Title: Route Aggregation Within Routeing Domain Confederations for IDRP

Reference: 1) SC6 N7196: ″Editor′s Revision of IDRP Text″
2) SC6 N7195: ″Disposition of Comments on CD 10747″

The disposition of comments for CD 10747 contained in SC6 N7195 asks national bodies to review the proposed text changes developed as a possible resolution to comment GB 1.5. This contribution proposes an alternative approach that accomplishes the goal of GB 1.5—namely, to allow a BIS to aggregate routes which have entered, but not yet exited, a confederation. On the whole, the proposed methods are believed to be less complex than the method outlined in SC6 N7195.

In formulating the method proposed in this paper, our goal is to provide a method that works for practical situations where aggregation will be beneficial and will reduce the amount of information. Secondly, our proposed method attempts to preserve as much ″RD_SEQ″ information as possible: that is, it uses RD_SETS only in those cases when necessary. This differs from the UK approach which begins by converting each_PATH attribute into a ″set″. Hence, the proposed USA approach preserves more useful routeing information in the aggregated attribute than does the UK method.

In developing this alternative, we have made a technical change to the encoding of the Entry Marker so that it now has an explicit ″type code″ within the RD_PATH attribute and is now known as ENTRY_LIST; we have tightened the requirement of the order in which overlapping confederations can be listed in the RD_PATH attribute; and we have reformulated the procedures for aggregating RD_PATH attributes in more precise terms that explicitly recognize each type of path segments (ENTRY_LIST, RD_SET, and RD_SEQ).

Having reviewed the manner in which the material on RD_PATH usage is presented in the the proposed UK text, we find it to be a more coherent style of presentation than in SC6 N7196. Therefore, we have organized the order of presentation of this new material along those lines: that is, all RD_PATH usage rules appear in a single contiguous body of text, rather than being dispersed throughout the text.

Finally, we have verified that the proposed RD_PATH aggregation procedures work correctly by applying them to the sample topology that the UK provided in their ballot comments.

1. **Encoding of RD_PATH Attribute:**

   The text on page 15 of SC6 N7196 for the RD_PATH attribute needs two editorial corrections: a) the term *UPDATE PDU* should be replaced with the term *route*, and b) the clause reference in the next-to-last paragraph should be to 8.1, not 8.1.2.1.

   We propose adding a new type of RD_PATH segment, ENTRY_LIST, which will have type code 3. The description of the new ENTRY_LIST in item ″c″ on page 15 of SC6 N7196 will be:

   > ENTRY_LIST: provides a list of the RDIs of the confederations that this route has entered but not yet exited.

   Finally, the new definitions and descriptions of ENTRY_LIST make the material in 8.13.4 (″Entry Marker for an RDC″) no longer relevant, Hence, it should be deleted in its entirety.

2. **Disjointed Presentation of Usage Rules for RD_PATH:**

   The text in SC6 N7196 presents the usage rules for the RD_PATH attribute in two different places, separated by several pages: 8.12.3 treats the case of no routeing domain confederations, and 8.13.5ff treats the case of routeing domain confederations. We suggest rearranging this material so that the information from both places is combined. The combined material will then appear as subclauses of 8.12.3.

   - Delete existing 8.12.3, 8.13.5, and 8.13.6. As noted below, portions of these clauses are used in constructing new clauses 8.12.3.1, 8.12.3.2, and 8.12.3.3.
   - The new 8.12.3 consists solely of the first paragraph of existing 8.12.3, with the words *UPDATE PDU* changed to *route*
   - The material in the third paragraph of existing 8.12.3 (When a BIS originates a route...) is combined with the material in existing 8.13.5, and is presented as a new subclause 8.12.3.1, as shown below:

     8.12.3.1 Generating the RD_PATH Attribute

     When a BIS originates a route to destinations contained within its own routeing domain or to destinations learned by means outside of the protocol (see 8.12.2), it shall examine the information contained in its managed object **RDC-CONFIG** to determine the ordering relationships among all the confederations of which the local routeing domain is a member. The local BIS shall then construct an RD_PATH attribute as follows:

     1. If the local routeing domain is a member of one or more confederations, the first path segment shall be of type ENTRY_LIST, and shall list all such confederations:

        a. If a confederation, RDC-B, is nested within another confederation, RDC-A, then the RDI of RDC-A shall precede that of RDC-B.

        b. The RDIs of overlapping confederations shall be listed in increasing order of the numerical value of the RDIs, as long as the order implied by any nesting relationships is maintained. The numerical value of the RDI is obtained by treating it as an unsigned binary integer.

     2. The next path segment shall be of type RD_SEQ, and it shall list the RDI of the BIS's routeing domain. If the local routeing domain is not a member of any confederation, then this path segment will be the only path segment in the RD_PATH attribute.

   - A new subclause will be constructed from the material in existing 8.13.6, to describe how to update the RD_PATH attribute of a received route. The only material taken from 8.13.6 is that which deals with reception (not advertisement) of a route. New material is added to prevent construction of two consecutive path segments of the same type: e.g., an RD_SEQ segment may not be followed immediately by another RD_SEQ segment. The proposed text is as follows:

     8.12.3.2 Updating a Received RD_PATH Attribute

     When a BIS receives a route from an adjacent BIS, the receiving BIS shall update the RD_PATH attribute of that route according to the following rules:

     1. If the route was received from a BIS located in the same routeing domain as the local BIS, then the RD_PATH attribute shall not be updated.

2. If the route was received from a BIS located in an adjacent routeing domain, the local BIS shall identify those confederations the route has just entered (see 8.13.3), and shall append new path segments to the RD_PATH of the received route, as follows:

   a. If the route has entered any confederations, the BIS shall append a path segment of type ENTRY_LIST that lists all the newly entered confederations:

      1) If a confederation, RDC-B, is nested within another confederation, RDC-A, then the RDI for RDC-A shall precede that for RDC-B.

      2) The RDIs for overlapping confederations shall be listed in increasing order of the numerical value of the RDIs, as long as the order implied by any nesting relationships is maintained. The numerical value of the RDI is obtained by treating it as an unsigned binary integer.

   b. Next, the BIS shall append a path segment of type RD_SEQ, and it shall list the RDI of the BIS′s routeing domain. If the newly received route has not entered any confederations, then this path segment will be the only path segment appended to the RD_PATH attribute.

   Note: The RDI of the local BIS′s routeing domain can appear twice in the updated RD_PATH attribute of a route received from a BIS located in an adjacent routeing domain: once because it was present in the route carried in the UPDATE PDU, and again as the final step in the procedure outlined above. Since such a route contains an instance of RD looping, the local BIS should not use such a route as an input to its Decision Process.

- A new subclause will be constructed by combining material from the second paragraph of 8.12.3 (When a BIS propagates a route...) and from item ″b″ of 8.13.6. This clause will address rules for modifying the RD_PATH attribute of a route that a BIS propagates to adjacent BISs. The proposed text is as follows:

  8.12.3.3 RD_PATH Attributes of a Propagated Route

  After receiving a route from a neighbor BIS, the local BIS will have modified its RD_PATH attribute in accordance with 8.12.3.2. If the BIS then chooses to advertise the route a neighbor BIS located in an adjacent routeing domain, modifications to the RD_PATH attribute are needed.

  The local BIS shall use the methods of 8.13.3 to determine which confederations, if any, must be exited in order to reach the adjacent BIS. If no confederations will be exited, then no modifications to the RD_PATH attribute are needed.

  If any confederations will be exited, the BIS shall apply the following procedures to each of the exited confederations. The procedures shall be applied first to any exited confederation that is located at the lowest level of the local partial order, as described in managed object **RDC-Config**. The procedures shall then be applied iteratively to the other exited confederations, always working from those at the lowest level to those at the higher levels. For purposes of this procedure, a path segment that lists multiple RDIs shall be treated as if it were multiple consecutive path segments, where each path segment lists a single RDI and the order of appearance of RDIs is maintained. For example, a path segment that listed RDIs X, Y, and Z (in that order) is treated as if it were a path segment listing X, followed by a path segment listing Y, followed by a path segment listing Z.

The local BIS shall scan the RD_PATH attribute, right to left starting at the last (highest numbered) octet looking for a ENTRY_LIST that names the exited confederation. We use the notation (X) to denote a path segment of type ENTRY_LIST that contains RDI X, and <X> to denote a path segment of type RD_SEQUENCE that contains RDI X.

For each exited confederation (for example, the confederation whose RDI is ″X″), the advertising BIS shall then update the RD_PATH of the route as follows:

1. If (X) is not present in the RD_PATH, then the route is in error, and the BIS shall issue an IDRP ERROR PDU which reports a Misconfigured_RDCs error.
2. If (X) is present, and there are no entry lists for other confederations between itself and the end of the RD_PATH attribute, then contents of the RD_PATH attribute, from (X) to the end of the attribute, shall be replaced by <X>.
3. If an intervening ENTRY_LIST, (Y), is found, and RDC-X is known to be nested within RDC-Y, then the route is in error. The local BIS shall send an IDRP_ERROR PDU that reports a Misconfigured_RDCs error.
4. If an intervening ENTRY_LIST, (Z), is found, but the local BIS′s routeing domain is not a member of RDC-Z, then the route is in error, and the local BIS shall send a IDRP ERROR PDU that reports a Misconfigured_RDCs error.
5. If an intervening ENTRY_LIST, (Z), is found, where RDC-X and RDC-Z are overlapping confederations, the contents of the RD_PATH attribute between (X) and (Z), inclusive, shall be replaced by <X>(Z).

If, at the completion of this procedure there are consecutive path segments of the same type, they shall be combined into a single path segment of the same type.

3. **Aggregation Rules for RD_PATH:**

SC6 N7196 places unnecessary restrictions on aggregating routes whose RD_PATH attribute contains an RDC Entry Marker. Specifically, the last paragraph in clause 8.12.3 states that ″... RDs which are members of a confederation shall not be permitted to collapse RD_SEQUENCE information into an RD_SET when the original sequence contains an entry marker (see clause 8.13.4), nor shall they be permitted to generate information expressed as an RD_SET.″ Likewise, clause 8.17.2.3 states that ″If a route has an RD_PATH attribute that contains a confederation entry marker (see 8.13.4), then that route shall not be aggregated with any other route″.

A more careful analysis shows that the only requirement that needs to be satisfied when aggregating RD_PATH attributes is that the order of entered, but not exited confederations, as recorded in the ENTRY_LIST path segments of the RD_PATH attribute, shall not be altered by the aggregation process. The RD_PATH attributes of all valid routes will have entered the same confederations, but they may not have entered them in the same order. Aggregation is permitted only when the RDIs of all entered confederations appear in the same order in each component attribute; otherwise, it is not.

We suggest the following changes:

a. The last paragraph of 8.12.3 has been removed from the revised text presented in this contribution.
b. The existing text in 8.17.2.3 that deals with aggregation of the RD_PATH attribute should be deleted in its entirety (including the duplicated first paragraph), and replaced with the following new text:

RD_PATH attributes: The individual RD_PATH attributes from which the aggregated RD_PATH attribute will be constructed are called the *component attributes*. Aggregation shall not be performed if each component RD_PATH attribute does not list the same entered confederations in the same order of appearance. For purposes of the following algorithms, a path segment that lists multiple RDIs shall be treated as if it were multiple consecutive path segments, where each path segment lists a single RDI and the order of appearance of RDIs is maintained.

The main procedure calls the subroutine defined below for aggregating RD_PATH attributes that contain no ENTRY_LISTS. In effect, it applies the subroutine to all segments that are located between ENTRY_LISTs, between an ENTRY_LIST and the end of a component attribute, or between the start of a component attribute and its first ENTRY_LIST.

The main procedure is as follows:

MAIN PROCEDURE

1. Set the aggregated RD_PATH to "empty".
2. Scanning from the back of each non-empty component attribute, locate the first ENTRY_LIST.
3. If no ENTRY_LIST is found, apply the subroutine for aggregating RD_PATHs with no ENTRY_LISTs, and prepend the result to the aggregated RD_PATH attribute.
4. If a ENTRY_LIST is found, prepend the following to the aggregated RD_PATH attribute, in the order indicated: the located ENTRY_LIST, followed by the path segments obtained by applying the subroutine for aggregation of RD_PATHs with no ENTRY_LISTs to the path segments that follow the located ENTRY_LIST in each component attribute. If a component attribute has no path segments following the located ENTRY_LIST, pass it to the subroutine as an empty set.
5. Delete from each component attribute all the path segments that were appended to the aggregated attribute in steps 3 or 4.
6. Repeat steps 2 through 5 until every component attribute is empty.

   If there are consecutive path segments of the same type, they shall be combined into a single path segment of the same type.

The subroutine for aggregating RD_PATH attributes with no entry lsits is as follows:

SUBROUTINE TO AGGREGATE ROUTES WITH NO ENTRY_LISTS

1. Set the aggregated RD_PATH to ″empty″.
2. Scanning from the back of each component attribute, locate the first identical longest sequence of path segments that occurs in every component attribute, including any that are empty.

   Note: It will not be possible to find an identical sequence in every component attribute if one or more of them are empty.
3. If there is no identical sequence, form a path segment of type RD_SET that contains every RDI in every non-empty component attribute. Prepend this list to the aggregated RD_PATH attribute.
4. If the identical sequence is the final sequence of every component attribute, prepend it to the aggregated route.
5. If the identical sequence is not the final sequence of every component attribute, form a path segment of type RD_SET that lists every RDI that occurs between the end of the identical sequence and the end of each non-empty component attribute. Prepend this list to the aggregated RD_PATH attribute.
6. Delete from each component attribute all path segments that were added to the aggregated RD_PATH attribute in step 3, 4, or 5.
7. If, after the deletions in step 6 have been made, an RDI is present in both the aggregated RD_PATH attribute and in any of the component attributes, then the accumulated RD_PATH attribute shall be replaced by a single path segment of type RD_SET that lists every RDI that was present in the component routes that were the input to this subroutine (before any deletions were made), and the subroutine terminates. Otherwise, repeat steps 2 through 6 until every component attribute is empty.

Appendix A. Example of RD_PATH Update and Aggregation Rules

This appendix illustrates the application of the proposed rules for updating RD_PATH attributes and for aggregating them. It uses Figure A-1 from Appendix A of the UK comments contained in SC6 N7089 (Summary of Voting). For convenience, that figure is reproduced here. The example OSIE consists of four confederations and nine BISs, each of which is located in a separate routeing domain. Confederation boundaries are represented by the dotted ellipses, and the BISs and their routeing domains are represented by the solid black squares. In this topology, confederations B, C, and D overlap one another; and confederations B, C, and D are each nested within confederation A.

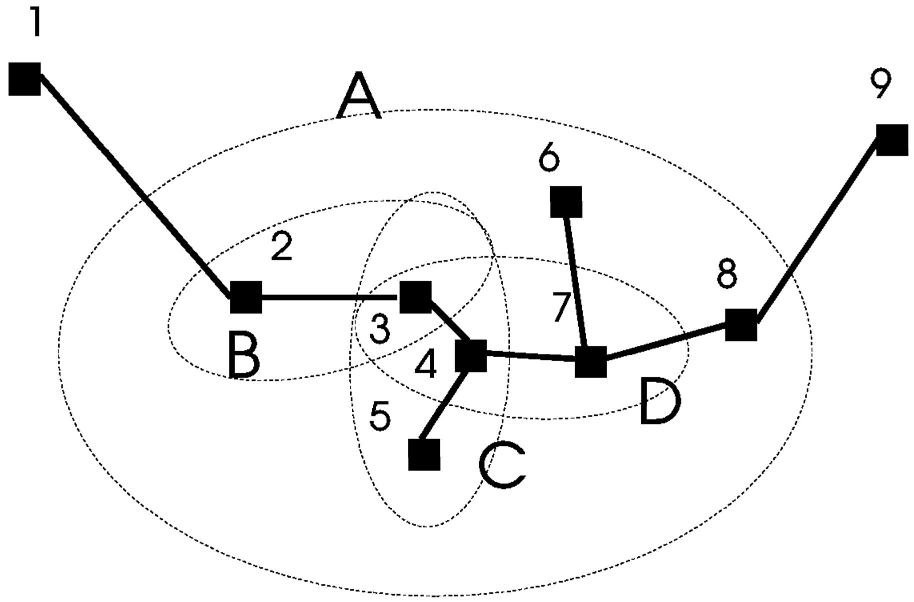

Figure 1. An Example Topology

The illustrative example assumes that a route is originated in RD1, and is subsequently propagated through RD2 and RD3 to RD4. Another route is originated by RD5 and is propagated to RD4. At RD4, these two routes are to be aggregated, and then propagated to RD7.

RD6 originates a route, and propagates it to RD7, where it will be aggregated with the route received from RD4 (which itself is an aggravated route). The aggregated route is then propagated from RD7 through RD8 to RD9.

In the following description, RDIs contained in path segments of type RD_SEQUENCE are enclosed in angle brackets <...>, those contained in path segments of type RD_SET are enclosed in curly braces {...}, and those contained in path segments of type ENTRY_LIST are enclosed in parentheses (...). The shorthand phrase *BIS n* means "the BIS that is located in RD n". Commas are used only for visual clarity only, and have no notational significance.

1. BIS 1 originates route α, creating its RD_PATH according to 8.12.3.1:

   < 1 >

2. BIS 5 originates route β, creating its RD_PATH according to 8.12.3.1:

   (A,C),<5>

   This reflects that BIS 5 is a member of both confederations A and C, and that C is nested within A.

3. BIS 6 originates route τ, creating its RD_PATH according to 8.12.3.1:

   (A),<6>

   This reflects that BIS 6 is a member of confederation A.

4. When BIS 2 receives route α, it updates the RD_PATH according to 8.12.3.2:

   <1>,(A,B),<2>

   This reflects that route α has entered confederations A and B, and that B is nested within A.

5. When BIS 2 propagates route α to BIS 3, BIS 2 updates its RD_PATH according to 8.12.3.3. Its RD_PATH remains unchanged, since no confederations have been exited.

6. When route α is received by BIS 3, it RD_PATH is updated according to 8.12.3.2:

   <1>,(A,B),<2>,(C,D),<3>

   This reflects that route α has entered confederations C and D and routeing domain 3. Since C and D overlap, they are listed in increasing numerical order (which for this example will be assumed to correspond to their alphabetical order).

7. When BIS 3 propagates route α to BIS 4, it amends the RD_PATH attribute according to 8.12.3.3:

   <1>,(A),<B>,(C,D),<3>

   This reflects that route α has exited confederation B.

8. When BIS 4 receives route α from BIS 3, it updates the RD_PATH attribute according to 8.12.3.2:

   <1>,(A),<B>,(C,D),<3,4>

   Since no confederations were entered, only 4′s RDI was appended.

9. When BIS 5 propagates route β to BIS 4, it updated the RD_PATH according to 8.12.3.3:

   (A,C),<5>

   Since no confederations were exited, no changes are made.

10. When BIS 4 receives route β, it updates its RD_PATH according to 8.12.3.2:

    (A,C),<5>,(D),<4>

    This reflects that route β has entered confederation D and routeing domain 4.

11. Now BIS 4 aggregates route α and route β to produce a new route, $\alpha\beta$, by applying the methods in the revised text for 8.17.2.3. The two routes contain:

    Route α: <1>,(A),<B>,(C,D),<3,4>

    Route β: (A,C),<5>,(D),<4>

Scanning from the back of each component attribute and recalling that the procedure treats (C,D) as if it were (C),(D), the main procedure will locate (D). Steps 4 and 5 of the main procedure then result in:

Route $\alpha\beta$: (D),{3},<4>

Route α': <1>,(A),<B>,(C)

Route β': (A,C),<5>

At the next iteration, the procedure locates (C) as the last entry list in routes α' and β'. Steps 4 and 5 of the main procedure then result in:

Route $\alpha\beta$: (C),{5},(D),{3},<4>

Route α': <1>,(A),<B>

Route β': (A)

At the next iteration, the procedure locates (A) as the last entry list in routes α' and β'. Steps 4 and 5 of the main procedure then result in:

Route $\alpha\beta$: (A),{B},(C),{5},(D),{3},<4>

Route α': <1>

Route β': empty

The final iteration finds no more entry lists, so the final result becomes:

Route $\alpha\beta$: {1},(A),{B},(C),{5},(D),{3},<4>

12. When BIS 4 propagates route $\alpha\beta$ to BIS 7, it will update the RD_PATH attribute according to 8.12.3.3:

    {1},(A),{B},<C>,(D),{3},<4>

    This reflects that route $\alpha\beta$ has exited confederation C.

13. When BIS 7 receives route $\alpha\beta$, it updates its RD_PATH according to 8.12.3.2:

    {1},(A),{B},<C>,(D),{3},<4,7>

    This reflects that route $\alpha\beta$ has entered routeing domain 7, but has entered no confederations.

14. When BIS 6 propagates route τ to BIS 7, it amends the RD_PATH attribute according to 8.12.3.3. The route attributes is not changed, since no confederations have been exited.

15. When BIS 7 receives route τ, it updates its RD_PATH attribute according to 8.12.3.2:

    (A),<6>,(D),<7>

    This reflects that route τ has entered confederation D and routeing domain 7.

16. Now BIS 7 aggregates route $\alpha\beta$ with route τ according to the revised text proposed for 8.17.2.3. The two component routes are:

    Route $\alpha\beta$: {1},(A),{B},<C>,(D),{3},<4,7>

    Route τ: (A),<6>,(D),<7>

    Scanning from the back, the main procedure will find that (D) is the last entry list. Steps 4 and 5 of the main procedure will result in:

Route $\alpha\beta\tau$: (D),{3,4},<7>

Route $\alpha\beta'$: {1},(A),{B},<C>

Route τ': (A),<6>

The next iteration will identify (A) as the final entry list. Steps 4 and 5 of the main procedure will result in:

Route $\alpha\beta\tau$: (A),{B,C,6},(D),{3,4},<7>

Route $\alpha\beta'$: {1}

Route τ': empty

The final iteration finds no more entry lists, so the final result becomes:

Route $\alpha\beta\tau$: {1},(A),{B,C,6},(D),{3,4},<7>

17. When BIS 7 propagates route $\alpha\beta\tau$ to BIS 8, it updates the RD_PATH attribute according to 8.12.3.3:

    {1},(A),{B,C,6},<D>

    This reflects that route $\alpha\beta\tau$ has exited from confederation D.

18. When BIS 8 receives route $\alpha\beta\tau$, it updates the RD_PATH attribute according to 8.12.3.2.

    {1},(A),{B,C,6}<D,8>

    This reflects that route $\alpha\beta\tau$ has entered domain 8.

19. When BIS 8 propagates route $\alpha\beta\tau$ to BIS 9, it updates the RD_PATH attribute according to 8.12.3.3:

    {1},<A>

    This reflects that route $\alpha\beta\tau$ has exited from confederation A.

20. When BIS 9 receives route $\alpha\beta\tau$, it amends the RD_PATH according to 8.12.3.2:

    {1},<A,9>