# Design of QoS-based Checkpoint Protocol for Multimedia Network Systems and Implementation on Java VM *

Kengo Hiraga and Hiroaki Higaki†
Tokyo Denki University ‡

## 1 Introduction

The advanced computer and network technologies have lead to the development of distributed systems. Here, an application is realized by multiple processes computing and communicating by exchanging messages through communication channels. Some mission-critical applications are required to be executed fault-tolerantly. One important method for fault-tolerance is checkpoint-recovery. For restarting execution correctly after recovery, a set of checkpoints taken by all the processes should form a *consistent global checkpoint* [1]. Distributed multimedia applications such as distance learning, tele-conference, tele-medicine and video on demand have recently been developed on communication networks. A multimedia message is so large that it is required to take checkpoints even during transmission and reception of the message. In addition, an application can accept a message even if a part of the message is lost. Based on the properties, the authors have proposed a measurement for consistency of a global checkpoint [2]. According to it, this paper discusses checkpoint protocols for multimedia network systems. These protocols are non-blocking for supporting realtime applications. Here, consistency and timeliness are QoS parameters.

## 2 Global Consistency

Let $\mathcal{S} = \langle \mathcal{V}, \mathcal{L} \rangle$ be a distributed system where $\mathcal{V} = \{p_1, \ldots, p_n\}$ is a set of processes $p_i$ and $\mathcal{L} \subseteq \mathcal{V}^2$ is a set of communication channels $\langle p_i, p_j \rangle$ from $p_i$ to $p_j$. During failure-free execution, $p_i$ takes a *checkpoint* $c_i$. A set $\{c_1, \ldots, c_n\}$ is a *global checkpoint* $C_{\mathcal{V}}$. Global consistency $GC$ is determined by channel consistency $CC$ for all the channels in $\mathcal{L}$. $CC$ for $\langle p_i, p_j \rangle$ is determined by message consistency $MC$ for all the messages transmitted through $\langle p_i, p_j \rangle$. Finally, MC for a message $m$ is determined by timing relation between $m$ and $C_{\{p_i,p_j\}} = \{c_i, c_j\}$. The measurement of consistency in [2] is upper compatible with the conventional one in [1].

**[Message consistency]** A multimedia message $m$ is decomposed into a sequence $\langle pa_1, \ldots, pa_l \rangle$ of packets for transmission. Suppose $p_i$ takes $c_i$ between transmissions of $pk_s$ and $pk_{s+1}$ and $p_j$ takes $c_j$ between receptions of $pk_t$ and $pk_{t+1}$. If $s > t$, $\{pk_{t+1}, \ldots, pk_s\}$ is a set of lost packets which are not retransmitted after recovery. Thus, lost packets decrease $MC$. On the other hand, if $s < t$, $\{pk_{s+1}, \ldots, pk_t\}$ is a set of orphan packets. These packets are surely retransmitted after recovery. Thus, orphan packets do not affect $MC$. Therefore, lost consistency is induced as a ratio of value of lost packets to value of $m$. Since $m$ is transmitted after compression, value of packets is not unique as in MPEG.

$$MC(m, s, t) = 1 - \frac{\sum_{k=t+1}^{s} value(pa_k)}{value(m)} \tag{1}$$

**[Channel consistency]** Here, $\mathcal{M}_{ij}$ is a set of messages transmitted through $\langle p_i, p_j \rangle$.

$$CC(\langle p_i, p_j \rangle) = \prod_{m \in \mathcal{M}_{ij}} MC(m) \tag{2}$$

**[Global consistency]**

$$GC(C_{\mathcal{V}}) = (\prod_{\langle p_i, p_j \rangle \in \mathcal{L}} CC(\langle p_i, p_j \rangle))^{1/|\mathcal{L}|} \tag{3}$$

## 3 Basic Checkpoint Protocol

A basic checkpoint protocol $\mathcal{P}_{\mathcal{B}}$ is designed where $GC$ is adapted as a QoS parameter. Though $\mathcal{P}_{\mathcal{B}}$ is based on a 3-phase coordinated checkpoint protocol, it is non-blocking. Each process is not required to suspend execution as in a conventional protocols for data communication systems. Here, a sequence number $seq(m)$ of $m$ and $rvalue(pa_k, m) = value(pa_k)/value(m)$ are piggied back to $pa_k$.

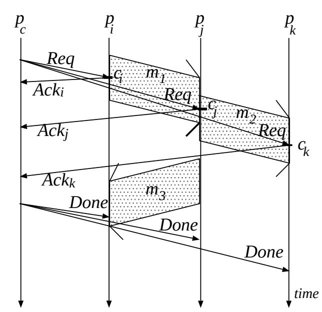


Figure 1: Basic protocol $\mathcal{P}_{\mathcal{B}}$.

**[Basic protocol $\mathcal{P}_{\mathcal{B}}$ (Figure 1)]**

1) Let $RC$ be required consistency. A coordinator $p_c$ sends a request message $Req$ to every $p_i \in \mathcal{V}$.
2) On receiving $Req$, $p_i$ takes a tentative checkpoint $tc_i$ and sends back an acknowledgement message $Ack_i$ to $p_c$. For every $\langle p_i, p_j \rangle$ ($\langle p_j, p_i \rangle$), $seq(m_{ij})$ ($seq(m_{ji})$) and $tvalue(m_{ij}) = \sum rvalue(pa_k, m_{ij})$ ($tvalue(m_{ji}) = \sum rvalue(pa_k, m_{ji})$) for $pa_k$ of the last message $m_{ij}$ ($m_{ji}$) sent (received) before $tc_i$ are piggied back to $Ack_i$.
3) On receiving all $Ack_i$, $p_c$ calculates $GC$. If $GC > RC$, $p_c$ sends $Done$ to $p_i$. Otherwise, $p_c$ sends $Cancel$ to $p_i$.
4) On receipt of $Done$, $p_i$ changes $tc_i$ to $c_i$. On receipt of $Cancel$, $p_i$ discards $tc_i$. □

---

*マルチメディアネットワークのための QoS に基づいたチェックポイントプロトコルの設計とその JavaVM への実装
†平賀 研吾, 桧垣 博章 {hira,hig}@higlab.k.dendai.ac.jp
‡東京電機大学

## 4 Extended Protocol

Though $\mathcal{P_B}$ is non-blocking for supporting realtime multimedia applications, it is not certain to take $C_{\mathcal{V}}$ with required global consistency. According to the definition of $GC$, the less lost packets are, the higher global consistency we archive. Thus, the following modification in step 2) of $\mathcal{P_B}$ for higher probability to take $C_{\mathcal{V}}$ and higher $GC(C_{\mathcal{V}})$ is introduced.

- If $p_i$ is sending a message, $p_i$ takes $tc_i$ soon.
- Otherwise, $p_i$ postpones taking $tc_i$ for $\Delta T_i$. If $p_i$ starts sending another message or $p_i$ starts receiving additional message while receiving a message, $p_i$ takes $tc_i$.

Here, we introduce an additional QoS parameter $\tau$ for timeliness. $p_c$ is required to receive $Ack_i$ within $\tau$ since the transmission of *Req*. Thus, $\Delta T_i = \tau - 2\delta_i$ where $\delta_i$ is transmission delay between $p_c$ and $p_i$.

**[Extended protocol $\mathcal{P_E}$]**

1) Let $RC$ and $\tau$ are required consistency and timeliness. $p_c$ sends *Req* to every $p_i \in \mathcal{V}$. $\tau$ is piggied back to *Req*.
2) On receiving *Req*, $p_i$ takes $tc_i$ as follows:
   2-1) If $p_i$ is sending a message, $p_i$ takes $tc_i$.
   2-2) Otherwise, $p_i$ postpones taking $tc_i$ for $\Delta T_i$. During this period,
      - if $p_i$ is receiving a message and starts sending another message, $p_i$ takes $tc_i$ immediately.
      - if $p_i$ is not communicating and starts sending or receiving a message, $p_i$ takes $tc_i$ immediately.

   On taking $tc_i$, $p_i$ sends $Ack_i$ as in step 2) of $\mathcal{P_B}$.
3) and 4) are same as in $\mathcal{P_B}$. □

## 5 Implementation and Evaluation

The proposed protocol for taking checkpoints in a multimedia network system is implemented on JavaVM. JavaVM is an interpreter for programs written in Java language and the execution environment is independent of the architecture, i.e. hardware architecture and software one including an operating system. Hence, it is possible to achieve wide interoperability if the proposed protocol is implemented in JavaVM.

For taking a checkpoint $c_i$ of a process $p_i$, i.e. storing state information of $p_i$ at $c_i$ for recovery from a certain failure in a system, serialization which is a function provided by Java environment is adopted. Here, each object for an application implements a Serializable class. By invoking a method WriteObject which is a method in ObjectOutputStream class, an object is transformed to a byte sequence from which an original object is achieved when a recovery procedure, ObjectOutputStream is invoked later. A serialized object, i.e. a byte sequence, is stored into a stable storage.

In this section, we evaluate an overhead for taking a checkpoint implemented by the above method to determine whether it is reasonable for a multimedia network system to take a checkpoint while an object is transmitting and/or receiving a multimedia message.

Here, it is assumed that a multimedia message $m$ is transmitted from a computer $M_i$ to another one $M_j$ both of which are connected to a Ethernet LAN. Hence, $m$ is decomposed into fully filled Ethernet frames, i.e. each of them is a 1500 byte packet. While transmitting $m$, $M_j$ takes a local checkpoint by using a serialization mechanism. If the size of a communication buffer is not sufficient, some packets are lost and taking a checkpoint during a communication event is not reasonable. Otherwise, our proposed method is acceptable.

The specification of two computers and NICs attached to the computers is as follows:

$M_i$: Celeron 1.2GHz CPU, 512MByte Memory, Windows2000 SP3 and Planex FNW-9800T 100Base-TX Ethernet NIC.

$M_j$: Crusoe 800MHz CPU, 256MB Memory, WIndowsXP Home Edition and Realtek RTL8139/810 Family PCI 100Base-TX Ethernet NIC.

Figure 2 shows the relation between size of an object and required time duration to store the serialized object into a storage. The time duration $T$ is almost proportional to the size $S$ of an object: $T = 92.5[msec/Mbyte]S$. The maximum size of an object supported by JavaVM is 60MByte. Even though the object size is 60MByte and packet transmission interval in $M_i$ is set as minimum value, i.e. setting a parameter as NoWait and the achieved interval is about 30.0msec, no packets are lost while taking a checkpoint. Therefore, it is reasonable to introduce our method to a multimedia network system.

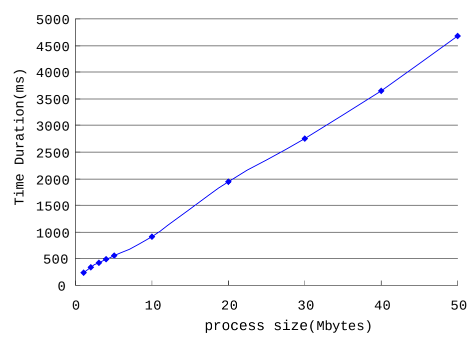


Figure 2: Time Duration for Taking Checkpoint.

## 6 Concluding Remarks

This paper proposed two checkpoint protocols for multimedia network systems. These protocols are based on a novel global consistency as a QoS parameter. For higher consistency, delayed checkpointing is introduced with another QoS parameter, timeliness. Finally, we evaluate the processing overhead to take a checkpoint and find that the proposed method is acceptable for multimedia network systems.

## References


[1] Chandy, K.M. and Lamport, L., "Distributed Snapshot: Determining Global States of Distributed Systems," ACM Trans. on Computer Systems, Vol. 3, No. 1, pp. 63–75 (1985).

[2] Hiraga, K. and Higaki, H., "Consistent Global Checkpoint in Multimedia Network Systems," Proc. of the 8th Workshop on Multimedia Communication and Distributed Processing Systems, pp. 253–258 (2000).