Consistency Issues in Distributed Checkpoints

Jean-Michel Hélary, Robert H.B. Netzer, *Member*, *IEEE Computer Society*, and Michel Raynal

Abstract—A *global checkpoint* is a set of local checkpoints, one per process. The traditional *consistency* criterion for global checkpoints states that a global checkpoint is consistent if it does not include messages received and not sent. This paper investigates other consistency criteria, *transitlessness,* and *strong consistency*. A global checkpoint is transitless if it does not exhibit messages sent and not received. Transitlessness can be seen as a dual of traditional consistency. Strong consistency is the addition of transitlessness to traditional consistency. The main result of this paper is a statement of the necessary and sufficient condition answering the following question: "*Given an arbitrary set of local checkpoints, can this set be extended to a global checkpoint that satisfies* $\mathcal{P}$" (where $\mathcal{P}$ is traditional consistency, transitlessness, or strong consistency). From a practical point of view, this condition, when applied to transitlessness, is particularly interesting as it helps characterize which messages do not need to be recorded by checkpointing protocols.

Index Terms—Checkpointing, consistency, strong consistency, transitlessness, distributed systems, fault-tolerance, rollback recovery.

1 Introduction

In a distributed system, a *local checkpoint* is a local state of a process and a *global checkpoint* is a set of local checkpoints, one from each process. The usual notion of *consistency* for global checkpoints stipulates that a message sent by a process after its local checkpoint A must not be received by its destination process before its local checkpoint B, for the global checkpoint including A and B to be consistent. A consistent global checkpoint is thus a snapshot of a state of the system that the actual execution passed through or could have passed through. The determination of consistent global checkpoints has many applications such as the detection of (stable or unstable) distributed properties [1], [3], [6], [10], the determination of distributed breakpoints [5], [12], and rollback-recovery [4] to name a few. Consequently, the determination of consistent global checkpoints has received wide attention.

Recently, the following problem has been addressed [14], [19]:

> *"Given a set of local checkpoints, can this set be extended to form a consistent global checkpoint?"*

This simple question is actually at the heart of many checkpointing problems. It has been shown that two local checkpoints being causally unrelated is a necessary but not sufficient condition for them to belong to the same consistent global checkpoint. Two local checkpoints can have a type of "hidden" dependence that prevents them from ever belonging to the same consistent global checkpoint. This problem was first addressed by Netzer and Xu [14] who introduced the notion of a *Z-path* between local checkpoints to capture both their causal and hidden dependencies. Building on their work, Wang [19] studies two canonical consistent global checkpoints to which a set of local checkpoints can belong (the minimum and maximum) and introduces the Rollback-Dependency Trackability (RDT) property which states that all dependencies among local checkpoints are trackable on-line. Baldoni, Hélary, Mostefaoui, and Raynal [2] present a protocol ensuring that local checkpoints taken during a computation satisfy the RDT property. The interested reader will find in [19] a set of problems whose solutions greatly benefit from the RDT theory (namely, software error recovery, guaranteed deadlock recovery, global checkpoints with mobile hosts, causal distributed breakpoints, and output commit).

A thorough investigation of consistency issues of global checkpoints shows that the usual consistency criterion described above represents only one side of the problem. This paper is devoted to such an investigation and studies two other consistency criteria for global checkpoints. The first one, called *transitlessness*, focuses on global checkpoints in which no messages are sent and not received. The second one, called *strong consistency*, is the addition of usual consistency and transitlessness. Definitions for these consistency criteria are provided and the following question is answered:

> *"Given a set of local checkpoints, can this set be extended to a global checkpoint that satisfies the consistency criterion* $\mathcal{P}$*?"*

(where $\mathcal{P}$ is traditional consistency, transitlessness, or strong consistency).

The approach used in this paper is based on a directed graph that represents the execution. Lamport has shown that a computation can be modeled as a graph representing the *happen-before* relation of potential causality among the computation's states and events [9]. We show how this graph can be transformed to produce what we call the $\mathcal{Z}$-graph, whose paths correspond exactly to Z-paths. It also appears that traditional consistency and transitlessness are duals of each other. Another transformation of the graph exposes this duality and leads to what we call the $\mathcal{T}$-graph, and the paths in the $\mathcal{T}$-graph are called *T-paths*. T-paths are

- *J.-M. Hélary and M. Raynal are with IRISA, Université de Rennes, Campus de Beaulieu, 35042 Rennes Cedex, France. E-mail: {helary, raynal}@irisa.fr.*
- *R.H.B. Netzer is with the Computer Science Department, Brown University, Providence, RI 02921. E-mail: rn@cs.brown.edu.*

Manuscript received 9 May 1997; revised 4 Feb. 1998.
Recommended for acceptance by W.H. Sanders.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number 105038.

duals of Z-paths. The $\mathcal{T}$-graph is obtained from $\mathcal{Z}$-graph by exchanging the send and receive events associated with each message. Finally, we define the $\mathcal{S}$-graph of an execution to be the union of the corresponding $\mathcal{Z}$-graph and $\mathcal{T}$-graph. These three graphs are particular instances of a more general graph defined over the set of communication events and checkpoints, and the three consistency criterion are particular instances of a generic consistency criterion which we call *acceptability*.

With this approach, a general necessary and sufficient condition for a set of local checkpoints to belong to a global checkpoint satisfying *acceptability* can be formulated in terms of paths of the generic graph.[1] We show that the necessary and sufficient conditions for a set of local checkpoints to belong to a global checkpoint satisfying traditional consistency, transitlessness, or strong consistency, are obtained as particular instances of the generic condition. This important result is thus filling a logical "hole" in the current state of knowledge about global states.

Albeit this paper is centered on a theoretical result, there are practical applications and implementations of this result. From a practical point of view, the necessary and sufficient condition characterizing transitlessness is particularly interesting as it shows when message recording can be avoided during checkpointing. Without such a condition, checkpointing protocols are required to save messages that could be in transit with respect to some global checkpoint [13], [17]. Moreover, among the applications that can benefit from transitlessness or strong consistency, replay mechanisms for distributed executions are particularly noticeable: Message recording can be reduced and, consequently, efficiency of replay mechanisms can be increased [15]. From an implementation point of view, Hélary, Mostefaoui, and Raynal [7] have used this condition to design a protocol ensuring that every local checkpoint belongs to a *strongly consistent* global checkpoint. This protocol belongs to the category of communication-induced protocols, i.e., 1) processes take basic local checkpoints at their own pace and 2) the protocol decides *on-the-fly*, upon each receipt of message, whether it has to force processes to take additional local checkpoints or to record the message in order to ensure strong consistency. At the operational level the protocol uses a timestamping mechanism and adds only one integer to each application message. Note that this approach is quite distinct from those based on explicit coordination, where protocols require additional control messages [3].

This paper is composed of seven sections. Section 2 defines our model of distributed computation and checkpointing. Section 3 provides definitions for traditional consistency, transitlessness, and strong consistency of global checkpoints. Section 4 first associates a generic graph with a communication and checkpoint pattern, then defines the *acceptability* abstract consistency criterion. Then, Section 5 states and proves a necessary and sufficient condition for a set of local checkpoints to be extended to an acceptable global checkpoint and shows how acceptability can be instantiated to get traditional consistency, transitlessness and strong consistency. Section 6 discusses the mathematical structure of acceptable global checkpoints. Finally, Section 7 shortly concludes the paper.

1. Netzer and Xu [14] and Wang [19] state and prove a necessary and sufficient condition for a set of local checkpoints to belong to a consistent global checkpoint. This paper can be seen as an extension of their work to other consistency criteria for global checkpoints.

2 Distributed Computations and Checkpoints

2.1 Distributed Computations

A distributed computation consists of a finite set P of n processes $\{P_1, P_2, \ldots, P_n\}$ that communicate and synchronize only by exchanging messages. We assume that each ordered pair of processes is connected by an asynchronous directed logical channel whose transmission delays are unpredictable but finite. Note that channels are not required to be FIFO. Each process runs on a processor; processors do not share memory; there is no bound on their relative execution speeds.

A process can execute internal and communication (send[2] and delivery) statements. An internal statement does not involve 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 *"deliver(m),"* it is blocked until at least one message directed to P_i has arrived; then a message is withdrawn from one of its input channels and delivered to P_i. Executions of internal, send, and delivery statements are modeled by internal, send, and deliver events.

Processes of a distributed computation are *sequential*, in other words, each process P_i produces a *sequence* of events $e_{i,1} \ldots e_{i,s} \ldots$ This sequence can be finite or infinite. Every process P_i has an initial local state denoted $\sigma_{i,0}$. The local state $\sigma_{i,s}$ $(s > 0)$ results from the execution of the sequence $e_{i,1} \ldots e_{i,s}$ applied to the initial state $\sigma_{i,0}$. More precisely, the event $e_{i,s}$ moves P_i from the local state $\sigma_{i,s-1}$ to the local state $\sigma_{i,s}$. By definition, we say that "$e_{i,x}$ *belongs to* $\sigma_{j,s}$" (denoted $e_{i,x} \in \sigma_{j,s}$) if $i = j$ and $x \leq s$.

Let H be the set of all the events produced by a distributed computation. This computation is modeled by the directed acyclic graph $\widehat{H} = (H, \overset{hb}{\rightarrow})$, where $\overset{hb}{\rightarrow}$ denotes the well-known Lamport's *happened-before* relation [9]. More precisely, $\overset{hb}{\rightarrow}$ is the relation defined as:

$$e_{i,s} \overset{hb}{\rightarrow} e_{j,t} \Leftrightarrow \begin{array}{l}(i = j \wedge t = s + 1) \vee \\ (e_{i,s} = send(m) \wedge e_{j,t} = deliver(m)) \vee \\ (\exists e_{k,v} : e_{i,s} \overset{hb}{\rightarrow} e_{k,v} \wedge e_{k,v} \overset{hb}{\rightarrow} e_{j,t})\end{array}$$

As internal events are not relevant for our study, we consider in the following the subgraph $\widehat{H}_{com}$ of $\widehat{H}$ from which internal events have been suppressed. H_{com} denotes the set of communication events.

2.2 Local and Global Checkpoints

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

2. We assume a process never sends messages to itself.

DEFINITION 2.1. *A communication and checkpoint pattern (CCP) is a pair* $(\widehat{H}_{com}, C_{\widehat{H}})$ *where* $\widehat{H}_{com}$ *is a distributed computation and* $C_{\widehat{H}}$ *is a set of local checkpoints defined on* $\widehat{H}$.

$C_{i,x}$ represents the xth local checkpoint of process P_i. The local checkpoint $C_{i,x}$ corresponds to some local state $\sigma_{i,s}$ with $x \leq s$. Fig. 1 shows an example of checkpoint and communication pattern.[3] We assume that each process P_i takes an initial local checkpoint $C_{i,0}$ (corresponding to $\sigma_{i,0}$), and after each event a checkpoint will eventually be taken. Thus, each process always begins, and ends, with a checkpoint.

Global checkpoints. A *global checkpoint* is a set of local checkpoints, one from each process. For example, $\{C_{k,1}, C_{j,1}, C_{i,1}\}$ and $\{C_{k,1}, C_{j,2}, C_{i,2}\}$ are two global checkpoints depicted in Fig. 1.

3 CONSISTENT, TRANSITLESS, AND STRONGLY CONSISTENT GLOBAL CHECKPOINTS

3.1 Consistent Global Checkpoints

Informally, a consistent global checkpoint is a global checkpoint that the execution either passed through or had the potential of passing through due to variations in message transfer delays and process speeds [3]. A consistent global checkpoint cannot exhibit messages received (from their receiver's point-of-view) but not yet sent (from their sender's point-of-view). Note that a consistent global checkpoint can exhibit messages sent but not received. We introduce the notion of an *orphan* message to formally define consistency.

A message m sent by a process P_i to a process P_j is called *orphan* with respect to the ordered pair of local checkpoints $(C_{i,x}, C_{j,y})$ iff the delivery of m belongs to $C_{j,y}$ ($deliver(m) \in C_{j,y}$) while its sending event does not belong to $C_{i,x}$ ($send(m) \notin C_{i,x}$). An ordered pair of local checkpoints is *consistent* if and only if there are no orphan messages with respect to this pair. For example, Fig. 1 shows the pair $(C_{k,1}, C_{j,1})$ is consistent, while the pair $(C_{i,2}, C_{j,2})$ is inconsistent (because of the orphan message m_5).

DEFINITION 3.1. *A global checkpoint is* consistent *if all its pairs of local checkpoints are consistent.*

In Fig. 1, the global checkpoint $(C_{k,1}, C_{j,1}, C_{i,2})$ is not consistent because of the orphan message m_2, while the global checkpoint $(C_{k,1}, C_{j,1}, C_{i,1})$ is consistent.

3.2 Transitless Global Checkpoints

Informally, a *transitless* global checkpoint does not exhibit messages as being sent but not yet received. The transitlessness criterion is a dual to the traditional notion of consistency. However, it may present messages received but not yet sent, depending on whether the transitless global checkpoint happens to be consistent. Determining the conditions under which transitless global checkpoints can exist is important. Such a condition would show precisely the context in which message recording can be avoided.[4] To our knowledge, such a condition has never been stated before. Actually, let us note that the recording of messages ensures a "virtual transitlessness": As soon as a message is recorded, it can be viewed as no longer preventing transitlessness as it can be retrieved if it must be restored.[5]

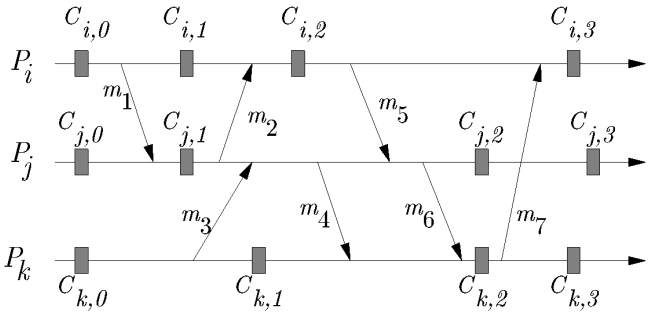


Fig 1. A checkpoint and communication pattern.

More formally, a message m is *in-transit* with respect to an ordered pair of local checkpoints $(C_{i,x}, C_{j,y})$ if $send(m) \in C_{i,x}$ and $deliver(m) \notin C_{j,y}$ (i.e., $send(m)$ occurs before $C_{i,x}$ and $deliver(m)$ occurs after $C_{j,y}$). The ordered pair of checkpoints $(C_{i,x}, C_{j,y})$ is *transitless* if it has no in-transit messages.

DEFINITION 3.2. *A global checkpoint is* transitless *if all its pairs of local checkpoints are transitless.*

For example, in Fig. 1, the global checkpoints $(C_{k,0}, C_{j,1}, C_{i,2})$ and $(C_{k,2}, C_{j,3}, C_{i,2})$ are transitless (but not consistent). $(C_{k,0}, C_{j,1}, C_{i,1})$ is both transitless and consistent. Due to the in-transit message m_1 the consistent global checkpoint $(C_{k,0}, C_{j,0}, C_{i,1})$ is not transitless. Finally, the global checkpoint $(C_{k,1}, C_{j,1}, C_{i,2})$ is neither transitless (m_3 is in-transit) nor consistent (m_2 is orphan).

3.3 Strongly Consistent Global Checkpoints

A strongly consistent global checkpoint is a global checkpoint that is both transitless and consistent. More formally:

DEFINITION 3.3. *A global checkpoint* is strongly consistent *if all its pairs of local checkpoints are consistent and transitless.*

In the running example, the global checkpoint $(C_{k,3}, C_{j,2}, C_{i,3})$ is strongly consistent.

4 A GENERIC APPROACH

This section presents a unique framework able to express the previous consistency criteria (traditional consistency, transitlessness, and strong consistency). Section 4.1 associates a generic graph with any communication and checkpoint pattern, and defines an abstract consistency criterion for global checkpoints called *acceptability*. Then, Section 4.2

3. This figure. uses the usual space-time diagram. Local checkpoints are indicated by black rectangular boxes; the other local states are not explicitly indicated.

4. Such a characterization is exact for software error recovery applications, where every consistent global checkpoint can potentially be used as a recovery line. In the case of hardware failure recovery applications, only the *maximum* consistent global check point can be used as the recovery line. For this case, an exact characterization of which message logs are useful can be found in [18].

5. A similar notion (namely *temporary dependency edges*) has been introduced in [17].

shows how this generic graph must be instantiated to obtain a graph on which a particular consistency criterion can be evaluated. In Section 5, this generic graph is used to state and prove a necessary and sufficient condition for a global checkpoint to be acceptable and then provide instantiations of the generic condition for the three consistency criteria defined above.

4.1 A Generic Graph

A directed graph $G = (V, E)$ is associated with a CCP $(\widehat{H}_{comm}, C_{\widehat{H}})$ in the following way.

- The set V of its vertices is composed of the set of communication events H_{com} and the set of local checkpoints $C_{\widehat{H}}$.
- The set E of edges is the following. An edge is either a *local* edge (denoted *l-edge*) or a *communication* edge (denoted *c-edge*).
  - l-edges:
    - If e_1 and e_2 are two consecutive communication events of a same process P_i, then both $(e_1, e_2) \in E$ and $(e_2, e_1) \in E$.
    - If C is a checkpoint of P_i and e is the first communication event of P_i that follows C, then $(C, e) \in E$.
    - If C is a a checkpoint of P_i and e is the last communication event of P_i that precedes C, then $(e, C) \in E$.
    - If C_1 and C_2 are two consecutive local checkpoints of a process P_i (with C_1 being the first) and there is no communication event between them, then $(C_1, C_2) \in E$.
  - c-edges:
    - A c-edge (e_1, e_2) is such that e_1 and e_2 are two communication events that belong to different processes and concern the same message. We show below that different definitions of c-edges correspond to different message properties (orphan or in-transit) and lead to different instantiations of this generic graph.

Notation. A path in the graph G from a vertex x to a vertex y is denoted $x \to y$. Let Σ and Σ' be two sets of vertices. Then, $\Sigma \to \Sigma'$ means that $\exists x \in \Sigma$ and $\exists x' \in \Sigma'$ such that $x \to x'$. If $\Sigma = \{x\}$ (respectively, $\Sigma' = \{x'\}$) then we will use the notation $x \to \Sigma'$ (respectively, $\Sigma \to x'$) instead of $\{x\} \to \Sigma'$ (respectively, $\Sigma \to \{x'\}$).

Consider a communication event e issued by a process P_i. *e.prec* (respectively, *e.succ*) denotes the last (respectively, first) checkpoint of P_i that precedes e (respectively, follows e).

Acceptability

DEFINITION 4.1. *Let (C_I, C_j) be an ordered pair of checkpoints, C_i belonging to P_i and C_j belonging to P_j, with $i \neq j$. The ordered pair (C_i, C_j) is acceptable if there is no c-edge (e_1, e_2) with e_1 issued by P_i, e_2 issued by P_j, $C_i \to e_1$, and $e_2 \to C_j$.*

DEFINITION 4.2. *Let $\Sigma = \{C_1, \cdots, C_n\}$ be a global checkpoint. Σ is acceptable if all ordered pairs (C_i, C_j) are acceptable.*

TABLE 1
INSTANTIATIONS OF THE GENERIC GRAPH

Graph	Generic G	$\mathcal{Z}(G)$	$\mathcal{T}(G)$	$\mathcal{S}(G)$
property	*acceptable*	*consistent*	*transitless*	*strongly consistent*
path	$\to$ (*path*)	$\overset{Z}{\to}$ (Z-path)	$\overset{T}{\to}$ (T-path)	$\overset{S}{\to}$ (S-path)

4.2 Instantiations

This section presents three instantiations of the graph G, namely $\mathcal{Z}(G)$, $\mathcal{T}(G)$, and $\mathcal{S}(G)$, on which the generic necessary and sufficient conditions, stated and proved in Section 5, will be instantiated to obtain necessary and sufficient conditions related to consistency, transitlessness and strong consistency, respectively, as summarized in Table 1.

Consistency. The instantiation $\mathcal{Z}(G)$ of the graph G is obtained in the following way:

$$((e_1, e_2) \text{ is a c-edge} \Leftrightarrow (e_1 = \textit{send}(m) \wedge e_2 = \textit{deliver}(m))$$

Let P_i and P_j be the processes that issued the events e_1 and e_2, respectively. Moreover, let C_i be a checkpoint of P_i taken before e_1 and C_j a checkpoint of P_j taken after e_2. The previous c-edge means that the message m is *orphan* with respect to the ordered pair (C_i, C_j). So, when we consider the instantiated graph $\mathcal{Z}(G)$, *acceptable* means *consistent*. Fig. 2 shows the $\mathcal{Z}(G)$ graph associated with the CCP depicted in Fig. 1. For example, message m_7 is orphan with respect to $(C_{k,2}, C_{i,3})$ (there is a path from $C_{k,2}$ to $C_{i,3}$).

Transitlessness. The instantiation $\mathcal{T}(G)$ of the graph G is obtained in the following way:

$$((e_1, e_2) \text{ is a c-edge} \Leftrightarrow (e_1 = \textit{deliver}(m) \wedge e_2 = \textit{send}(m))$$

Let P_i and P_j be the processes that issued the events e_1 and e_2, respectively. Moreover, let C_i be a checkpoint of P_i taken before e_1 and C_j a checkpoint of P_j taken after e_2. The previous c-edge means that the message m is *in-transit* with respect to the ordered pair (C_i, C_j). So, when we consider the instantiated graph $\mathcal{T}(G)$, *acceptable* means *transitless*. Fig. 3 shows the $\mathcal{T}(G)$ graph associated with the CCP depicted Fig. 1. For example, message m_3 is in-transit with respect to $(C_{j,1}, C_{k,1})$ (there is a path from $C_{j,1}$ to $C_{k,1}$).

Strong Consistency. The instantiation $\mathcal{S}(G)$ of the graph G is obtained in the following way:

$$((e_1, e_2) \text{ is a c-edge} \Leftrightarrow ((e_1 = \textit{send}(m) \wedge e_2 = \textit{deliver}(m))$$
$$\vee (e_1 = \textit{deliver}(m) \wedge e_2 = \textit{send}(m))$$

In other words, $\mathcal{S}(G)$ is the *union* of the two graphs $\mathcal{Z}(G)$ and $\mathcal{T}(G)$. So, when we consider the instantiated graph $\mathcal{S}(G)$, *acceptable* means *consistent* **and** *transitless*, i.e., *strongly consistent*. Fig. 4 shows the $\mathcal{S}(G)$ graph associated with the CCP depicted Fig. 1. For example, there is a path from $C_{k,1}$ to $C_{i,2}$, through the communication events *deliver*(m_4), *send*(m_4), *send*(m_2), and *deliver*(m_2).

5 Necessary and Sufficient Conditions

5.1 Conditions for Acceptability

This section addresses the generic problem: Given a set M of one or more local checkpoints, is it possible to extend M to a global checkpoint Σ that is acceptable? A necessary and sufficient condition is stated and proved.

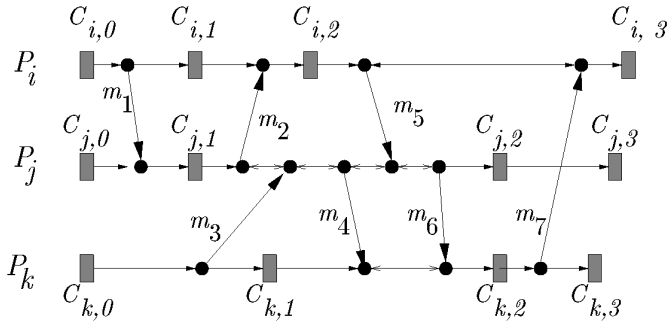


Fig. 2. $\mathcal{Z}(G)$ graph. o = communication events; [] = checkpoints.

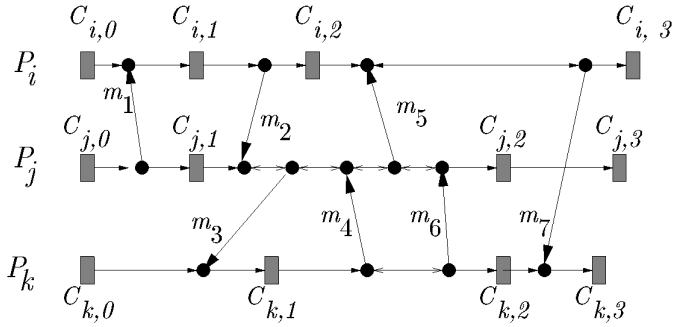


Fig. 3. $\mathcal{T}(G)$ graph. o = communication events; [] = checkpoints.

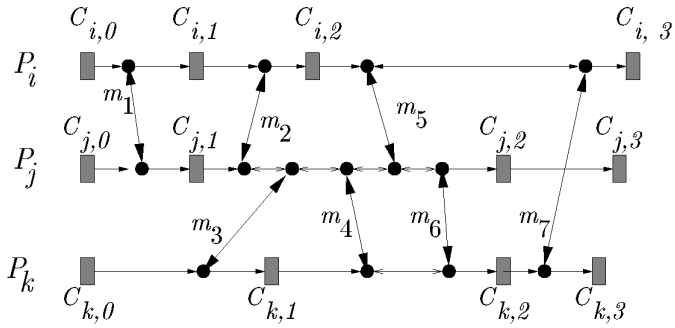


Fig. 4. $\mathcal{S}(G)$ graph. o = communication events; [] = checkpoints.

LEMMA 5.1. *A global checkpoint* Σ *is acceptable if and only if* $\neg(\Sigma \rightarrow \Sigma)$.

PROOF. For each process P_i, let $\Sigma(i)$ denote the local checkpoint of Σ taken by P_i. We will say that an event e of P_i is *before* (respectively, *after*) Σ if $e \rightarrow \Sigma(i)$ (respectively, $\Sigma(i) \rightarrow e$). Note that any event is necessarily before or after Σ. □

Necessity. Σ is acceptable $\Rightarrow \neg(\Sigma \rightarrow \Sigma)$. We prove that $(\Sigma \rightarrow \Sigma) \Rightarrow (\Sigma$ is *not* acceptable). An *internal* path is a path that is formed of only local edges.

If $(\Sigma \rightarrow \Sigma)$, it means that there exist i and j such that $\Sigma(i) \rightarrow \Sigma(j)$. Thus, there exists a path starting with an event e of P_i after Σ and ending with an event e' of P_j before Σ. Let e'' denote the *last* event of this path being after Σ (occurring on P_k) and e''' denote the event following e'' on the path, as depicted on Fig. 5. Since e is after Σ and e' is before Σ, e'', and e''' necessarily exist (maybe, $e = e''$ or $e''' = e'$ or both). Since e'' is after Σ and e''' is before Σ, there is no internal path from e'' to e''' and thus e''' may not occur on P_k. Thus, (e'', e''') is a c-edge with $\Sigma(k) \rightarrow e''$ and $e''' \rightarrow \Sigma(l)$. This shows that the ordered pair of local checkpoints $(\Sigma(k), \Sigma(l))$ is not acceptable and, consequently, the global checkpoint Σ is not acceptable.

Sufficiency. $\neg(\Sigma \rightarrow \Sigma) \Rightarrow (\Sigma$ is acceptable). We show that (Σ is *not* acceptable) $\Rightarrow (\Sigma \rightarrow \Sigma)$. If Σ is not acceptable, then there exist i and j such that the ordered pair of local checkpoints $(\Sigma(i), \Sigma(j))$ is not acceptable. Consequently, there exists a c-edge (e, e') where e is an event of P_i, e' is an event of P_j, $\Sigma(i) \rightarrow e$ and $e' \rightarrow \Sigma(j)$. Thus, by transitivity, $\Sigma(i) \rightarrow \Sigma(j)$ from which $\Sigma \rightarrow \Sigma$.

THEOREM 5.2. *Let* M *be a set of local checkpoints, each from a different process.* M *can be extended to an acceptable global checkpoint if and only if* $\neg(M \rightarrow M)$.

PROOF. □

Necessity. (There exists an acceptable global checkpoint Σ and $M \subseteq \Sigma) \Rightarrow \neg(M \rightarrow M)$. We prove that $(M \rightarrow M) \Rightarrow$ (no global checkpoint Σ with $M \subseteq \Sigma$ can be acceptable). In fact, $(M \rightarrow M)$ and $M \subseteq \Sigma$ implies that $\Sigma \rightarrow \Sigma$ and thus, from Lemma 5.1, Σ is not acceptable.

Sufficiency. $\neg(M \rightarrow M) \Rightarrow$ (there exists an acceptable global checkpoint Σ and $M \subseteq \Sigma$). We construct a global checkpoint Σ defined as follows. Note that this construction is structurally identical to Netzer and Xu's original proof [14] but it is applied with a general $\rightarrow$ relation. For each i:

1) If $M(i)$ exists then $\Sigma(i) = M(i)$.
2) If $M(i)$ does not exist then $\Sigma(i)$ is the first checkpoint taken by P_i such that $\neg(\Sigma(i) \rightarrow M)$. Note that $\Sigma(i)$ is always defined: If P_i has no event e such that $e \rightarrow M$, then $\Sigma(i) = C_{i,0}$ (the initial checkpoint on P_i). Otherwise, let e be the last event of P_i such that $e \rightarrow M$. Then $\Sigma(i) = e.succ$. Note that in the extreme case, $\Sigma(i)$ is the last checkpoint of P_i, beyond which no events exist.

By construction $M \subseteq \Sigma$. Now, we prove that $\neg(\Sigma \rightarrow \Sigma)$. Let $\Sigma(i)$ and $\Sigma(j)$ $(i \neq j)$ be two local checkpoints of Σ. We will show that $\forall i, \forall j, \neg(\Sigma(i) \rightarrow \Sigma(j))$. Four cases have to be considered:

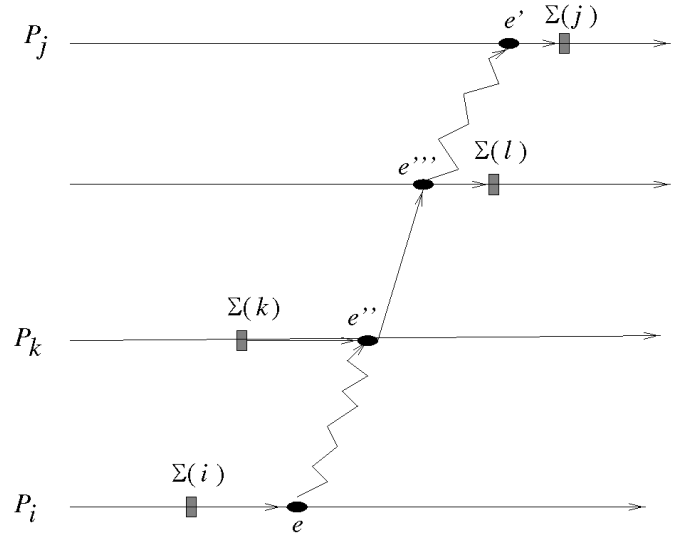


Fig. 5. Proof of Lemma 5.1.

1) $\Sigma(i) \in M$ and $\Sigma(j) \in M$. Then, by assumption, $\neg(\Sigma(i) \rightarrow \Sigma(j))$.
2) $\Sigma(i) \notin M$ and $\Sigma(j) \in M$. Then, by construction, $\neg(\Sigma(i) \rightarrow \Sigma(j))$.
3) $\Sigma(i) \in M$ and $\Sigma(j) \notin M$. If $\Sigma(j) = C_{j,0}$ then, by definition, $\neg(\Sigma(i) \rightarrow \Sigma(j))$. Otherwise, suppose that $\Sigma(i) \rightarrow \Sigma(j)$. It means that there exists a path starting with an event e of P_i and ending with an event e' of P_j, with $\Sigma(i) \rightarrow e$ and $e' \rightarrow \Sigma(j)$ (Fig. 6). Let $C_j = e'.prec$ (C_j exists since $\Sigma(j)$ is not P_j's initial checkpoint). Since C_j is taken before $\Sigma(j)$, by construction (due to point 2)), there exists k such that $\Sigma(k) \in M$ and $C_j \rightarrow \Sigma(k)$. Consequently, there exists a path starting with event e" of P_j and ending with event e''' of P_k with $C_j \rightarrow e$" and $e''' \rightarrow \Sigma(k)$. Since $C_j \rightarrow e$" and $C_j = e'.prec$ there is no checkpoint between the two events e' and e'' and thus there is an internal path from e' to e" (whether e' occurs before or after e''). Summarizing, we have: $\Sigma(i) \rightarrow e$ and $e \rightarrow e'$ and $e' \rightarrow e$" and e"$\rightarrow e'''$ and $e''' \rightarrow \Sigma(k)$, from which we have $\Sigma(i) \rightarrow \Sigma(k)$ with $\Sigma(i) \in M$ and $\Sigma(k) \in M$, contradicting the assumption.
4) $\Sigma(i) \notin M$ and $\Sigma(j) \notin M$. Suppose that $\Sigma(i) \rightarrow \Sigma(j)$. The situation is the same than the one depicted Fig. 6, except that $\Sigma(i) \notin M$. It has the same consequence, namely $\Sigma(i) \rightarrow \Sigma(k)$ with $\Sigma(i) \in M$ and $\Sigma(k) \in M$, contradicting the assumption.

COROLLARY 5.3. *Let C be a local checkpoint. Then C can be member of an acceptable global checkpoint if and only if $\neg(C \rightarrow C)$.*

PROOF. Obvious from Theorem 5.2 applied with $M = \{C\}$. □

5.2 Instantiating the Condition

The previous generic result can be easily instantiated to state the necessary and sufficient conditions for a global checkpoint to be consistent, transitless, and strongly consistent, respectively.

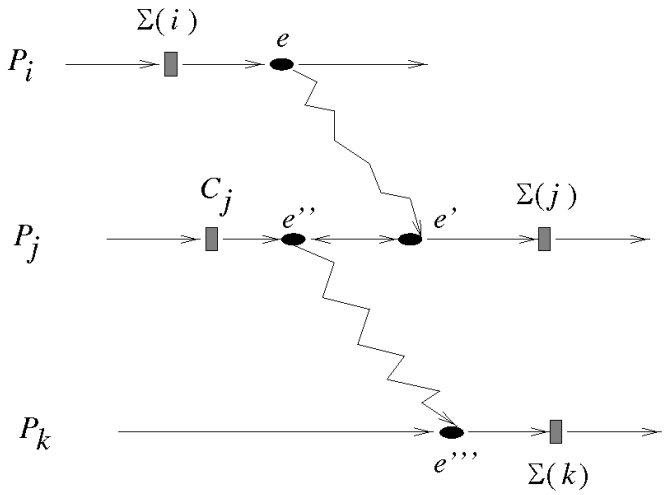


Fig. 6. Proof of Theorem 5.2.

5.2.1 Necessary and Sufficient Conditions for Consistent Checkpoints

LEMMA 5.4. *A global checkpoint Σ is consistent if and only if $\neg(\Sigma \xrightarrow{Z} \Sigma)$.*

THEOREM 5.5. *Let M be a set of local checkpoints from different processes. M can be extended to a consistent global checkpoint if and only if $\neg(M \xrightarrow{Z} M)$.*

COROLLARY 5.6. *Let C be a local checkpoint. Then C can be member of a consistent global checkpoint if and only if $\neg(C \xrightarrow{Z} C)$.*

These conditions were first stated and proved by Netzer and Xu [14]. Here they are obtained as a simple consequence of a more general result. The next two sections concern new results.

5.2.2 Necessary and Sufficient Conditions for Transitless Checkpoints

Similar results hold for transitlessness, where $\rightarrow$ is instantiated as $\xrightarrow{T}$. For example:

THEOREM 5.7. *Let M be a set of local checkpoints from different processes. M can be extended to a transitless global checkpoint if and only if $\neg(M \xrightarrow{T} M)$.*

In the context of distributed debugging, this theorem is used by one of the authors [15] to show when a determistic replay from a global checkpoint is possible without recording any messages during the original execution.

5.2.3 Necessary and Sufficient Conditions for Strongly Consistent Checkpoints

To obtain analogous results for strong consistency, we instantiate $\rightarrow$ as $\xrightarrow{S}$.

THEOREM 5.8. *Let M be a set of local checkpoints from different processes. M can be extended to a strongly consistent global checkpoint if and only if $\neg(M \xrightarrow{S} M)$.*

6 Structure of Acceptable Global Checkpoints

6.1 A Lattice Structure

Let $\mathcal{L}$ be the set of all acceptable global checkpoints of a distributed computation. This section explores the mathematical structure of this set.

DEFINITION 6.1. *Let C_1^i and C_2^i be two local checkpoints produced by P_i. As P_i is sequential, these checkpoints are produced one after the other. Let $min(C_1^i, C_2^i)$ and $max(C_1^i, C_2^i)$ denote the first and the last of them, respectively.*

DEFINITION 6.2. *Let $\Sigma_1 = (C_1^1, C_2^1, \cdots, C_n^1)$ and $\Sigma_2 = (C_1^2, C_2^2 \cdots, C_n^2)$ be two acceptable global checkpoints.*

- *$MAX(\Sigma_1, \Sigma_2)$ denotes the global checkpoint $(C_1, C_2, \cdots, C_n)$ such that $\forall i : C_i = max(C_i^1, C_i^2)$.*

- *$MIN(\Sigma_1, \Sigma_2)$ denotes the global checkpoint $(C_1, C_2, \cdots, C_n)$ such that $\forall i : C_i = min(C_i^1, C_i^2)$.*

DEFINITION 6.3. *Let $\Sigma_1 = (C_1^1, C_2^1, \cdots, C_n^1)$ and $\Sigma_2 = (C_1^2, C_2^2, \cdots, C_n^2)$ be two acceptable global checkpoints. $\Sigma_1 \preccurlyeq \Sigma_2$ iff $\Sigma_1 = MIN(\Sigma_1, \Sigma_2)$ (or equivalently, $\Sigma_2 = MAX(\Sigma_1, \Sigma_2)$).*

THEOREM 6.1. *$(\mathcal{L}, \preccurlyeq)$ is a lattice.*

PROOF. Let $\Sigma_1 = (C_1^1, C_2^1, \cdots, C_n^1)$ and $\Sigma_2 = (C_1^2, C_2^2, \cdots, C_n^2)$ be two acceptable global checkpoints. From the previous definitions we have:

- $MIN(\Sigma_1, \Sigma_2)$ and $MAX(\Sigma_1, \Sigma_2)$ are unique.
- $MIN(\Sigma_1, \Sigma_2) \preccurlyeq \Sigma_1 \preccurlyeq MAX(\Sigma_1, \Sigma_2)$.
- $MIN(\Sigma_1, \Sigma_2) \preccurlyeq \Sigma_2 \preccurlyeq MAX(\Sigma_1, \Sigma_2)$.

To prove that $(\mathcal{L}, \preccurlyeq)$ is a lattice, we must show that $MIN(\Sigma_1, \Sigma_2)$ and $MAX(\Sigma_1, \Sigma_2)$ are acceptable global checkpoints. We show that $MAX(\Sigma_1, \Sigma_2)$ is acceptable (the proof for $MIN(\Sigma_1, \Sigma_2)$ is analogous). Let $MAX(\Sigma_1, \Sigma_2) = (C_1, C_2, \cdots, C_n)$. We show that any pair C_i, C_j is acceptable. We suppose that C_i (respectively, C_j) is produced by P_i (respectively, P_j).

- If both C_i and C_j belong to Σ_1 (or both belong to Σ_2), then as Σ_1 (respectively, Σ_2) is acceptable, the ordered pair (C_i, C_j) is acceptable.
- Suppose now that C_i belongs to Σ_1 (i.e., $C_i = C_i^1$) and C_j belongs to Σ_2 (i.e., $C_j = C_j^2$).

  As $C_i = C_i^1 = max(C_i^1, C_i^2)$, we have $C_i^2 = C_i^1$, or $C_i^2 \rightarrow C_i^1$. Moreover, as Σ_2 is acceptable, there cannot exist a c-edge (e_1, e_2) with e_1 (respectively, e_2) issued by P_i (respectively, P_j), $C_i^2 \rightarrow e_1$ and $e_2 \rightarrow C_i^2$ (Definition 4.1). Hence, it follows that there is no c-edge (e_1, e_2) with $C_i^1 \rightarrow e_1$ and $e_2 \rightarrow C_j^2$, i.e., with $C_i \rightarrow e_1$ and $e_2 \rightarrow C_j$.
- The other cases are obtained by simple index substitutions. □

6.2 Extending Previous Results

Let $\mathcal{L}_{cons}$, $\mathcal{L}_{tr\text{—}less}$ and $\mathcal{L}_{sc}$ be instantiations of $\mathcal{L}$ where *acceptable* means *consistent*, *transitless*, and *strongly consistent*, respectively. It follows from Theorem 6.1 that each of these sets is a lattice. In the case of $\mathcal{L}_{cons}$, this was already shown in [8], [16]. The previous theorem extends this result to the sets of of transitless and strongly consistent global checkpoints.

Consider a set M of local checkpoints that can be extended to an acceptable global checkpoint, i.e., $\neg(M \rightarrow M)$ (Theorem 5.2). Moreover, let $\mathcal{L}(M)$ be the sublattice of $\mathcal{L}$ including all acceptable global checkpoints that contain M. In the same way, let $\mathcal{L}_{cons}(M)$, $\mathcal{L}_{tr\text{—}less}(M)$, and $\mathcal{L}_{sc}(M)$ be the sublattices of consistent, transitless and strongly consistent global checkpoints each of them containing M, respectively.

As far as $\mathcal{L}_{cons}(M)$ is concerned, Wang [19] describes several algorithms to compute the maximum and the minimum global checkpoints. In the same context, [11] describes an algorithm that computes all consistent global checkpoints including M. The previous theorem shows that these algorithms can be extended to transitless global checkpoints and to strongly consistent global checkpoints.

7 CONCLUSION

This paper has investigated consistency criteria for Distributed Checkpointing, namely *traditional consistency*, *transitlessness*, and *strong consistency*. Main results of this paper are a formulation and a proof of a necessary and sufficient condition answering the following question:

"Given an arbitrary set of local checkpoints, can this set be extended to a global checkpoint that satisfies $\mathcal{P}$?"

(where $\mathcal{P}$ is traditional consistency, transitlessness or strong consistency). This theoretical condition is independant of particular communication models (FIFO or not FIFO, causally ordered or not, etc.). It only assumes reliable channels. Protocols producing checkpoint and communication patterns that satisfy this condition can belong either to the family of the coordinated protocols (e.g., [3]) or to the family of communication-induced protocols (e.g., [7]).

ACKNOWLEDGMENTS

The authors thank Yi-Min Wang for interesting discussions on checkpointing problems, and the anonymous referees, whose constructive comments have helped improve the presentation.

REFERENCES

[1] Ö Babaoğlu, E. Fromentin, and M. Raynal, "A Unified Framework for the Specification and Run-time Detection of Dynamic Properties in Distributed Computations," *J. Systems Software*, vol. 33, pp. 287-298, 1996.

[2] R. Baldoni, J.M. Hélary, A. Mostefaoui, and M. Raynal, "A Communication-Induced Checkpointing Protocol that Ensures Rollback-Dependency Trackability," *Proc. 27th IEEE Symp. Fault-Tolerant Computing (FTCS'27)*, pp. 68-77, Seattle, Washington, June 1997.

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

[4] E.N. Elnozahy, D.B. Johnson, and Y.M. Wang, "A Survey of Rollback-Recovery Protocols in Message-Passing Systems," Technical Report CMU-CS-96-181, Carnegie Mellon Univ., 1996.

[5] J. Fowler and W. Zwaenepoel, "Distributed Causal Breakpoints," *Proc. 10th IEEE Int'l Conf. Distributed Computing Systems*, Paris, pp. 134-141, May 1990.

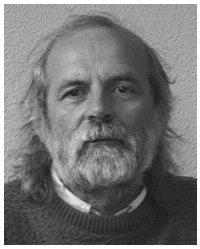
[6] J.M. Hélary, C. Jard, N. Plouzeau, and M. Raynal, "Detection of Stable Properties in Distributed Applications," *Proc. Sixth ACM Symp. Principles of Distributed Computing*, pp. 125-136, Vancouver, 1987.

[7] J.M. Helary, A. Mostefaoui, and M. Raynal, "Communication-Induced Determination of Consistent Snapshots," *Proc. Int'l Symp. Fault-Tolerant Computing, FTCS-28*, pp. 208-217, Munich, Germany, June 1998. ftp://ftp.irisa.fr/techreports/1997/PI-1126.ps

[8] D.B. Johnson and W. Zwaenepoel, "Recovery in Distributed Systems Using Optimistic Message Logging and Checkpointing," *J. Algorithms*, vol. 11, no. 3, pp. 462-491, Sept. 1990.

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

[10] T.H. Lai and T.H. Yang, "On Distributed Snapshots," *Information Processing Letters*, vol. 25, pp. 153-158, 1987.
[11] D. Mannivannan, R.B.H. Netzer, and M. Singhal, "Finding Consistent Global Checkpoints in Distributed Computations," *IEEE Trans. Parallel and Distributed Systems*, vol. 8, no. 6, pp. 623-627, 1997.
[12] B. Miller and J. Choi, "Breakpoint and Halting in Distributed Programs," *Proc. Eighth IEEE Int'l Conf. Distributed Computing Systems*, San Jose, Calif., pp. 316-323, May 1988.
[13] A. Mostefaoui and M. Raynal, "Efficient Message Logging for Uncoordinated Checkpointing Protocols," *Proc. Second European Dependable Computing Conf. (EDCC'2)*, pp. 353-364, Lecture Notes In Computer Science 1150, Taormina, Italy, Springer-Verlag, Oct. 1996.
[14] R.H.B. Netzer and J. Xu, "Necessary and Sufficient Conditions for Consistent Global Snapshots," *IEEE Trans. Parallel and Distributed Systems*, vol. 6, no. 2, pp. 165-169, 1995.
[15] R.H.B. Netzer, and Y. Xu, "Replaying Distributed Programs Without Message Logging," *Proc. Sixth IEEE Int'l Symp. High Performance Distributed Computing,"* Portland Oreg., Aug. 1997.
[16] R. Schwarz and F. Mattern, "Detecting Causal Relationships in Distributed Computations: In Search of the Holy Grail," *Distributed Computing*, vol. 7, pp. 149-174, 1994.
[17] Y.-M. Wang, and W.K. Fuchs, "Optimistic Message Logging for Independent Checkpointing in Message Passing Systems," *Proc. 11th IEEE Symp. Reliable Distributed Systems*, pp. 147-154, Oct. 1992.
[18] Y.-M. Wang and W.K. Fuchs, "Optimal Message Log Reclamation for Uncoordinated Checkpointing," *Fault-Tolerant Parallel and Distributed Systems*, Pradhan, D.R. Avresky, eds., IEEE CS Press, Los Alamitos Calif., 1995.
[19] 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.



Jean-Michel Hélary received his Docteur troisiéme cycle degree in applied mathemetics (numerical analysis of partial differential equations) from the University of Paris in 1968, then his Habilitation Doctor degree from the University of Rennes in 1988. He is currently a professor of computer science at the University of Rennes 1, France. His research interests are distributed algorithms, protocols, programming languages, graph algorithms, and parallelism. He is a member of the INRIA research team "Algorithmes Distribués et Protocoles," led by Prof. Michel Raynal. On this team, he is most specifically interested in the methodological aspects of distributed algorithms design.

Robert H.B. Netzer received the BSE degree in computer science from the University of Florida, and the MS and PhD degrees, both in computer science, from the University of Wisconsin at Madison, where he also pursued studies in civil engineering. He is currently an assistant professor of computer science at Brown University. His research involves instrumentation techniques to support tools for programming, debugging, and understanding sequential and distributed systems. He is a member of the IEEE Computer Society.?

Michel Raynal received his "doctorat d'Etat" in computer science from the University of Rennes 1, France, in 1981. In 1981, he founded the computer science department in the telecommunications engineer school (ENST) in Brest. Since 1984, he has been a professor of computer science at the University of Rennes 1. At IRISA (CNRS-INRIA-University joint computing research laboratory located in Rennes), he leads the research group Algorithmes distribués et protocoles (Distributed Algorithms and Protocols). His research interests include distributed algorithms, operating systems, computing systems, protocols research and dependability. His main interest lies in the fundamental principles that underly the design and the construction of distributed systems. He has been the principal Investigator for a number of research grants in these areas (grants from French private companies, France Telecom, CNRS, NFS-INRIA, and ESPRIT BRA).