Specification for NACK Behavior in NLSR

Nick Gordon

October 14, 2016

Abstract

This document describes what behavior NLSR should take when receiving NACKS. The optimal behavior differs from case to case. All important cases are given here.

1 Introduction

There are multiple cases to consider with respect to Interest type when describing NLSR's optimal behavior with regards to NACK processing. Generally, there are two broad strategies. We identify five different NACK situations and the responses for each. Further, in each case, the mention of logging implies a higher level than normal; all important operations of the code generally emit log messages.

It is also important to consider that if NLSR receives a NACK, then all of the outgoing Faces have been NACKed, not just one.

In all cases, "no reason" is also the recommended behavior for unlisted reasons. Additionally, it is important to note that "congestion" as a NACK reason has not been implemented yet. As such, we have elected to treat it as part of "no reason" until the semantics and scenarios associated with it are established.

2 LSA, key, and validator Interests

This is the most detailed case. Generally, all three of these situations can be handled identically. They are:

- Congestion: If the network is congested, then we want to wait some time and then resend the Interest. A backoff algorithm should be used to prevent further congesting the network. A retry limit should not be used, as these three types of data are core data, and NLSR cannot function without them. As such, once we reach the ceiling of the backoff, we continue until outside intervention or the data is gotten.

  However, no strategy currently implements this NACK reason, and this is because the semantics and situations in which it occurs are not well-understood.

- No route: This may be caused by a link failure, and so does not represent a total failure of routing; if some Face is destroyed, that will be reflected in adj. LSAs and reactions will propagate accordingly. In the case that we have no route, the best approach is to wait a short while and then resend.
- Duplicate: We should resend the Interest in the case we get a duplicate NACK. Of note is that it is impossible to have an Interest sent with multicast strategy NACK every Face at its originating router. In other words, if all Faces from router A converge at router B, then at least one Face out of router A will remain un-NACKed, and the strategy at A will not deliver to the consumer, i.e. NLSR, a NACK. This means that when NLSR receives a NACK with duplicate reason, in fact the "tipping point"-NACK was for another reason. However, currently we can't know what that reason was, so we should simply resend and hope it doesn't happen again.
- No reason: We can't do much about this NACK. The best recourse is to log a threatening, ominous message and hope that an operator figures out what's going on.

3 Hello Interests

This is the second interesting case. However, we note that only two cases can occur. That is, only the "No route" and "No reason" NACKs are possible:

- No route: A NACK with this reason for a Hello Interest indicates that NLSR is down on that router or that the link out of this router is down. That is, NFD has gotten the Interest, but does not know where to forward it. This will only happen if NLSR's prefixes are damaged or not registered altogether. We should emit a particular error message and treat that neighbor as down. We will retry as normal at the next Hello interval.
- No reason: The same strategy that is taken in the LSA case is taken here.

4 Sync Interests

We should probably ignore all NACKs gotten by Sync. This is a practical reason, as NLSR currently uses a forked version of ChronoSync. Currently, the original ChronoSync does not process NACKs, either, and so we do not want to write patches for outdated code that will be obviated by a merge back, anyway.

5 FaceStatus Dataset

When NLSR starts, it requests a FaceStatus dataset from NFD, which theoretically provides the information NLSR needs to configure its neighbors. In the

case that NLSR cannot fetch this dataset, the response should be the same for all NACK reasons; do nothing. This is because NLSR is also listening to the FaceEvent stream, and will get Face information as it as added to NFD. Additionally, NLSR will refetch the dataset at long intervals of typically an hour, to catch the cases where NFD Face events get lost for some reason.

6 Nlsrc

In the case of Nlsrc-originating interests, we should only emit errors for the operator. This is because nlsrc, as a command-line utility, should not autoresolve issues like this. These are the NACK reasons:

- No route: In this case, NLSR is not running on the localhost, and so we should log an appropriate error, but do nothing else.
- No reason: We should log some kind of error, but do nothing else.