Synchronizing DRNI state

Mick Seaman

This note describes a simple sequencing protocol that allows peers communicating using idempotent protocols and connected by media that can, at times, misorder, duplicate, and delay packets for periods in excess of the intervals between changes to the variables they wish to communicate to be sure that their peers have the latest copy of those variables. The protocol deals with the possibility of a participant being rebooted without that being apparent to its peers, without requiring non-volatile storage for transient parameters. This capability distinguishes it from the simple use of a very large sequence space to identify successive changes to a variable set.

1. Background

IEEE Std 802.1AX Link Aggregation specifies a Distributed Resilient Network Interconnect (DRNI). A pair of cooperating DRNI Systems can aggregate links connected to either of them into a single Link Aggregation Group (LAG). Each of those aggregated links can be can be connected to one or other of another pair of DRNI Systems. The resulting configuration is resilient to link and system failure, and is deployed where high availability is required, e.g. to support network to network interconnect (NNI). In this application DRNI supports dynamic service provisioning and administrative independence for each of the connected networks. A given customer service can be allocated to any one of the aggregated links that interconnect the networks with reallocation as required if that link fails, while transmission into and out of each of the connected networks can be allocated to a port on either of that network's DRNI System pair. The paired DRNI systems are connected by an intra-relay connection (IRC) that allows a frame received by one of the paired DRNI Systems to be transmitted to the other, so it can be relayed on the correct link or network port.[1]

Each data frame that a DRNI System receives from the IRC can be destined for the local network (an Up frame that its neighbor—its partner DRNI System—has received, on one of its aggregated links, from the other network or destined for one of the receiving systems aggregated links (a Down frame that its neighbor has received from the local network). To avoid the definition of a new tag, and to allow the IRC to be supported by a wide range of existing technologies, these data frames do not carry an additional tag to identify each of them as an Up frame or a Down frame. However it is important that the receiving DRNI System not mistake one for the other, as returning an Up frame to the other network or returning a Down frame to the local network can create a forwarding loop or disrupt other aspects of network operation. Correct Up/Down forwarding depends on synchronizing changes to all or some of the variables that determine how the frames associated with each conversation are allocated to links and local ports. A DRNI System that has made a change, as a result of administrative action or as a result of a link failure or restoration, but is not sure whether its neighbor has updated its own copy of the relevant variables, can discard a data frame that it would otherwise have forwarded across the IRC and/or discard a data frame that it has received from the IRC. Various forwarding rules are possible, though they all depend on synchronizing some variables.

As mentioned above, an IRC can be provided in a number of ways. Paired DRNI systems can be adjacent in the same rack, separated so that they can benefit from independent power and environmental support, or separated by a considerable distance so that the two sets of links used by a network-to-network link aggregation group are completely route diverse. The IRC itself can be a link aggregation group, and can use its own provisioned service provider connection through bridges in its home network.[2] These alternatives mean that the IRC can, in some deployments at least, introduce occasional misordering and frame loss and delays that may exceed the intervals between configuration changes resulting from link failure or restoration or from automated management.

The Distributed Relay Control Protocol (DRCP) that paired DRNI Systems use to synchronize their state variables also runs over the DRC. While very rare frame loss and misordering might be tolerable for

[1] 802.1AX-Rev/D1.0 provides a detailed explanation with diagrams.

[2] Any set of technologies or protocols capable of providing the MAC Internal Sublayer Service (802.1AC

customer data frames, a DRC error that resulted in looping frames is not, particularly if an error in one of an NNI's paired DRNI Systems resulted in returning a frame to the other network. The effects of a failure in one network's half of an NNI are meant to be confined to that network.

The design challenge then is to select rules and DRCP protocol elements that deal with IRC misordering without being so complex that implementors and administrators will avoid their use in simple cases, e.g. when the IRC between paired DRNI Systems is very short and thought to be proof against interruption. When rapid changes are made to variables that are to be synchronized, each DRNI System can assume that they are synchronized only when the latest change has propagated to its neighbor, without the possibility of intervening variables values subsequently appearing at that neighbor.

Of course a network administrator might choose to support the IRC with a TCP tunnel, but that would provide an unnecessary guarantee (the sequential delivery of all intervening variable changes when only the final state is required) at the cost of additional delay when failures occur.

2. Synchronization protocol

The proposed protocol consists of two layered elements. First a protocol that discards out of order frames, second a set of distinct labels for each successive version of a set of variables so that the transmitter of those variables can be sure that the most up to date copy has been received. In the absence of any future changes, the misordering prevention will ensure that received copy can be trusted.

The important design point so far as the misordering prevention is concerned is that it satisfies a cardinal rule of distributed system protocol design: i.e. after a known bounded (and acceptable) time during which the participants in the protocol behave correctly and all transmitted frames are received in order, the protocol participants will reach a desired legal state that is determined by the values of the variables that the protocol is intended to convey, irrespective of the initial values of the variables (however ridiculous) used by the protocol in its operation. This accommodates the possibility of one of the participants being reinitialized and returning the protocol specific variables to their initial values. This criteria is not met by simply assign a packet number, starting from some initial value, to each transmitted packet and discarding all received packets carrying a number that is not greater than the highest number previously received.

Each packet (DRCPDU) includes a packet number assigned by the transmitter and a reflected packet number for each receiver. The protocol description below considers only two participants, Alice and Bob (say), as is the case for DRCP, though it should be easy to see how it can be applied to more.

Alice receives a packet from Bob, containing the two sequence numbers:

```
numbered     // assigned by Bob
reflected    // from a packet Bob received from Alice
```

and checks them against three variables that she maintains:

```
highest_numbered   // packet received from Bob
highest_reflected  // in a packet received from Bob
highest_sent       // previously, by Alice to Bob
```

accepting (or discarding the packet), and conditionally updating those variables, and the variable

```
reflect            // to be reflected to Bob
```

The following code fragment returns True if the received packet is to be accepted, and False otherwise:

```
bool   accept = False;
if   (numbered    > highest_numbered)
{    accept = True;
     highest_numbered = numbered;
}
if   (    (reflected >   highest_reflected)
     && (reflected <= highest_sent)
     )
{    accept = True;
     highest_reflected = reflected;
}
return (accept);
```

when Alice transmits a packet she increments the value of highest_sent and places that value in the packet's numbered field and the value of reflect in the packet's reflected field.

The sequence number fields can be linear, with a large enough sequence space for all the packets that could be sent by a continuously operational DRNI system, at least 32 bits, or can be circular. In the latter case '>' '<' and '<=' are interpreted as denoting the relevant half of the sequence space relative to value to be tested. A modest sized space, with half the space covering the maximum number of packets that could be in flight between Alice and Bob and back again, is sufficient. 8 bits should be more than enough, with further transmit opportunities denied if the values of highest_reflected and highest_sent are too far apart. In

any case the IRC should be pronounced ‘down’ if that criterion holds up transmission for more than a second or so (data may be flowing, but the neighboring DRNI System is probably brain dead).

The responsibilities of the upper, variable set labelling, element of the synchronization protocol have already been described. The important point is that the labels potentially in use at any time be distinct, even if they are labelling the same information. If, for example, one of the DRNI System’s set of active link changes (from set A to set B, say) then changes back again (to set A) it is not sufficient for that system to know that its neighbor’s view of those links is that of set A. That knowledge doesn’t preclude the neighbors view from unexpectedly changing to B (for a while at least). Contrariwise if the successive versions of the set are numbered 1, 2, 3, then the current situation is clear. The version numbering space can be quite small (if circular), just sufficient to uniquely label the distinct version that can be extant at any one time.

3. Is this really necessary

The synchronization protocol described above can be used in a number of applications, so the question is do we need to use it (or something with equivalent functionality) to support DRCP. We probably need something to ensure that a neighboring DRNI system has not become brain dead. There is little point in specifying DRCP if it needs supplementing with unspecified mechanisms and so doesn’t provide interoperability. Apart from that the need is principally driven by the fact that the set of active DRNI LAG links can change frequently, on timescales comparable with those associated with higher delay IRCs. Changes to the available and preferred Gateway ports could be restricted to occur at a lower rate (with the possible exception of those made by the DRClient—referred to as NetworkControl in the D1.0 specification).

In the D1.0 specification frames received from the IRP are handled by the DR_IRP state machine (Figure 9-10) and the conditions for transition to the PASS TO AGGREGATOR and PASS TO GATEWAY both involve using neighbor state. The result is to place more synchronization responsibility on the two DRNI Systems than necessary. Consider what would happen if we added a check to discard a data frame received from the IRC if the local system’s port conversation id and gateway conversation id and the related masks allowed that frame to pass through both the local gateway port and one of the local aggregation ports[3]. When reflected configurations (i.e. those identical if, for one of them, System A is relabelled as System B) are discarded there are only two correct possibilities for the path taken by a given frame, either both its gateway port and its aggregation port are in the same DRNI System or they are in different systems. It is not possible to move from one of these correct configurations to an unwanted Down-to-Down or Up-to-Up configuration without at least one of Systems changing its idea of which of them should provide the gateway port for that frame (either neither, admitting the Down-to-Down possibility, or both, admitting the possibility of Up-to-Up. It follows that we only have to synchronize gateway port[4] changes to prevent these unwanted forwarding scenarios. The intervals between those changes could be restricted to be much longer than any possible IRC delays, thus avoiding the need for explicit synchronization.

[3]Of course we need to make sure that an frame is not forwarded across the IRC if the local System believes its gateway port and aggregation ports are both local.

[4]And conversation id mapping changes, though these should take place much less frequently.