Cluster Comput (2013) 16:979–988
DOI 10.1007/s10586-012-0239-2

Quorum-based synchronization protocols for multimedia replicas

Tadateru Ohkawara · Ailixier Aikebaier · Tomoya Enokido · Makoto Takizawa

Received: 8 July 2012 / Accepted: 19 November 2012 / Published online: 9 January 2013

Abstract Multiple replicas of multimedia objects are distributed to peers in overlay networks. In quorum-based (QB) protocols, every replica may not be up-to-date and the up-to-date replica can be found in the version counter. Multimedia objects are characterized in terms of not only data structure but also quality of service (QoS) parameters like frame rate. A transaction reads a parameter of a replica while there is a type of read operation to read a whole state of a replica. Each parameter of a replica is changed through a write operation. Thus, the data structure and QoS parameters of a replica are independently manipulated. In the multimedia quorum-based (MQB) protocol, multiple replicas of a multimedia object are synchronized based on the newness precedent relation. An object is an encapsulation of data and abstract operations for manipulating the data. There are enriching and impoverishing types of write operations. Some data is added to a replica in an enriching operation. On the other hand, some data in a replica is removed in an impoverishing operation. In order to reduce the overhead to write every replica in a quorum, we take an approach that the state of each replica is not always updated. If a transaction issues an enriching write operation, every replica in the write quorum is updated in the same way as the QB protocol. On the other hand, if an impoverishing write operation is issued, every replica is not updated in the quorum. Impoverishing operations are just recorded in replicas. On receipt of a read operation to read a whole state, impoverishing operations recorded are performed on a replica. The MQB protocol is evaluated in terms of the processing overhead of replicas. We show that the processing overhead of each replica can be reduced in the MQB protocol.

Keywords Quorum · Multimedia object · Replication · Impoverishing operation · Enriching operation · Multimedia quorum

T. Ohkawara (✉) · M. Takizawa
Department of Computer and Information Science, Seikei University, 3-3-1 Kichijoji-kitamachi, Musashino-shi, Tokyo 180-8633, Japan
e-mail: tadateru.ohkawara@gmail.com

M. Takizawa
e-mail: makoto.takizawa@computer.org

A. Aikebaier
National Institute of Information and Communications Technology, Tokyo, Japan

T. Enokido
Department of Faculty of Business Administration, Rissho University, Tokyo, Japan

1 Introduction

In scalable distributed systems like cloud computing systems [1, 5, 10, 11] and peer-to-peer (P2P) overlay networks [12, 15, 16, 19, 20], resource objects like files and databases are replicated in multiple computers in order to increase the performance, reliability, and availability. Multimedia objects in addition to simple objects like files are distributed in networks. Especially, multimedia objects are in nature autonomously distributed to computers through peer-to-peer communication like downloading in P2P overlay networks. There are many discussions on how to maintain the mutual consistency of multiple replicas of a simple object like a file [7]. There is at least one up-to-date replica in a quorum and a pair of a read quorum and a write quorum include at least one common replica in the QB protocols [2, 6, 9, 17, 18]. Up-to-date replicas in a quorum can be found in version counters [3]. Every replica in a write quorum is updated to be the newest one each time a write operation is issued. For a read operation, a newest replica o_i in a read quorum is read. Then, every replica which is older than the newest replica o_i is changed with the same state as the replica o_i.

Multimedia objects are characterized in terms of quality of service (QoS) in addition to data structure. Hence, each replica o_i of a multimedia object o is characterized in terms of parameters $p_0, p_1, \ldots, p_l$ $(l \geq 1)$ where the first parameter p_0 shows the data structure, i.e. subobjects and another parameter p_k indicates a QoS parameter like frame rate $(k = 1, \ldots, l)$. Compared with simple objects like files, a larger volume of data is transmitted in networks and manipulated in a multimedia object. Computation resource is spent to manipulate multimedia replicas in computers. Hence, it is critical to discuss how to reduce the overhead, especially processing overhead of each replica. The multimedia quorum-based (MQB) protocol is discussed to reduce the overhead in the papers [13, 14]. Here, replicas are partially ordered in the version vector $\langle vc_0, vc_1, \ldots, vc_l \rangle$ of version counters. Each version counter vc_k is used for each parameter p_k $(k = 0, 1, \ldots, l)$. Each version counter vc_k is incremented so as to be larger than the maximum value in a write quorum each time the parameter p_k is updated. A replica with the maximum version vector is newest, i.e. up-to-date in each quorum.

In order to increase the performance of the MQB protocol, we propose a novel synchronization mechanism where the state of a replica is not always changed while the parameters are changed. Each parameter p_k in a replica o_i is read and written in a *read* operation r^k and *write* operation $w^k(x)$, respectively.

There are *enriching* and *impoverishing* types of write operations [14]. Some new data not in a replica is added in an enriching operation. For example, a subobject is added to a replica. On the other hand, some data in a replica is deleted in an impoverishing operation. For example, a QoS parameter, say the number of colours is reduced. The newer state of a replica shown by the parameters can be obtained by reducing data in the physical state. Hence, we take the following approach:

1. In an enriching type of write operation $w^k(x)$, every replica is updated in a write quorum Q_{kw}.
2. In an impoverishing type of write operation $w^k(x)$, the new value x of the parameter p_k is just recorded but each replica o_i is not updated in a write quorum Q_{kw}.

The replica o_i is materialized, i.e. physically updated to be up-to-date by changing the state with one shown by the parameters. If a read operation r^k is issued to read the kth parameter p_k of a replica, the parameter p_k of an up-to-date replica in a read quorum Q_{kr} is read. In a read operation r^0 to read the data structure parameter p_0, a replica o_i is materialized and a whole physical state of the replica o_i is read.

We also discuss an MQB-RM (read materialization) protocol where every replica is updated after a transaction reads a newest replica in a read quorum.

In the MQB and MQB-RM protocols, the processing overhead of each replica o_i can be reduced since every replica may not be written in a write quorum, just parameters are changed. In this paper, we evaluate the MQB and MQB-RM protocols compared with the QB protocol in terms of the processing overhead. We show the processing overhead of each replica can be reduced in the MQB protocol compared with the QB protocol.

In Sect. 2, we discuss enriching and impoverishing types of operations of multimedia objects. In Sect. 3, we present procedures for read and write operations in the MQB and MQB-RM protocols. In Sect. 4, we evaluate the MQB and MQB-RM protocols compared with the QB protocol.

2 Multimedia objects

2.1 Parameters

An object o is an encapsulation of data structure and operations for manipulating the data structure. A multimedia object o is characterized in terms of not only data structure parameter but also quality of service (QoS) parameters. An object is characterized in a tuple $\langle p_0, p_1, \ldots, p_l \rangle$ of logical parameters. The first logical parameter p_0 stands for the data structure scheme which shows the *part_of* structure of subobjects. For example, an object o is composed of subobjects a, b, and c. Here, the logical parameter p_0 of the object o is $\langle a, b, c \rangle$. The other logical parameters $p_1, \ldots, p_l$ indicate QoS parameters $(l \geq 1)$. For example, the second logical parameter p_1 shows the frame rate. If the frame rate of an object o is 40 fps, the logical parameter p_1 of the object o is 40. Here, the object o is shown in a tuple $\langle \langle a, b, c \rangle, 40 \rangle$ of the logical parameters p_0 and p_1. In this paper, we assume every subobject of an object o has the same QoS parameters for simplicity.

Let x and y be a pair of values to be taken by a logical parameter p_k. If p_k is a QoS parameter $(k \geq 1)$, the value x is poorer than y $(x \prec y)$ if $x < y$. For example, the frame rate 20 fps is *poorer* than 40 fps $(20 \prec 40)$. For a data structure parameter p_0 $(k = 0)$, x is *poorer* than y $(x \prec y)$ iff every subobject of x is included in y $(x \subset y)$. For example, an object is composed of three subobjects $\langle a, b, c \rangle$ and another object is composed of two subobjects $\langle a, b \rangle$. Here, $\langle a, b \rangle$ is poorer than $\langle a, b, c \rangle$. $x \preceq y$ iff $x \prec y$ or $x = y$. x is richer than y iff $x \succ y$. If $x \preceq y$, y includes additional data which is not in x.

An object o is replicated in multiple computers. Let o_i be a replica of the object o $(i = 1, \ldots, n)$. Each replica o_i supports the same operations and parameters as the object o.

Each replica o_i is also composed of the same logical parameters $\langle p_0, p_1, \dots, p_l \rangle$ as the object o. Let $o_i.p_k$ indicate a logical parameter p_k of a replica o_i.

2.2 Types of operations

Each operation op^k is performed to manipulate a logical parameter p_k of a replica o_i. There are two types of operations, *read* (r^k) and *write* (w^k) on a logical parameter p_k, i.e. $op \in \{r, w\}$. A transaction reads a logical parameter p_k of a replica o_i in a read operation r^k ($k \geq 1$). A read operation r^0 is used to read the whole state of a replica o_i denoted by the logical parameters $\langle p_0, p_1, \dots, p_l \rangle$.

In the write operation $w^k(x)$, the value x is overwritten to the logical parameter p_k of the replica o_i. For example, suppose a logical parameter p_k shows the frame rate of a replica o_i. In a write operation $w^k(20)$, the frame rate parameter p_k of the replica o_i is changed with 20 fps. Here, the physical state of the replica o_i is changed so that the frame rate is 20 fps in a traditional write operation.

We consider two types of write operations $w^k(x)$ on a parameter p_k of a replica o_i in this paper:

1. Materialization type of write.
2. Unmaterialization type of write.

In one type of write operation $w^k(x)$, not only the logical parameter p_k but also the physical state of the replica o_i are changed. This type of write operation is referred to as *materialization* write which is a traditional write operation. In another type of write operation $w^k(x)$, only the logical parameter p_k is changed although the physical state is not changed in the replica o_i. This type of write operation is referred to as *unmaterialization* write. We here introduce a *physical* parameter s_k which shows the value of the kth parameter of the physical state of a replica o_i. A physical state of a replica o_i means a state which is physically stored in a computer. Thus, a tuple $\langle s_0, s_1, \dots, s_l \rangle$ of the physical parameters shows the physical state of a replica o_i. On the other hand, a tuple $\langle p_0, p_1, \dots, p_l \rangle$ of the logical parameters denotes a logical state of a replica o_i which shows a current state but may be different from the physical state. If a materialization write operation $w^k(x)$ is performed on a replica o_i, both the logical parameter p_k and the physical parameter s_k of the replica o_i are changed with a new value x. That is, $p_k = s_k$ in the replica o_i. On the other hand, if an unmaterialization write $w^k(x)$ is performed on a replica o_i, only the logical parameter p_k is changed with a new value x but the physical parameter s_k is not changed. The change of the physical parameter s_k means that the physical state of a replica o_i is changed.

A replica o_i where $p_k = s_k$ for every logical parameter p_k is referred to as *materialized*. In an unmaterialized replica o_i, $p_k \neq s_k$ for some logical parameter p_k. Here, an unmaterialized replica o_i is referred to as *materializable* iff the logical parameter p_k is equal to or poorer than the physical parameter s_k ($p_k \preceq s_k$) for every logical parameter p_k. Suppose a replica o_i is not materialized, i.e. $p_k \neq s_k$ for some logical parameter p_k which shows the frame rate. Suppose the logical parameter p_k of a frame rate is 20 fps and the physical parameter s_k is 40 fps. That is, the physical parameter s_k is richer than the logical parameter p_k ($p_k \preceq s_k$). Here, we can obtain the current physical state of the replica o_i just by decreasing the frame rate without obtaining additional data not in the physical state. That is, the replica o_i is materializeable. On the other hand, if $p_k = 40$ fps and $s_k = 20$ fps, i.e. the logical parameter p_k is poorer than the physical parameter s_k ($p_k \succ s_k$), the current physical state of the replica o_i cannot be obtained without additional frame data which is not in the physical state of the replica o_i.

For a pair of different logical QoS parameters p_k and p_h ($k \neq h$), a read operation r^k is compatible with a write operation w^h and a write operation w^k is also compatible with a write operation w^h. A read operation r^0 conflicts with a write operation w^k of every logical parameter p_k ($k \geq 0$) since the whole state of the replica o_i is read in a read operation r^0. Let O be a set of replicas of an object o in a system S. Let $Q_{k,op} (\subseteq O)$ shows a quorum of replicas where an operation op^k on a parameter p_k is performed. For every pair of operations op_1^k and op_2^k, if op_1^h and op_2^k conflict with one another, $Q_{k,op_1} \cap Q_{h,op_2} \neq \phi$ and $Q_{k,op_1} \cup Q_{h,op_2} = O$.

Write operations are further classified into the following types [13, 14] with respect to whether some data is added or removed on a replica o_i:

1. Enriching type of write.
2. Impoverishing type of write.

In an enriching type of write operation $w^k(x)$, some data not in a replica o_i is required to be added to the replica o_i. For example, colour data has to be added to a monochromatic replica to change with a coloured replica. That is, the value x is richer than the physical parameter s_k of a replica o_i ($x \succ s_k$). On the other hand, a replica o_i can be updated just by removing data in the replica o_i in an impoverishing type of write operation $w^k(x)$. That is, $x \preceq s_k$. For example, a coloured replica can be changed with a monochromatic one just by removing the colour data.

3 Multimedia quorum-based (MQB) protocol

3.1 Parameters

Let O be a set $\{o_1, \dots, o_n\}$ of replicas of a multimedia object o in a system S. Each replica o_i is characterized in terms of logical parameters $\langle p_0, p_1, \dots, p_l \rangle$ ($l \geq 1$). The first logical parameter p_0 stands for the data structure of the replica

o_i which shows a *part_of* relation of subobjects. For example, a replica o_i is composed of three subobjects a, b, and c. Here, the logical parameter p_0 is a tuple $\langle a, b, c\rangle$ of the subobjects in the replica o_i. The other logical parameters $p_1, \dots, p_l$ show QoS parameters. For each logical parameter p_k, there are a pair of operations r^k and w^k to read and write the parameter p_k of a replica o_i, respectively ($k = 0, 1, \dots, l$). For example, the *colour* parameter p_1 takes one of values; fc (fully coloured), gs (gray-scaled), and mc (monochromatic). The parameter p_2 is a QoS parameter which shows the frame rate, e.g. 40 fps. Here, suppose a replica o_i is composed of fully coloured movie subobjects a, b, and c with frame rate 40 fps. A logical state of the replica o_i is given in a tuple $\langle\langle a, b, c\rangle, fc, 40\rangle$ of logical parameters, where $p_0 = \langle a, b, c\rangle$, $p_1 = fc$, and $p_2 = 40$. In the write operation $w^0(x)$, a subobject x is deleted, added, or modified. For example, suppose the subobject c in the replica o_i is deleted in a delete operation $w^0(c)$. The replica o_i is changed with a new state $\langle\langle a, b\rangle, fc, 40\rangle$. A *delete* is an impoverishing type of write operation and *add* is an enriching type of write operation. Suppose a QoS parameter p_2 stands for frame rate. In the write operation $w^2(20)$, the frame rate parameter p_2 of a replica o_i is changed with 20 fps. This is an impoverishing write operation since $40 \succeq 20$. The replica o_i is changed with a new state $\langle\langle a, b\rangle, fc, 20\rangle$.

Each replica o_i is characterized in a tuple $\langle s_0, s_1, \dots, s_l\rangle$ of physical parameters in addition to the logical parameters $\langle p_0, p_1, \dots, p_l\rangle$. Initially, $s_k = p_k$ for each parameter p_k in a replica o_i. A tuple $\langle s_0, s_1, \dots, s_l\rangle$ of the physical parameters shows a physical state of a replica o_i which is really stored in a computer. Hence, each parameter s_k is referred to as physical parameter of a replica o_i.

On receipt of a write operation $w^k(x)$, the logical parameter p_k of a replica o_i is updated with a new value x. However, the physical state of the replica o_i is not changed if w^k is an impoverishing type of write operation in our approach to reducing the processing overhead. On the other hand, the physical state of the replica o_i is changed in an enriching type of write operation, i.e. not only the logical parameter p_k but also the physical parameter s_k are updated. In an impoverishing type of write operation $w^k(x)$, the logical parameter p_k and the version counter vc_k are updated in a replica o_i while the physical parameter s_k is not updated. Thus, a tuple $\langle p_0, p_1, \dots, p_l\rangle$ of the logical parameters shows a current logical state of a replica o_i. On the other hand, a tuple $\langle s_0, s_1, \dots, s_l\rangle$ of the physical parameters denotes a current physical state of the replica o_i which is really stored in a computer.

If a physical parameter s_k is the same as the logical parameter p_k, the logical parameter p_k is referred to as *materialized*. A tuple $\langle p_0, p_1, \dots, p_l\rangle$ of the logical parameters shows a newest state of a replica o_i. The logical parameter p_k is materialized in an enriching write operation w^k while not materialized in an impoverishing type of write operation w^k. If every logical parameter p_k of a replica o_i is materialized, the replica o_i is referred to as *materialized*, where $p_k = s_k$ for every logical parameter p_k. It is noted the logical parameter p_k is equal to or richer than the physical parameter s_k in a replica o_i ($o_i.p_k \succeq o_i.s_k$) since the replica o_i is materialized each time an enriching write operation w^k is performed but is not materialized, just the logical parameter p_k is changed in an impoverishing write operation.

Next, suppose a read operation r^0 is issued to a replica o_i to read the logical data structure parameter p_0. A newest replica o_i is first selected in a read quorum Q_{0r} and then is materialized. The whole state of the replica o_i is read in the read operation r^0.

Let us consider a replica $o_i = \langle\langle a, b, c\rangle, fl, 40\rangle$ of a *movie* object o which is composed of three subobjects a, b, and c which are fully coloured with 40 fps. Here, the logical parameters $\langle p_0, p_1, p_2\rangle$ are the same as the physical parameters $\langle s_0, s_1, s_2\rangle$ in the replica o_i. First, the frame rate parameter p_2 is changed with 20 fps. Then, the subobject c is deleted in a write operation $w^0(c)$ which is also an impoverishing type. Here, the logical parameter p_0 of data structure is changed with $\langle a, b\rangle$. The physical parameters $\langle s_0, s_1, s_2\rangle$ are still $\langle\langle a, b, c\rangle, fl, 40\rangle$ while the logical parameters $\langle p_0, p_1, p_2\rangle$ are changed with $\langle\langle a, b\rangle, fl, 20\rangle$. The physical data structure parameter $s_0 = \langle a, b.c\rangle$ is richer than the logical parameter $p_0 = \langle a, b\rangle$ ($s_0 \succeq p_0$) and the QoS parameters $p_2 = 40$ fps is richer than $s_2 = 20$ fps ($s_2 \preceq p_2$). A tuple $\langle\langle a, b, c\rangle, fl, 40\rangle$ of the physical parameters indicates a current physical state of the replica o_i. A tuple $\langle\langle a, b\rangle, fl, 20\rangle$ of the logical parameters denotes a current logical state of the replica o_i to be changed. Here, the replica o_i is not materialized. The physical state of a replica o_i shown by the physical parameters $\langle s_0, s_1, \dots, s_l\rangle$ is older than the logical state denoted by the logical parameters $\langle p_0, p_1, \dots, p_l\rangle$ if $\langle s_0, s_1, \dots, s_l\rangle \neq \langle p_0, p_1, \dots, p_l\rangle$.

Suppose a replica o_i is not materialized but each logical parameter p_k can be richer than a physical parameter s_k. There is a materialization procedure $mat(o_i)$ by which the physical state of a replica o_i is changed from $\langle s_0, s_1, \dots, s_l\rangle$ to the new state $\langle p_0, p_1, \dots, p_l\rangle$, i.e. the replica o_i is materialized. Here, the physical state of the replica o_i is really changed. Computation resources are spent to change the physical state of the replica o_i, i.e. materialize the replica o_i. For example, data in the replica o_i is decoded and encoded. Hence, we try to reduce the number of materializations to be done in replicas in this paper. Then, the physical state $\langle s_0, s_1, \dots, s_l\rangle$ of the replica o_i is changed with $\langle p_0, p_1, \dots, p_l\rangle$. For example, the physical state of a replica o_i is $\langle\langle a, b, c\rangle, fc, 40\rangle$ which is composed of three subobjects a, b and c with QoS parameters $p_1 = fl$ and $p_2 = 40$ fps. A tuple of the logical parameters $\langle p_0, p_1, p_2\rangle$ of the replica o_i is $\langle\langle b, c\rangle, fc, 20\rangle$. Here, the replica o_i can be materialized to the physical state $\langle\langle a, b\rangle, fc, 20\rangle$ by removing

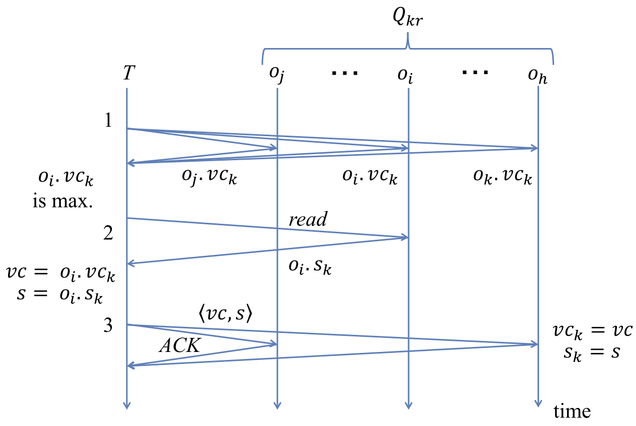

Fig. 1 Read procedure r^k

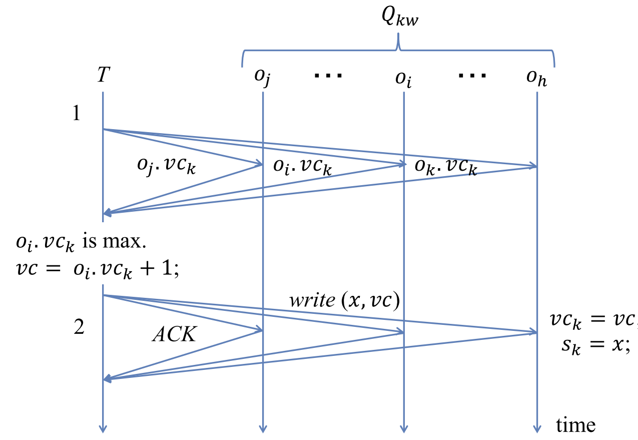

Fig. 2 Write procedure $w^k(x)$

the subobject c and decreasing the frame rate to 20 fps. Here, the physical parameters $\langle s_0, s_1, s_2\rangle$ get the same as the logical parameters $\langle p_0, p_1, p_2\rangle = (\langle\langle b, c\rangle, fc, 20\rangle)$, i.e. the replica o_i is materialized.

3.2 Version vector

For each logical parameter p_k, there is a version counter vc_k [14]. Initially, the version counter vc_k of each logical parameter p_k is 0 in each replica o_i. Let $o_i.vc_k$ stand for the version counter vc_k of a replica o_i, respectively. $o_i.V$ shows a vector $\langle vc_0, vc_1, \dots, vc_l\rangle$ of the version counters in a replica o_i. Suppose a transaction T issues an operation op^k to manipulate the parameter p_k in a quorum $Q_{k,op}$. If a write operation $w^k(x)$ is performed on a replica o_i, the version counter $o_i.vc_k$ is incremented by one. Here, a replica o_i is newer than a replica o_j iff $o_i.V > o_j.V$. In a quorum $Q_{k,op}$, a replica o_i whose version counter vc_k is maximum has the newest parameter p_k. If a read operation r^k is issued, the logical parameter p_k of a newest replica in a read quorum Q_{kr} is read.

3.3 Read and write procedures of QoS parameters

We discuss how to manipulate replicas in a quorum. We first consider a read operation r^k and a write operation $w^k(x)$ for a logical QoS parameter p_k ($k = 1, \dots, l$). Here, the transaction T obtains the newest value of the logical parameter p_k by the following procedure (refer to Fig. 1).

[Read procedure of r^k]

1. Find a newest replica o_i in a read quorum Q_{kr} whose version counter $o_i.vc_k$ is maximum, i.e. $o_i.vc_k = \max(o_j.vc_k \mid o_j \in Q_{kr})$. $vc = o_i.vc_k$.
2. Read the logical parameter $o_i.p_k$ in the replica o_i.
3. For every replica o_j ($j \neq i$) in the quorum Q_{kr}, $o_j.p_k = o_i.p_k$ and $o_j.vc_k = vc$.

The transaction T finds a replica o_i which has the newest value of the logical parameter p_k in the read quorum Q_{kr}. That is, the replica o_i has the largest version counter vc_k in the quorum Q_{kr}. Then, the transaction T reads the logical parameter p_k of the replica o_i.

Next, a transaction T issues a write operation $w^k(x)$ to write a value x in a QoS parameter p_k of replicas in a write quorum Q_{kw} (refer to Fig. 2).

[Write procedure of $w^k(x)$]

1. Find a replica o_i in a write quorum Q_{kw} whose version counter $o_i.vc_k$ is maximum, i.e. newest replica o_i. $vc = o_i.vc_k + 1$.
2. For every replica o_j ($\neq o_i$) in the quorum Q_{kw}, the value x is written to the logical parameter p_k of the replica o_j and the version vector vc_k is changed with the maximum value vc, i.e. $o_j.p_k = x$ and $o_j.vc_k = vc$.
3. If $w^k(x)$ is an enriching type of write operation, i.e. the logical parameter p_k is richer than the physical parameter s_k ($p_k \succ s_k$), every replica o_j in the quorum Q_{kw} is materialized by the materialization procedure $mat(o_j)$. The physical state of the replica o_i is changed with a new state shown by a tuple $\langle p_0, p_1, \dots, p_l\rangle$ of the logical parameters. Now, the physical state of the replica o_i is up-to-date.

It is noted that the new parameter value x is written to the logical parameter p_k of a replica o_i in a write operation $w^k(x)$. If $w^k(x)$ is an enriching write operation, the value x is written to the physical parameter s_k of the replica o_i in addition to the logical parameter p_k, i.e. the state of the replica o_i is materialized.

3.4 Read and write procedures of a data structure parameter

Next, we consider a read operation r^0 and a write operation $w^0(x)$ for the data structure parameter p_0. First, a transaction T issues a write operation $w^0(x)$ to write a value x in the data structure parameter p_0 in a write quorum Q_{0w}. In fact, $w^k(x)$ means an add or delete operation of a subobject x in a replica o_i. A transaction T writes a value x to the data structure parameter p_0 of a replica as follows.

[**Write procedure of** $w^0(x)$]

1. Find a newest replica o_i whose version counter vc_0 is maximum in a write quorum Q_{0w}. $vc = o_i.vc_0 + 1$. $o_i.p_0 = x$ and $o_i.vc_0 = vc$.
2. For every replica o_j in the quorum Q_{0w}, $o_j.p_0 = x$ and $o_j.vc_0 = vc$.
3. If $w^0(x)$ is an enriching write operation, every replica o_i is materialized by the materialization procedure $mat(o_i)$ in the quorum Q_{0w}.

If a write operation $w^0(x)$ is an impoverishing type of write operation like delete of a subobject, the value x is just recorded in the logical parameter p_0 but the physical parameter s_0 of the replica o_i is not updated. On the other hand, each replica o_i is materialized in an enriching type of write operation w^0. The version counter v_0 in every replica o_i is increased to the maximum value vc.

Next, a transaction T issues a read operation r^0 to a read quorum Q_{0r}. Here, it is noted the transaction T has to read a whole state of a newest replica in the read quorum Q_{0r} while only a logical parameter p_k is read in another read operation r^k ($k > 0$). The transaction T reads the data structure parameter p_0 of a replica in the read quorum Q_{0r} as follows.

[**Read procedure of** r^0]

1. Find a newest replica o_i such that $o_i.vc_k \geq o_j.vc_k$ for every parameter p_k and for every replica o_j in a read quorum Q_{0r}. If found, $vc = o_i.vc_0$ which is the maximum value of the version counter vc_0 in the quorum Q_{0r}. The transaction T reads the whole state of the replica o_i and go to step 4.
2. If not found, find a replica o_i whose version counter vc_0 is maximum in the read quorum Q_{0r}. If the replica o_i is found, $vc = o_i.vc_0$. For each logical parameter p_k ($k \neq 0$), find a replica o_j whose version counter vc_k is maximum, i.e. o_j has the newest value of the logical parameter p_k. $o_i.s_k = o_j.s_k$ and $o_i.vc_k = o_j.vc_k$.
3. The replica o_i is materialized by the materialization procedure $mat(o_i)$. The transaction T reads the whole state of the replica o_i.
4. For every replica o_j ($\neq o_i$) in Q_{0r}, $o_j.p_k = o_i.p_k$ and $o_j.vc_k = o_i.vc_k$ for every logical parameter p_k.

In a read operation r^0, a newest materialized replica o_i is first found in the read quorum Q_{0r}. If not found, a replica o_i whose version counter vc_0 is maximum in the read quorum Q_{0r} is found. If some logical parameter p_k of the replica o_i is not newest, a replica o_j with a newest value x of the logical parameter p_k is found in the quorum Q_{0r}. The logical parameter $o_i.p_k$ is changed with the newest value x. Then, the replica o_i is materialized. The transaction T reads the materialized, newest replica o_i in the quorum Q_{0r}. The logical parameter p_k and version counter vc_k in every replica o_j are updated with the same values after step 4 as the newest replica o_i in the quorum Q_{0r} after step 4.

Here, there are two ways to do for the other replicas than the newest replica o_i after step 4.

1. Read-materialization (*RM*).
2. Just-read (*R*).

In one way, every replica o_j in the quorum Q_{0r} is materialized by the materialization procedure $mat(o_j)$. This means the physical state of every replica in the quorum Q_{0r} gets the newest after the transaction T reads the replica o_i. This strategy is referred to as *read-materialization* (*RM*). *MQB-RM* stands for the MQB protocol with RM strategy.

In another way, only the replica o_i is materialized. Here, only one replica o_i is materialized in a read operation while the logical parameters $p_0, p_1, \ldots, p_l$ of every other replica are newest values in a read quorum Q_{0r}. This strategy is referred to as *just-read* (*R*) one. In the MQB protocol, the *R* strategy is taken. Since only a newest replica o_i which a transaction reads is materialized, the processing overhead of replicas can be reduced.

4 Evaluation

4.1 Environment

We evaluate the MQB and MQB-RM protocols compared with the QB protocol in terms of processing overhead of each replica. In the evaluation, we assume there is a set O ($= \{o_1, \ldots, o_n\}$) of n (≥ 1) replicas $o_1, \ldots, o_n$. Each replica o_i has logical parameters $\langle p_0, p_1, \ldots, p_l \rangle$ where p_0 shows a data structure parameter and each p_k is a QoS parameter ($k = 1, \ldots, l$). In the QB protocol, every replica is updated, i.e. materialized in a write quorum Q_{kw} each time a write operation w^k is performed on a logical parameter p_k ($k = 0, 1, \ldots, l$). That is, every write operation is a materialization type. In a read operation r^k, the logical parameter p_k in the newest replica o_i is first read in a read quorum Q_{kr}. Then, every other replica o_j is updated to be the newest one in the read quorum Q_{kr}. On the other hand, a replica is not materialized in an impoverishing write operation w^k with the MQB protocol. In a read operation r^0, a newest replica o_i is first found. Then, the replica o_i has to has materialized if o_i is not materialized.

Let γ (≤ 1) show the *read* ratio, i.e. the ratio of the number of read operations to the total number of operations issued by transactions. Here, $(1 - \gamma)$ indicates the write ratio.

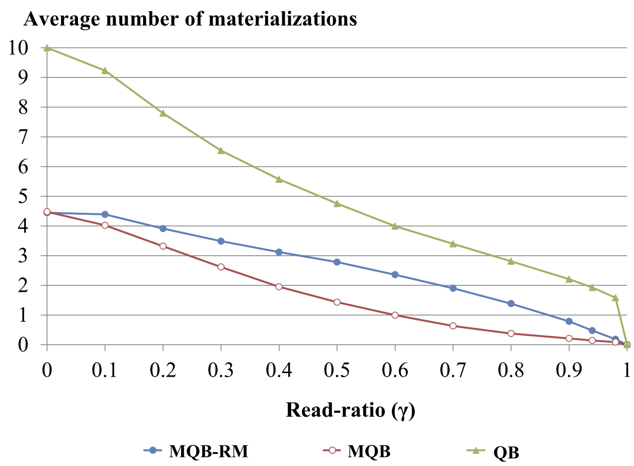

Fig. 3 Average number of materializations for one operation ($n = 10$, $l = 5$)

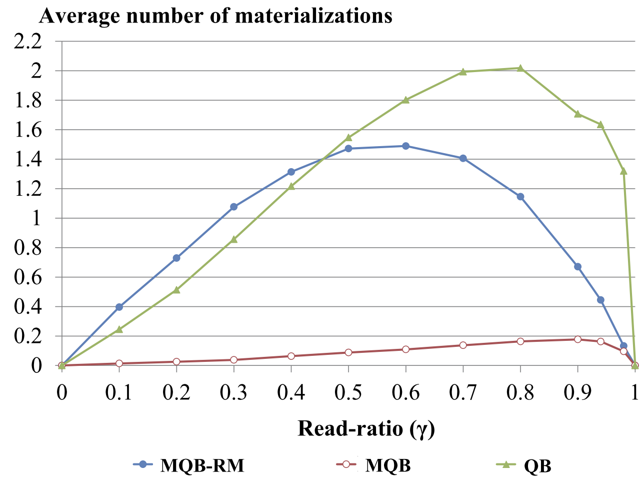

Fig. 4 Average number of materializations for read ($n = 10$, $l = 5$)

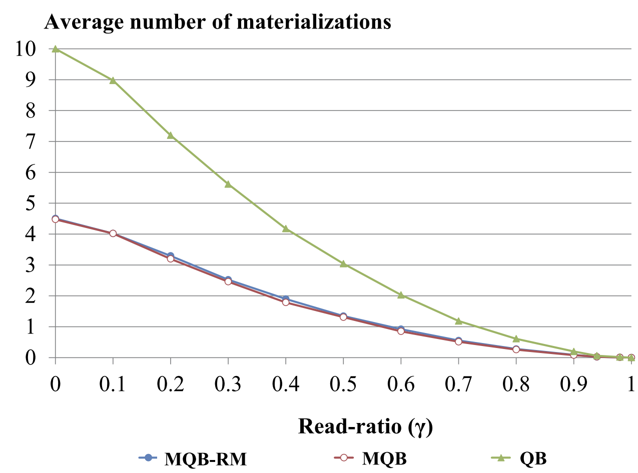

Fig. 5 Average number of materializations for write ($n = 10$, $l = 5$)

In this paper, we assume the sizes $|Q_{kr}|$ and $|Q_{kw}|$ of the quorums Q_{kr} and Q_{kw} are in inverse proportion to the read ratio γ and write ratio $(1 - \gamma)$. That is, if a read operation r^k is more frequently issued than a write operation w^k, the size of the read quorum Q_{kr} is smaller than the write quorum Q_{kw}. We assume $|Q_{kr} \cap Q_{kw}| = 2$ in this evaluation. Each time an operation op^k on the logical parameter p_k is issued, the number of replicas are randomly selected to be included in a quorum $Q_{k,op}$. In this evaluation, we assume the number l of QoS parameters is five, i.e. $l = 5$.

In the simulation, we assume one transaction issues one operation op^k. The totally 2,000 transactions are serially issued. A logical parameter p_k to be manipulated in an operation op^k is randomly selected ($k \in \{0, 1, \ldots, l\}$). A type of operation $op \in \{r, w\}$ is also randomly selected so that the read ratio γ is satisfied. Then, replicas to be in a quorum $Q_{k.op}$ are randomly selected in the replica set O for each operation op^k. In each write operation $w^k(x)$, a value x is written to the logical parameter p_k of a replica o_i. Here, the value x is randomly selected as $x \in \{0, \ldots, 99\}$. If $w^k(x)$ is an enriching type, the value x is also written to the physical parameter s_k in the MQB protocol. If the value x is smaller than the physical parameter s_k, i.e. current physical value of the kth parameter in the replica o_i, the write operation $w^k(x)$ is considered to be an impoverishing type. Otherwise, $w^k(x)$ is an enriching type of write operation. Here, the value x is written to the logical parameter p_k as well as the physical parameter s_k. In a read operation r^k, a transaction reads a newest replica o_i in a read quorum Q_{kr}. If a newest replica o_i is not materialized, the replica o_i is materialized and then is read by the transaction.

4.2 Evaluation results

Figures 3, 4, and 5 show the average numbers of materializations of replicas done for each operation in the MQB protocol, MQB-RM, and QB protocols. Figure 3 shows the average number of materializations of replicas for each operation in the MQB, MQB-RM, and QB protocols for the read ratio γ with $n = 10$ and $l = 5$. The number of materializations of replicas in the MQB and MQB-RM protocols can be reduced to about 30 % and 50 %, respectively, of the QB protocol for $\gamma = 0.5$ as shown in Fig. 3. This means, the MQB and MQB-RM protocols imply the smaller processing overhead in each replica than the QB protocol. The processing overhead in the MQB protocol is the smallest.

Figures 4 and 5 show the average numbers of materializations for one read operation and one write operation, respectively, for read ratio γ in the MQB, MQB-RM, and QB protocols. Here, $n = 10$ and $l = 4$. In the MQB protocol, the average number of materializations for a read operation can be drastically reduced. For example, the average number of materializations in the MQB protocol is almost 10 % of the QB protocol for $\gamma = 0.8$. On the other hand, the average number of materializations for a write operation in the MQB

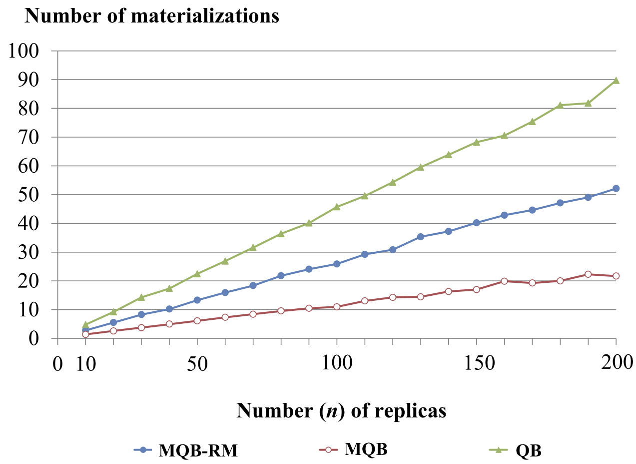

Fig. 6 Average number of materializations ($l = 5$, $\gamma = 0.5$)

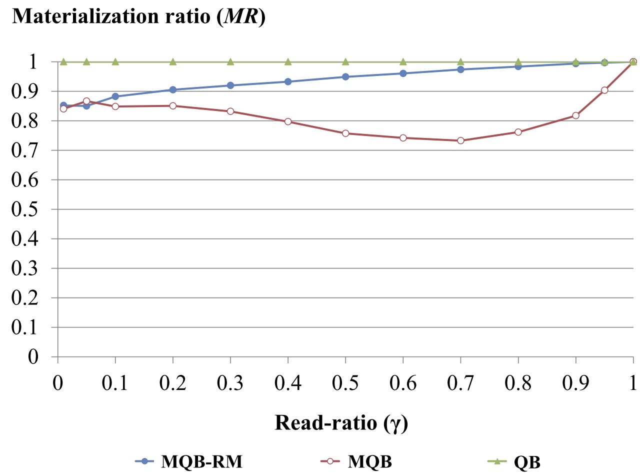

Fig. 7 Materialization-ratio ($n = 10$, $l = 5$)

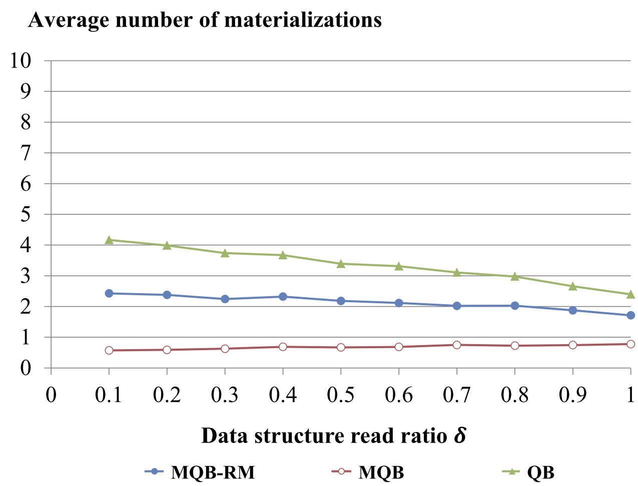

Fig. 8 Average number of materialization for r^0 ($n = 10$, $l = 5$, $\gamma = 0.6$)

protocol is the same as the MQB-RM and can be reduced by 55 % compared with the QB protocol.

Figure 6 shows the number of materializations in the MQB, MQB-RM, and QB protocols for the total number n of replicas where $l = 5$ and $\gamma = 0.5$. As the number n of replicas increases, the number of materializations of replicas linearly increases in every protocol. For example, the numbers of materializations in the MQB and MQB-RM protocols are 20 % and 50 % of the QB protocol for $n = 100$, respectively. The processing overheads of the MQB and MQB-RM protocols are smaller than the QB protocol. The MQB protocol supports the smallest processing overhead in the protocols.

In the MQB protocol, there is a newest replica o_i but the replica o_i may not be materialized in a read quorum Q_{0r} when a read operation r^0 is issued. Let MR be the read materialization ratio, i.e. the ratio of read operations in which a newest, materialized replica is found to the total number of read operations issued ($0 \leq MR \leq 1$). Here, it is noted $MR = 1$ for every read ratio γ in the QB protocol. That is, a transaction can necessarily find a newest, materialized replica in a read quorum with the QB protocol. In the MQB protocol, MR is smaller than the QB protocol since replicas are not necessarily materialized in a quorum. For example, $MR = 0.65$ for $\gamma = 0.7$ are $MR = 0.83$ for $\gamma = 0.4$ in the MQB protocol. That is, if 70 % ($\gamma = 0.7$) of operations are read ones, there is probability 0.35 that a replica which is read is not materialized in the MQB protocol. $MR = 0.93$ and $MR = 0.98$ for $\gamma = 0.4$ and $\gamma = 0.7$, respectively, in the MQB-RM protocol. It takes time to materialize a replica. For example, for $\gamma = 0.5$, if one hundred read operations are issued, we have to materialize a replica to perform 35 read operations in the MQB protocol while 5 read operations in the MQB-RM protocol. Hence, it takes a longer time to read a replica in the MQB protocol than the MQB-RM and QB protocols. One idea is that the MQB protocol is taken if the read ratio γ is smaller, e.g. $\gamma \leq 0.4$ and the MQB-RM protocol is taken for $\gamma > 0.4$.

Next, we assume a read operation r^0 for the data structure parameter p_0 is randomly issued with probability δ while every write operation w^k is randomly issued as discussed here. Another read operation r^k ($1 \leq k \leq l$) is randomly issued to read the logical parameter p_k with probably $(1 - \delta)/l$. Here, $\delta = 1/(l + 1)$ means that every read operation r^k ($k = 0, 1, \dots, l$) is randomly issued as evaluated in Figs. 4–7. The larger the ratio δ is, the more often the whole state of a replica o_i is read. In order to read the whole state of a replica o_i, the replica o_i has to be materialized. Figure 8 shows the average number of materializations in the MQB, MQB-RM, and QB protocols for the data structure read operation ratio δ where $n = 10$, $l = 5$, and $\gamma = 0.6$. The average number of materializations can be reduced in the MQB are MQB-RM protocols than the QB protocol. In the MQB protocol, the average number of materializations is almost independent of the data structure read ratio δ.

5 Concluding remarks

In this paper, we discussed how to reduce the processing overhead of each replica of a multimedia object in the multimedia quorum-based (MQB) protocol. A multimedia object is characterized in terms of not only data structure parameter p_0 but also QoS parameters $p_1, \dots, p_l$. There are read and write operations r^k and w^k for each parameter p_k $(k = 0, 1, \dots, l)$. There are enriching and impoverishing types of write operations. Some data has to be added to a replica in an enriching write operation like *add*. On the other hand, just data in a replica is removed in an impoverishing write operation like *delete*. In order to increase the performance of the MQB protocol, impoverishing write operations are just recorded in every replica while enriching operations are performed on every replica in a quorum. In a read operation to read a whole state of a replica, if a newest replica o_i is not materialized in a read quorum, the replica o_i is read after materialized. In the MQB-RM protocol, replicas in a read quorum are updated after a transaction reads the newest replica. We evaluated the MQB protocol and the MQB-RM protocol compared with the traditional QB protocol in terms of processing overhead of each replica. We showed the number of materializations of replicas can be reduced in the MQB and MQB-RM protocols compared with the QB protocol. The MQB protocol implies the minimum average number of materializations, i.e. smallest processing overhead.

In scalable systems, we have to more reduce the processing overhead of each replica. We are now discussing an extended MQB protocol where only some number, not all of replicas in a wrote quorum are materialized in a enriching type of write operation. We are also evaluating the communication overhead in addition to the processing overhead in a scalable system.

References

1. Chuang, C., Kao, S.: Adjustable flooding-based discovery with multiple QoSs for cloud services acquisition. Int. J. Web Grid Serv. **7**(2), 208–224 (2011)
2. DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., Vogels, W.: Dynamo: Amazon's Highly Available Key-value Store. ACM SIGOPS Oper. Syst. Rev. **14**(6), 205–220 (2007)
3. Enokido, T., Higaki, H., Takizawa, M.: Group protocol for distributed replicated objects. In: Proc. of the 27th International Conference on Parallel Processing (ICPP-98), pp. 570–577 (1998)
4. Enokido, T., Hori, K., Takizawa, M., Raynal, M.: Quorum-based multi-invocation model for replicated objects. J. Concurr. Eng. Res. Appl. **12**(3), 185–194 (2004)
5. Flahive, A., Taniar, D., Rahayu, W.: Ontology as a service (OaaS): a case for sub-ontology merging on the cloud. J. Supercomput. (2012, to appear). doi:10.1007/s11227-011-0711-4
6. Gifford, D.K.: Weighted voting for replicated data. In: Proc. of the 7th Symposium on Operation Systems Principles (SOSP '79), pp. 150–162 (1979)
7. Gray, J.: Notes on database operating systems. In: Lecture Notes in Computer Science, vol. 60. Springer, Berlin (1978)
8. Helal, A., Bhargava, B.: Performance evaluation of the quorum consensus replication method. In: Proc. of Computer Performance and Dependability Symposium on Computer Performance and Dependability Symposium, pp. 165–172 (1995)
9. Herlihy, M.: A quorum-consensus replica method for abstract data types. ACM Trans. Comput. Syst. **4**(1), 32–53 (1986)
10. Hofmann, P., Woods, D.: Cloud computing: the limits of public clouds for business applications. IEEE Internet Comput. **14**, 90–93 (2010). ISBN 1089-7801
11. Kim, W.: Could computing adoption. Int. J. Web Grid Serv. **7**(3), 225–245 (2011)
12. Nghiem, T.P., Waluyo, A.B., Tanier, D.: A pure peer-to-peer approach for kNN query processing in mobile ad hoc networks. Pers. Ubiquitous Comput. (2012, to appear). doi:10.1007/s00779-012-0545-y
13. Ohkawara, T., Aikebaier, A., Enokido, T., Takizawa, M.: Quorums-based replication of multimedia objects in distributed systems. In: Proc. of the International Conference on Network-Based Information Systems (NBiS-2011), pp. 333–340 (2011)
14. Ohkawara, T., Aikebaier, A., Enokido, T., Takizawa, M.: Completable quorums of multimedia objects. In: Proc. of IEEE the 26th International Conference on Advanced Information Networking and Applications (AINA-2012), pp. 597–604 (2012)
15. Schollmeier, R.: A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In: Proc. of the First International Conference on Peer-to-Peer Computing (P2P-2001), pp. 101–102 (2001)
16. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: a scalable peer-to-peer lookup service for Internet applications. In: Proc. of ACM the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '01), pp. 149–160 (2001)
17. Tanaka, K., Hasegawa, K., Takizawa, M.: Quorum-based replication in object-based systems. J. Inf. Sci. Eng. **6**(7), 317–331 (2000)
18. Tanaka, K., Takizawa, M.: Quorum-based locking protocol for replicas in object-based systems. In: Proc. of IEEE the 5th International Symposium or Autonomous Decentralized Systems (ISORC-2001), pp. 196–203 (2001)
19. Waluyo, A.B., Taniar, D., Rahayu, W., Aikebaier, A., Takizawa, M., Srinivasan, B.: Trustworthy-based efficient data broadcast model for P2P interaction in resource-constrained wireless environments. J. Comput. Syst. Sci. (2012, to appear). doi:10.1016/j.jcss.2011.10.019
20. Waluyo, A.B., Taniar, D., Rahayu, W., Aikebaier, A., Takizawa, M., Srinivasan, B.: Mobile peer-to-peer data dissemination in wireless ad-hoc networks. J. Inf. Sci. (2012, to appear). doi:10.1016/j.ins.2012.07.035

Tadateru Ohkawara received his B.E. Degree in computer science from Seikei University, Tokyo, Japan, in 2011. He is graduate student in Seikei University, Tokyo, Japan, from 2011. His research embarks on replication of multimedia object and focused on how to maintain mutually consistency of replicas in P2P overlay networks.

Ailixier Aikebaier received his B.E. Degree in Computers and Systems Engineering from XinJiang University, China, in 2000, and M.E. Degree in Computers and Systems Engineering from Tokyo Denki University, Japan in 2009. He got his Ph.D. in Computer Science at Seikei University, 2011. He is currently working for National Institute of Information and Communications Technology (NICT), Japan. He won the best paper award at CISIS2008 and CISIS2010. His research interests include distributed systems, P2P networks, trust and reputation problems in wireless sensor networks, and fault-tolerant systems.

Tomoya Enokido (M'02) received B.E. and M.E. Degrees in Computers and Systems Engineering from Tokyo Denki University, Japan in 1997 and 1999, respectively. After that he worked for NTT Data Corporation, he joined Tokyo Denki University in 2002. He received his D.E. Degree in Computer Science from Tokyo Denki University in 2003. After that he worked for Computers and Systems Engineering as a research associate, he joined Faculty of Business Administration of Rissho University in 2005. He is an associate professor in the Faculty of Business Administration, Rissho University. His research interests include distributed systems. He is a member of IEEE.

Makoto Takizawa (M'87) received the D.E. Degree in computer science from Tohoku University, Sendai, Japan. He is a Professor with the Department of Computer and Information Science, Seikei University, Tokyo, Japan. He was a Professor and the Dean of the Graduate School of Science and Engineering, Tokyo Denki University, Saitama, Japan. He was a Visiting Professor at Gesellschaft für Mathematik und Datenverarbeitung-Integrated Publication and Information Systems Institute (GMD-IPSI), Keele University, Keele, UK, and at Xidian University, Xi'an, China. His research interests include distributed systems and computer networks. Dr. Takizawa was on the Board of Governors and is a Golden Core Member of the IEEE CS. He is a Fellow of Information Processing Society of Japan (IPSJ). He chaired many international conferences such as the IEEE International Conference on Distributed Computing Systems (ICDCS), International Conference on Parallel and Distributed Systems (ICPADS), and International Conference on Database and Expert Systems Applications (DEXA). He founded IEEE International Conference on Advanced Information Networking and Applications (AINA).