How Big (or How Small) Should (or Might) NFSv4 Minor Versions Be?
Exploring the Constraints

David Noveck
IETF 77
March 23, 2010
Introduction

• Purposes
  – Exploring the issues, technical and otherwise, with the size of minor versions
  – In particular, thinking about very small minor versions
  – Exploring the document structure for “normal” (whether big or small) minor versions.
  – Stimulating group discussion

• Non-purposes:
  – Coming to any immediate conclusion on these issues
Normal (?) Minor Versions

• What things make a minor version non-normal?
  – Initiating the protocol as whole, or,
  – Violating any of the minor version rules
    • Containing mandatory new features
    • Making things mandatory-to-not-implement immediately

• So, using this definition
  – v4.0 and v4.1 are not normal minor versions
  – v4.2 is the first normal minor version
    • We should think carefully about the issues
    • We don’t have precedents to go by
    • We will be establishing precedents (due to inertia rather than *stare decisis*)
  – Expect most new minor versions to be “normal”
Constraints Taken for Granted

• No more non-normal minor versions
  – At least for quite a while
• No more 600+ page documents
• No more versions that take about 700 pages to describe (RFCs 5661 & 5662)
• No more versions as big as v4.1
  – Even if they have smaller documents due to a different document strategy
Defining the Maximum

• Can’t use pages, affected by doc. strategy
• Let’s look at big changes and guess at size
• In v4.1:
  – Sessions (including trunking) [2.0]
  – pNFS (including file layout type) [2.0]
  – Directory delegation [1.0]
  – Multi-server namespace (+new attributes) [1.0]
  – New compliance attributes [0.2]
  – New stateid stuff [0.3]
• Group should have some sense of rough maximum [2.5]? [3.0]? [3.5]?
What About a Minimum?

• How small can/should a minor version be?
• Smallest would be to correct omissions
  – “How could we have forgotten …”
  – But it isn’t an erratum
• Without arguing about whether this is an example, consider commit level
  – Why can’t WRITE tell you that you don’t need a LAYOUTCOMMIT?
  – Duh. Because we forgot to add it to the enum
When Would a v4.x be too Small?

• Issues of overhead
  – Document writing (depends on doc. strategy)
  – Group last call
  – IETF last call
  – RFC editor
  – Non-trivial. WG needs to compare to benefits

• What isn’t a big issue for small versions
  – Overhead of writing a client (as for v4.1)
  – A small v4.2 is more like a v4.1.1
    • A v4.1 client that accepts 2 in the version field conforms
    • Then the issue is implementing a small feature
Some Models for Minor Versions

• Three models discussed below
  – Marquee Feature Model
  – Timed Model
  – Maintenance Model

• Not mutually exclusive
  – Working group can adopt one or more than one
Marquee Feature Model

- Requires one or more marquee features
  - Big enough to generate interest
- Version ready when marquee feature(s) are ready
  - Plus whatever else is ready at the time
    - Should be able to credibly defer things not quite ready
  - Most similar to v4.1
    - Although we weren’t really prepared to drop things
Timed Version Model

• Decide on a minor version cadence
  – Attempt to stick to it
  – Can modify it, if it is too fast or slow
  – But generally not for individual features

• Allows people to plan
  – If a feature take longer than expected, it is deferred
  – Other features are not held up

• Client implementations can also plan
Maintenance Version Model

• To correct generally recognized omissions or mistakes
  – Which aren’t errata. Not editing mistakes.
  – Will be dispute about how important the issue is, but not about the fact that wrong choice was made.

• If there is rough consensus,
  – Group creates a small minor version, for that/those alone
  – Up to group but other sorts of things add risk, even if they seem generally OK/ready
Document Strategy

• Avoid big documents
  – One approach is to just document delta between v4.x and v4.x+1 in single v4.x+1 RFC

• Problems:
  – Gets unwieldy when x > 3
    • Each document may modify others
    • Don’t know where to go for the truth about v4.x
    • No XDR file for v4.x
  – X > 3 may happen quickly if maintenance versions

• Can reissue big RFC’s every so often, or …
Alternate Document Strategy

• Here is an alternate document strategy
• Definitely a first pass
• Appreciate working group comments
• Divides documentation up:
  – Feature documents (become RFC’s)
  – Version documents (also become RFC’s)
    • Done very late in process
Feature RFC’s

• Documents features in feature RFC’s, not the version RFC
  – Makes it easier to split up work appropriately
  – Makes it easier to put off decision on what is ready until that decision is necessary

• Consists of:
  – New sections explaining new feature
  – Descriptive sections for new ops (same format as RFC 5661)
  – Changed versions of sections from RFC5661 and earlier feature RFCs.
  – To avoid delta scanning nightmare, require full section changes:
    • If you change section a.b feature RFC has a new version, not “section a.b is the same except except for … and …”
    • In particular, if you change an operation, you have a revised version of that operation in feature RFC
Version RFC’s

• Contains:
  – Full XDR for minor version
    • Implicitly contain XDR for all versions
    • Uses “#if MINOR_VERSION > n”
    • Can programmatically check for compatibility
  – Updated OP-vs.-error tables to reflect
    • New ops, ops becoming mandatory, deprecated, mandatory-to-not-implement
  – Version document index
    • For each a.b-level section, including op and cb definitions
      – Specifies where correct (i.e. latest) version is to be found
        » RFC 5661
        » Feature RFC for this version
        » Feature RFC for previous version
Can Write Validation Tools

- Check that the XDR source processed with -DMINOR_VDRSION=n matches XDR for minor version n.
- Report on the differences in error table with regard to existing ops
- That all major sections of feature RFCs are referenced as the most current version of something in the index.
- Report on diffs when new section replaces old
- Should reduce the gap between decision on contents and the version document last-call
Should be Able to

• Have scripts which scan index and with other RFCs, produce:
  – An explanation RFC-style document, like first half of RFC5661
  – An ob/cb RFC-style document with ops listed either in numeric or alphabetical order

• Should be able to create a web site to produce minor version documents or html drafts when you type in the version number.
If we have time

• Questions
• Comments
• In any case, discussion needed on working group list