gamma$.*

Definition 15. *A Z-path $\zeta = \mu_1 \cdot \mu_2 \cdot \ldots \cdot \mu_l$ belongs to a consistent cut Γ if all the causal components of ζ, i.e., all the causal paths μ_i for $1 \le i \le l$ belong to Γ.*

Definition 16. *A Z-path ζ is* left-doubled *by a causal path μ in relation to a consistent cut Γ if and only if both ζ and μ belong to Γ.*

Namely, if a Z-path is left-doubled by another causal Z-path in relation to a consistent cut, these two Z-paths are in the left of this consistent cut. For example, the Z-path $[m_5, m_2]$ in Fig. 1 is left-doubled by the causal path $[m_5, m_3]$ in

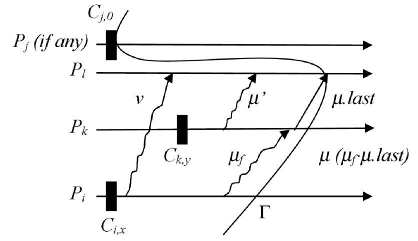

Fig. 3. The scenario of Lemma 1.

relation to the consistent cut Γ_2, but not for the consistent cut Γ_1 since $[m_5, m_3]$ does not belong to Γ_1. Besides, from Theorem 2, if a PCM-path $\mu \cdot m$ is visibly doubled by a causal path μ', then $\mu \cdot m$ is also left-doubled by μ' in relation to any consistent cut which $\mu \cdot m$ belongs to because μ' is in the causal past of $\mu.last$. The notions of consistent cuts and left-doubling will be exploited to construct the proofs in this paper and can also be used to give a new definition for the RDT property from the viewpoint of progressive views.

Definition 17. *A consistent cut Γ satisfies the RDT property* if and only if *all Z-paths that belong to Γ are left-doubled [15].*

Besides, according to the foregoing discussions quoted from [15], we can derive the following theorem:

Theorem 7. *A checkpoint and communication pattern produced by an online protocol satisfies the RDT property* if and only if *all consistent cuts satisfy the RDT property.*

3 Properties of RDT-PXCM-Paths

In this section, we will investigate some interesting properties of RDT-PXCM-paths. First, we will give a definition for a prime path "containing" another prime path.

Definition 18. *Let A-paths and B-paths be any two types of PXC-paths. An A-path $[m_1, m_2, \ldots, m_q]$ $(q \geq 1)$ is said to* contain *a B-path if and only if the message sequence $[m_i, m_{i+1}, \ldots, m_q]$ is a B-path for some i, $1 \leq i \leq q$.*

In [16], it has been proven that a PC-path contains an EPSC-path. Next, we further demonstrate that a PC-path met by an online RDT protocol also contains a PM-path.

Lemma 1. *In a checkpoint and communication pattern, every PC-path encountered by an online RDT protocol contains a PM-path. Namely, its last message is prime.*

Proof. By contradiction, assume there exists a PC-path μ (from $I_{i,x}$ to P_l) encountered by an online RDT protocol, which does not contain a PM-path. This means that its last message, supposed from $I_{k,y}$ to P_l, is not prime. Then, there must exist a prime causal path μ' from $I_{k,y'}$ to P_l such that $y \leq y'$ and $receive(\mu'.last) \overset{hb}{\rightarrow} receive(\mu.last)$. Now, let the causal path composed of all messages of μ except $\mu.last$ be μ_f. We have that $send(\mu'.first) \overset{hb}{\rightarrow} receive(\mu_f.last)$, as shown in Fig. 3; otherwise, the resultant causal path $\mu_f \cdot \mu'$ will conflict the primality of μ. It is possible to construct a consistent cut Γ that only includes the causal past of $receive(\mu.last)$ and the initial checkpoint $C_{j,0}$ of the process P_j (if any), in which no event belongs to the causal past of $receive(\mu.last)$, as depicted in Fig. 3. Note that Γ does not include the event $receive(\mu.last)$. Besides, Γ needs to satisfy RDT because the online RDT protocol must have ensured RDT in every consistent cut in the left of the event $receive(\mu.last)$ before the delivery of $\mu.last$. Hence, by Definition 17, the Z-path $\mu_f \cdot \mu'$ is left-doubled by a causal path ν in relation to Γ. The causal path ν is from $I_{i,x'}$ to P_l such that $x \leq x'$, according to Definition 3. Moreover, $receive(\nu.last) \overset{hb}{\rightarrow} receive(\mu.last)$ since ν belongs to Γ. This conflicts the primality of μ and then leads to a contradiction. □

We can consequently deduce the following theorem from the previous lemma.

Theorem 8. *In a checkpoint and communication pattern, if an online RDT protocol encounters a PCM-path at a point, it also encounters a PMM-path there.*

Let U-paths be a type of RDT-PXCM-path. We obtain the following corollary by applying Theorem 6.

Corollary 2. *In a checkpoint and communication pattern, if an online RDT protocol encounters a PCM-path at a point, it also encounters a U-path there.*

Because U-paths are a subset of PCM-paths, it seems that U-paths would occur less frequently than PCM-paths. However, Corollary 2 reveals that U-paths in fact occur at the same points where PCM-paths occur. Therefore, if an RDT protocol breaks all U-paths, it actually breaks all PCM-paths, not only some certain PCM-paths. That is, it has the same performance with **FDAS**, which exactly breaks all PCM-paths [7]. But, **FDAS** may piggyback less control information on a message to distinguish its checkpoint-inducing condition because **FDAS** does not need to distinguish the extra properties that U-paths possess more than PCM-paths. (See [9], [13], [18] for more details about the size of piggybacked control information to distinguish a condition.) Here, let V-paths be another type of RDT-PXCM-path. From Theorem 6 and Corollary 2, we can derive the following general result.

Corollary 3. *In a checkpoint and communication pattern, if an online RDT protocol encounters a U-path at a point, it also encounters a V-path there.*

Hence, we can conclude that an RDT protocol will meet every type of RDT-PXCM-paths at the same point in a checkpoint and communication pattern.

Now, we will demonstrate another interesting property on RDT-PXCM-paths in the following lemma:

Lemma 2. *In a checkpoint and communication pattern, for every PCM-path $\mu \cdot m$ that an online RDT protocol encounters, if the PMM-path $\mu.last \cdot m$ is visibly doubled, $\mu \cdot m$ is also visibly doubled.*

Proof. Let the causal path doubling $\mu.last \cdot m$ be μ' and the causal path composed of all messages of μ except $\mu.last$ be μ_f. There are two cases to be considered:

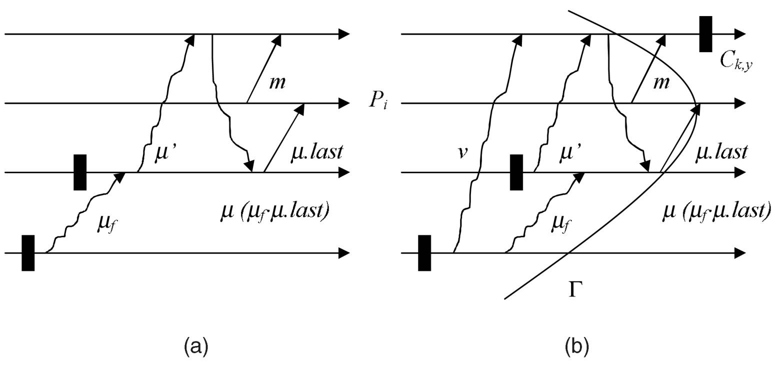

Fig. 4. The scenario of Lemma 2.

1. If $receive(\mu_f.last) \overset{hb}{\rightarrow} send(\mu'.first)$, it is trivial that $\mu \cdot m$ is visibly doubled by the causal path $\mu_f \cdot \mu'$, as shown in Fig. 4a.
2. Now, suppose $send(\mu'.first) \overset{hb}{\rightarrow} receive(\mu_f.last)$. It is possible to construct a consistent cut Γ that only includes the causal past of $receive(\mu.last)$ and the initial checkpoint $C_{j,0}$ of the process P_j (if any) in which no event belongs to the causal past of $receive(\mu.last)$, as depicted in Fig. 4b. Note that Γ does not include the event $receive(\mu.last)$, but includes both μ_f and μ' because these two causal paths are in the causal past of $receive(\mu.last)$. Besides, if the message m is from P_i to the interval $I_{k,y}$, Γ does not contain the checkpoint $C_{k,y}$. Otherwise, there is a causal path μ_l that started after $C_{k,y}$ and reached P_i between the two events $send(m)$ and $receive(\mu.last)$. The resultant Z-path $\mu_l \cdot m$ is a nondoubled Z-cycle and has to be broken by a forced checkpoint taken between $send(m)$ and $receive(\mu.last)$. Then, the considered PMM-path $\mu.last \cdot m$ and PCM-path $\mu \cdot m$ will also be broken. So, we do not have to consider such a case. Here, Γ needs to satisfy RDT because the online RDT protocol must have ensured RDT in every consistent cut in the left of the event $receive(\mu.last)$ before the delivery of $\mu.last$. Hence, by Definition 17, the Z-path $\mu_f \cdot \mu'$ is left-doubled by a causal path ν in relation to Γ. Obviously, ν can also causally double $\mu \cdot m$ due to $receive(\nu.last) \overset{hb}{\rightarrow} C_{k,y}$. And, ν is in the causal past of $\mu.last$ since ν belongs to Γ. Thus, ν can be detected before the delivery of $\mu.last$. This means that $\mu \cdot m$ is visibly doubled by the causal path ν.□

Likewise, we can have the following theorem according to Lemma 2.

Theorem 9. *In a checkpoint and communication pattern, if an online RDT protocol encounters an invisibly doubled PCM-path at a point, it also encounters an invisibly doubled PMM-path there.*

Moreover, we can also deduce the following corollary from Theorem 6.

Corollary 4. *In a checkpoint and communication pattern, if an online RDT protocol encounters an invisibly doubled PCM-path at a point, it also encounters an invisibly doubled U-path there.*

Consequently, if an RDT protocol breaks every invisibly doubled U-path, it actually breaks all invisibly doubled PCM-paths. Thus, it has the same behavior with **BHMR**, which exactly breaks all invisibly doubled PCM-paths [9], but often needs more control information piggybacked on every message to distinguish the extra properties that U-paths possess more than PCM-paths. Again, we can obtain the following general result.

Corollary 5. *In a checkpoint and communication pattern, if an online RDT protocol encounters an invisibly doubled U-path at a point, it also encounters an invisibly doubled V-path there.*

The protocol **BHMR** introduced in [9] was implemented as piggybacking $O(n^2)$ control information on a message. In [13], the protocol breaking all invisibly doubled EPSCM-paths also needs to piggyback $O(n^2)$ control information on each message (but more than BHMR's), and the authors claimed that this protocol is optimal with respect to the size of control information. However, it has been shown that the protocol breaking all invisibly doubled PMM-paths only requires $O(n)$ control information piggybacked on a message in a later work [19]. Therefore, we have another interesting property obtained from the foregoing results now. Since we have demonstrated that breaking all invisibly doubled PMM-paths is equivalent to breaking all invisibly doubled EPSCM-paths, this means that it is only necessary to piggyback the $O(n)$ control information found in [19] to implement the protocol breaking all invisibly doubled EPSCM-paths instead of the $O(n^2)$ control information found in [13]. Also, BHMR can exploit such $O(n)$ control information to break all invisibly doubled PCM-paths. So, although detecting a type of RDT-PXCM-path other than the PCM-path expectedly requires piggybacking more control information on a message to distinguish the extra properties, sometimes we can perhaps design a more efficient protocol by finding a new type of RDT-PXCM-path. But, so far in the literature, only the PMM-path is known to be able to achieve this benefit, for the sake of being constituted by two messages.

4 Properties of Visibly Doubling

In advance, we 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, we also call this V-path a *V-cycle*. Trivially, if

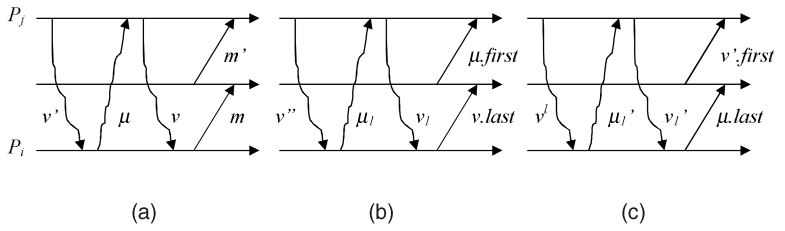

Fig. 5. The scenario of Theorem 10.

$x' < x$, this V-cycle cannot be causally doubled and, thus, is called a *nondoubled* V-cycle. For example, the path $[m_5, m_4]$ in Fig. 1 is a PMM-cycle and is nondoubled. Another interesting contribution of this paper is a property on visibly doubling. Now, let U-paths be a type of RDT-PXCM-path. We will show that if an online RDT checkpointing protocol breaks all invisibly doubled U-paths and all U-cycles, it will break all visibly doubled U-paths. This result allows us to identify the redundant control information used by such a protocol to detect visibly doubled U-paths. It has been proven that PCM-paths and EPSCM-paths have the foregoing property in [10] and [16], respectively. In the following, we continue to reveal that PMM-paths have this property, too.

Theorem 10. *Given any checkpoint and communication pattern, if an online RDT protocol breaks all invisibly doubled PMM-paths and all PMM-cycles, it also breaks all visibly doubled PMM-paths.*

Proof. Suppose there exists an online RDT protocol that breaks all invisibly doubled PMM-paths and all PMM-cycles, but does not break some visibly doubled PMM-paths. Let $m \cdot m'$ denote such a PMM-path. As shown in Fig. 5a, $m \cdot m'$ must have a doubling causal path μ and a causal path ν that had brought the knowledge about μ to the sender of m before m was sent. Without loss of generality, we can assume that μ is prime. And, trivially, the messages $\mu.last$ and $\nu.last$ are in the causal past of $send(m)$. We will show that the absence of a forced checkpoint to break $m \cdot m'$ leads to a contradiction. So, the protocol must break all visibly doubled PMM-paths.

First, suppose the causal path ν is prime. According to Lemma 1, we have that the message $\nu.last$ is prime. Besides, the PMM-path $\nu.last \cdot \mu.first$ cannot be a PMM-cycle or an invisibly doubled PMM-path that has to be broken by a forced checkpoint since such a forced checkpoint will make $m \cdot m'$ unable to be causally doubled by μ. Thus, $\nu.last \cdot \mu.first$ is a visibly doubled PMM-path and has a doubling causal path μ_1 and a causal path ν_1 which brought the knowledge about μ_1 to the sender of $\nu.last$ before $\nu.last$ was sent, as depicted in Fig. 5b. Obviously, the messages $\mu_1.last$ and $\nu_1.last$ are in the causal past of $send(\nu.last)$ and, consequently, in the causal past of $send(m)$.

Now, suppose the causal path ν is not prime and started from the interval $I_{j,x}$. Then, there must be another causal path ν' that reached P_i before $receive(\nu.last)$ and started in the interval $I_{j,x'}$ with $x \leq x'$. If $receive(\mu.last) \xrightarrow{hb} send(\nu'.first)$, we have the same scenario with the case that ν is prime because $receive(\nu'.last)$ must occur after $send(\mu.first)$; otherwise, there would be causal cycle. So, we only have to consider the case of $send(\nu'.first) \xrightarrow{hb} receive(\mu.last)$, as shown in Fig. 5a. Also, by Lemma 1, we have that the message $\mu.last$ is prime since μ is prime. Besides, if the PMM-path $\mu.last \cdot \nu'.first$ is a PMM-cycle or an invisibly doubled PMM-path, it has to be broken by a forced checkpoint. Such a forced checkpoint will make ν prime since we can pick ν to be the one arriving P_i first among all the causal paths which reached P_i and started from P_j posterior to $receive(\mu.last)$. Thus, $\mu.last \cdot \nu'.first$ is a visibly doubled PMM-path and has a doubling causal path μ_1' and a causal path ν_1' that brought the knowledge about μ_1' to the sender of $\mu.last$ before $\mu.last$ was sent, as depicted in Fig. 5c. Likewise, the messages $\mu_1'.last$ and $\nu_1'.last$ are in the causal past of $send(\mu.last)$ and, thus, in the causal past of $send(m)$.

So far, we have shown that, for the assumed protocol, an unbroken, visibly doubled PMM-path $m \cdot m'$ implies the existence of another distinct unbroken, visibly doubled PMM-path $\nu.last \cdot \mu.first$ (or $\mu.last \cdot \nu'.first$), where $\nu.last$ (or $\mu.last$) is in the causal past of $send(m)$. Similarly, the unbroken, visibly doubled PMM-path $\nu.last \cdot \mu.first$ also implies the existence of another distinct, unbroken, visibly doubled PMM-path $\nu_1.last \cdot \mu_1.first$ (or $\mu_1.last \cdot \nu_1''.first$), where $\nu_1.last$ (or $\mu_1.last$) is in the causal past of $send(\nu.last)$. Alternatively, the unbroken, visibly doubled PMM-path $\mu.last \cdot \nu'.first$ implies a similar scenario, too. Hence, by repeatedly applying the above argument, the existence of the unbroken $m \cdot m'$ implies an infinite number of distinct unbroken visibly doubled PMM-paths in the causal past of $send(m)$. This contradicts the fact that the causal history must be finite. □

Because a PMM-path is also a U-path by Theorem 6 and, thus, a PMM-cycle is also a U-cycle, an online RDT protocol breaking all invisibly doubled U-paths and all U-cycles will break all invisibly doubled PMM-paths and all PMM-cycles. Hence, it will break all visibly doubled PMM-paths from the previous theorem. Therefore, this protocol will break all PMM-paths. Moreover, according to the discussions in Section 3, breaking all PMM-paths is equivalent to breaking all U-paths. So, such a protocol will in fact break all U-paths and, consequently, all visibly doubled U-paths. So, we have the following corollary:

Corollary 6. *Given any checkpoint and communication pattern, if an online RDT protocol breaks all invisibly doubled U-paths and all U-cycles, it also breaks all visibly doubled U-paths.*

Let us define the causal past of a U-path $\mu \cdot m$ as the causal past of $receive(\mu.last)$. From the process of proving Theorem 10, we can find that if an online RDT protocol allows the causal past of a visibly doubled PMM-path to contain a doubled PMM-cycle, such a PMM-path will not induce infinite numbers of visibly doubled PMM-paths. Therefore, if an RDT protocol breaks all PMM-cycles, no PMM-path can be visibly doubled. The proofs for PCM-paths and EPSCM-paths in [10] and [16] also exhibit similar scenarios. We can also conclude that a visibly doubled U-path must contain a doubled U-cycle in the causal past from Corollary 6.

5 Systematic Techniques for Comparisons

There have been many online RDT checkpointing protocols proposed in the literature [7], [9], [10], [13], [18] and the performance of most existing RDT protocols have also been compared, both with theoretical analyses and simulations [9], [10], [13], [16], [18]. In this section, we will give a systematic technique to compare the performance of online RDT protocols theoretically, applying the results found in the previous works [10], [16] and the results newly introduced in this paper. We want to point out that this systematic technique can compare the *existing* RDT protocols correctly, so we conjecture that it can also apply to all ordinary online RDT protocols correctly. However, because there may be dozens of online RDT protocols and some of them may be weird, it is difficult to give a formal and concise proof for the foregoing conjecture in the context now. It is still an open problem that we will try to solve in the future.

Here, suppose the two online RDT protocols to be compared are **PTLA** and **PTLB**, of which checkpoint-inducing conditions are C_A and C_B, respectively. Besides, we have that C_B implies C_A. Namely, the protocol PTLB is based on a stronger condition than PTLA's. Now, let U-paths be a type of RDT-PXCM-path. Then, the commendatory systematic technique for comparing RDT protocols is introduced in the following:

1. A protocol breaking all U-paths is equivalent to **FDAS**.
2. A protocol breaking all invisibly doubled U-paths and all U-cycles is equivalent to **FDAS**.
3. A protocol breaking all invisibly doubled U-paths is equivalent to **BHMR**.
4. If **PTLA** and **PTLB** are equivalent to **FDAS** by applying Step 1 or 2, they are equivalent.
5. If **PTLA** and **PTLB** are equivalent to **BHMR** by applying Step 3, they are equivalent.
6. If **PTLA** is equivalent to **FDAS** by applying Step 1 or 2, **PTLB** outperforms **PTLA** because protocols that force a checkpoint at stronger conditions outperform **FDAS** [10].
7. If **PTLB** is equivalent to **FDAS** by applying Step 1 or 2, **PTLB** outperforms **PTLA** because **FDAS** outperforms protocols that force a checkpoint at weaker conditions [10].
8. If both C_A and C_B are weaker than the checkpoint-inducing condition of FDAS, PTLB outperforms **PTLA** because all the existing RDT protocols based on weaker conditions than FDAS's have been shown comparable in [10] and another RDT protocol with a weaker condition than FDAS's has never been proposed so far.

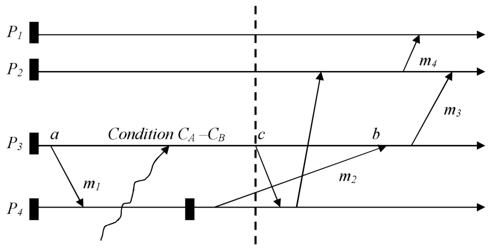

Fig. 6. The checkpoint and communication pattern *ccpat*.

9. If **PTLA** does not break all U-cycles, **PTLA** and **PTLB** are *incomparable*. That is, even though the protocol **PTLB** is based on a stronger condition, it may force more checkpoints than **PTLA** does for some communication patterns. We can construct the checkpoint and communication pattern *ccpat*, depicted in Fig. 6, to demonstrate the foregoing observation. Such a pattern has the following properties:

    a. Any RDT protocol must force the process P_3 in *ccpat* to take at least one checkpoint between point a and b since the Z-cycle $[m_2 \cdot m_1]$ is nondoubled.
    b. Any RDT protocol which forces P_3 in *ccpat* to take a checkpoint between point c and b must also force P_2 to take another forced checkpoint because the message m_3 will become prime and the Z-path $[m_3 \cdot m_4]$ is noncausally doubled.
    c. There is a $(C_A - C_B)$ condition which means a condition belonging to C_A but not belonging to C_B, between point a and c in *ccpat*.
    d. **PTLA** will take a forced checkpoint when it encounters such a condition and only needs this forced checkpoint to make *ccpat* satisfy RDT.
    e. **PTLB** will force a checkpoint before the delivery of m_2 and thus needs another forced checkpoint to break $[m_3 \cdot m_4]$.

So, for the pattern *ccpat*, PTLA outperforms than PTLB. However, for the cut pattern shown as the portion to the left of the dotted line in Fig. 6, the online protocol **PTLA** must still take exactly one forced at the point of meeting the condition $(C_A - C_B)$, whereas **PTLB** does not take any forced checkpoint. Hence, PTLA and PTLB are incomparable.

Checkpoint and communication patterns used in [10], [16], [18] to show that two RDT protocols are incomparable can all be constructed by applying Step 9. Besides, as an illustration, let us compare the two protocols **RDT-Partner** (breaking all PMM-paths except doubled PMM-cycles) [18] and **No-EPSCM-Path** (breaking all EPSCM-paths except doubled EPSCM-cycles) [13] now. Obviously, the checkpoint-inducing condition of **RDT-Partner** is stronger than that of **No-EPSCM-Path**. But, since **No-EPSCM-Path** does not break all EPSCM-cycles, these two protocols are incomparable. We can use the systematic technique introduced previously to construct a checkpoint and communication pattern to reveal this result. Such a pattern is shown in Fig. 7. Figs. 7a and 7b show the resultant patterns of applying **RDT-Partner** and **No-EPSCM-Path**, respectively, where rectangular boxes

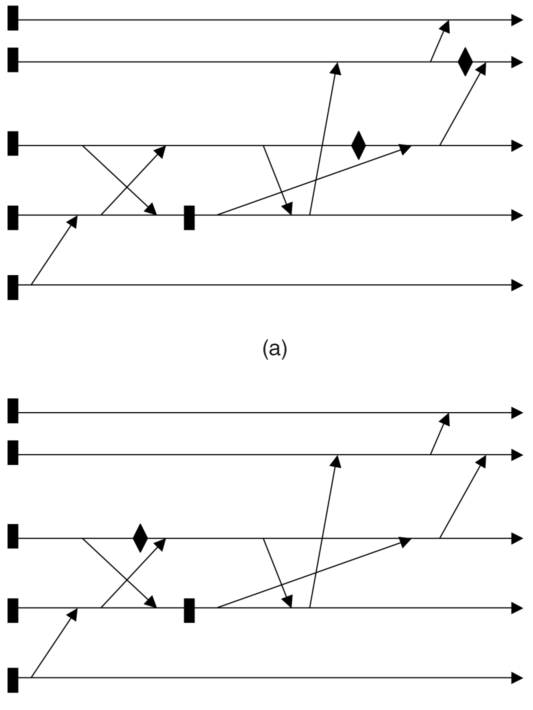

Fig. 7. (a) Applying **RDT-Partner**, (b) applying **No-EPSCM-Path** (or applying **No-PCM-Path**).

represent basic checkpoints and diamond boxes represent forced checkpoints. The figure indicates that there exists a pattern for which **RDT-Partner** must take two forced checkpoints to satisfy RDT, while **No-EPSCM-Path** needs only one. Moreover, such a pattern can also be used to compare **RDT-Partner** and **No-PCM-Path** [13], which breaks ll PCM-paths except a PCM-cycle $\mu \cdot m$ with $send(\mu.first) \overset{hb}{\rightarrow} receive(m)$. The result of applying **No-PCM-Path** to this pattern has the same scenario as Fig. 7b. Hence, these two protocols are also incomparable because **No-PCM-Path** does not break all PCM-cycles.

Although our previous systematic technique only finds a special pattern for performance comparison as two RDT checkpointing protocols are incomparable, it can still offer a good theoretical basis for analyzing the behaviors of the considered protocols. The constructed pattern can help us further investigate which protocol has a better behavior in a certain pattern. Then, we can derive which protocol we should use for a particular application in order to obtain better performance.

6 Related Work

Some communication-induced checkpointing protocols can only achieve the property of *Domino-Effect Freedom (DEF)* [20], [21], [22], [23]. Such protocols guarantee no domino effect, but there still exist hidden, zigzag dependencies among local checkpoints. Note that RDT protocols ensure that all dependencies are made online trackable. DEF communication-induced checkpointing protocols can be classified into two distinct categories: *index-based* and *model-based* [4]. An index-based protocol associates local checkpoints with sequence numbers similar to Lamport's logical clocks [17] in a way that checkpoints with the same sequence numbers are forced to be consistent. Comparatively, a model-based protocol does not use a timestamping function, but prevents the formation of special checkpoint and communication patterns in the execution. DEF protocols proposed in [20], [21] belong to the index-based category, whereas those proposed in [22], [23] belong to the model-based category.

In [24], several simulation experiments were conducted to compare the performance of some index-based and model-based DEF protocols. The authors found that if the communication patterns under study mimic a periodic broadcast, model-based protocols appear to be "eager" in taking forced checkpoints to prevent the formation of Z-cycles compared to index-based ones. In [25], a few theoretical results on performance comparisons of some DEF protocols were introduced. The discussions in [25] can be exploited to explain the foregoing simulation result more than those described in [24]. In [24], each process running the simulation has the same rate of taking basic checkpoints such that the sequence number of every process is almost equivalent at the same instant. Besides, those parallel applications under study use a common iterative structure to solve a computation-intensive problem in which processes change results and resume. Thus, a process often sends a message to some process with a larger or equivalent sequence number and has received a message from another process in the previous interval. This leads to the fact that the checkpoint-inducing condition of a model-based protocol will occur more frequently than that of an index-based one [25]. To the contrary, if a process usually sends a message to another process with a sequence number not larger than its own, then model-based protocols outperform index-based ones for such a pattern [25].

7 Conclusions

Since the RDT property has been introduced, several characterizations of such a property in terms of checkpoint and communication patterns have been proposed. These crucial characterizations can be represented as some type of RDT-PXCM-path. Here, let the U-path and V-path be any two types of RDT-PXCM-paths. In the context, we have found some properties of these RDT characterizations. First, we demonstrated that if an online RDT protocol meets an invisibly doubled U-path at a point of a pattern, it also meets an invisibly doubled V-path there. So, we can obtain a conclusion that breaking all invisibly doubled U-paths is equivalent to breaking all invisibly doubled V-paths for an online RDT protocol. Moreover, we also showed that a visibly doubled U-path must contain a doubled U-cycle in the causal past. From the previous results, we revealed that some different protocols in fact have the same behavior for all possible patterns. Therefore, we can find the equivalent but more efficient protocol for a certain protocol. Last but not least, we presented a commendatory systematic technique for comparing the performance of ordinary online RDT protocols. Such a technique provides a good theoretical basis for analyzing the behavior of RDT protocols.

ACKNOWLEDGMENTS

The author wishes to express his sincere thanks to the anonymous referees for their valuable comments. This work was supported by the National Science Council, Taiwan, Republic of China, under Grant NSC 90-2218-E-005-007.

REFERENCES

[1] Y.M. Wang, A. Lowry, and W.K. Fuchs, "Consistent Global Checkpoints Based on Direct Dependency Tracking," *Information Processing Letters,* vol. 50, no. 4, pp. 223-230, May 1994.

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

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

[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, no. 3, pp. 375-408, Sept. 2002.

[5] R. Koo and S. Toueg, "Checkpointing and Rollback-Recovery for Distributed Systems," *IEEE Trans. Software Eng.,* vol. 13, no. 1, pp. 23-31, Jan. 1987.

[6] B. Janssens and W.K. Fuchs, "Experimental Evaluation of Multiprocessor Cache-Based Error Recovery," *Proc. Int'l Conf. Parallel Processing,* no. 1, pp. 505-508, 1991.

[7] 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.

[8] 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.

[9] R. Baldoni, J.M. Helary, A. Mostefaoui, and M. Raynal, "A Communication-Induced Checkpointing Protocol that Ensures Rollback-Dependency Trackability," *Proc. IEEE Fault-Tolerant Computing Symp.,* pp. 68-77, 1997.

[10] 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.

[11] 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.

[12] R. Baldoni, J.M. Helary, and M. Raynal, "Rollback-Dependency Trackability: Visible Characterizations," *Proc. 18th ACM Symp. Principles of Distributed Computing,* pp. 33-42, May 1999.

[13] 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, Mar. 2001.

[14] 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, no. 2, pp. 105-111, Oct. 2001.

[15] I.C. Garcia and L.E. Buzato, "On the Minimal Characterization of the Rollback-Dependency Trackability Property," *Proc. 21st IEEE Int'l Conf. Distributed Computing Systems,* pp. 342-349, Apr. 2001.

[16] J. Tsai, S.Y. Kuo, and Y.M. Wang, "Some Characteristics of Communication-Induced Checkpointing Protocols with RollBack-Dependency Trackability," *J. Parallel and Distributed Computing,* Mar. 2003.

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

[18] I.C. Garcia, G.M.D. Vieira, and L.E. Buzato, "RDT-Partner: An Efficient Checkpointing Protocol that Enforces Rollback-Dependency Trackability," *Proc. 19th Brazilian Symp. Computer Networks,* May 2001.

[19] I.C. Garcia and L.E. Buzato, "A Linear Approach to Enforce the Minimal Characterization of the Rollback-Dependency Trackability Property," Technical Report IC-01-17, Univ. of Campinas, Brazil, Dec. 2001.

[20] A. Mostefaoui, J.M. Helary, R.H.B. Netzer, and M. Raynal, "Communication-Based Prevention of Useless Checkpoints in Distributed Computations," *Distributed Computing,* vol. 13, no. 1, pp. 29-43, Jan. 2000.

[21] G.M.D. Vieira, I.C. Garcia, and L.E. Buzato, "Systematic Analysis of Index-Based Checkpointing Algorithms Using Simulation," *Proc. IX Brazilian Symp. Fault-Tolerant Computing,* 2001.

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

[23] I.C. Garcia and L.E. Buzato, "Checkpointing Using Local Knowledge about Recovery Lines," Technical Report, TR-IC-99-22, Univ. of Campinas, Brazil, 1999.

[24] L. Alvisi, E. Elnozahy, S. Rao, S.A. Husain, and A. De Mel, "An Analysis of Communication-Induced Checkpointing," *Proc. IEEE Fault-Tolerant Computing Symp.,* pp. 242-249, 1999.

[25] J. Tsai and J.W. Lin, "On Characteristics of DEF Communication-Induced Checkpointing Protocols," *Proc. Pacific Rim Int'l Symp. Dependable Computing,* pp. 29-36, 2002.

Jichiang Tsai received the BS 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 PhD degree in electrical engineering in 1999. Dr. Tsai served as a postdoctoral research fellow at 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, computer architecture, operating systems, and computer networks.