Sequence ID control for supporting task gaps without recovery procedure

Akihiro Shimura, CANON INC.

August 10, 1998

This document proposes the alternative model of error recovery based on the sequence identifier which handles task gaps caused by Abort Task.

1.Gap support based on "enforced sequential command tag"

Task gap support based on "enforced sequential command tag" requires following special process on both initiator and target;

- Target aborts all successive commands subsequent to aborted task.
- Initiator recovers (retries) all aborted commands again.

This approach will be inefficient due to its recovery overhead (abort and retry) even in normal operation because Abort Task may not be so extremely infrequent and it is not error handling but normal operation for the initiator.

2.Gap support proposal without recovery procedure

To avoid this inefficiency, we propose alternative gap support based on the sign of distance between sequence identifiers.

This proposal briefly consists of following rules.

- Target determines whether the command is ahead or behind from current position based on the sign of distance between them.
- Initiator is responsible to keep the actual sequence of tasks in the task list to be recognized correctly by the target.

3.Sequence Identifier Comparison

The actual sequence identifier space is finite. This space ranges from 0 to 2^{16} - 1.

Since the space is finite, all arithmetic dealing with sequence identifiers must be performed modulo 2^{16}. This unsigned arithmetic preserves the relationship of sequence identifiers as they cycle from 2^{16} - 1 to 0 again. There are some subtleties to computer modulo arithmetic, so great care should be taken in the comparison of such values. In the following description, the symbol "=<" means "less than or equal" (modulo 2^{16}).

4.Model

4.1Target Model

Target recognizes whether a request is already executed, not executed yet or intermediately executed by comparing the sequence identifier contained in the request and the sequence identifier the target is keeping as current position.

If the sequence identifier contained in the request is greater than the sequence identifier of the current position, the target identifies the request as not executed yet. If the sequence identifier contained in the request is less than the sequence identifier of the current position, the target identifies the request as already executed. Otherwise, if the sequence identifier contained in the request is equal to the sequence identifier of the current position, the target identifies the request as intermediately executed.

If the target identified the request as already executed, the target avoids the duplicated execution of the request, and may return completion status according with notify bit in the ORB. If the target identified the request as not executed yet, target updates the sequence identifier of current position to the value contained in the request, and executes it. If the target identified the request as intermediately executed, the target avoids the partially duplicated execution of the request based on the state the target is keeping with the current position.

4.2Initiator Model

Initiator is responsible to keep the actual sequence of tasks in the task list to be recognized correctly by the target. To do this, initiator applies additional constraint in appending new ORB destined to certain queue in addition to the constraint of MAX_TASK_SET_SIZE parameter. Initiator assumes the current position of the target execution to the position immediately follows the position the initiator received completion status most recently for the queue. The constraint is that initiator waits to append ORB until sequence identifier in the ORB becomes greater than or equal to the sequence identifier initiator is assuming as current position of the target.

Because a series of abort task may increase sequence identifier of ORB to be appended beyond the value that satisfy the above condition, there exists a case that initiator cannot append ORB anymore infinitely. To avoid this occurrence, initiator examines if there are outstanding tasks for the queue in the task list. If there is no outstanding task for the queue, the initiator appends ORB with adjusting its sequence identifier to meet above condition. By waiting until there is no outstanding task for the queue, initiator can ensure that the positions of initiator and target are synchronized, and can safely seed adjusted sequence identifier to the target.

5.Examples

Following figure shows an example usage of the model. To simplify the figure, sequence ID range is limited to 0-16, the control by the MAX_TASK_SET_SIZE parameter is eliminated and task list shows only task for one queue.

The first bar represents the ID range the initiator can append ORBs.

The minus portion of second bar represents the ID range the target believes as already executed, and the plus portion of second bar represents the ID range the target believes as not executed.

The third is sequence ID scale.

The fourth bar represents the possible ORB range in the task list.

The fifth boxes are ORBs and the corresponding sequence Ids are in the box.

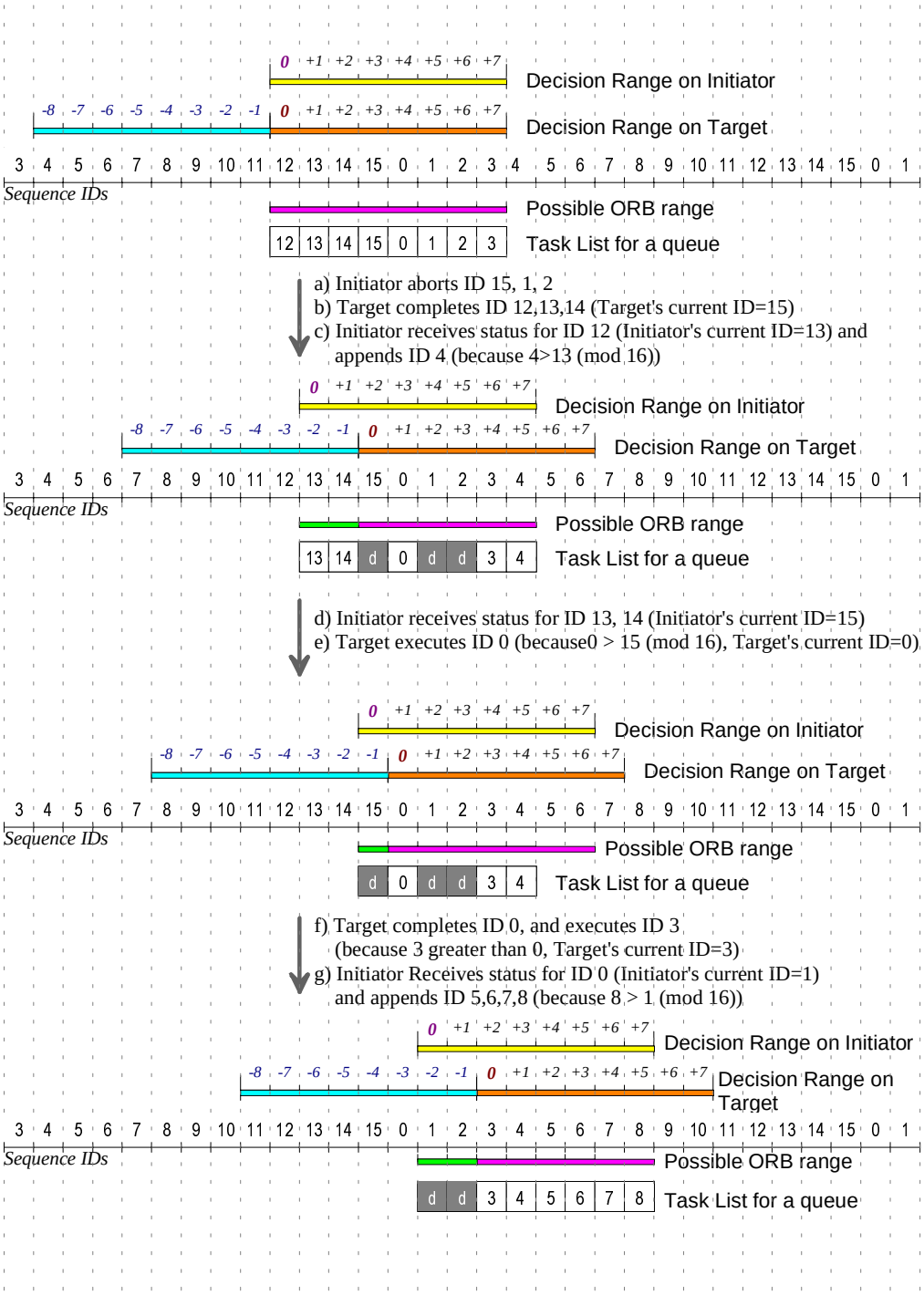
This example starts with both initiator and target recognizing current ID as 12.

a) Initiator aborts ID 15, 1 and 2. As a result, these ORBs become dummy ORBs (represented “d” in the figure).

b) Target completes ID 12, 13 and 14. As a result, target’s current ID becomes 15, and the decision range of the target moves to right three times.

c) Initiator receives status for ID 12. As a result, initiator’s current ID becomes 13, the ORB for ID 12 is discarded and the decision range of the initiator moves to right once. Because 4>13 (mod 16), initiator appends new ORB which has ID 4.

The example figure continues several steps as described in the figure.


0 +1 +2 +3 +4 +5 +6 +7
Decision Range on Initiator
-8 -7 -6 -5 -4 -3 -2 -1 0 +1 +2 +3 +4 +5 +6 +7
Decision Range on Target
3 4 5 6 7 8 9 10 11 12 13 14 15 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1
Sequence IDs
Possible ORB range
12 13 14 15 0 1 2 3
Task List for a queue
a) Initiator aborts ID 15, 1, 2
b) Target completes ID 12,13,14 (Target's current ID=15)
c) Initiator receives status for ID 12 (Initiator's current ID=13) and
appends ID 4 (because 4>13 (mod 16))
0 +1 +2 +3 +4 +5 +6 +7
Decision Range on Initiator
-8 -7 -6 -5 -4 -3 -2 -1 0 +1 +2 +3 +4 +5 +6 +7
Decision Range on Target
3 4 5 6 7 8 9 10 11 12 13 14 15 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1
Sequence IDs
Possible ORB range
13 14 d 0 d d 3 4
Task List for a queue
d) Initiator receives status for ID 13, 14 (Initiator's current ID=15)
e) Target executes ID 0 (because0 > 15 (mod 16), Target's current ID=0)
0 +1 +2 +3 +4 +5 +6 +7
Decision Range on Initiator
-8 -7 -6 -5 -4 -3 -2 -1 0 +1 +2 +3 +4 +5 +6 +7
Decision Range on Target
3 4 5 6 7 8 9 10 11 12 13 14 15 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1
Sequence IDs
Possible ORB range
d 0 d d 3 4
Task List for a queue
f) Target completes ID 0, and executes ID 3
(because 3 greater than 0, Target's current ID=3)
g) Initiator Receives status for ID 0 (Initiator's current ID=1)
and appends ID 5,6,7,8 (because 8 > 1 (mod 16))
0 +1 +2 +3 +4 +5 +6 +7
Decision Range on Initiator
-8 -7 -6 -5 -4 -3 -2 -1 0 +1 +2 +3 +4 +5 +6 +7
Decision Range on Target
3 4 5 6 7 8 9 10 11 12 13 14 15 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1
Sequence IDs
Possible ORB range
d d 3 4 5 6 7 8
Task List for a queue