# Necessary and Sufficient Conditions for Consistent Global Snapshots

Robert H.B. Netzer and Jian Xu

Department of Computer Science
Brown University
Providence, Rhode Island 02912

**CS-93-32**
September 1993

*July 5, 1993*

# Necessary and Sufficient Conditions for Consistent Global Snapshots

*Robert H. B. Netzer*
*rn@cs.brown.edu*

*Jian Xu*
*jx@cs.brown.edu*

Dept. of Computer Science
Brown University
Box 1910
Providence, RI 02912

## Abstract

Consistent global snapshots are important in many distributed applications. We prove the exact conditions for an arbitrary checkpoint, or a set of checkpoints, to belong to a consistent global snapshot, a previously open problem. To describe the conditions, we introduce a generalization of Lamport's happened-before relation called a *zigzag path*.

Keywords: causality, global checkpoints, distributed systems, consistent global states, Lamport's happened-before relation.

### 1. Introduction

Checkpoints are used in many applications including parallel debugging[6, 9], distributed simulation[3], and fault-tolerant computing[1, 4]. A checkpoint is a saved intermediate state of an execution and can be analyzed post-mortem or used to restart the execution from that point. A global snapshot[3] of a distributed computation is a set of local checkpoints, one for each process. A global snapshot is *consistent* if no local checkpoint *happens before* another; that is, there is no causal path from one checkpoint to another in the sense that a message (or sequence of messages) sent after one checkpoint is received before the other[5]. As a result, a consistent snapshot is a set of local states that actually occurred simultaneously during execution or had the potential of occurring simultaneously. Determining when individual local checkpoints can be combined with others to form a consistent snapshot is an important problem. In this paper we prove the exact conditions under which an arbitrary checkpoint, or a set of checkpoints, can be combined with other checkpoints to form a consistent global snapshot. In the

[1] This research was partly supported by ONR Contract N00014-91-J-4052, ARPA Order 8225.

[2] R.H.B. Netzer and J. Xu are with the Dept. of Computer Science, Brown University, Box 1910, Providence, RI 02912. e-mail: {rn,jx}@cs.brown.edu

[3] The terms *global snapshot* and *global checkpoint* are used interchangeably in the literature. We use the term ''global snapshot'' to avoid overloading the term ''checkpoint'' with two meanings (one global and one local).

past only the necessary conditions have been known; the sufficient conditions have been an open problem.

Our main result is a proof of the necessary and sufficient conditions. For example, given two checkpoints, the definition of consistency says that neither can happen before the other if they are to belong to the same consistent snapshot; this condition is necessary for the checkpoints to be consistent. In Figure 1, neither of the checkpoints $C_{1,1}$ and $C_{3,1}$ happen before each other, and they can be combined with $C_{2,1}$ to form a consistent snapshot (as shown by the dashed line in Figure 1). However, as noticed by Wang, Lowry and Fuchs[8], this condition is not sufficient. In Figure 1, checkpoints $C_{1,1}$ and $C_{3,2}$ also do not happen before each other, but they *cannot* be combined with any other checkpoint to form a consistent snapshot. A consistent snapshot contains one checkpoint per process, and in this example no checkpoint in $p2$ can be combined with both $C_{1,1}$ and $C_{3,2}$ while maintaining consistency. Because of message $m4$, $C_{3,2}$ cannot be consistent with $C_{2,1}$ or any earlier checkpoint in $p2$, and because of message $m3$, $C_{1,1}$ cannot be consistent with $C_{2,2}$ or any later checkpoint in $p2$. Thus, no checkpoint in $p2$ is available to form a consistent snapshot.

To describe the necessary and sufficient conditions, we define a generalization of Lamport's happened-before relation called a *zigzag path*. The happened-before relation shows causal paths among checkpoints; one checkpoint happens before another if a sequence of messages exists from one checkpoint to another with each

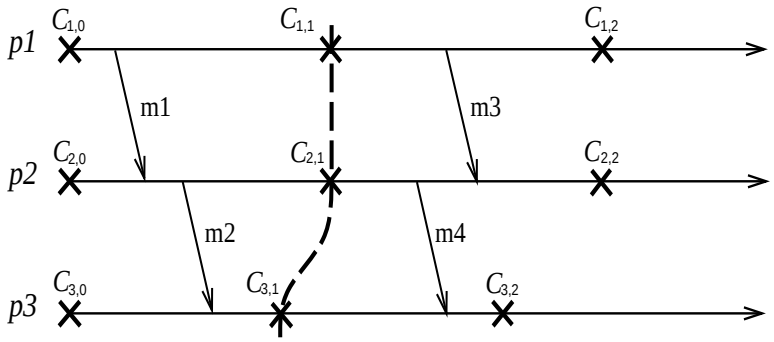


**Figure 1. Consistent and inconsistent snapshots ("×"s indicate checkpoints): $C_{1,1}$, $C_{2,1}$ and $C_{3,1}$ form a consistent snapshot; $C_{1,1}$ and $C_{3,2}$ cannot belong together in a consistent snapshot.**

message sent after the previous one in the sequence is received. A zigzag path between two checkpoints is like a causal path, but a zigzag path allows a message to be sent *before* the previous one in the path is received. As we later prove, a zigzag path exists between two checkpoints exactly when they cannot belong to the same consistent snapshot. For example, in Figure 1 the checkpoint $C_{1,0}$ happens before $C_{3,1}$ because of the causal path formed by messages $m1$ and $m2$. Although a causal path does not exist from $C_{1,1}$ to $C_{3,2}$, a zigzag path does exist, formed by messages $m3$ and $m4$ (the path forms a zigzag shape in the graph). This zigzag path means that $C_{1,1}$ and $C_{3,2}$ can never belong together in any consistent snapshot.

Zigzag paths have applications to any problem that requires saving or analyzing consistent snapshots. For example, in a previous paper [10] we used zigzag paths to virtually eliminate rollback propagation in a checkpointing and rollback recovery scheme for fault tolerance[2]. In such a scheme, the state of a distributed system is saved periodically in checkpoints during normal execution. Upon detecting a fault, recovery is performed by rolling the system back to a previous state not affected by the fault, and resuming normal execution. The snapshot from which restart occurs must be consistent to ensure no process is restarted from a state that has recorded the receipt of a rolled back message. In an independent checkpointing scheme (where there is no checkpoint coordination among processes), not all checkpoints are guaranteed to belong to a consistent snapshot, and in the worst case, rollback propagates and requires restarting from the execution's beginning[7]. We virtually eliminate rollback propagation by tracking zigzag paths on-line, allowing each process to adaptively take checkpoints at moments that minimize the number of checkpoints that cannot belong to a consistent snapshot[10].

Another application for consistent snapshots and consistent states in general is parallel and distributed debugging. A consistent state represents some state that could have occurred during an execution regardless of the relative speed of process execution or message delays. Only consistent states can be used to evaluate some global predicates or answer debugging queries. We believe the work presented here also provides a better understanding of consistent global states.

## 2. Model

We next outline our model for representing executions of message-passing programs. A *program execution* is a pair, $P = \langle E, \xrightarrow{\text{HB}} \rangle$, where $E$ is a finite set of *events* and $\xrightarrow{\text{HB}}$ is the *happened-before* relation defined over $E$[5]. An event represents the execution instance of a send, receive, or checkpoint operation.

We assume that a fixed number of processes exist during execution, and that each process periodically saves (or *checkpoints*) its state in stable storage. We denote the $i^{\text{th}}$ checkpoint in process $p$ as $C_{p,i}$. We also assume each process $p$ takes an initial checkpoint $C_{p,0}$ immediately after execution begins. In addition, the *checkpoint interval*

$S_{p,i}$ is all the computation performed in process $p$ between checkpoints $C_{p,i}$ and $C_{p,i+1}$ (and includes $C_{p,i}$ but not $C_{p,i+1}$).

The happened-before relation, $\xrightarrow{\text{HB}}$, shows how events potentially affect one another[5], and is defined as the irreflexive transitive closure of the union of two other relations: $\xrightarrow{\text{HB}} = (\xrightarrow{\text{XO}} \cup \xrightarrow{\text{M}})^+$. The $\xrightarrow{\text{XO}}$ relation shows the order in which events in the same process execute. The $i^{\text{th}}$ event in any process $p$ (denoted $e_{p,i}$) always executes before the $i+1^{\text{st}}$ event: $e_{p,i} \xrightarrow{\text{XO}} e_{p,i+1}$. The $\xrightarrow{\text{M}}$ relation shows the order in which messages are delivered: $a \xrightarrow{\text{M}} b$ means that $a$ sent a message that $b$ received. An event $a$ is said to *happen before* (or have a *causal path* to) an event $b$ iff $a$ could affect $b$ because they belong to the same process or because a sequence of messages was sent from $a$ (or a following event) to $b$ (or a preceding event).

A *global snapshot* is a set of local checkpoints, one per process (for simplicity, we assume the channel states can be inferred from the local states). A *consistent global snapshot* is a global snapshot in which no message is recorded received but not yet sent. That is, for any two checkpoints $C_{p,i}$ and $C_{q,j}$ in a consistent global snapshot, $C_{p,i} \xleftrightarrow{\text{HB}/} C_{q.j}$ (i.e., the two checkpoints are *unordered* by the $\xrightarrow{\text{HB}}$ relation)[4].

## 3. Zigzag Paths and Consistent Global Snapshots

We now present our results. We first introduce *zigzag paths* as a generalization of Lamport's happened-before relation, and then use them to characterize exactly when an arbitrary set of checkpoints can all belong to the same consistent snapshot. We then present examples of two important special cases: the conditions for an arbitrary checkpoint to be useful in the sense that some consistent snapshot exists to which it belongs, and the conditions for two arbitrary checkpoints to both belong to the same consistent snapshot.

### 3.1. Zigzag Paths

Recall that a global snapshot is consistent iff none of its component checkpoints happen before one another (they are all mutually unordered). However, as shown in Figure 1, having two checkpoints that do not happen before each other is not sufficient for ensuring that they can belong together in some consistent snapshot. Even if neither happens before the other, they can be precluded from ever belonging to a consistent snapshot by their rela-

---

[4] We use superscripted arrows to denote relations, and write $a \not\longrightarrow b$ as a shorthand for $\neg(a \longrightarrow b)$, and $a \not\longleftrightarrow b$ as a shorthand for $\neg(a \longrightarrow b) \wedge \neg(b \longrightarrow a)$.

tionships to checkpoints in other processes. We define the notion of a *zigzag path* as a generalization of Lamport's happened-before relation to express this situation.

*Definition 1*

A *zigzag path* exists from $C_{p,i}$ to $C_{q,j}$ iff there are messages $m_1, m_2, \cdots, m_n$ ($n \geq 1$) such that

(1) $m_1$ is sent by process $p$ after $C_{p,i}$,

(2) if $m_k$ ($1 \leq k < n$) is received by process $r$, then $m_{k+1}$ is sent by $r$ in the same or a later checkpoint interval (although $m_{k+1}$ may be sent before *or* after $m_k$ is received), and

(3) $m_n$ is received by process $q$ before $C_{q,j}$.

Checkpoint $C$ is involved in a *zigzag cycle* iff there is a zigzag path from $C$ to itself.

It is important to note the difference between a zigzag path and a causal path. Lamport's $\xrightarrow{\text{HB}}$ relation shows causal paths. A causal path exists from $C_{p,i}$ to $C_{q,j}$ iff there is a chain of messages starting after $C_{p,i}$ and ending before $C_{q,j}$ with each message sent after the previous one in the chain is received. For a zigzag path, we allow such a chain, but also allow any message in the chain to be sent *before* the previous one is received, as long as the send and receive are in the same checkpoint interval. Thus, a causal path is always a zigzag path, but a zigzag path may not be a causal path (and thus may not cause one checkpoint to happen before the other). Figure 1 illustrates such a difference. A causal path exists from $C_{1,0}$ to $C_{3,1}$ formed by the chain of messages $m1$ and $m2$, the receive of $m1$ happening *before* the send of $m2$; this causal path is also a zigzag path. A zigzag path exists from $C_{1,1}$ to $C_{3,2}$ formed by the chain of messages $m3$ and $m4$, the receive of $m3$ happening *after* the send of $m4$; this zigzag path is not a causal path, since it does not cause $C_{1,1}$ to happen before $C_{3,2}$.

Another difference between zigzag paths and causal paths is that zigzag paths do not always represent causality, so it is possible for a zigzag path to exist from a checkpoint back to itself (called a zigzag cycle). In contrast, causal paths can never form cycles. A zigzag cycle can exist because in a zigzag path we allow a message to be sent before the previous message in the path is received (as long as the send and receive are in the same checkpoint interval). Figure 3 shows a zigzag cycle involving $C_{2,1}$, formed by messages $m4$ and $m3$; $m3$ is sent before $m4$ is received, and both are in the same checkpoint interval ($S_{1,1}$).

### 3.2. Consistent Global Snapshots

Our definition of zigzag paths generalizes the happened-before relation since causal paths are always zigzag paths but not *vice versa*. Making such a generalization allows us to capture exactly what constrains checkpoints to

be in the same consistent snapshot. In this section, we prove that a consistent snapshot can be formed that includes an arbitrary set $S$ of checkpoints iff no zigzag path exists between any two checkpoints in $S$ (including a zigzag cycle from any checkpoint to itself).

Intuitively, if a zigzag path exists between a pair of checkpoints, and that zigzag path is also a causal path, then the checkpoints are ordered. Because consistency requires checkpoints to be unordered, they cannot be consistent. However, even if the zigzag path is not a causal path, a consistent snapshot cannot be formed that contains both checkpoints. The zigzag nature of the path causes any snapshot that includes the two checkpoints to be inconsistent. To visualize the effect of a zigzag path, consider any *snapshot line* (a line drawn through a set of checkpoints) through the two checkpoints. Figure 2 shows two snapshot lines through $C_{1,1}$ and $C_{3,2}$. Because of the zigzag between the two checkpoints, such a line always crosses at least one message in the zigzag path, and this message causes one of the checkpoints in the snapshot to happen before another. In Figure 2, the zigzag path from $C_{1,1}$ to $C_{3,2}$ renders any snapshot line through these checkpoints inconsistent (shown by the dashed lines in Figure 2).

Conversely, if no zigzag path exists between two checkpoints (including a zigzag path from a checkpoint to itself), then it is *always* possible to construct a consistent snapshot containing them. By including the first check-

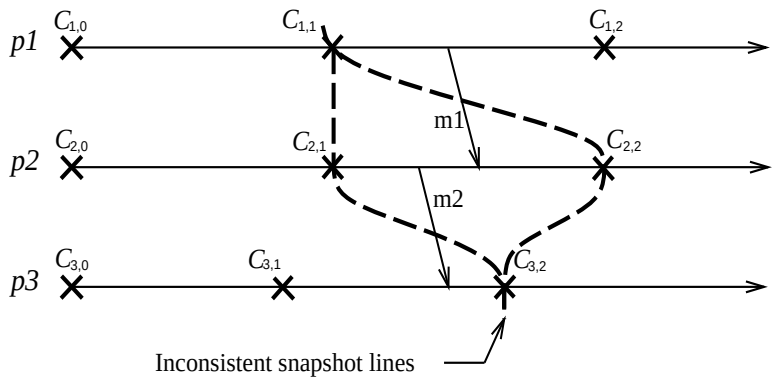


**Figure 2. The zigzag path from $C_{1,1}$ to $C_{3,2}$ renders any snapshot line through $C_{1,1}$ and $C_{3,2}$ inconsistent. A message that orders the checkpoints crosses any snapshot line, making the snapshot inconsistent.**

point in each process that has no zigzag path to either checkpoint, a consistent snapshot is formed. The snapshot is guaranteed to be consistent since none of its component checkpoints can happen before one another (otherwise, at least one of its checkpoints has a zigzag path to one of the two checkpoints). The following theorem proves these observations.

*Theorem 1 (Consistency Theorem)*

A set of checkpoints, $S$, from different processes can belong to the same consistent global snapshot iff no checkpoint in $S$ has a zigzag path to any other checkpoint (including itself) in $S$.

*Proof.*

*If Part.* We will show that the absence of the zigzag paths implies that checkpoints in $S$ can belong to the same consistent checkpoint. We construct a consistent global snapshot $G$ that contains the checkpoints in $S$ and one checkpoint in each other process as follows:

(1) For each process that has no checkpoint in $S$ but that has a zigzag path to some checkpoint in $S$, we include in $G$ its first checkpoint that has no zigzag path to any checkpoint in $S$.

(2) For each other process that has no checkpoint in $S$, we include its initial checkpoint in $G$. Note that no other checkpoint can happen before this initial checkpoint.

We claim that all checkpoints in $G$ are mutually unordered (so $G$ is consistent). To establish a contradiction, assume that one checkpoint happens before another.

*Case 1.* $C_{p,i} \in S$ happens before $C_{q,j} \in S$. Since $C_{p,i}$ happens before $C_{q,j}$, a zigzag path also exists, contradicting the assumption that no zigzag path exists between any two checkpoints in $S$.

*Case 2.* $C_{p,i} \in S$ happens before $C_{q,j} \in G - S$. A zigzag path must exist from $C_{q,j-1}$ to $C_{r,k}$ for some $C_{r,k} \in S$. This zigzag path, plus the messages that cause $C_{p,i}$ to happen before $C_{q,j}$, establish a zigzag path from $C_{p,i}$ to $C_{r,k}$, contradicting the assumption.

*Case 3.* $C_{p,i} \in G - S$ happens before $C_{q,j} \in S$. Since $C_{p,i}$ happens before $C_{q,j}$, a zigzag path also exists, contradicting the way $G$ was constructed.

*Case 4.* $C_{p,i} \in G - S$ happens before $C_{q,j} \in G - S$. A zigzag path must exist from $C_{q,j-1}$ to $C_{r,k}$ for some $C_{r,k} \in S$. As in case 2, this zigzag path, plus the messages that cause $C_{p,i}$ to happen before $C_{q,j}$, also establish a zigzag path from $C_{p,i}$ to $C_{r,k}$. But this contradicts the assumption that $C_{p,i}$ was the first checkpoint in $p$ with no zigzag path to any checkpoint in $S$.

*Only-If Part.* We will show that if a zigzag path exists between any two checkpoints in $S$, then they cannot belong to the same consistent snapshot. We first prove the case where there is a zigzag path from $C_{p,i}$ to $C_{q,j}$, and then consider the case of a zigzag cycle from some checkpoint $C_{p,i}$ to itself.

Assume that a zigzag path exists from $C_{p,i}$ to $C_{q,j}$. We use induction on the number $n$ of messages comprising the path to show that $C_{p,i}$ and $C_{q,j}$ cannot belong to the same consistent snapshot.

*Basis* ($n = 1$). A zigzag path consisting of one message is also a causal path, so in this case $C_{p,i} \xrightarrow{\text{HB}} C_{q,j}$ , implying that they cannot belong to the same consistent shapshot.

*Induction.* Assume that a zigzag path of $n$ messages $m_1 \cdots m_n$ from $C_{p,i}$ to $C_{r,k}$ implies that $C_{p,i}$ and $C_{r,k}$ cannot belong to the same consistent snapshot. We will show that a zigzag path of $n+1$ messages $m_1 \cdots m_n m_{n+1}$ from $C_{p,i}$ to $C_{q,j}$ implies that $C_{p,i}$ and $C_{q,j}$ cannot belong to the same consistent snapshot.

To establish a contradiction, assume that $C_{p,i}$ and $C_{q,j}$ can belong to the same consistent snapshot, and that message $m_{n+1}$ is sent by process $r$ during its checkpoint interval $S_{r,k}$. By the definition of zigzag path, $m_n$ must be received in the same or previous checkpoint interval as $m_{n+1}$ is sent. Thus messages $m_1 \cdots m_n$ constitute a zigzag path from $C_{p,i}$ to $C_{r,k+1}$. According to the hypothesis, $C_{p,i}$ cannot be in the same consistent snapshot with $C_{r,k'}$ for all $k' \geq k+1$. In addition, because message $m_{n+1}$ is sent after $C_{r,k}$ and received before $C_{q,j}$, $C_{r,k}$ and its preceding checkpoints are all ordered before $C_{q,j}$. Thus $C_{q,j}$ cannot be in the same consistent snapshot with $C_{r,k'}$ for all $k' \leq k$. Therefore, no checkpoint in process $r$ can be combined with both $C_{p,i}$ and $C_{q,j}$ to form a consistent snapshot.

We now show that a zigzag path from $C_{p,i}$ to itself means that $C_{p,i}$ cannot belong to any consistent snapshot, and thus checkpoints in $S$ cannot belong to the same consistent snapshot. To establish a contradiction, assume that $C_{p,i}$ belongs to a consistent snapshot, $G$, but there exists a zigzag path from $C_{p,i}$ to itself consisting of messages $m_1 \cdots m_n$ (note that $n \geq 2$ because a single message cannot constitute a zigzag cycle). We show that the zigzag cycle implies $G$ cannot be consistent.

Suppose message $m_n$ is sent by process $q$ during its checkpoint interval $S_{q,j}$. Message $m_n$ causes $C_{q,j}$ and all the preceding checkpoints in process $q$ to be ordered before $C_{p,i}$, so $C_{q,j}$ or any preceding checkpoint cannot belong to $G$. However, according to the proof of the previous case, the zigzag path consisting of messages $m_1 \cdots m_{n-1}$ from $C_{p,i}$ to $C_{q,j+1}$ implies that $C_{q,j+1}$ or any subsequent checkpoint in $q$ cannot belong to $G$. Thus, no checkpoint in $q$ can belong to $G$, so $G$ cannot be consistent. QED

### 3.3. Examples

The Consistency Theorem shows that the zigzag notion captures exactly the conditions for an arbitrary set of checkpoints, $S$, to all belong together in the same consistent snapshot. Two interesting special cases include the conditions for an arbitrary checkpoint to be useful in the sense that some consistent snapshot exists to which it belongs (i.e., $|S| = 1$), and the conditions for two arbitrary checkpoints to both belong to the same consistent snapshot ($|S| = 2$). In this section, we present these special cases as corollaries, and give examples to illustrate them.

*Corollary 1*

Checkpoint $C_{p,i}$ can belong to a consistent snapshot iff $C_{p,i}$ is involved in no zigzag cycle.

In Figure 3, there is a zigzag path from $C_{2,2}$ to itself (formed by messages $m4$ and $m3$). This zigzag cycle implies that $C_{2,2}$ cannot belong to *any* consistent shapshot. Because of $m3$ (the message that completes the zigzag cycle), $C_{2,2}$ is inconsistent with $C_{1,1}$ or any preceding checkpoint in process $p1$. Similarly, the Consistency Theorem shows that the zigzag path from $C_{2,2}$ to $C_{1,2}$ (consisting only of message $m4$) means $C_{2,2}$ cannot be in any consistent shapshot with $C_{1,2}$ or any subsequent checkpoint in $p1$. Thus, $C_{2,2}$ cannot be combined with any checkpoint in $p1$ to form a consistent shapshot. Conversely, as illustrated below, if a checkpoint is involved in no

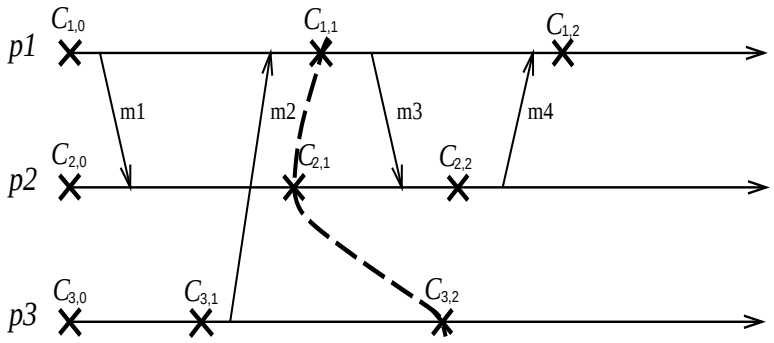


**Figure 3. A zigzag cycle from $C_{2,2}$ back to itself, and a consistent snapshot including $C_{1,1}$ and $C_{3,2}$ (between which no zigzag path exists).**

zigzag path, then it is always possible to include the checkpoint in a consistent snapshot.

*Corollary 2*

Checkpoints $C_{p,i}$ and $C_{q,j}$ ($p \neq q$) can belong to the same consistent global checkpoint iff

(1) no zigzag cycle involving $C_{p,i}$ or $C_{q,j}$ exists, and

(2) no zigzag path exists between $C_{p,i}$ and $C_{q,j}$.

Corollary 2 states that, for two checkpoints to belong together in the same consistent snapshot, no zigzag path can exist between the checkpoints, including a zigzag path from either checkpoint to itself (otherwise, by Corollary 1, they could not belong to any consistent snapshot). In Figure 3, there is no zigzag path between $C_{1,1}$ and $C_{2,1}$, and neither is involved in a zigzag cycle. Thus, a consistent snapshot exists in which both $C_{1,1}$ and $C_{2,1}$ appear. We can construct such a snapshot, $G$, by including the first checkpoint in process $p3$ that has no zigzag path to either $C_{1,1}$ or $C_{2,1}$. $C_{3,2}$ is such a checkpoint, since $C_{3,1}$ is the last checkpoint in $p3$ that has a zigzag path to either $C_{1,1}$ (formed by $m2$) or $C_{2,1}$ (formed by $m2$ and $m1$). Note that a zigzag path is formed from $C_{3,1}$ to $C_{2,1}$ by messages $m2$ and $m1$, since $m2$ is received in the same checkpoint interval as $m1$ is sent. Thus, $G = \{C_{1,1}, C_{2,1}, C_{3,2}\}$, shown by the dotted line in Figure 3. Several observations show that all checkpoints in $G$ are mutually unordered and thus $G$ is consistent. First, no zigzag paths between $C_{1,1}$ and $C_{2,1}$ means that neither happens before the other. Second, no zigzag path from $C_{3,2}$ to either $C_{1,1}$ or $C_{2,1}$ ensures that $C_{3,2}$ does not happen before either of these checkpoints. Finally, neither $C_{1,1}$ or $C_{2,1}$ can happen before $C_{3,2}$; otherwise, a zigzag path would exist from one of them to the other, contradicting the assumption. For example, if $C_{1,1}$ happened before $C_{3,2}$, then the messages comprising this causal path, plus message $m2$, would form a zigzag cycle from $C_{1,1}$ back to itself.

## 4. Conclusion

We have shown that the zigzag notion, as a generalization of Lamport's happened-before relation, captures exactly the conditions for a set of checkpoints to belong to the same consistent global checkpoint. We proved in this paper that a set of checkpoints can belong to the same consistent global checkpoint iff no zigzag path exists from a checkpoint to any other.

The results can be used in applications that requires saving or analyzing consistent checkpoints. We have shown in a previous paper one application of our results to reducing rollback propagation for independent checkpointing. We envision the results can also be used in other applications, such as analyzing global states, and reducing space overhead for message logs and checkpoints.

**5. Acknowledgements**

We thank Yi-Min Wang (Bell Labs) for discussions about the results.

**References**

[1] B. Bhargava and S. R. Lian, ‘‘Independent checkpointing and concurrent rollback for recovery - An optimistic approach,’’ *Proc. IEEE Symp. on Reliable Distr. System*, pp. 3-12 (1988).

[2] K. M. Chandy and C. V. Ramamoorthy, ‘‘Rollback and recovery strategies for computer programs,’’ *IEEE Trans. on Computers* **21** pp. 546-556 (June 1972).

[3] D. R. Jefferson, ‘‘Virtual Time,’’ *ACM Trans. on Programming Languages and Systems*, **7**(3) pp. 404-425 (July 1985).

[4] R. Koo and S. Toueg, ‘‘Checkpointing and rollback-recovery for distributed systems,’’ *IEEE Trans. on Software Engineering* **SE-13**(1) pp. 23-31 (January 1987).

[5] L. Lamport, ‘‘Time, Clocks, and the Ordering of Events in a Distributed System,’’ *Comm. of the ACM* **21**(7) pp. 558-565 (July 1978).

[6] R.H.B. Netzer and J. Xu, ‘‘Adaptive Message Logging for Incremental Replay of Message-Passing Programs,’’ *To appear in Supercomputing ’93*, Portland, OR, (November 1993).

[7] B. Randell, ‘‘System Structure for Software Fault Tolerance,’’ *IEEE Trans. on Software Engineering* **SE-1**(2) pp. 220-232 (June 1975).

[8] Y. M. Wang, A. Lowry, and W. K. Fuchs, ‘‘Consistent Global Checkpoints Based on Direct Dependency Tracking,’’ *Research Report RC 18465, IBM T.J. Watson Research Center, Yorktown Heights, New York*, (Oct. 1992).

[9] L. D. Wittie, ‘‘Debugging Distributed C Programs by Real Time Replay,’’ *SIGPLAN/SIGOPS Workshop on Parallel and Distributed Debugging*, pp. 57-67 Madison, WI, (May 1988).

[10] J. Xu and R.H.B. Netzer, ‘‘Adaptive Independent Checkpointing for Reducing Rollback Propagation,’’ *Brown University Computer Science Dept. Technical Report CS-93-25*, (May 1993).