FIB Update Summary

I. FIB Update Failure

This section discusses causes of FIB update failures and necessary actions to handle the failures.

A. Causes

It is critically important that after the FIB update process, the FIB and RIB are consistent with one another. If they are not consistent, issues with future registration and unregistration may occur. There are two possible cases where failures may occur in the middle of the FIB update process: timeouts and invalid faces.

Assume you have the following RIB:

Current State of RIB

Name	Face ID	Cost	CHILD_INHERIT	CAPTURE
/a/b	1	5	FALSE	FALSE
/a/c	1	5	FALSE	FALSE

and assume you have the following RIB update batch:

RIB Update Batch

Name	Face ID	Cost	CHILD_INHERIT	CAPTURE
/a	2	10	TRUE	FALSE

This will generate the following FIB updates:

FIB Updates

Name	Face ID	Cost	Action
/a	2	10	ADD
/a/b	2	10	ADD
/a/c	2	10	ADD

If the updates for /a and /a/b succeed but the update for /a/c fails due to timeout, the entire RibUpdateBatch will fail and the RIB will not be updated.

The problem caused by this is that the updates for /a and /a/b are in the FIB, but there will be no way to unregister them. If an unregistration command comes for /a, the entry does not exist in the RIB and there will be no way to know what or if longer prefixes inherited this route.

Another potential case is the issue that FIB updates may contain any number of different Face IDs (for one or multiple RIB updates batched based on Face ID). For example:

Current State of RIB

Name	Face ID	Cost	CHILD_INHERIT	CAPTURE
/	1	50	TRUE	FALSE
/a	2	25	FALSE	FALSE
/a/b	3	10	FALSE	FALSE

If the following RIB update is applied:

RIB Update Batch

Name	Face ID	Cost	CHILD_INHERIT	CAPTURE
/a	4	20	FALSE	TRUE

The following FIB updates will be computed:

FIB Updates

Name	Face ID	Cost	Action
/a	4	20	ADD
/a	1	-	REMOVE
/a/b	1	-	REMOVE

Note that while the RIB update is for Face ID 4, two updates were generated that alter entries with Face ID 1.

Even if the FIB updates are successful for one Face ID, other updates may fail for different Face IDs. So, though the FIB does not add the route with the invalid Face ID, it will not remove the earlier successful updates even though they belong to the same FIB update grouping.

These issues can occur with just one RIB update, as shown above, or multiple RIB updates batched together.

B. Handling Failures

In the case where an update continuously times-out, the FIB Updater can try to apply the update a set number of times before declaring the batch a failure. When this failure occurs, the RIB daemon should crash and both NFD and NRD should restart (Task #1585). Once both processes are restarted, the FIB and RIB will be synchronized.

In the case where an invalid Face causes an update failure, the FIB Updater can reverse the previously successful updates to synchronize the FIB and the RIB. Once all of the updates have been reversed, the RIB update batch can be abandoned. If the updates cannot be reversed due to timeout, the previous timeout solution can be applied.

In non-recoverable error cases (e.g., key is not valid), the RIB Daemon can simply crash.

II. Operation Ordering in FIB Updater

One important decision is whether or not FIB updates should occur before or after the RIB is updated.

One option is to update the RIB before FIB updates are calculated and sent. If an error occurs, changes will need to be reversed in the RIB. If nothing is done to reverse the incorrect state of the RIB, future registrations may be handled incorrectly: If a prefix is in the RIB but not in the FIB, an application that registers the same prefix will not generate FIB updates. To recover from this, the RIB update corresponding to the failed FIB updates would need to be reversed in the RIB, and any FIB updates that previously succeeded would need to be reversed or the RIB Daemon and NFD must restart to synchronize the FIB and RIB.

Alternatively, in Junxiao's proposed design, FIB updates are calculated, sent, and then the RIB is updated. If a FIB update fails due to an INVALID_FACE or timeout, the rest of the FIB updates are abandoned and some previously successful FIB updates may remain in the FIB. To recover from this, any FIB updates that previously succeeded would need to be reversed in the FIB or the RIB Daemon and NFD must restart to synchronize the FIB and RIB.

In summary, (a) both schemes require the reversal of successful updates applied to the FIB; (b) when recovering from errors, the scheme that updates the RIB first requires an additional step, reversing changes to the RIB; (c) the scheme that updates the FIB first may require more memory to correctly calculate the FIB updates. Note that (c) is an issue only when the RIBUpdates are batched.

III. Batching RIB Updates

This section discusses potential issues in batching RIB updates.

A. Calculating FIB Updates

While the proposed design eliminates the step of reversing the change in the RIB, the difficulty of calculating the FIB updates is greatly increased when multiple RIB updates are batched due to the restriction of not altering the RIB throughout the update process. Not only do updates have to be calculated based on the current state of the RIB but also based on the effects of previous RIB updates.

For example, assume you have the following RIB:

Current State of RIB

Name	Face ID	Cost	CHILD_INHERIT	CAPTURE
/	2	5	TRUE	FALSE
/a	2	10	TRUE	TRUE
/a/b	2	15	FALSE	TRUE
/a/c	2	15	FALSE	FALSE
/a/d	2	15	FALSE	FALSE

And this pending RIB update batch:

RIB Update Batch

Name	Face ID	Cost	CHILD_INHERIT	CAPTURE
/a	2	10	TRUE	FALSE
/	2	5	FALSE	FALSE

Processing the first RibUpdate turns off CAPTURE on /a and generates the following FibUpdates:

FIB Updates

Name	Face ID	Cost	Action
/a	2	5	ADD
/a/c	2	5	ADD
/a/d	2	5	ADD

Processing the second RibUpdate (without considering the first RibUpdate) turns off CHILD_INHERIT for / and generates no FibUpdates because in the current RIB, /a still has CAPTURE set. This is incorrect as /a, /a/c, and /a/d should no longer inherit Face ID 2 with cost 5. The FibUpdater will now need to check previously calculated updates to guarantee the correctness of FibUpdates, because the CAPTURE flag or CHILD_INHERIT flag may have changed on another name during a previous RibUpdate.

The correct FibUpdates should be:

FIB Updates

Name	Face ID	Cost	Action
/a/c	2	10	ADD
/a/d	2	10	ADD

But, without maintaining some sort of additional state, the FibUpdater does not know how one RibUpdate may affect the next RibUpdate in the same batch.

One potential idea to ease this calculation is to use a shadow RIB that copies portions of the RIB. The shadow RIB can be updated during the FIB update calculation process and used to simulate modification of the RIB. With the shadow RIB, the current FIB update algorithms can still be used.

Suggestions are welcome on how to calculate the FIB updates efficiently.

B. Delay in Responding to Applications

Another issue is the effect that batched and queued updates will have on the RibManager response delay. If an application registers a prefix, the RIB update may be added to a batch at the end of the access control queue. The response to this application will need to wait for all updates in the queue to be processed, all FIB updates to be calculated, all FIB updates sent and verified, and the RIB to be updated. This process assumes the best case scenario; it is possible that FIB updates may fail for certain batches where they may need to be retried or abandoned with the FIB alterations reversed. There is already a timeout issue caused by OSX's slow keychain and the added delay may only further increase the problem.