ORACLE

RFC 3530: Persistent State and Lessons in Protocol Engineering

Bill Baker and Piyush Shivam

Overview

- Implementation experiences with NFSv4 persistent state
- Highlight flaws (fatal?) in our current process of protocol engineering
- Present and brainstorm mechanisms to fix the flaws

NFSv4 Protocol State in 3530

- NFSv4 is a stateful protocol
  - Locking state: opens/locks/delegations
- Loss of server locking state causes GRACE period
- Persistent locking state
  - Save and restore all locking state to/from stable storage

You got to be kidding.. really?

- Resilience to edge conditions – a more “graceful” server
  - *9.6.3.4.3.. “The amount of information the server records in stable storage is in inverse proportion to how harsh the server wants to be whenever the edge conditions occur.”*
- 3530-bis recommends it
  - *9.14.1..“In the case of migration, the servers involved in the migration of a filesystem SHOULD transfer all server state from the original to the new server”.*
- 90 seconds downtime is not always acceptable.
- Try recovering state for a 100,000 state objects
  - 90 seconds may not be sufficient

But, is there any real use case?

- Whenever the server loses volatile server state
  - File system migration (killer feature of 3530?)
  - NFS service restart (therapeutic service reboot?)
  - Full system restart (system upgrades)
  - Rolling upgrades: Failover/Failback (clustered services)

Persistent state: two-key concepts

- State-save
  - Record all NFSv4 state in memory to disk
  - MUST be consistent with the state in volatile memory
  - MUST not regress regular NFS performance

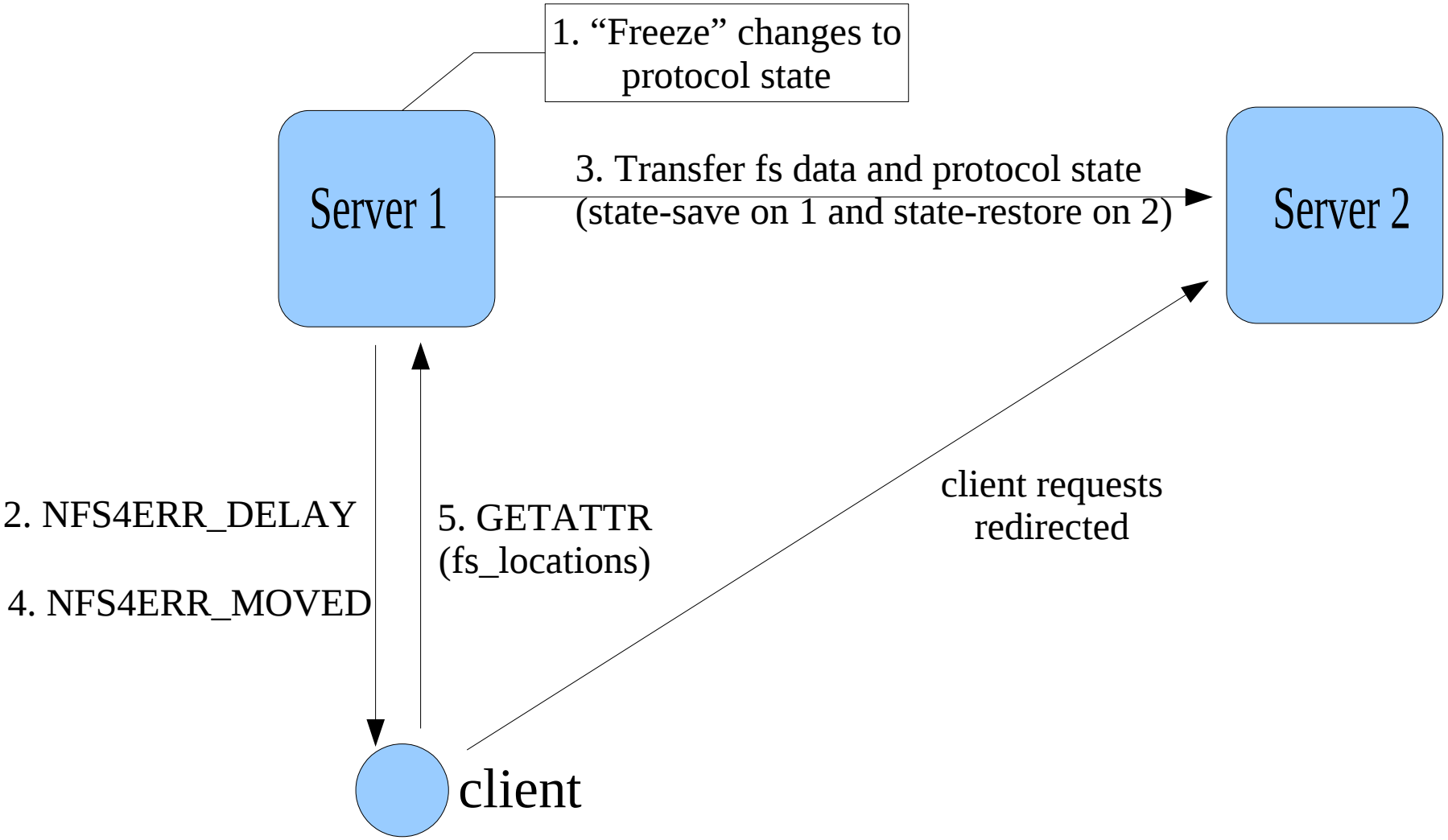
- State-restore
  - Restoring all NFSv4 state from disk to memory
  - MUST never disable GRACE period spuriously

Persistent State Implementation

- First implemented under file system migration
  - Save and restore protocol state only for the file system(s) in play

- Extended it to full service restart
  - For all file systems having NFS protocol state
  - Unplanned; born in a moment of frustration with GRACE period
  - Non-stop rebooting server (every 10 seconds) on the event floor

File System Migration: High Level Design

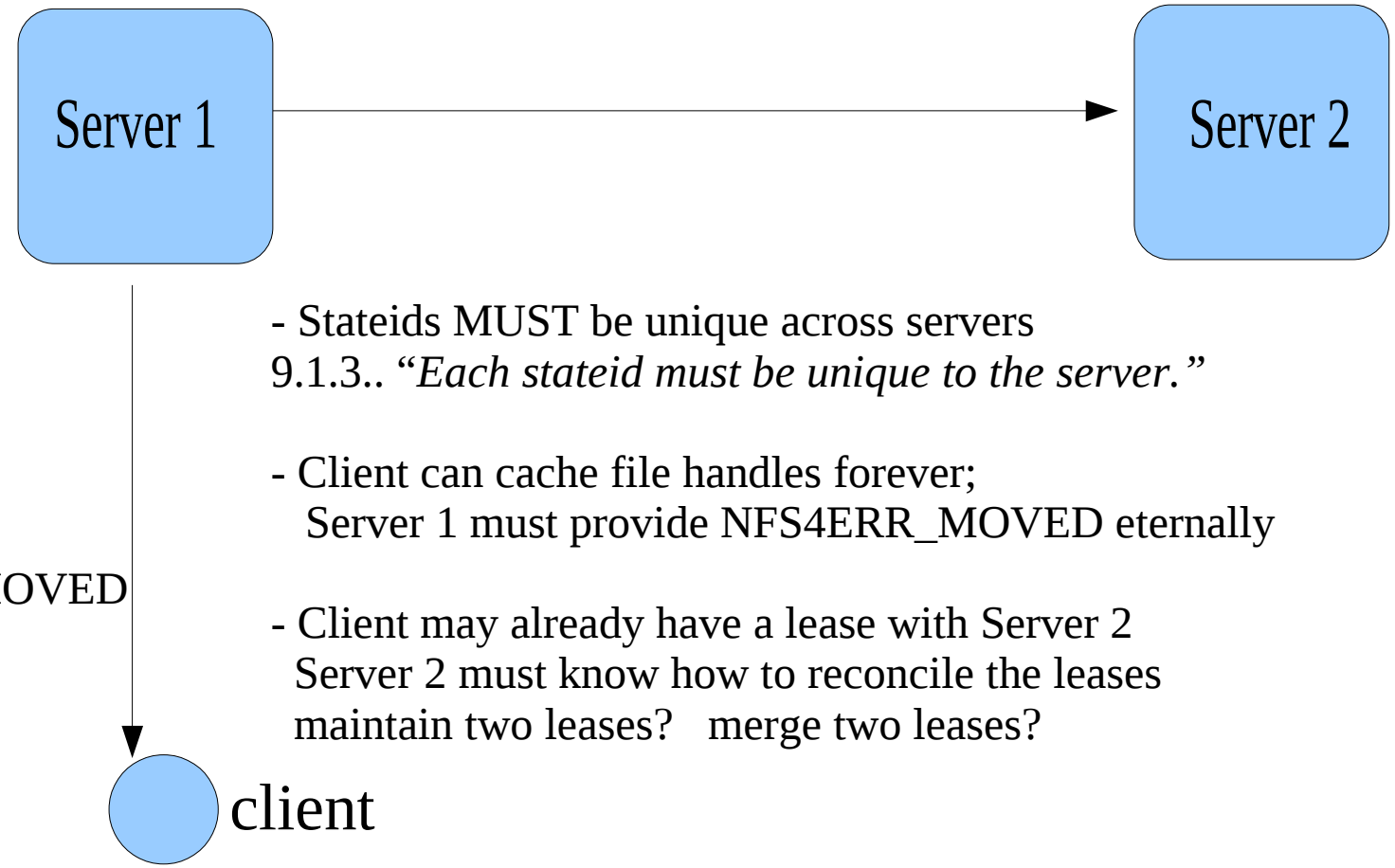


Challenges

- Ran into a myriad of issues: implementation and protocol

- See Dave Noveck's ID (currently at 40 pages)
  http://tools.ietf.org/html/draft-ietf-nfsv4-rfc3530-migration-update-01

Interesting Implementation Issues

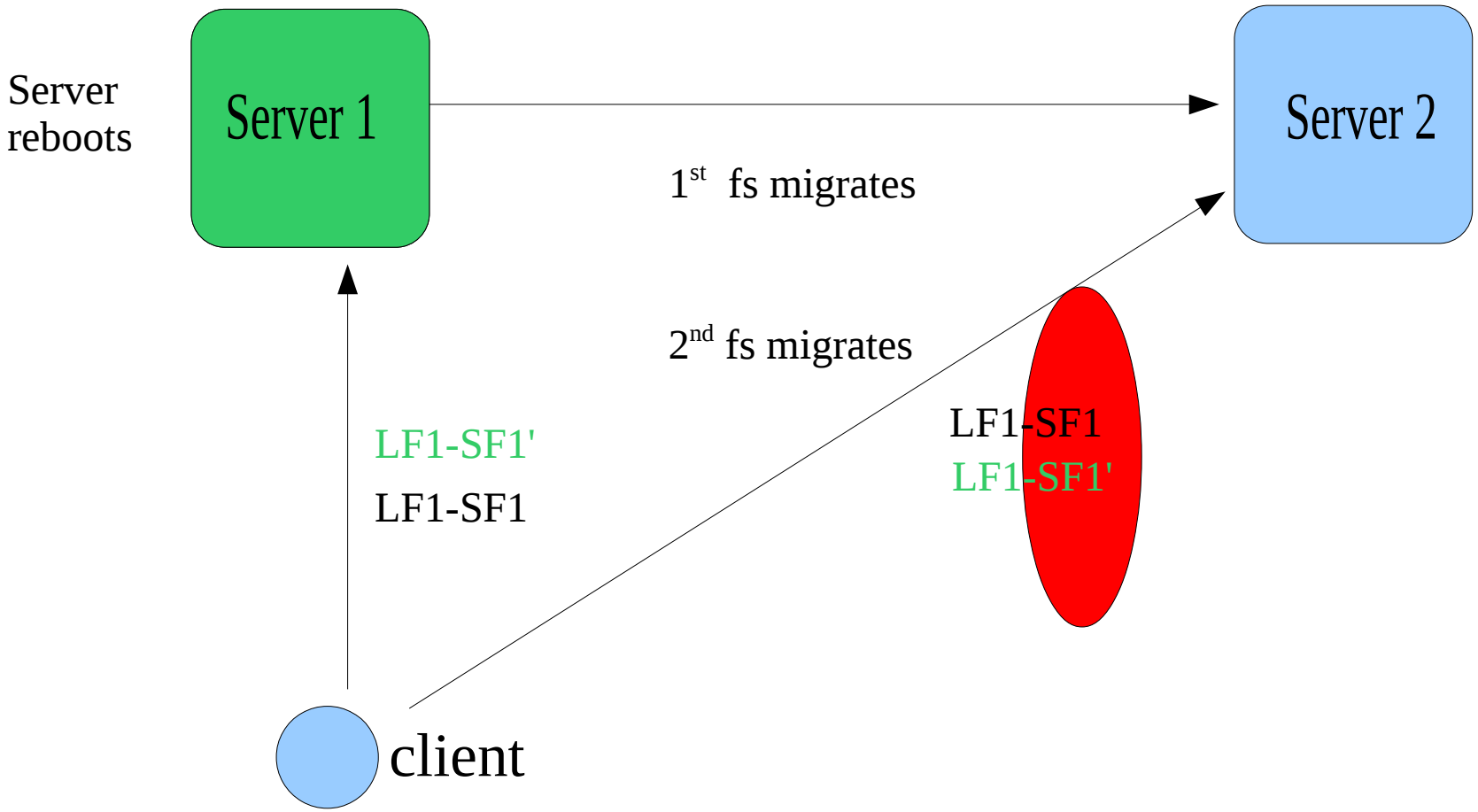


- Stateids MUST be unique across servers
9.1.3.. "*Each stateid must be unique to the server.*"

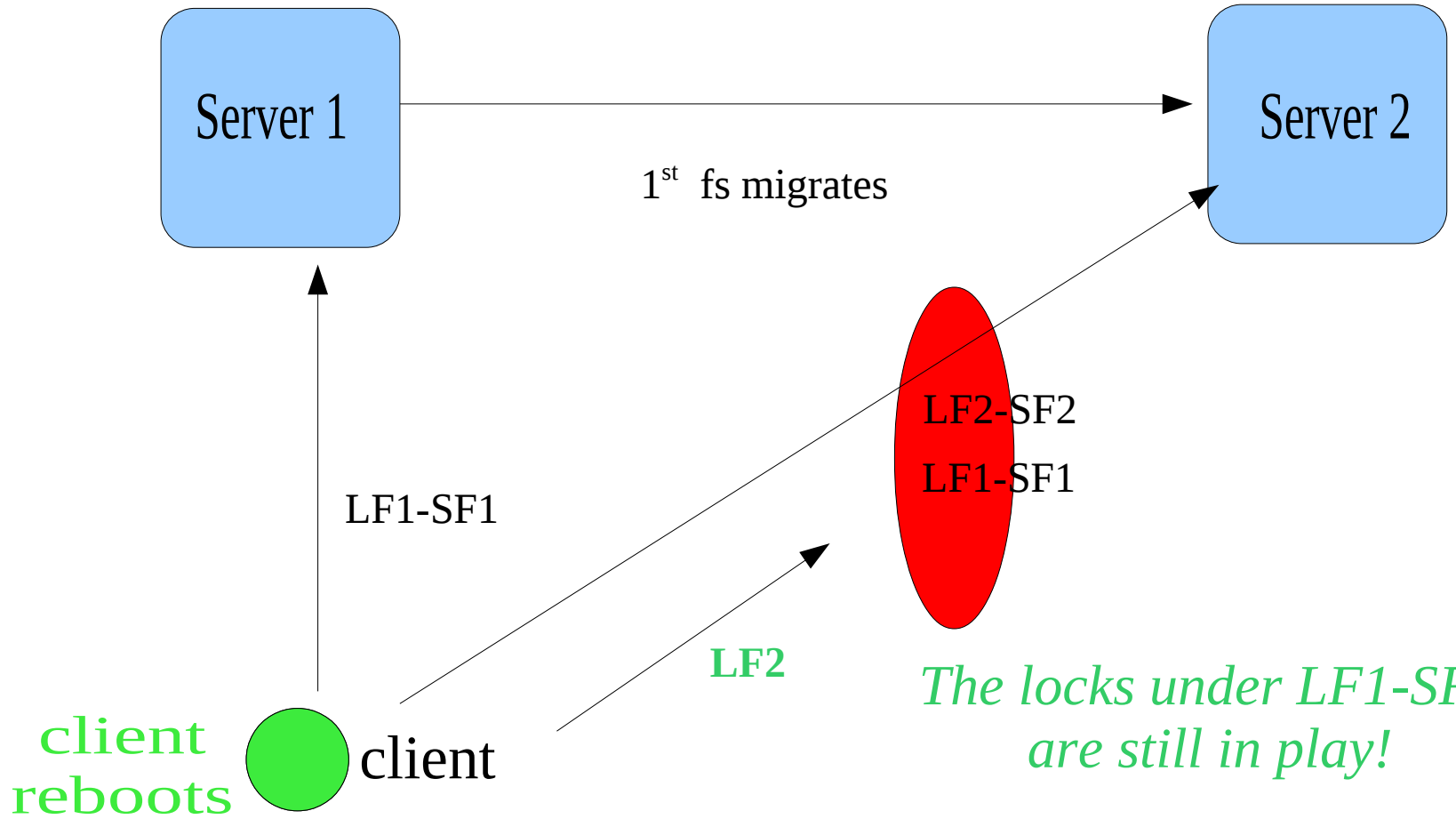
- Client can cache file handles forever;
Server 1 must provide NFS4ERR_MOVED eternally

- Client may already have a lease with Server 2
Server 2 must know how to reconcile the leases
maintain two leases? merge two leases?

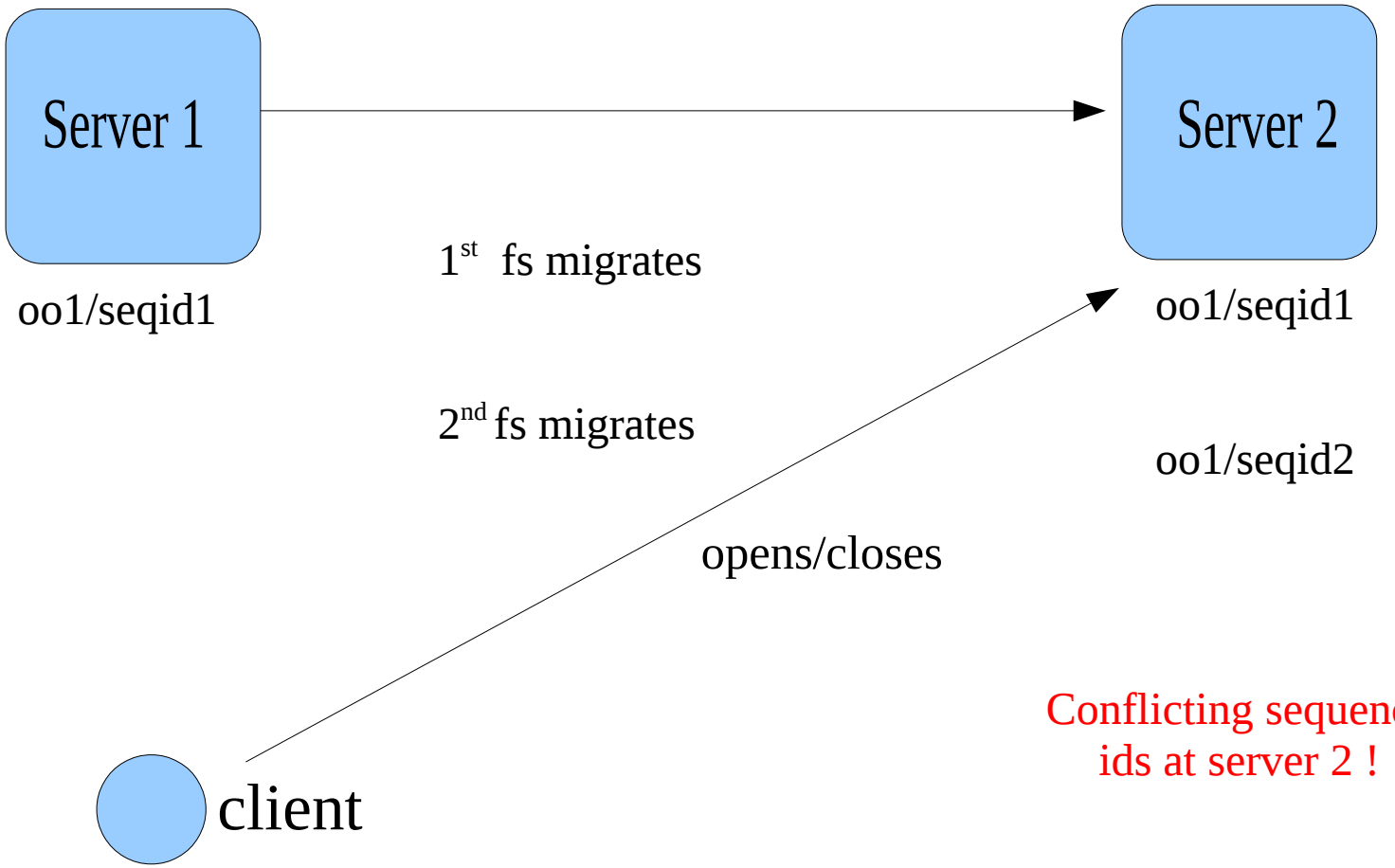
Multiple leases with a single client long form



The problem with multiple leases



Diverging open owner sequence ids



What a mess.. how do we fix all this?

- Community got together to fix the issues
- 3530-bis has some of the fixes
  - openowners must be per file system for migration
- Dave's internet draft address other issues in detail
  http://tools.ietf.org/html/draft-ietf-nfsv4-rfc3530-migration-update-01
- Serious implementations: 3530-bis + migration update draft

Fixing simple protocol problems

- Creative Interpretation of the spec
- Protocol abuse
  - NFS4ERR_LEASE_MOVED
  - How does a client clear the LEASE_MOVED condition?
    - { PUTFH, GETATTR fs_locations }
    -
    - {PUTFH, GETATTR fs_locations, RENEW clid }
  - This is a **HACK.** Why are we reduced to this?

Beyond the simple case ...

- Convention wisdom:

  minor versions, RFC-ng

- Ok, but largely ineffective

  So, to make feature X work in RFC 3530, wait for MV2?

- The feature was DOA

  For example, transparent state migration in 3530

The truth about NFSv4 minor versions

- The minor version mechanism is generally good enough to fix protocol flaws

- Good, but what really happened?

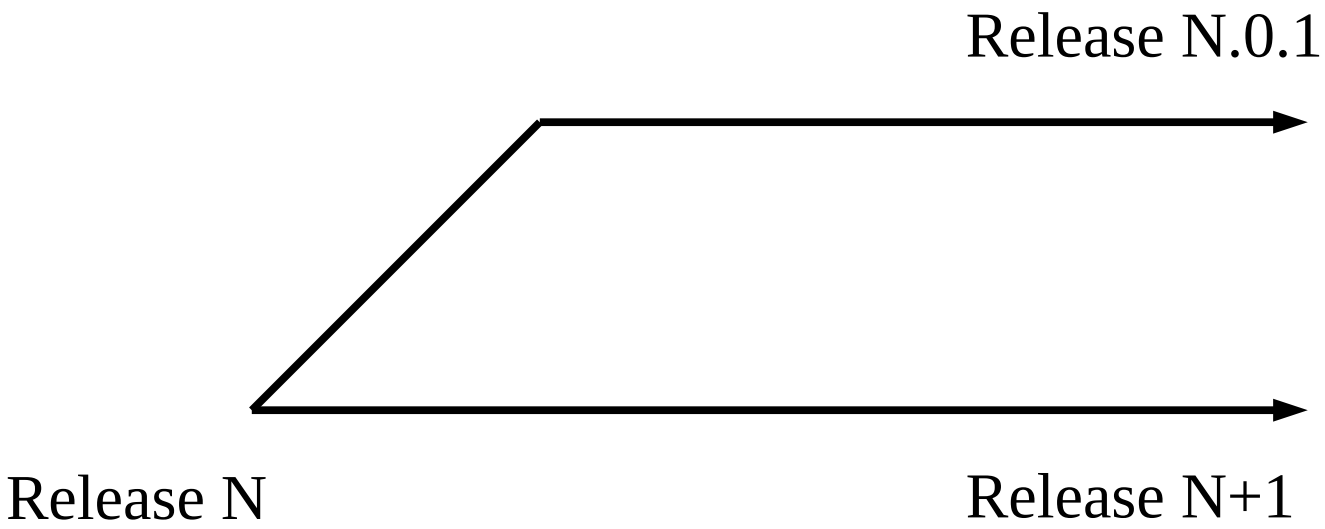
  minor versions were hijacked as a means of adding new features

  closed the door on it a means to fix protocol bugs

- This **IS** the bug!

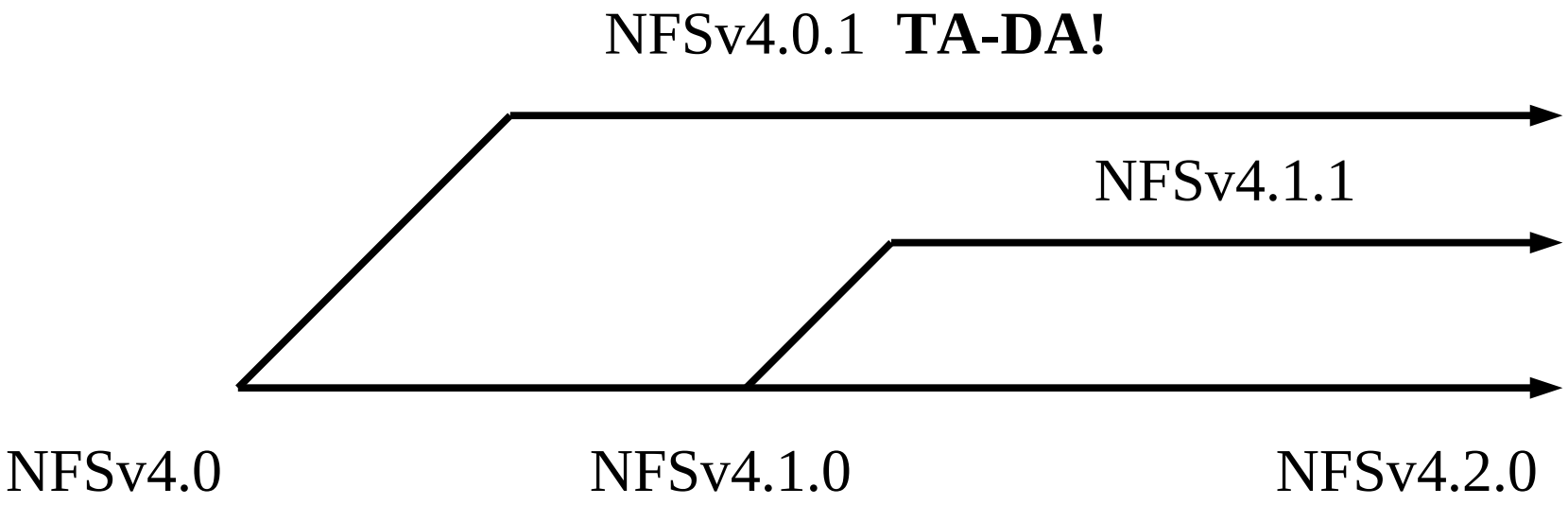
Maintenance vs. development trains

- Everyone knows this model, eh?



- Why aren't we using this model?

Maintenance vs. development trains



Minor version numbers

Same rules as minor versions, in fact, even more strict, only to fix protocol bugs that cannot be fixed otherwise, no new features, only to fix features already present in the RFC

0	NFSv4.0
1	NFSv4.1
401	NFSv4.0.1
2	NFSv4.2
411	NFSv4.1.1

Conclusions & Summary

- Implementation experience with persistent state exposed serious protocol issues in 3530
- Expect bugs, both in code & specification

“Only Human”

We have to have a way to fix bugs in a timely fashion, without resorting to hacks.


ORACLE®