NFSv4 Minor Versioning

Issues we need to look at

David Noveck

Nfsv4 Working Group meeting at IETF87

July 31, 2013

Overview

- Intense look at minor versioning and how well it is or isn’t working, for:
  - Fixing protocol bugs
  - Adding protocol features
- Will talk about problems in both these areas
- Will limit mention of “trains” or “à la carte”
  - Please do the same as regards questions
- Will not propose adjustments for now
  - Would take a long time to come to consensus
  - Might make sense to discuss in Vancouver or London

Why does NFSv4 have minor versioning?

- Because major versioning was just too frightening:
  - And we found, doing v4.1, that we really didn't need it:
    - We did major-version-like infrastructural changes using an xdr extension model (and survived)
  - But we didn't follow the intended paradigm
- To add features without changing much else
  - Was the original idea
  - Now, can only be done *after* mandatory v4.1 changes
  - Might need adjustment as we go forward.
- Other possible reasons:
  - Some on ***Next Slide*.**
  - Suggestions, anyone?

Things you'd like to do with minor versioning

- Fix some small protocol mistakes
  - As opposed to specification mistakes
  - After all, we do make them 😐 (big ones too? ☹)
  - v4.1 barrier raises micro-versioning issue
  - We could fix v4.1 mistakes in v4.2, but never have
    - Reasons probably have to do with "*feature batching*"
- Growing protocol by moving features toward mandatory status.
  - Part of the paradigm but we've never done it.
  - Not sure we ever will

Analyzing the (current) minor versioning paradigm

- Xdr extension relation among versions
- Mapping between versions and version numbers
  - Seems natural but …
    - Mapping prevents protocol bug fixes if done "too late"
      - This prevents all such fixes to v4.0, since v4.1 exists.
    - Could do others as very minor features.
      - But we never have even thought of that (outside the current paradigm).
- Feature batching
  - No features outside a feature batch
  - Will discuss ***Later*** after looking at protocol bug fix issues.

Some mistake/correction case studies

- We’ll look at:
  - Migration
  - Internationalization
  - Delegation reclamation
  - Do people know of other protocol bugs?
    - Maybe an informational document makes sense
    - Volunteers to edit/contribute?
- In each case, ability to fix protocol mistakes:
  - Would not eliminate the problem
  - But would make dealing with the problem easier

Migration case study

- RFC3530's handling of client id strings
  - Broke transparent state migration
- Were able to fix it without a protocol change
- All it took was:
  - Conceptual reworking of client identity in v4.0
  - See draft-ietf-nfsv4-rfc3530-migration-update
    - Demonstration that RFC3530 (+bis) was wrong about issue
    - Ten pages of description/clarification of trunking detection
- Worthwhile work
  - Glad we did it. We learned a lot, but …

Migration case study, continued

- In retrospect, it wasn’t the most efficient way to proceed
  - It was a high-risk endeavor. Luckily we succeeded.
- A protocol change would have addressed this in a simpler way, if we were able do it:
  - E.g. SETCLIENTID_PLUS that returned a server id string.
  - Would allow simpler uniform-client-string clients
  - Would fit in a minor-version-like extension.
- We just didn’t think of it.
  - I thought about various hacky op sequences
  - But my co-authors talked me out of it ☺
  - An added op is obvious, once you take that blindfold off
    - We had no provision for fixing protocol mistakes.
    - We didn’t think we’d ever make any. We were wrong ☹

Internationalization case study

- Even though it is not clear how to solve this,
  - We probably have to do something.

Client would be helped by attributes describing server handling. For example:

  - Preferred_normalization_form
  - Normalization_preserving
  - Normalization_sensitive

- Would be best as a v4.0 extension
  - V4.3 seems awfully late to start on this.
  - With current rules, that’s where we are ☹

Delegation reclaim case study

- In v4.0, no way to reclaim (only) a delegation
  - A small protocol mistake
  - Could have been fixed in a bug-fix minor version
- Was fixed in v4.1 but,
  - There's nothing v4.1-related about it
  - Was merged with WANT_DELEGATION
    - One case was an experimental/optional new feature
      - Which hasn't been implemented so far.
    - Other two cases do delegation reclamation, which should be a mandatory/recommended feature
      - It might well be a v4.0 bug fix in a v4.0.1
      - If that possibility were not foreclosed. ☹
    - The two functions don't fit well together in a single op

Feature Batching

- Not the same as minor versioning
  - It is a part of current approach to minor versioning
  - More troublingly, it is the *totality* of our current approach to minor versioning
  - Need to understand where it is and isn’t helpful
- What does it consist of?
  - Deciding in advance that a set of features be defined together
  - Documenting them all together (in a single RFC)

Feature Batching
What’s it good for?

- Makes sense when features are related
  - Use common facilities
  - Variations on a common theme
  - Interact in complicated ways
- Good match when features are present/absent together
  - Mandatory/infrastructural features
  - Or features where having one requires another

Ironically, our current minor versioning approach is most well-adapted to the major-versioning-like case

Feature Batching
Problems with unrelated features

- Decision on feature batch made very early
  - Hard to adapt when features take longer than expected.
  - Even worse when batch content put in charter
- Many features in batch not implemented when spec is done
  - Hard to get them ***all*** implemented in time
  - Increases protocol bug rate. We have trouble fixing such bugs.
  - No way to defer a feature (or make it experimental)
  - Results in features put in speculatively so people don’t have to wait for years, for the next feature batch.
    - That can make above issues worse. Vicious circle possible.
- Possible dilution-of-responsibility issue
  - Proposer urgency can evaporate upon feature acceptance
  - Getting feature done can become somebody else’s job

Considering Paradigm Change

Issues that need to be resolved

- Enum reservation
- Feature interaction
- Feature promotion/demotion
- Feature discovery extensions?
- Documentation approach
- General control issue
  - Working group needs a level of control but needs to avoid:
    - unhelpful centralization/serialization of decisions
    - premature decisions
  - War-of-the-metaphors seems to be about this issue

War of the Metaphors, Final Chapter

With some help from moviequotes.com

Train metaphors reflect concerns about inflexibility of current model. Trains can be efficient if circumstances are right, but don't respond well when they encounter unexpected circumstances.

- "My, my, my, my, my, my, my. What a mess"
  - Deputy U.S. Marshall Sam Gerard (the Tommy Lee Jones character), surveying the results of a railroad derailment, in <u>The Fugitive</u>.

Food service metaphors appeal to our desire/need for maximal flexibility about what we eat. Of course, we may need mechanisms in place to avoid unpleasant consequences of bad choices.

- "Somewhere in the world the wrong pig met up with the wrong bat"
  - Dr. Ally Hextal (the Jennifer Ehle character), contemplating influenza DNA, in <u>Contagion</u>

At the end of the movie, this does turn out to be a food service metaphor