Multimedia Checkpoint Protocol based on Recovery Overhead

Shinji Osada and Hiroaki Higaki
Department of Computers and Systems Engineering
Tokyo Denki University
E-mail: {shinji,hig}@higlab.k.dendai.ac.jp

In order to achieve fault-tolerant distributed systems, checkpoint-recovery has been researched and many protocols have been designed. A global checkpoint taken by the protocols have to be consistent. For conventional data communication networks, a global checkpoint is defined to be consistent if there is no inconsistent message in any communication channel. For multimedia communication networks, there are different requirements for time-constrained failure-free execution and large-size message transmissions where lost of a part of the message is acceptable. This paper proposes novel criteria, consistency and duplexity, for global checkpoints in multimedia communication networks.

リカバリオーバヘッドに基づいたマルチメディアチェックポイントプロトコル

東京電機大学 理工学部 情報システム工学科
長田 慎司　桧垣 博章
E-mail: {shinji,hig}@higlab.k.dendai.ac.jp

近年ネットワークや、コンピュータ技術の急速な発達に伴ないマルチメディア情報をネットワーク上で取り扱うサービスが普及している。また、このようなネットワーク上でのサービスにおいて耐故障性を保証するための手法の 1 つであるチェックポイントリカバリ手法の開発も広く進められている。チェックポイントリカバリの手法では、ネットワークに存在するプロセスは故障の無い時の情報をチェックポイントとして保存し、システムで故障が発生した際にはチェックポイントの情報より、システムを故障前の状態戻すことによって耐故障性を保証している。ここで、マルチメディアネットワークでは、実時間処理を伴なうことがあるため、従来のチェックポイントリカバリの手法を適用することが必ずしも適切であるとは言えない。本論文ではマルチメディアネットワークに適した新しいチェックポイントリカバリのためのグローバルチェックポイントの評価指標を提案する。

1 Introduction

Advanced computer and network technologies have lead to the development of computer networks. Here, an application is realized by multiple processes located on multiple computers connected to a communication network such as the Internet. Each process computes and communicates with other processes by exchanging messages through communication channels. Mission-critical applications are required to be executed fault-tolerantly. That is, even if some processes fail, execution of an application is required to be continued. One of the important methods to realize fault-tolerant networks is *checkpoint-recovery*. During failure-free execution, each process sometimes takes local checkpoints by storing state information into a stable storage. If a certain process fails and recoveries, e.g. by replacing a crashed hardware and rebooting an operating system on which the process depends and by doing nothing for timing dependent problem such as deadlocks, the processes restart from the checkpoints by restoring the state information from the stable storage. For restarting execution of applications correctly in conventional data communication networks, a set of local checkpoints taken by all the processes and from which the processes restart should form a *consistent global checkpoint* [3]. A global checkpoint is defined to be consistent if there is neither *orphan* nor *lost message*. However, in a multimedia communication network, applications require transmission of large-size multimedia messages and low overhead failure-free execution rather than complete consistency. Hence, this paper proposes novel criteria for global checkpoints based on properties of multimedia communication networks and applications.

2 Conventional Consistency

Let $\mathcal{N} = \langle \mathcal{V}, \mathcal{L} \rangle$ be a computer network 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 a process p_i to another process p_j. Execution of an application in p_i is modeled by a sequence of occurrences of *events*. p_i transits from one *state* to another by an occurrence of an event. There are two kinds of events; *local events* and *communication events*. At a local event, state transition in p_i is caused by local computation without exchanging a message. At a communication event, p_i communicates with another process by exchanging a message and the state of p_i

is transitted. There are two kinds of communication events; a *message sending event* $s(m)$ and a *message receipt event* $r(m)$ for a message m. In order to realize a fault-tolerant network, there are two kinds of methods; checkpoint-recovery and *replication*. In replication [7, 9, 12, 17, 20], each process is replicated and placed on multiple computers. Even if a certain process fails, other replicated processes continue to execute an application. On the other hand, checkpoint-recovery is widely available [1, 2, 5, 6, 8, 11, 13, 15, 19]. Here, during failure-free execution, each process p_i sometimes takes a *local checkpoint* c_i by storing current state information into a *stable storage* [16]. After p_i fails and recovers, p_i restarts execution of an application from c_i by restoring the state information from the stable storage. If p_i restarts independently of the other processes, there may be two kinds of *inconsistent messages*; *lost messages* and *orphan messages* [10].

[Inconsistent message] Suppose that processes p_i and p_j take local check points c_i and c_j, respectively. A message m transmitted through $\langle p_i, p_j \rangle$ is *inconsistent* if m is a lost message or an orphan message for a set $C_{\{p_i,p_j\}} = \{c_i, c_j\}$ of local checkpoints. m is a *lost message* iff $s(m)$ occurs before taking c_i in p_i and $r(m)$ occurs after taking c_j in p_j. m is an *orphan message* iff $s(m)$ occurs after taking c_i in p_i and $r(m)$ occurs before taking c_j in p_j. □

In order to correctly recover $\mathcal{N}$ from a process failure, there should be no inconsistent message in any communication channel in $\mathcal{L}$. Thus, in case of a failure in a process p_i, not only p_i but also other processes restart from local checkpoints. Hence, a *global checkpoint* $C_{\mathcal{V}} = \{c_1, \ldots, c_n\}$ which is a set of local checkpoints of all the processes in $\mathcal{V}$ should be *consistent* [10].

[Consistent global checkpoint] A global checkpoint $C_{\mathcal{V}}$ in $\mathcal{N} = \langle \mathcal{V}, \mathcal{L} \rangle$ is consistent iff there is no inconsistent message in any communication channel in $\mathcal{L}$. □

3 Multimedia Networks

Recently, multimedia network applications such as distance learning, tele-conference, tele-medicine and video on demand have been developed on communication networks [18]. Here, messages with multimedia data including text, voice, sound, picture and video are exchanged among processes. These messages are larger than those with conventional data. Hence, it takes longer time to transmit and receive them. As shown in Figure 1, the following four *pseudo events* are defined for a multimedia message $\overline{m}$ transmitted through a communication channel $\langle p_i, p_j \rangle$

- $sb(\overline{m})$: p_i begins sending $\overline{m}$.
- $se(\overline{m})$: p_i ends sending $\overline{m}$.
- $rb(\overline{m})$: p_j begins receiving $\overline{m}$.
- $re(\overline{m})$: p_j ends receiving $\overline{m}$.

A message sending event $s(\overline{m})$ for $\overline{m}$ begins at $sb(\overline{m})$ and ends at $se(\overline{m})$ in p_i. A message receipt event $r(\overline{m})$ for $\overline{m}$ begins at $rb(\overline{m})$ and ends at $re(\overline{m})$ in p_j. Computer network protocols, e.g. TCP/IP protocols [4], are hierarchically composed. A message in an upper layer is decomposed into multiple packets in a lower layer. For example, an IP datagram may be decomposed into multiple Ethernet frames in a sender process since an MTU (Maximum Transfer Unit) for an IP datagram is 64kbyte and one for an Ethernet frame is 1.5kbyte. These frames are reassembled in a receiver process. Thus, a multimedia message m is assumed to be decomposed into a sequence $\langle pa_1, \ldots, pa_l \rangle$ of multiple *packets* for transmission as in Figure 1. Here, $s(pa_k)$ is a *packet sending event* and $r(pa_k)$ is a *packet receipt event* for a packet pa_k.

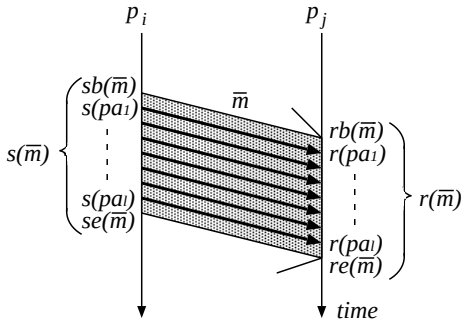

Figure 1: Multimedia message transmission.

For a message m with conventional data, communication events $s(\overline{m})$ and $r(\overline{m})$ are assumed to be atomic. Here, each local checkpoint c_i in a process p_i is taken only when no other event occurs in p_i. However, a multimedia message is so larger than a conventional data message that it takes longer time to transmit and receive the message. Thus, if a process is required to take a local checkpoint during a communication event, it has to wait until an end of the event. Hence, timeliness requirement in a checkpoint protocol is not satisfied and communication overhead in recovery is increased. Therefore, a local checkpoint should be taken immediately when a process is required to take it even during a communication event. That is, a process p_i sending a multimedia message $\overline{m} = \langle pa_1, \ldots, pa_l \rangle$ takes a local checkpoint c_i between packet sending events $s(pa_s)$ and $s(pa_{s+1})$ and another process p_j receiving $\overline{m}$ takes a local checkpoint c_j between packet receipt events $r(pa_r)$ and $r(pa_{r+1})$. In addition, part of a multimedia message may be lost in a communication channel for an application, e.g. MPEG data transmission. Such an application requires not to retransmit lost packets in recovery but to transmit packets with shorter transmission delay. Hence, an overhead for taking a checkpoint during failure-free execution is required to be reduced.

4 Criterion (1) - Consistency

As discussed in section 2, since the conventional criterion for a global checkpoint is based on an architecture of conventional data communication networks, novel criteria should be introduced into multimedia communication networks. *Global consistency Gc* for

a global checkpoint $C\nu = \{c_1, \ldots, c_n\}$ denotes degree of consistency for C_ν in $\mathcal{N} = \langle \mathcal{V}, \mathcal{L} \rangle$. In a conventional data communication network, Gc is defined as follows:

$$Gc = \begin{cases} 1 & \text{no inconsistent message in } \mathcal{L}. \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

In a multimedia communication network, a local checkpoint is taken even during a communication event and it is acceptable for an application to lose part of a multimedia message. Hence, a domain of Gc is a closed interval $[0, 1]$ instead of a discrete set $\{0, 1\}$. Here, Gc should be compatible with the conventional criterion, i.e. (1) should be satisfied.

Gc is determined by timing-relation among local checkpoints and communication event for messages transmitted through communication channels. Thus, Gc is calculated by *channel consistency* Cc_{ij} for all the communication channels $\langle p_i, p_j \rangle \in \mathcal{L}$. Furthermore, Cc_{ij} for a communication channel $\langle p_i, p_j \rangle$ is calculated by *message consistency* Mc^u_{ij} for all multimedia messages $\overline{m_u}$ transmitted through $\langle p_i, p_j \rangle$. Finally, Mc^u_{ij} for a multimedia message $\overline{m_u}$ transmitted through $\langle p_i, p_j \rangle$ is induced by timing-relation between communication events for $\overline{m_u}$ and local checkpoints $C_{\{p_i, p_j\}} = \{c_i, c_j\}$.

4.1 Message Consistency

Message consistency Mc^u_{ij} is degree of consistency for a set $C_{\{p_i, p_j\}} = \{c_i, c_j\}$ of local checkpoints and a multimedia message m_u transmitted through a communication channel $\langle p_i, p_j \rangle$. Here, c_i and c_j are taken by processes p_i and p_j, respectively. Here, we define an *inconsistent multimedia message*.

[**Inconsistent multimedia message**] A multimedia message m_u is inconsistent iff m_u is a *lost multimedia message* as in Figure 2 or an *orphan multimedia message* as in Figure 3. m_u is a lost multimedia message iff $se(\overline{m_u})$ occurs before taking c_i in p_i and $rb(\overline{m_u})$ occurs after taking c_j in p_j. m_u is an orphan multimedia message iff $sb(\overline{m_u})$ occurs after taking c_i in p_i and $rb(\overline{m_u})$ occurs before taking c_j in p_j. □

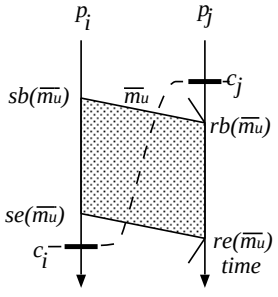

Figure 2: Lost multimedia messages.

If m_u is a lost multimedia message, all the packets of m_u have been already sent by p_i but none of them is received by p_j in recovery. If m_u is an orphan multimedia message, m_u might not be retransmitted after recovery due to non-deterministic property of p_i even though p_j has already received part of m_u. For compatibility with (1), $Mc^u_{ij} = 0$ for $\overline{m_u}$.

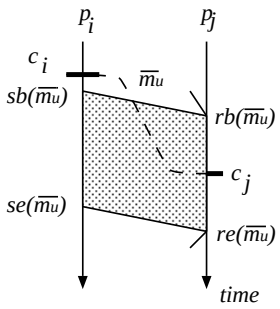

Figure 3: Orphan multimedia messages.

[**Consistency for inconsistent multimedia message**] $Mc^u_{ij} = 0$ for an inconsistent multimedia message $\overline{m_u}$. □

If c_i and c_j are taken before $sb(\overline{m_u})$ and $rb(\overline{m_u})$, $Mc^u_{ij} = 1$. In addition, if c_i and c_j are taken after $se(\overline{m_u})$ and $re(\overline{m_u})$, $Mc^u_{ij} = 1$. Thus, it is compatible with (1). As discussed in the previous section, a multimedia message $\overline{m_u}$ is decomposed into a sequence of multiple packets $\langle pa_1, \ldots, pa_l \rangle$. Thus, $s(\overline{m_u})$ is composed of a sequence $\langle sb(\overline{m_u}),\ s(pa_1), \ldots, s(pa_l),\ se(\overline{m}) \rangle$ of packet sending events and pseudo events and $r(\overline{m_u})$ is composed of a sequence $\langle rb(\overline{m_u}),\ r(pa_1), \ldots, r(pa_l),\ re(\overline{m_u}) \rangle$ of packet receipt events and pseudo events.

[**Lost and orphan packets**] Suppose that local checkpoints c_i and c_j are taken between $s(pa_s)$ and $s(pa_{s+1})$ and between $r(pa_r)$ and $r(pa_{r+1})$ for a multimedia message $\overline{m_u} = \langle pa_1, \ldots, pa_l \rangle$, respectively. pa_k is a *lost packet* iff $s(pa_k)$ occurs before taking c_i in p_i and $r(pa_k)$ occurs after taking c_j in p_j. pa_k is an *orphan packet* iff $s(pa_k)$ occurs after taking c_i in p_i and $r(pa_k)$ occurs before taking c_j in p_j.□

If $s = r$, there is no lost and orphan packet. Hence, $Mc^u_{ij} = 1$.

If $s > r$, $\{pa_{r+1}, \ldots, pa_s\}$ is a set of lost packets. These packets are not retransmitted after recovery. Lost packets in a conventional data communication network are restored by logging them in failure-free execution [1,15]. However, in a multimedia communication network, less overhead in failure-free execution is required since applications require time-constrained execution. For example, storing a message log in a stable storage makes transmission delay and jitter in MPEG data transmission larger. In addition, even if part of a multimedia message is lost in recovery, an application accepts the message. The less packets are lost, the higher message consistency is achieved. A multimedia message is usually compressed for transmission. Thus, value of packets for a message is not unique. For example in an MPEG data transmission, value of a packet for an I-picture is higher than value of a packet for a B-picture. Therefore, message consis-

tency depends on total value of lost packets as follows:

$$\frac{\partial Mc_{ij}^{u}}{\partial l\text{-}value} < 0$$
$$\text{where } l\text{-}value = \sum_{lost\ packets\ pa_k} value(pa_k). \quad (2)$$

Here, a domain of Mc_{ij}^{u} is an open interval $(0, 1)$.

If $s < r$, $\{pa_{s+1}, \ldots, pa_r\}$ is a set of orphan packets. An orphan multimedia message might not be retransmitted after recovery due to non-deterministic property of a process. However, orphan packets are surely retransmitted after recovery since c_i and c_j are taken during transmission and receipt of $\overline{m_u}$ and the content of $\overline{m_u}$ being carried by a sequence $\langle pa_1, \ldots, pa_l \rangle$ of packets is not changed even after recovery. Orphan packets are received twice, once in failure-free execution and once after recovery. By assigning a sequence number to each packet, it never occurs for an application to receive a packet more than once. Hence, message consistency does not depend on orphan packets.

[**Message consistency**] Let $value(\overline{m_u})$ be total value of packets $pa_1, \ldots, pa_l$ of $\overline{m_u}$.

$$Mc_{ij}^{u} = 0 \quad \text{if } l\text{-}value = value(\overline{m_u}).$$
$$Mc_{ij}^{u} = 1 \quad \text{if } l\text{-}value = 0. \quad (3)$$
$$\frac{\partial Mc_{ij}^{u}}{\partial l\text{-}value} < 0 \text{otherwise.} \quad \square$$

[**Example 1**] In Figure 4, the reduction of message consistency is proportional to total value of lost packets. Here, message consistency is as follows:

$$Mc_{ij}^{u} = 1 - \frac{l\text{-}value}{value(\overline{m_u})} = 1 - \frac{\sum_{k=r+1}^{s} value(pa_k)}{value(\overline{m_u})} \quad (4)$$

$\square$

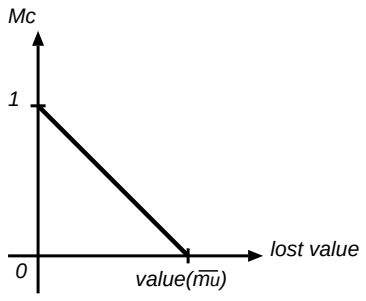

Figure 4: Message Consistency Example(1).

[**Example 2**] In Figure 5, if most of the packets of

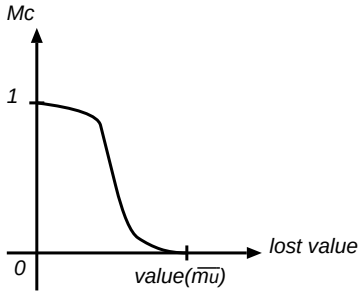

Figure 5: Message Consistency Example(2).

a message $\overline{m_u}$ are not lost, message consistency is almost 1. On the other hand, if most of the packets of $\overline{m_u}$ are lost, message consistency is almost 0. For example in MPEG data transmission, even if small part of a message are lost, applications accept the message. On the other hand, if most part of a message are lost, applications do not accept the message. Hence, according to increase of total value of lost packets, message consistency decreases first gradually, then rapidly and finally gradually. Hence, message consistency is as follows:

$$Mc_{ij}^{u} = \frac{\frac{tan^{-1}(-\alpha(lostvalue - \frac{value(\overline{m_u})}{2}))}{tan^{-1}(\alpha\frac{value(\overline{m_u})}{2})} + 1}{2} \quad (5)$$
$$\alpha : constant$$

$\square$

4.2 Channel Consistency

Based on the message consistency for multimedia messages $\overline{m_u}$ and local checkpoints c_i and c_j in processes p_i and p_j respectively, channel consistency Cc_{ij} is defined as degree of consistency for a set $C_{\{p_i,p_j\}} = \{c_i, c_j\}$ of local checkpoints in a communication channel $\langle p_i, p_j \rangle \in \mathcal{L}$. Cc_{ij} is calculated by using message consistency Mc_{ij}^{u} for every message $\overline{m_u}$ transmitted through $\langle p_i, p_j \rangle$. For compatibility with (1), if message consistency for every message transmitted through $\langle p_i, p_j \rangle$ is 1, channel consistency is also 1. On the other hand, if message consistency for at least one message transmitted through $\langle p_i, p_j \rangle$ is 0, channel consistency is also 0. In addition, channel consistency monotonically increases for consistency of messages transmitted through $\langle p_i, p_j \rangle$.

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

$$Cc_{ij} = 1 \quad \text{if } \forall \overline{m_u} \in \mathcal{M}_{ij} \quad Mc_{ij}^{u} = 1.$$
$$Cc_{ij} = 0 \quad \text{if } \exists \overline{m_u} \in \mathcal{M}_{ij} \quad Mc_{ij}^{u} = 0. \quad (6)$$
$$\forall \overline{m_u} \in \mathcal{M}_{ij} \quad \frac{\partial Cc_{ij}}{\partial Mc_{ij}^{u}} > 0 \quad \text{otherwise.} \quad \square$$

[**Example 3**] Cc_{ij} is calculated by multiplication of Mc_{ij}^u for all the messages $\overline{m_u}$ transmitted through $\langle p_i, p_j \rangle$. This satisfies (6).

$$Cc_{ij} = \prod_{\overline{m_u}} Mc_{ij}^u \quad \Box \tag{7}$$

4.3 Global Consistency

Based on the channel consistency for communication channels $\langle p_i, p_j \rangle \in \mathcal{L}$ and global checkpoint $C_{\mathcal{V}} = \{c_1, \ldots, c_n\}$, global consistency Gc is defined as degree of the global checkpoint. Gc is calculated by using channel consistency Cc_{ij} for every communication channel $\langle p_i, p_j \rangle \in \mathcal{L}$. For compatibility with (1), if channel consistency for every channel in $\mathcal{L}$ is 1, global consistency is also 1. On the other hand, if consistency for at least one communication channel is 0, global consistency is also 0. In addition, channel consistency monotonically increases for consistency of the communication channels in $\mathcal{L}$.

[**Global consistency**]

$$\begin{aligned} Gc &= 1 \quad \text{if } \forall \langle p_i, p_j \rangle \in \mathcal{L} \;\; Cc_{ij} = 1. \\ Gc &= 0 \quad \text{if } \exists \langle p_i, p_j \rangle \in \mathcal{L} \;\; Cc_{ij} = 0. \end{aligned} \tag{8}$$

$\forall \langle p_i, p_j \rangle \in \mathcal{L} \;\; \dfrac{\partial Gc}{\partial Cc_{ij}} > 0$ otherwise. $\Box$

[**Example 4**] Gc is calculated by multiplication of Cc_{ij} for all the communication channels $\langle p_i, p_j \rangle \in \mathcal{L}$. For independence of system scale, normalization factor $|\mathcal{L}|$ is applied. This satisfies (8).

$$Gc = \Big(\prod_{\langle p_i, p_j \rangle} Cc_{ij} \Big)^{|\frac{1}{\mathcal{L}}|} \quad \Box \tag{9}$$

5 Criterion (2) - Duplexity

For taking a checkpoint during failure-free execution, each process stores state information into a stable storage. Here, the state information consists of state variables of an application and communication buffers. There are two communication buffers in a process p_i; a transmission buffer tb_i and a receipt buffer rb_i. When a process transmits a message $\overline{m}$ to another process, $\overline{m}$ is decomposed into multiple packets $pa_1, \ldots, pa_k$. Then, these packets are stored into tb_i. Packets in tb_i are transmitted in FIFO order. In a process, multiple threads of control may be invoked and transmit messages simultaneously. Hence, packets for these messages are interleaved in tb_i. On the other hand, received packets are stored into rb_i and processed in FIFO order.

As discussed in the previous section, the newly introduced consistency of a global checkpoint is induced only by the number of lost packets. In recovery, before a process p_i receiving a message when it takes a local checkpoint restarts execution of an application, p_i has to receive all orphan packets of $\overline{m}$ On the other hand, a process p_j sending $\overline{m}$ sends all orphan packets of $\overline{m}$ before packets of another message even though p_i has already received since the packets have already been stored into a transmission buffer tb_i of p_j. Thus, the more orphan packets are, the longer recovery time is. Hence, we introduce an additional criterion for a global checkpoint based on the number of orphan packets.

As the discussion of consistency in section 4, we calculate the global duplexity by using local duplexity induced by the state of transmission and receipt buffers.

[**Duplexity td_i for tb_i**] When a process p_i takes a checkpoint, a transmission buffer tb_i of p_i is stored into a stable storage as state information. In recovery, for restarting to execute an application, p_i has to transmit all the orphan messages stored in tb_i. However, p_i sends the message in tb_i only in the order of tb_i. Hence, duplexity td_i for tb_i is denoted as follows:

$$\begin{aligned} td_i = F_1(&number\ of\ messages\ m \\ &before\ the\ last\ orphan\ message\ in\ tb_i) \end{aligned} \tag{10}$$

[**Duplexity rd_i for rb_i**] Before restarting an application, a process p_i has to receive all the orphan messages destined to p_i from p_j where $\langle p_j, p_i \rangle \in \mathcal{L}$. rd_i^j denotes recovery overhead caused by orphan messages transmitted through $\langle p_j, p_i \rangle$ after recovery.

$$rd_i^j = F_2(number\ of\ orphan\ messages\ intd_i) \tag{11}$$

Thus, rd_i is induced by a function whose parameters are rd_i^j and monotonically increases for rd_i^j.

$$\begin{aligned} rd_i &= F_3(rb_1, rb_2, \ldots, rb_n) \\ \frac{\partial rd_i}{\partial |rb_j|} &\geq 0 \end{aligned} \tag{12}$$

[**Duplexity dup_i for p_i**] Using td_i and rd_i, dup_i is defined as follows:

$$\begin{aligned} dup_i &= F_4(td_i, rd_i) \\ \frac{\partial dup_i}{\partial td_i} &\geq 0 \quad \frac{\partial dup_i}{\partial rd_i} \geq 0 \end{aligned} \tag{13}$$

[**Duplexity rd_i for rb_i**] Global duplexity Gd is calculated by using dup_i for all the processes $p_i \in \mathcal{V}$.

$$\begin{aligned} Gd &= F_5(dup_1, \ldots, dup_i) \\ \frac{\partial Gd}{\partial dup_i} &\geq 0 \quad \Box \end{aligned} \tag{14}$$

6 Conclusion and Remarks

This paper has proposed two novel criteria, consistency and duplexity, for a global checkpoint in multimedia network systems. The authors will design QoS-based checkpoint protocols based on these criteria.

References

[1] Agbaria, A., Attiya, H., Friedman, R. and Vitenberg, R., "Quantifying rollback propagation in distributed checkpointing," The 20th International Symposium on Reliable Distributed Systems, pp. 36–45 (2001).

[2] Avril, H., Tropper, C., "On rolling back and checkpointing in time warp," IEEE Transactions on Parallel and Distributed Systems, Vol. 12, No. 11, pp. 1105–1121 (2001).

[3] Baldoni, R., Raynal, M., Cioffi, G. and Helary, J.M., "Direct Dependency-Based Determination of Consistent Global Checkpoints," The 3rd International Conference on Principles of Distributed Systems, pp. 11–28 (1999).

[4] Douglas, E.C., "Internetworking with TCP/IP," Prentice-Hall (1991).

[5] Elozahy, E.N., Johnson, D.B. and Wang, Y.M., "A Survey of Rollback-Recovery Protocols in Message-Passing Systems," Technical Note of Carnegie Mellon University, CMU-CS-96-181 (1996).

[6] Gendelman, E., Bic, L.F. and Dillencourt, M.B., "An efficient checkpointing algorithm for distributed systems implementing reliable communication channels," The 18th International Symposium on Reliable Distributed Systems, pp. 290–291 (1999).

[7] Higaki, H., Nemoto, N., Tanaka, K. and Takizawa, M., "Protocol for Groups of Pseudo-Active Replication Objects," International Workshop on Object Oriented Realtime Distributed Systems, pp. 35–41 (1999).

[8] Hiraga, K. and Higaki, H., "Consistent Global Checkpoints in Multimedia Netowrk Systems," The 15th International Conference on Information Networking (ICOIN), pp.271–276 (2001).

[9] Homburg, P., Steen, M.V. and Tanenbaum, A.S., "An Architecture for A Wide Area Distributed System," In Proceedings 7th ACM SIGOPS European Workshop, (1996).

[10] Janakiraman, G. and Tamir, Y., "Coordinated Checkpointing-Rollback Error Recovery for Distributed Shared Memory Multicomputers," In Proc. of the 13th Symp. on Reliable Distributed Systems, pp. 42–51 (1994).

[11] Koo, R. and Toueg, S., "Checkpointing and Rollback-Recovery for Distributed Systems," IEEE Trans. on Software Engineering, Vol. SE-13, No. 1, pp. 23–31 (1987).

[12] Little, M.C., McCue, D.L. and Shrivastava, S.K., "Maintaining Information about Persistent Replicated Objects in a Distributed System," Appeared in the Proceedings of the 13th International Conference on Distributed Computing System, pp. 491–498 (1993).

[13] Manivannan, D. and Singhal, M., "A Low-overhead Recovery Technique Using Quasi-Synchronous Checkpointing," Proceedings of the 16th International Conference on Distributed Computing Systems, pp. 100–107 (1996).

[14] Osada, S., Hiraga, K. and Higaki, H., "Qos based Checkpointing Protocol in Multimedia Network Systems," The Annual IEEE International Workshop on Fault-Tolerant Parallel and Distributed Systems, CD-ROM (2001).

[15] Pankaj, J., "Fault Tolerance in Distributed Systems," Prentice Hall, pp.185–213 (1994).

[16] Plank, J.S., "An Overview of Checkpointing in Uniprocessor and Distributed Systems, Focusing on Implementation and Performance," Technical Report of Department of Computer Science, (1997).

[17] Rangarajan, S., Garg, S. and Huang, Y., "Checkpoints-on-demand with active replication," The 17th International Symposium on Reliable Distributed Systems, pp. 75–83 (1998).

[18] Rejaie, R., Handley, M., and Estrin, D., "RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet," In Proc. IEEE INFOCOM, pp. 1337–1345 (1999).

[19] Taesoon, P. and Yeom, H.Y., "An asynchronous recovery scheme based on optimistic message logging for mobile computing systems," Proceedings 20th International Conference on Distributed Computing Systems, pp. 436–443 (2000).

[20] Zhuang, S.Q., Zhao, B.Y., Joseph, A.D., Katz, R.H. and Kubiatowicz, J., "Bayeux: An Architecture for Scalable and Fault-tolerant Wide-Area Data Dissemination," NOSSDAV, (2001).