Agreement protocol sequencing

Mick Seaman

This note details the use of the Agreement Number (AN) and Discarded Agreement Number (DAN) in the Agreement Protocol specified for P802.1aq Shortest Path Bridging[1]. The protocol's Agreement Digest summarizes the physical topology of the network, and is computed in a way that ensures that the risk of protocol participants with different views of that topology computing the same Digest is negligible. As a protocol participant encounters topology changes, it[2] successively limits the frames it forwards to a subset determined by a set of loop-free forwarding rules[3] for those successive topologies. Each participant forwards using the full set of active topologies corresponding to its currently perceived physical topology (i.e. ignores prior topologies) only when its Digest *matches*[4] that of its neighbours and sequence number conditions are met: it can then be sure that they are also forwarding using that topology, or one of its subsequent subsets.

The need for the AN and DAN sequence numbers is due to the buffering between protocol participants. SPB may run over long distance services that can exhibit significant delays and a non-negligible, if small, risk of misordering. A participant needs to know that a prior Digest value is not 'in flight' before declaring a *topology match*. Otherwise a neighbour might use that value as a starting point after communicating a different prior value. The sequence numbering also deals with misordering, without requiring additional mechanisms to recover from arbitrary disruption or participant re-initialization.

In addition to providing an overview of the protocol and its operation, this note provides a proof of correctness (guaranteeing loop-free behavior independent of message transit delays) when protocol participants are connected by a service that either does not misorder frames or where misordering can be detected by the specified protocol mechanisms (i.e. when misordered messages are no more than one set of changes out of date.)

1. Introduction

In the absence of the AN and DAN sequence numbers, the scenario illustrated in Figure 1 would be possible. It starts with participants A(lice) and B(ob) basing their calculations on the physical topologies represented by digests '1' and '2' respectively. Then (more or less at the same time) each receives link state information communicating the other topology. Agreement protocol messages originating with the prior topology views cross, and each adopts full forwarding based on two topologies whose arbitrary combination might cause a loop.

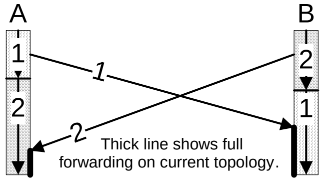

Figure 1—Crossing agreements

While such a loop would have been resolved as soon as Alice and Bob converge (using IS-IS mechanisms) on the same physical topology, and the circumstances that would cause it might be thought unlikely, they are certainly not impossible if there are a number of flapping links. Further agreement messages do not resolve the issue by themselves: see Figure 2.

When each participant receives the second message in Figure 2 (for topology '1' in the case of Alice receiving) it has no longer has a detailed record of the old topology—it might differ slightly or considerably from that current. Certainly the receiving participant should not stop forwarding entirely because its peer sees a different topology: indeed an agreement

[1] P802.1aq/D2.7 clauses 13.17, 13.27, 13.29.14, 13.29.28, 14, 28.11.3.5.1.

[2] There is a participant per Bridge Port and forwarding is limited only on a per port basis, so the neighbours concerned are only those attached to a single LAN (often just one). A single match does not have to propagate throughout the network before forwarding is improved.

[3] See Link state agreement, March18th 2010.

[4] This note uses the term *match* rather than sync, synchronization, agree etc. as the latter already have meanings in the context of 802.1Q Clause 13. A 'digest match' or 'matching digests' is used to mean that the Digests transmitted and received by a participant have the same value. A 'topology match' is only declared if the digests match and the AN/DAN conditions specified in this note are met.

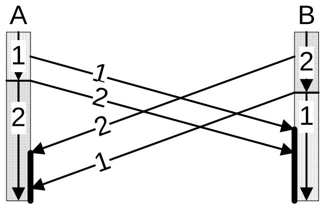

Figure 2—Further agreement messages

message with an unknown digest should be cached so each participants need send only one message and receive only one to move from complete forwarding on one topology to complete forwarding on its successor, as in Figure 3.

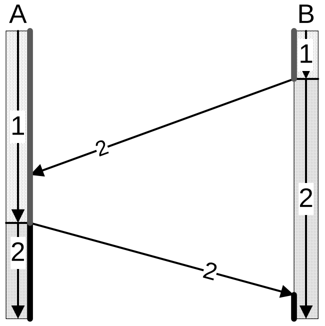

Figure 3—Normal topology progression

The scenario in Figure 1 is prevented by including the AN (agreement number) and DAN (discarded agreement number) in each agreement message. When a participant calculates a new topology and the accompanying digest[1], it increments its own AN and checks for a topology match. When the participant receives an agreement protocol message, it sets its transmitted DAN to the received AN, and then checks for a topology match. If, on checking for a topology match, the participant finds that the last received and currently transmitted digests are equal, it sets its transmitted DAN equal to the last received AN plus 1, and if the last received DAN is also equal to its AN or AN plus 1, it has matched topologies. In other words:

```
onTopologyUpdate()
{   tx.digest = calculatedDigest; tx.an++;
    checkTopologyMatch();
}

onMessageReception()
{   rx.digest = msg.digest; rx.an = msg.an;
    rx.dan = msg.dan; tx.dan = rx.an;
    checkTopologyMatch();
}

checkTopologyMatch()
{   if   (rx.digest == tx.digest)
    {   tx.dan = rx.an+1;
        if   ((rx.dan == tx.an) || (rx.dan == tx.an+1))
        {   topologyMatched();
}   }   }
```

Figure 4 shows how the sequence numbering handles the Figure 1 scenario.

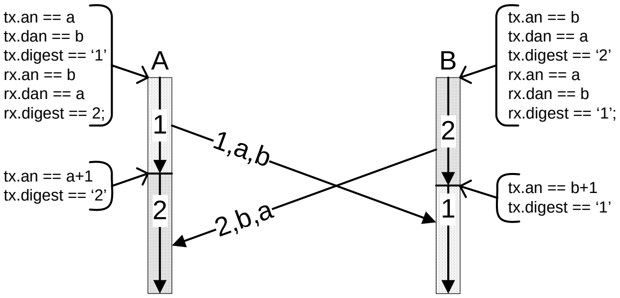

Figure 4—Handling crossing agreements

When the crossing messages are received, their DANs lag rather than precede or equal the receivers' ANs, so a false topology match is not declared.

Figure 6 shows the influence of further messages, as IS-IS converges on a stable physical topology.

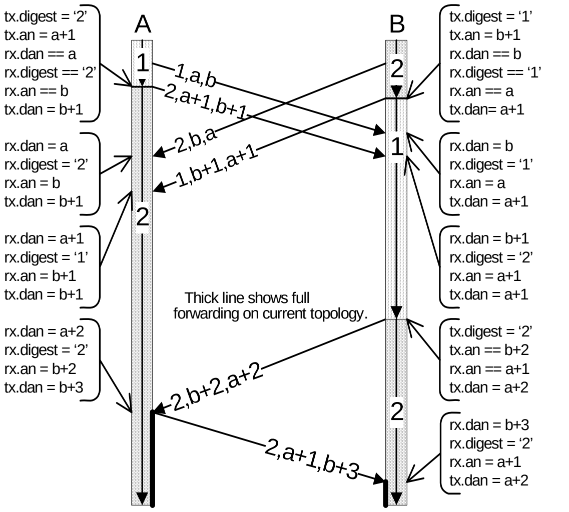

Figure 5—Further messages and stable topology

Figure 6 shows the normal topology progression introduced in Figure 3, including the last periodic messages transmissions for the initial topology so the initial state of the participants is clear. The actual change to complete forwarding on the new topology takes just two messages—one from each of the participants.

[1] This description is a little simplistic, it ignores the fact that the new Digest cannot be transmitted until the forwarding changes required for the loop-free rules have been made.

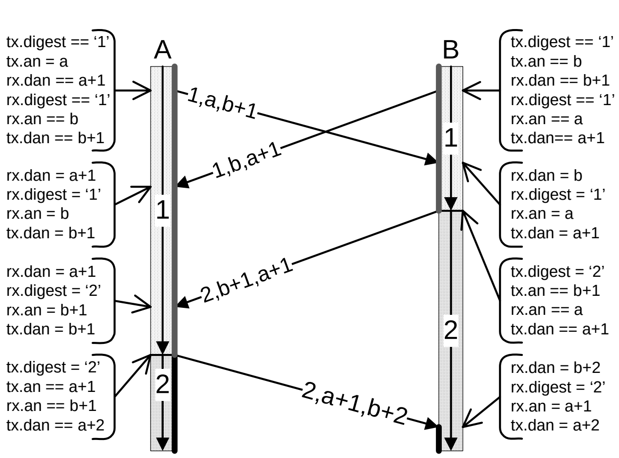

Figure 6—Handling normal topology progression

Figure 7 shows what happens when there is a 'glitch' in the topology, a temporary change that is noted by only one of the participants[1].

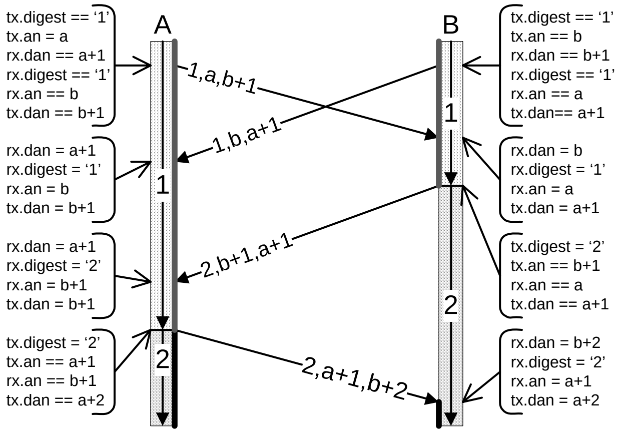

Figure 7—Recovering from a glitch

2. Misordering

Without further refinement, the protocol as described so far in this note handles a number of cases of message reordering without creating potential topology conflicts. The received DAN checking is often a sufficient defence. However such conflicts are possible, as illustrated in Figure 8.

These conflicts could be prevented by simply discarding out-of-order frames, but if one participant gets out of sync with the other (perhaps because it is reinitialized) connectivity will be lost permanently. It is highly undesirable to have to introduce additional mechanisms to recover from this eventuality. The solution is to restrict topology matches following out-of-order reception to those where the received DAN is the transmit AN plus one. This guarantees that the received message was sent after both the receiving and transmitting participants have settled on the same topology. Such a match will always occur eventually provided that IS-IS does cause both participants to settle on the same topology. Subsequent matches can use either the received DAN equals the received AN and the 'plus one' condition.

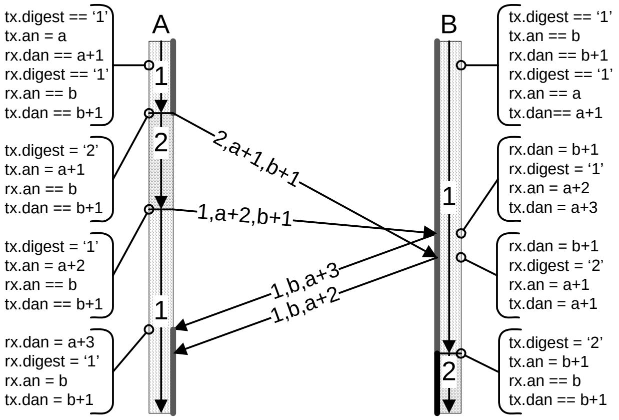

Figure 8—Misordering and conflict

```
onTopologyUpdate()
{   tx.digest = calculatedDigest; tx.an++;
    checkTopologyMatch();
}
onMessageReception()
{   if  (msg.an < rx.an) outOfOrder = True;
    rx.digest = msg.digest; rx.an = msg.an;
    rx.dan = msg.dan; tx.dan = rx.an;
    checkTopologyMatch();
}
checkTopologyMatch()
{   if  (rx.digest == tx.digest)
    {   tx.dan = rx.an+1;
        if  (   ((rx.dan == tx.an) && !outOfOrder)
                ||  ( rx.dan == tx.an+1))
        {   topologyMatched(); outOfOrder = False;
}   }   }
```

3. Sequence number space

So far the protocol description has ignored the limitations imposed by the small (two-bit) AN sequence number space. In order to distinguish old out-of-order messages from AN increments resulting from several closely spaced changes with possible message loss, the use of fresh AN's needs to be modified by feedback from each participant's peer. This is done by using the received DAN to rotate the sequence number window. The transmit AN can be increased as far as the received DAN plus one. Thus if Alice's AN is currently a, and her digest matches with

[1] The Agreement Digest reflects the physical topology, not the whole of the IS-IS state, in particular it omits IS-IS sequence numbers so one participant might simply process the latest of a number of LSPs and thus miss temporary topology perturbations visible to others.

Bob’s she would naturally expect to receive a message with a DAN of a+1, giving her permission to transmit with an AN of a+1 or a+2[1] before a further message from Bob is required to rotate the window once more. Thus a+3, which is indistinguishable from a-1, is temporarily outside the widow, and Bob can identify misordered messages as long as they are no more than one topology change out of date.

```
onTopologyUpdate()
{   if ((tx.digest != calculatedDigest) &&
        (tx.an+1 == tx.dan) || (tx.an+1 == tx.dan+1))
    {   tx.digest = calculatedDigest; tx.an++;
        checkTopologyMatch();
}   }
```

```
onMessageReception()
{   if   (msg.an == rx.an+3) outOfOrder = True;
    rx.digest = msg.digest; rx.an = msg.an;
    rx.dan = msg.dan; tx.dan = rx.an;
    onTopologyUpdate();
}
```

```
checkTopologyMatch()
{   if  (   (rx.digest == tx.digest) &&
            (tx.digest == calculatedDigest))
    {   tx.dan = rx.an+1;
        if  (   ((rx.dan == tx.an) && !outOfOrder)
            ||  ( rx.dan == tx.an+1))
        {   topologyMatched(); outOfOrder = False;
}   }   }
```

Note that the checks performed on a topology update are now also performed following message reception, since the latter may have advanced the transmit AN window so that the tx.digest can be updated.

4. When to transmit

Transmissions should occur periodically, so that message loss does not delay topology agreement indefinitely. When the Agreement Digest is carried in BPDUs, the natural Hello Time of 2 seconds suffices for this purpose. When it is desirable to also carry the Agreement Digest in IS-IS Hellos (other message) a refresh frequency suited to that of the carrying protocol should be chosen. Transmission should also be scheduled whenever the transmit AN and transmit DAN are updated, but not otherwise. The result is that (in the absence of message loss) any digest change that will result in a match will result in the necessary exchange of messages to take place, but the message sequence will not be prolonged when that is not an immediate possibility. Figures 5 through 7 illustrate the desired behavior.

5. Multiple participants

Although not explicitly detailed so far in this note, I hope it is reasonably clear that the protocol easily accommodates groups of participants connected over shared or pseudo-shared media[2], at the cost of four bits per participant[3] in each protocol message multicast by each participant to all the others. A topology match is generally declared only when the digest and received AN and DAN from each and every participant meets the specified criteria.

In this way the agreement protocol can also be used in non IS-IS scenarios where topology information is directly carried in messages, and there is some set of well defined rules as to what behavior is expected in any configuration and how that behavior should change in successive periods of change before the next match is declared. The protocol is naturally efficient in those scenarios, as its rules for matching involve the minimal possible message exchange.

[1] If the peer participant has declared a topology match with the current digest, otherwise the window advances by one.

[2] Or over some more complex set of links.

[3] The cost of identifying which participant each set of bits belong to can be avoided as the participants are easily placed in order (by system id or MAC address). The digest itself ensures that different participants agree on which participants are participating and which bits go with which.

6. Proofs

6.1 Loop-free in the absence of misordering

First we show that, unless protocol messages are misordered, one participant will only declare a topology match and remain forwarding on that topology without further subsetting if the other has declared a match or subsetted his forwarding using that same topology.

The development of some simple terminology and temporal logic allows us to see the wood for the trees when representing the changing states of the protocol participants and their inter-dependencies. We treat the protocol as having an unlimited supply of sequence numbers that are never reused, with only the real integer part modulo 4 being carried in the two-bit message fields. Since each increase in a transmitter's two-bit AN field is limited, knowledge of the prior extended value allows a protocol observer to determine the corresponding extended values without ambiguity. It is also convenient to view the ANs generated by Alice in any protocol run, together with the DANs returned by Bob, as being drawn from a different number space (AN_A) from Bob's ANs and Alice's DANs (AN_B)[1]. Let:

$$
\begin{aligned}
&a, a', a'' \in AN_A \\
&b, b', b'' \in AN_B \\
&\forall\, x \in AN_A\ (x+1 \in AN_A) \\
&\forall\, x \in AN_B\ (x+1 \in AN_B) \\
&AN_A \cap AN_B = \varnothing
\end{aligned}
$$

Since each participant increments its AN when and only when the transmitted digest changes, we can identify the digest value by its AN and define the function $D(x,y)$ as being True iff the digests identified by $x,\ y \in AN_A \cup AN_B$ have the same value. Further:

$$\forall\, x\ \forall\, y{:}(D(x,y))\ (\neg D(x\text{-}1,y) \wedge \neg D(x+1,y))$$

In addition each message can be represented simply by its <an>.<dan> tuple, for example:

a.b+1

If we know to which participant we refer, the separate sequence number spaces also facilitate identification of receive and transmit variables[2]. These change, their value in a logical expression is a function of time[3] t:

$a.b{+}1_A$ — is True iff Alice's ((tx.an == a) $\wedge$ (tx.dan == b+1)) at time t

$b.a_A$ — refers to Alice's receive variables

$b.a_B$ — refers to Bob's transmit variables

$a{+}1.{*}_A$ — refers to Alice's tx.an alone, and is True iff her ((tx.an = a) at t, independent of tx.dan.

Strictly speaking:

$$a{+}1.{*}_A \equiv \exists x{:}\ (a{+}1.x_A)$$

and similarly for all other expressions including '*'.

If parts of any given logical statement are not qualified as to time, the entire statement is considered as being evaluated at the same time, e.g if p and q are declared to be time dependent variables:

$$p \Rightarrow q$$

means that q is True when p is True. Note that the individual members of AN_A and AN_B are time independent (and are not boolean variables).

Let M_{ab} denote the declaration of a topology match by Alice at time t when her AN is a and the last message she received from Bob had AN b. From the definition of the protocol:

$$
\begin{aligned}
M_{ab} \equiv \quad & & \text{......}\ (1)\\
& (a.{*}_A \wedge D(a,b)) & \text{.....}\ (1a)\\
& \wedge\ (\quad (b.a_A) & \text{....}(1b.1)\\
& \qquad \vee\ (b.a{+}1)_A) & \text{....}(1b.2)
\end{aligned}
$$

Equation 1 says that Alice declares a topology match at time t when her transmit AN is a and the last message she received from Bob had AN b iff: (a) her transmit AN is indeed a, and the digests for a and b match, and either: (b.1) the received AN is b (as required) and the received DAN a; or (b.2) the received AN is b and the received DAN a+1.

We wish to prove that:

$$
\begin{aligned}
& \exists t_1{:}((t = t_1) \wedge M_{ab}) & \text{......}\ (2)\\
& \Rightarrow \quad \exists t_0{:}(& \\
& \qquad\qquad \forall\, t{:}((t = t_0) \wedge (t_0 < t_1)) & \\
& \qquad\qquad\qquad (b.a_B \vee b.a{+}1_B) & \text{.....}\ (2a)\\
& \qquad\quad \wedge\ \forall\, t{:}((t_0 < t) \wedge (t < t_1))(& \\
& \qquad\qquad\qquad\quad \forall\, b'{:}(\neg D(b,b'))(\neg M_{b'*})) & \text{...}\ (2b)
\end{aligned}
$$

Equation 2 (required to prove) says that Alice's declaration of a topology match with transmit AN a and received AN b at some time t_1 implies that at some earlier time t_0 : (a) Bob's transmit AN was b, and his

[1] The sets generated by a_1 = 1 and b_1 = 1+i and the successor function for each set are suitable.

[2] The messages are only interesting in that they permit one participant's variables to result in a later change in the others. Conventional advice is to model the channel(s) containing the messages within the overall system state, which causes the latter to be rather complex. Here we model the channel purely in terms of possible changes to receive variables, or requirements on transmit variables if receive variables are to change.

[3] Formally there is a set of times, with an ordering relation, and a set of tuples for each time-dependent variable, with one element for each time value with that value as part of the tuple and the value of the variable at that time as the other part of the tuple. Spelling this out every time we want the value of the variable at a particular time is just a little tedious.

transmit DAN a or a+1—these are necessary conditions for a message transmitted by Bob at that time to result in the topology match M_{ab} (see eqn. 1); and (b) Bob did not declare a conflicting topology match (with a digest that didn't match that for b) in the intervening time. These are sufficient conditions for loop-freeness: the rules for transmitting any Digest require Bob to reduce his forwarding to that (or a subset of that) for the topology identified by the Digest, and not to increase it unless he declares a topology match. The equation (once proven) can be applied (with the substitution of different free variables in place of a and b) to successive topology matches by both Alice and Bob, so a loop-free condition will persist for ever.

Whenever t_0 and t_1 are used in this proof they refer to time values that satisfy eqn. 2.

Alice's declaration M_{ab} requires her prior reception of b.a or b.a+1 so proving (2a)—the existence of a suitable t_0—is trivial, it follows directly from:

$$\begin{aligned}&\exists t_j:((t = t_j) \wedge y.x_X) \;\Rightarrow &\ldots\ldots(3)\\&\exists t_i:((t_i < t_j) \;\wedge \forall\, t:(t = t_i) y.x_{Y\neq X})&\end{aligned}$$

Equation 3: one participant's receive AN and DAN will not take given values unless those have previously been the values of the other participant's transmit AN and DAN respectively[1]. This causality is generally assumed without explicit reference below. The rules of the protocol also require that each participant's (transmit) AN only increase over time:

$$\begin{aligned}&\exists t_i:(x.*_X) &\ldots\ldots(4)\\&\Rightarrow \quad \forall\, t:(t \geq t_i)(\forall\, y:(y < x)(\neg\, y.*_X))&\end{aligned}$$

Equation 4: If a participant's (X's) transmit AN is x at time t_i, then it cannot be y less than x later. Conversely it was x at time t_j it cannot be y greater than x earlier. If messages are not delivered out of order, or out of order messages are detected and discarded on receipt then eqn. 4 means that the recipient's receive AN can also only increase over time. Any reference below to the behavior of the protocol in the absence of misordering may assume these properties.

Looking at (2a), we have from the protocol definition:

$$b.a_B \wedge D(a,b) \quad \Rightarrow \quad (a\text{-}1.b''_B) :(b'' \leq b) \quad \ldots\ldots(5)$$

Equation 5: If Bob's transmit AN and DAN were respectively b and a (with matching digests), then his receive AN was a-1[2], with a digest matching that for Bob's transmit AN at or after the time of receipt but before Bob's transmit AN became b, since D(a-1,b) is False (see above). As a consequence Bob's received DAN (b" above) cannot be greater than b[3]. Furthermore (from eqn. 1):

$$\forall\, t:(t = t_1)(a.*_A \wedge \; b.*_A) \quad \ldots\ldots(6)$$

so:

$$\neg\exists\, t:(\exists\, b':(a\text{-}1.*_A \wedge b'.*_A \;\wedge (b' > b)) \quad \ldots\ldots(7)$$

and therefore (from (5) and (7)):

$$\forall\, t(\neg\exists\, b':((b' > b) \;\wedge a\text{-}1.b'_B)) \quad \ldots\ldots(8)$$

Also[4] looking at (2a), from the protocol definition:

$$b.a{+}1_B \wedge D(a,b) \Rightarrow \quad a.*_B \vee a{+}1.*_B \quad \ldots\ldots(9)$$

In the absence of misordering:

$$\begin{aligned}&\forall\, t:((t_0 \leq t) \wedge (t \leq t_1))(&\ldots\ldots(10)\\&\qquad(\forall\, b':(b'.*_B)(b' \geq b))&\\&\quad\wedge(\forall\, a':(a'.*_B)((a' = a) \vee (a' = a\text{-}1))&\end{aligned}$$

at and between t_0 and t_1 Bob's transmit AN was greater than or equal to b, and his receive AN was a or a-1. A conflicting topology match could have only taken place if these were not b and a respectively, so using eqn. 1 and 10:

$$\begin{aligned}&\forall\, t:((t_0 < t) \wedge (t < t_1))(\forall\, a' \;\forall\, b': &\ldots\ldots(11)\\&\quad(\neg D(b,b') \;\wedge M_{b'a'})&\\&\quad(b'.*_B \quad \wedge (b' > b) \;\wedge (a' = a\text{-}1)&\\&\qquad\qquad \wedge \exists b'':(a'.b''_B \wedge ((b'' = b') \;\vee (b'' = b'{+}1))&\end{aligned}$$

Equation 11: At any time between t_0 and t_1, for any transmit AN values a', b' such that Bob declares a topology match conflicting with that for b (and, by extension, with that for a), Bob's transmit AN is b' (greater than b), his receive AN is a-1, and his receive DAN b" (equal to b' or b'+1). But, eqn. 8 stated that Bob's receive variables can never be set to that combination of values. So Bob cannot declare a conflicting topology match.

Q.E.D.

[1] Messages can be lost, otherwise the implication '$\Rightarrow$' would be bi-directional.

[2] Only the reception of a-1 (with a digest match at or after the time of receipt) and a could result in Bob having a transmit DAN of a, and if D(a.b) the latter would result in a DAN of a+1, as in the following equation. Reception of an in-order message always results in tx.dan being set, it is not carried forward as it can be on a topology update.

[3] If a participant's AN is b-1 then its DAN cannot be greater than (b-1)+1.

[4] See eqn. 5 above.