Quality-based Flexible Distributed Systems *

5G－8

Tetsuo Kanezuka and Makoto Takizawa †
Tokyo Denki University ‡
e-mail{kane, taki}@takilab.k.dendai.ac.jp

1 Introduction

This paper discusses how to make a distributed object system flexible so as to satisfy the application's requirement in the change of the system environment. Each object supports other objects with quality of service (QoS). The change of the system is modeled to be the change of not only types of service but also QoS supported by the objects. We discuss equivalency and compatibility relations among the operations with respect to QoS. By using the QoS-based relations, we newly discuss a QoS-based compensating way to recover the object from the less qualified state. Finally, we discuss QoS-based replication of objects to make required QoS available even if some replicas get less qualified. Here, the replicas are not necessarily the same.

2 System Model

2.1 System configuration

A system is composed of multiple objects, $o_1, \ldots, o_n$. The objects communicate with other objects by the reliable network. Each object o_i is an encapsulation of the data structure and a collection of abstract operations $op_{i1}, \ldots, op_{il_i}$. o_i can be manipulated only through $op_{i1}, \ldots, op_{il_i}$. Operations change the state of o_i and output some data as the responses. Let $op_{ij}(s_i)$ denote a state of o_i obtained by applying op_{ij} to a state s_i of o_i. $[op_{ij}(s_i)]$ denotes the *view* of s_i by op_{ij}, i.e. the response data obtained by $op_{ij}(s_i)$. $op_{ij} \circ op_{ik}$ means that op_{ik} is computed after op_{ij}.

2.2 Quality of service (QoS)

Each object o_i supports applications with service. The service can be obtained by issuing the operations supported by o_i. Each type of service is characterized by parameters like level of resolution, number of frames, and number of colors. Quality of service (QoS) supported by o_i is given by the parameters. Even if two objects o_i and o_j support the same types of service, they may provide different levels of QoS.

The *scheme* of QoS is given a tuple of attributes $\langle a_1, \ldots, a_m\rangle$ where each attribute a_i shows a parameter. Let dom(a_i) be a *domain* of a_i, i.e. a set of possible values to be taken by a_i. A QoS *instance* q of the scheme $\langle a_1, \ldots a_m\rangle$ is given in a tuple of values, i.e. $\langle v_1, \ldots, v_m\rangle \in \mathrm{dom}(a_1) \times \ldots \times \mathrm{dom}(a_m)$. Let $a_i(q)$ show v_i in q. The values in dom(a_i) are partially ordered by a precedence relation $\preceq \subseteq \mathrm{dom}(a_i)^2$. A value v_1 *precedes* v_2 ($v_1 \succeq v_2$) in dom(a_i) if v_1 shows better QoS than v_2. For example, 120 × 100 [pixels] $\preceq$ 160 × 120 [pixels] for the *resolution* attribute. Let q_1 and q_2 show QoS instances of the scheme $\langle a_1, \ldots, a_m\rangle$. Let A be a subset $\langle b_1, \ldots, b_k\rangle$ of $\langle a_1, \ldots, a_m\rangle$ where $b_k \in \{a_1, \ldots, a_m\}$ and $k \leq m$. A projection $[q]_A$ of q on A is $\langle w_1, \ldots, w_k\rangle$ where $w_i = b_i(q)$ for $i = 1, \ldots, k$. A QoS *instance* q_1 of a scheme A_1 *partially dominates* q_2 of A_2 iff $a(q_1) \succeq a(q_2)$ for every attribute a in $A_1 \cap A_2$. q_1 *subsumes* q_2 ($q_1 \supseteq q_2$) iff q_1 partially dominates q_2 and $A_1 \supseteq A_2$. Let Q be a set of QoS *instances*. $q_1 \cup q_2$ and $q_1 \cap q_2$ show a *least upper bound* (*lub*) and a *greatest lower bound* (*glb*) of q_1 and q_2 in Q on $\preceq$, respectively.

2.3 Multimedia objects

In this paper, we consider multimedia objects. QoS of an object o_i has two aspects: *state* QoS, i.e. QoS obtained from the state s_i and *operation* QoS, i.e. QoS supported through the operations of o_i. For example, let us consider a video object *video* with a *display* operation as shown in Figure 1. A state s_i of o_i supports video data of 30 fps, which is a state QoS $Q(s_i)$. However, *display* can display the view $[display(s_i)]$ of the video data from s_i only at 20 fps. This is an operation QoS $Q([display(s_i)])$.

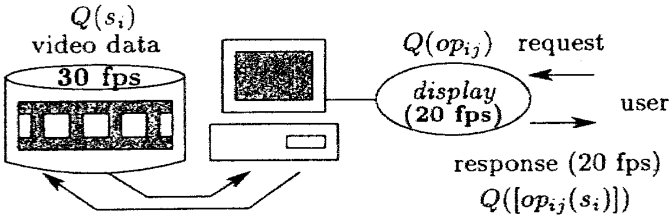


Figure 1: QoS of video object.

Let s_i denote a state of o_i and op_{ij} be an operation supported by o_i. Let $Q(s_i)$ denote the state QoS of s_i of o_i. Let $Q(op_{ij})$ denote QoS supported by op_{ij}. QoS of o_i can be viewed through the operation of o_i. Here, let $Q([op_{ij}(s_i)])$ denote QoS viewed by applying op_{ij} to the state s_i. Let $\langle s_i\rangle$ denote $\langle [op_{i1}(s_i)], \ldots, [op_{il_i}(s_i)]\rangle$, i.e. *view* of s_i. $Q(\langle s_i\rangle)$ is defined to be a tuple $\langle Q([op_{i1}(s_i)]), \ldots, Q([op_{il_i}(s_i)])\rangle$, i.e. operation QoS. $Q(\langle s_i\rangle)$ shows QoS of o_i which the users can view through the operations.

$Q(\langle s_i\rangle)$ *subsumes* $Q(\langle s_j\rangle)$ ($Q(\langle s_i\rangle) \supseteq Q(\langle s_j\rangle)$) iff there is some operation op_{ik} in o_i such that $Q([op_{ik}(s_i)]) \succeq Q([op_{ik}(s_j)])$ for every op_{ik} in o_j.

Suppose op_{ij} inserts some data d_{ij} to the state s_i of o_i. If $Q(s_i) \preceq Q(d_{ij})$, d_{ij} can be added to s_i. We consider case that $Q(s_i) \succ Q(d_{ij})$ [Figure 2(1)]. If QoS of d_{ij} is worse than s_i, d_{ij} cannot be inserted in s_i. However, users can get service from o_i through the operations of o_i. If QoS of d_{ij} viewed through an operation op_{ij} subsumes $Q(s_i)$, the users have no problem even if d_{ij} is inserted in s_i [Figure 2(2)].

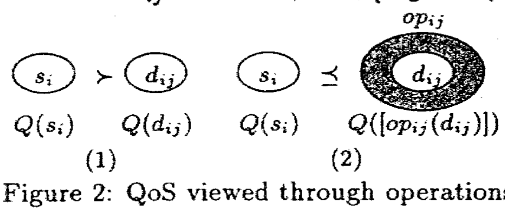


Figure 2: QoS viewed through operations.

---

*品質を考慮した柔軟な分散型システム
†金塚 哲郎　滝沢 誠
‡東京電機大学

3 QoS-Related Operations

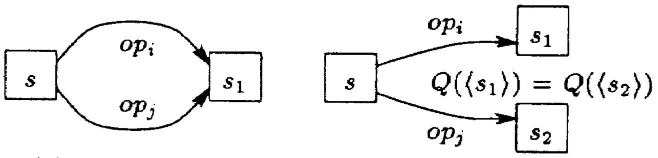
We discuss how operations $op_1, \ldots, op_l$ supported by an object o are related with respect to QoS.

3.1 Equivalency

First, we discuss equivalent relations among operations op_i and op_j supported by o. op_i is *equivalent* with op_j iff $op_i(s) = op_j(s)$ and $[op_i(s)] = [op_j(s)]$ for every state s of o [Figure 3(1)]. That is, op_i and op_j not only output the same data but also change o to the same state.

[Definition] op_i is *QoS-equivalent* with op_j iff $Q(\langle op_i(s)\rangle) = Q(\langle op_j(s)\rangle)$ for every state s of an object o. □

That is, $op \circ op_i(s)$ and $op \circ op_j(s)$ support the same view for every operation op [Figure 3(2)]. op_i is *QoS-equivalent* with op_j if $Q(\langle op_i(s)\rangle) = Q(\langle op_j(s)\rangle)$.



(1) *equivalency* (2) *QoS-equivalency*

Figure 3: Equivalent operations.

3.2 Compatibility

Next, we discuss in which order two operations op_i and op_j supported by the object o can be computed in order to keep the state of o consistent. According to the traditional theory, op_i *conflicts* with op_j if the result obtained by computing op_j after op_i is different from op_i after op_j. op_i is *compatible* with op_j unless op_i conflicts with op_j.

We now define a QoS-compatible relation among the operations op_i and op_j.

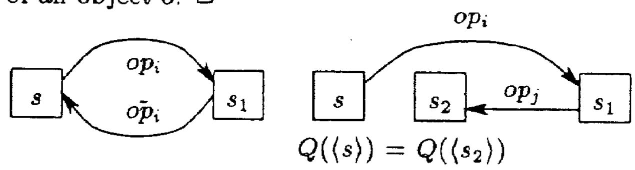
[Definition] op_i is *QoS-compatibility* with op_j iff $Q(\langle op_i \circ op_j(s)\rangle) = Q(\langle op_j \circ op_i(s)\rangle)$ for every state s of an object o.□

4 Compensation

In multimedia applications, there is a case that users undo the work done, for example, to redesign movies. One way to undo the work is to compute some operations to remove the effect done by the operations computed. op_j is a *compensating* operation of op_i if $op_i \circ op_j(s) = s$ for every state s of an object o [1]. Let $\tilde{op}_i$ denote a compensating operation of op_i. Let s' be a state obtained by computing op_i on a state s of o, i.e. $s' = op_i(s)$. Here, o can be rolled back to s if $\tilde{op}_i$ is computed on s'. For example, *append* is a compensating operation of *delete*. A pair of states s and s' of o may be considered to be equivalent from the application point of view even if s and s' are not the same.

Here, suppose a state s_1 is obtained by applying an operation op_i to a state s of an object o. Let us consider how to roll the object o back to s from s_1. One way is to compute the compensating operation $\tilde{op}_i$ of op_i on s_1 since $op_i \circ \tilde{op}_i\ (s) = s$ [Figure 4(1)]. Here, suppose there exists an operation op_j such that $op_i \circ op_j(s) = s_2$ where $s \neq s_2$ but $Q(\langle s_2\rangle) = Q(\langle s\rangle)$. s_2 is not the same as s. However, s_2 is *QoS-equivalent* with s [Figure 4(2)].

[Definition] op_j is *QoS-compensating* operation of op_i iff $Q(\langle op_i \circ op_j(s)\rangle) = Q(\langle s\rangle)$ for every state s of an object o. □



(1) *compensation* (2) *QoS-compensation*

Figure 4: Compensating operation.

5 QoS-Based Replication

The system is composed of multiple objects $o_1, \ldots, o_n$. In the traditional systems, objects are replicated in order to increase the reliability, availability, and performance. The applications would like to use objects which support QoS required by the applications.

First, suppose that an application would like to get some snapshot $[op_{ij}(s_i)]$ from a state s_i of an object o_i by an operation op_{ij}. o_i has to support the application with not only $[op_{ij}(s_i)]$ which satisfies the qualification specified in op_{ij} but also enough QoS $Q([op_{ij}(s_i)])$. Here, let R denote the application's requirement QoS of the snapshot which the application would like to derive from the object. If there exists some object o_i supporting an operation op_{ij} such that $R \subseteq Q([op_{ij}(s_i)])$, the application can access o_i to derive the snapshot data by using op_{ij}.

For example, suppose there are three objects o_1, o_2, and o_3 where $Q([op_1(s_1)]) \subset R$, $Q([op_2(s_2)]) \supseteq R$, and $Q([op_3(s_3)]) \supseteq R$. o_2 or o_3 can be manipulated by the applications because they satisfy R. If $Q([op_2(s_2)]) \supseteq Q([op_3(s_3)])$, o_2 is selected to be accessed.

Suppose there are two objects o_1 and o_2 which support operations op_1 and op_2, respectively. Suppose that an application is accessing o_1 though op_1. If o_1 is faulty or o_1 cannot support QoS subsuming RoS R, the application can no longer use o_1. Here, if $Q([op_2(s_2)]) \supseteq R$ for a state s_2 of o_2, the application can access o_2 on behalf of o_1.

[Definition] An object o_j is a ***QoS-based replica*** of o_i iff there is one operation op_j and state s_j of o_j such that $Q([op_j(s_j)]) \supseteq Q([op_i(s_i)])$ for every operation op_i and state s_i of o_i. □

Here, if $Q([op_j(s_j)]) = Q([op_i(s_i)])$ for every op_j and s_j of o_j, o_j is a ***QoS-based full replica*** of o_i. Even if two objects o_1 and o_2 support the same data and operations. o_1 and o_2 are not ***QoS-based full replica*** unless they do not support the same QoS.

6 Concluding Remarks

This paper has discussed how to make the distributed system flexible with respect to QoS supported by the objects. We have discussed the novel equivalent and compatible relations among the operations on the basis of QoS. We have also discussed the compensating method to undo the work done with respect to QoS. We have also discussed the QoS-based replication to support required QoS in the QoS change of objects.

References

[1] Korth, H. F., Levy, E., and Silberschalz, A., "A Formal Approach to Recovery by Compensating transactions," *Proc. of the VLDB*, 1990, pp.95–106.