NFSv4
Internet-Draft
Intended status: Informational
Expires: November 17, 2017

D. Noveck, Ed.
NetApp
P. Shivam
C. Lever
B. Baker
ORACLE

May 16, 2017

NFSv4 Migration and Trunking: Implementation and Specification Issues
draft-ietf-nfsv4-migration-issues-13

Abstract

This document discusses a range of implementation and specification issues concerning features related to the use of location-related attributes in NFSv4. These include migration, which transfers responsibility for a file system from one server to another, and trunking which deals with the discovery and control of the set of network addresses to use to access a file system. The focus of the discussion, which relates to multiple minor versions, is on providing appropriate clarification and correction of existing specifications.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This is an informational document that discusses a number of related issues in multiple versions of NFSv4.

Many of these relate to the migration feature of NFSv4, which provides for moving responsibility for a single filesystem from one server to another, without disruption to clients. A number of problems in the specification of this feature in NFSv4.0 were resolved by the publication of [RFC7931], which added trunking detection to NFSV4.0. However, NFSv4.0 remains without an appropriate discussion of trunking discovery, which has many important connections with migration. As a result, NFSv4.0 requires clarification of how the client is to respond to changes in the trunking arrangements to use, both when migration occurs and when it does not.

In addition, there are specification issues to be resolved with regard to the NFSv4.1 version of these features which are discussed in this document.

All of the issues discussed relate to the handling and interpretation of the location-related attributes fs_locations and fs_locations_info and to the proper client and server handling of changes in the values of these attributes

These issues are all related to the protocol features for effecting file system migration, or to trunking discovery but it is not possible to treat each of these features in isolation. These features are inherently linked because migration needs to deal with the possibility of multiple server addresses in location attributes and because location attributes, which provide trunking-related information, may change, which might or might not involve migration.

2. Language

2.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2.2. Use of Normative Terms

This document, which deals with existing issues/problems in standards-track documents, is in the informational category, and while the facts it reports may have normative implications, any such normative significance reflects the readers' preferences. For example, we may report that the existing definition of migration for NFSv4.1 does not properly describe how migrating state is to be merged with existing state for the destination server. While it is to be expected that client and server implementers will judge this to be a situation that it would be appropriate to resolve, the judgment as to how pressing this issue should be considered is a judgment for the reader, and eventually the nfsv4 working group to make.

We do explore possible ways in which such issues can be dealt with, with minimal negative effects, given that the working group has decided to address these issues, but the choice of exactly how to address these is best given effect in one or more standards-track documents and/or errata.

In the context of this informational document, these normative keywords will generally occur in the context of a quotation, most often direct but sometimes indirect. The context will make it clear whether the quotation is from:

- The base definition of the NFSv4.0 protocol [RFC7530].

- The document updating the handling of migration in NFSv4.0 [RFC7931].

- The current definition of the NFSv4.1 protocol [RFC5661].

An additional possibility is that these terms may appear in a proposed or possible text to serve as a replacement for a current protocol specification. Sometimes, a number of possible alternative texts may be listed and benefits and detriments of each examined in turn.

2.3. Terminology Used in this Document

In this document the phrase "client ID" always refers to the 64-bit shorthand identifier assigned by the server (a clientid4) and never to the structure which the client uses to identify itself to the server (called an nfs_client_id4 or client_owner in NFSv4.0 and NFSv4.1 respectively). The opaque identifier within those structures is referred to as a client id string".

Regarding trunking of network addresses, we use the following terminology:

- o Trunking detection refers to ways of deciding whether two specific network addresses are connected to the same NFSv4 server. The means available to make this determination depends on the protocol version, and, in some cases, on the client implementation.

- o Two network addresses connected to the same server are said to be server-trunkable.

- o Two network addresses connected to the same server such that those addresses can be used to support a single common session are referred to as session-trunkable. Note that two addresses may be server-trunkable without being session-trunkable.

- o Trunking discovery is a process by which a client using one network address can obtain other addresses that are trunkable (either server-trunkable or session-trunkable) with it.

Regarding terminology relating to attributes used in trunking discovery and other multi-server namespace features:

- o Location attributes include the fs_locations and fs_locations_info attributes.

- o Location entries are the individual file system locations in the location attributes.

- o Location elements are derived from location entries. If a location entry specifies an IP address there is only a single corresponding location element. Location entries that contain a host name, are resolved using DNS, and may result in one or more location elements. All location elements consist of a location address which is the IP address of an interface to a server and an fs name which is the location of the file system within the server's pseudo-fs. The fs name is empty if the server has no pseudo-fs and only a single exported file system at the root filehandle.

- o Two location elements are said to be server-trunkable if they specify the same fs name and the location addresses are such that the location addresses are server-trunkable.

- o Two location elements are said to be session-trunkable if they specify the same fs name and the location addresses are such that the location addresses are session-trunkable.

Each set of server-trunkable location elements defines the available access paths to a particular file system. When there are multiple such file systems, each of these, which contains the same data, is a replica of the others. Logically, such replication is symmetric, since the fs currently in use and an alternate fs are replicas of each other. Often, in other documents, the term "replica" is not applied to the fs currently in use, despite the fact that the replication relation is inherently symmetric.

3. Multi-version Issues and Their Resolution

3.1. Multi-Version Issues

The source of these issues rises from a lack of clarity regarding the meaning of and proper handling for location attributes that specify more than a single server address. Such situations can arise as a result of multiple entries in the same attribute or because a single entry has a server name which, when processed by DNS, is mapped to multiple server addresses.

Both [RFC7530] and [RFC5661] indicate that multiple addresses may be present and that these addresses may be different paths to the same server as well as different copies of the same data. However, the following issues have, for both protocols, interfered with the recognition of the existing location attributes as a way of providing a trunking discovery function:

- o There is no discussion of the use of these attributes when a file system is first accessed, giving the impression that they are only to be used as a way of overcoming access difficulties.

- o The treatment of migration (and in the case of NFSv4.1 of file system transitions in general) is written as if only a single server address will be accessed.

- o Although location attributes can contains the addresses of migration targets and of additional replicas as well, the issues that arise when both of these are specified are not clearly discussed.

In addition, there are factors that relates to specific protocol versions and documents:

- In NFSv4.0, as described solely by [RFC7530], trunking is treated as a problem to be avoided, making the whole matter moot.

- In NFSv4.0, as described by [RFC7530] together with [RFC7931], the situation is different. There is a means of trunking detection suggested in [RFC7931] but it is a suggestion only valid when the client chooses to use the uniform client id string model.

- For NFSv4.1, as described by [RFC5661], there is a standard method of trunking detection, which can be relied upon.

The issues that need to be resolved for both versions are:

- Provision of a trunking discovery facility to allow a client to find out about other addresses that may be used to access the current server.

- Better integration of migration with trunking changes, including situations in which the set of addresses to access the same server changes (without migration) and those in which there is a shift to a different server, but trunking of addresses on either the source or destination is involved

3.2. Resolution of Multi-Version Issues

Although the specifics of addressing these issues will be different for different versions, there are some common aspects discussed in the subsections below:

- The trunking discovery function, except for the additional location attribute fs_locations_info, is to be addressed in substantially the same way in both versions, as explained in Section 3.2.1.

- The interaction of trunking and migration is discussed in general terms in Section 3.2.2. However, the specifics of the NFSv4.1 client's response to NFS4ERR_MOVED are discussed in Sections 5.4.3, 5.4.4, and 5.4.5.

3.2.1. Providing Trunking Discovery

A client can discover a set network addresses to use to access a file system using an NFSv4 server in a number of ways:

- o If the client is accessing a server using its name, that name can be mapped to a set of IP addresses using DNS and if multiple addresses are available, those addresses can generally be used together to access the server.

- o A client connected to a server without knowledge of its name can obtain the value of a location attribute (e.g. fs_locations). Where an entry within that attribute specifies a server name, DNS can be used to obtain one or more network addresses corresponding to that name. In cases in which one of those is the address being used, the others that corresponding to that name can also be used to access the server.

- o A client can obtain the value of a location attribute (e.g. fs_locations) and use location entries that specify network addresses. When there is a means of trunking detection available (see below), all of addresses that are determined to correspond to the same server can be used to access the server.

Note that the last two of these are usable in situations in which NFS4ERR_MOVED was returned. Note that this does not necessarily mean that migration has occurred since there may be a shift in the set of network addresses to be used without changing to a different server. See Section 3.2.2 for further discussion.

Which of the above means of providing trunking information is appropriate to use in a given environment will depend on security considerations, the possible need for the server to direct different clients to different sets of addresses, and the availability of trunking detection facilities on the clients.

With regard to security, the possibility that requests to determine the set of network addresses corresponding to a given server might be interfered with or have their responses corrupted needs to be taken account of. As a result, when use of DNSSEC is not available, it might not be advisable to present server names in location attributes and present the network addresses directly, eliminating the need to use DNS to effect this translation. Fetching of location attributes should be done with integrity protection.

In some situations, the server will want to direct clients to use specific sets of network addresses, to effect load balancing, or to meet quality-of-service goals. In such environments, presentation of network addresses directly in the location attribute can help give the server the necessary control over the paths to be used when accessing particular file systems. When such techniques are used, servers typically present their own network addresses in the location

attribute while adding the names of other servers, such as those used to access replicas.

Trunking detection allows the client to determine whether two network addresses can be used to access the same server. The availability of trunking detection depends on the protocol version, and, in some case, on client implementation choices:

- o For NFSv4.0, a means by which it can be determined if two network addresses correspond to the same server is suggested in [RFC7931]. However is it is optional and only available to clients using the uniform client id string approach.

- o For NFSv4.1, the client can compare the server_owner returned in the response to EXCHANGE_ID to determine if two network addresses correspond to the same server.

As a result, direct presentation of network addresses in location entries may be problematic for NFSv4.0, since some clients might not have the trunking detection facilities that allow them to take advantage of this information. For further discussion of issues related to NFSv4.0, see Section 4.4.

3.2.2. Interaction of Trunking and Migration

When the set of network addresses designated by a location attribute changes, NFS4ERR_MOVED may or may not result, and in some of the cases in which it is returned migration will occur, while in others there a shift in the network addresses used to access a particular file system with no migration.

- o When the list of networks addresses is a superset of that previously in effect, there is no need for migration or any other sort of client adjustment. Nevertheless, the client is free to use an additional address if it provides another path to the same server. If, on the other hand, it does not do so, the client may treat it as it does a replica, to be used if the current server addresses become unavailable.

- o When the list of networks addresses is a subset of that previously in effect, immediate action is not needed if the address was not being used. The client should avoid using it in the future, whether the address is for a replica or a potential additional path to the server being used.

- o When an address being removed is one of a number of paths to the current server, the client can cease to use it but it can continue to use it until NFS4ERR_MOVED is received. This is not considered

a migration event, unless it is the last available path to the server that has become unusable.

When migration does occur, multiple addresses may be in use on the server previous to migration and multiple addresses may be available for use on the destination server.

With regard to the server in use, it may be that return of NFS4ERR_MOVED indicates that a particular network address is no longer to be used, without implying that migration of the file system to a different server is needed In light of this possibility, clients are best off not concluding that migration has occurred until concluding that all the network addresses known to be associated with the server are not usable.

It should be noted that the need to defer this determination is not absolute. If a client is not aware of all network addresses for any reason, if may conclude that migration has occurred when it has not and treat a switch to a different server address as if it were a migration event. This is generally harmless since the use of the same server via a new address will appear as a successful Transparent State Migration.

While significant harm will not arise from this misapprehension, it can give rise to disconcerting situations. For example, if a lock has been revoked during the address shift, it will appear to the client as if the lock has been lost during migration, normally calling for it to be recoverable via an fs-specific grace period associated with the migration event.

With regard to the destination server, it is desirable for the client to be aware of all the valid network addresses that can be used to access the destination server. However, there is no need for this to be done immediately. Implementations can process the additional location elements in parallel with normal use of the first valid location entry found to access the destination.

Because a location attribute may include entries relating to the current server, the migration destination and possible replicas to use, scanning for available network addresses could potentially be a long process. The following list of helpful practices, here presented as suggestions, could become RECOMMENDATIONs or REQUIREMENTs in future standards-track documents

- Servers are well advised to place location entries that represent addresses usable with the current server or a migration target before those associated with replicas.

- A client can cease scanning for trunkable location entries once it encounters one whose fs_name differs from the current fs name.

- A client can cease scanning for trunkable location entries once it encounters a location element whose address in not server-trunkable with the one it is using.

4. NFSv4.0 Issues

4.1. Core NFSv4.0 Migration Issues

Many of the problems seen with Transparent State Migration derived from the inability of NFSv4.0servers to determine whether two client IDs, issued on different servers, corresponded to the same client. This difficulty derived in turn from the common practice, recommended by [RFC7530], in which each client presented different client identification strings to different servers, rather than presenting the same identification string to all servers.

This practice, later referred to as the "non-uniform" client id string approach, derived from concern that, since NFSv4.0 provided no means to determine whether two IP addresses correspond to the server, a single client connected to both might be confused by the fact that state changes made via one IP address might unexpectedly affect the state maintained with respect to the second IP address, thought of as a separate server

To avoid this unexpected behavior, clients used the non-uniform client id string approach. By doing so, a client connected to two different servers (or to two IP addresses connected to the same server) appeared to be two different servers. Since the server is under the impression that two different clients are involved, state changes made on each distinct IP address cannot be reflected on another.

However, by doing things in that way, state migrated from server to server cannot be referred to the actual client which generated it, leading to confusion.

In addition to this core problem, the following issues with regard to Transparent State Migration needed to be addressed:

- Clarification regarding the ability to merge state from different leases even though their expiration times might not be precisely synchronized.

- Clarifying the treatment of client IDs since it is not always clear when clientid4 and when nfs_client_id4 was intended.

- Clarifying the logic of returning NFS4ERR_LEASE_MOVED.

- Clarifying the handling NFS4ERR_CLID_INUSE.

4.2. Resolution of Core Migration Protocol Difficulties in NFSv4.0

The client string identification issue was addressed in [RFC7931] as follows:

- Defining both the uniform and non-uniform client id string approaches as valid choices but indicating that the latter posed difficulties for Transparent Stare Migration.

- Providing a way that clients using the uniform approach could use to determine whether two IP addresses are connected to the same server.

- Allowing clients using the uniform approach to avoid negative consequences due to otherwise unexpected behavior since behavior that is a consequence of known trunking relationships is not unexpected.

- As a result, servers migrating state can be aware of the fact that the same client is associated with two different items of state even when that state was originally created on two different servers.

Since all of the other issues noted in Section 4.1 were also addressed by [RFC7931], publication of that document updating [RFC7530] addressed all issues with Transparent State Migration in NFSv4.0 known at that time.

4.3. Additional NFSv4.0 Issues

In light of the fact that a large set of migration-specific issues were addressed by the publication of [RFC7931], the remaining issues derive from those mentioned in Section 3.1. These include:

- Introducing facilities for trunking discovery.

- Clarifying the relationship between migration and trunking.

4.4. Resolution of Additional NFSv4.0 Issues

One possible approach to addressing these issues would entail publication of an additional standards-track document updating [RFC7530].

Fortunately, it appears that all of the material to be updated appears in Section 8 of that document, whether it concerns the provision of trunking discovery or the interaction of trunking and migration. It also appears that none of the material to be updated is in sections updated by [RFC7931].

A review of the existing Section 8 of [RFC7530], shows the following sections as requiring significant attention:

o The existing Section 8.1 requires a considerable expansion to explain the various uses of the fs_locations and the possible interactions among them.

o The existing Section 8.4 may require substantial re-organization to reflect the facts that fs_locations has multiple functions and may be referenced on multiple occasions.

o The existing Section 8.5 follows the previous approach for NFSv4.0 in assuming that trunking simply cannot and should not happen. For example, the last paragraph says:

> If a single location entry designates multiple server IP addresses, the client should choose a single one to use. When two server addresses are designated by a single location entry and they correspond to different servers, this normally indicates some sort of misconfiguration, and so the client should avoid using such location entries when alternatives are available. When they are not, clients should pick one of the IP addresses and use it, without using others that are not directed to the same server.

As written, this seems to foreclose any use of trunking in connection with migration. In retrospect, it appears that this section should have been revised as part of [RFC7931], but since that was not done then, the issue needs to be addressed now.

Overall, it appears that, in addition to the revision of Section 8.1, Sections 8.4 and 8.5 need to be reorganized. One likely approach is to divide the material into three main sections based on the circumstances in which the attribute is accessed:

A section dealing with initial use and how it is trunking discovery and preparation for replication.

A section dealing with subsequent attribute change and how it affects trunking discovery and preparation for replication.

A section dealing with how the attribute is used to deal with communication problems including receiving NFS4ERR_MOVED but also including other communication difficulties driving selection of a new replica.

5. Issues for NFSv4.1

5.1. Issues to Address for NFSv4.1

Because NFSv4.1 embraces the uniform client-string approach, as advised by section 2.4 of [RFC5661], addressing migration issues is simpler, in that a shift in client id string models is not required. Instead, NFSv4 returns information in the EXCHANGE_ID response to enable trunking relationships to be determined by the client.

Despite this simplification, there are substantial issues that need to be dealt with:

- o The other necessary part of addressing migration issues, providing for the server's merger of leases that relate to the same client, is not currently addressed by [RFC5661] and changes need to be made to make it clear that state needs to be appropriately merged as part of migration, to avoid multiple client IDs between a client-server pair.

- o The current discussion (in [RFC5661]), of the possibility of server_owner changes is incomplete and confusing.

- o As with NFSV4.0, the interaction of trunking with migration and other aspects of multi-server namespace needs to be clarified.

Addressing migration in NFSv41 will also require adaptation of the approaches used in [RFC7931] to the NFSv4.1 environment including:

- o The use of EXCHANGE_ID needs to be accommodated including issues associated with the expected confirmation status of client IDs transferred by Transparent State Migration.

- o The use of sessions needs to be addressed including discussion of the proper use of the status bits returned by the SEQUENCE operation.

In addition, there are a number of new features within NFSv4.1 whose relationship with migration needs to be clarified. Some examples:

- o There needs to be some clarification of how migration, and particularly Transparent State Migration, should interact with pNFS layouts.

- There are a number of issues related to the migration of sessions that need to be addressed.

Discussion of how to resolve these issues will appear in the sections below.

5.1.1. Addressing State Merger in NFSv4.1

The existing treatment of state transfer in [RFC5661], has similar problems to that in [RFC7530] in that it assumes that the state for multiple filesystems formerly on different servers will not be merged so that it appears under a single common client ID. We've already seen the reasons that this is a problem with regard to NFSv4.0.

Although we don't have the problems stemming from the non-uniform client-string approach, there are a number of complexities in the existing treatment of state management in the section entitled "Lock State and File System Transitions" in [RFC5661] that make this non-trivial to address:

- Migration is currently treated together with other sorts of filesystem transitions including transitioning between replicas without any NFS4ERR_MOVED errors.

- There is separate handling and discussion of the cases of matching and non-matching server scopes.

- In the case of matching server scopes, the text calls for an unrealistic degree of transparency, suggesting that the source and destination servers need to cooperate in stateid and client ID assignment.

- In the case of non-matching server scopes, the text does not mention the possibility of the transparent migration of state at all, resulting in a functional regression from NFSV4.0

5.1.2. Addressing pNFS Relationship with Migration

This is made difficult because, within the pNFS framework, migration might mean any of several things:

- Transfer of the MDS, leaving DS's as they are.

  This would be minimally disruptive to those using layouts but would require the pNFS control protocol being used to support the DS being directed to a new MDS.

- Transfer of a DS, leaving everything else in place.

Such a transfer can be handled without using migration at all. The server can recall/revoke layouts, and issue new ones, as appropriate.

- o Transfer of the filesystem to a new filesystem with both MDS and DS's moving.

  In such a transfer, an entirely different set of DS's will be at the target location. There may even be no pNFS support on the destination filesystem at all.

Migration needs to support both the first and last of these models.

5.1.3. Addressing Server_owner Changes in NFSv4.1

Section 2.10.5 of [RFC5661] states the following.

> The client should be prepared for the possibility that eir_server_owner values may be different on subsequent EXCHANGE_ID requests made to the same network address, as a result of various sorts of reconfiguration events. When this happens and the changes result in the invalidation of previously valid forms of trunking, the client should cease to use those forms, either by dropping connections or by adding sessions. For a discussion of lock reclaim as it relates to such reconfiguration events, see Section 8.4.2.1.

While this paragraph is literally true in that such reconfiguration events can happen and clients have to deal with them, it is confusing in that it can be read as suggesting that clients have to deal with them without disruption, which in general is impossible.

A clearer alternative would be:

> It is always possible that, as a result of various sorts of reconfiguration events, eir_server_scope and eir_server_owner values may be different on subsequent EXCHANGE_ID requests made to the same network address.
>
> In most cases such reconfiguration events will be disruptive and indicate that an IP address formerly connected to one server is now connected to an entirely different one.
>
> Some guidelines on client handling of such situations follow:
>
> - o When eir_server_scope changes, the client has no assurance that any id's it obtained previously (e.g. file handles) can be validly used on the new server, and, even if the new server

accepts them, there is no assurance that this is not due to accident. Thus it is best to treat all such state as lost/stale although a client may assume that the probability of inadvertent acceptance is low and treat this situation as within the next case.

- o When eir_server_scope remains the same and eir_server_owner.so_major_id changes, the client can use filehandles it has and attempt reclaims. It may find that these are now stale but if NFS4ERR_STALE is not received, he can proceed to reclaim his opens.

- o When eir_server_scope and eir_server_owner.so_major_id remain the same, the client has to use the now-current values of eir_server-owner.so_minor_id in deciding on appropriate forms of trunking.

5.1.4. Addressing Confirmation Status of Migrated Client IDs in NFSv4.1

When a client ID is transferred between systems as a part of migration, it has never been clear whether it should be considered confirmed or unconfirmed on the target server. In the case in which an associated session is transferred together with the client ID, it is clear that the transferred client ID needs to be considered confirmed, as the existence of an associated session is incompatible with an unconfirmed client ID.

The case in which a client ID is transferred without an associated session is less clear-cut, particularly since the treatment of EXCHANGE_ID in [RFC5661] assumes that CREATE_SESSION is the only means by which a client id may be confirmed. While this assumption is valid in the absence of Transparent State Migration, implementation of migration means that if this assumption is maintained, it is not clear how migrated client ID s can be a accommodated. If this assumption were maintained, we would have to choose between the following two alternatives, regarding whether the client ID to be reported as confirmed when EXCHANGE_ID is used to register an already-known client_owner with the server.

- o Report the client ID unconfirmed, because of the lack of an associated session. This makes it simpler for the client to determine whether there is an associated session transferred at the same time. However, it is inconsistent with the fact there are stateids which have been transferred with the client ID.

- o Report the client ID as confirmed, because it was confirmed on the source server and the transfer is not considered to have affected that. Given the current description of EXCHANGE_ID in [RFC5661],

some modification in the treatment of client id confirmation is called for. In particular, provision would have to be made to enable the client id slot sequence id to be used by the client to be determined.

Although the first approach makes it simpler for the client to determine whether there is an associated session transferred at the same time, it makes it more difficult to determine whether Transparent State Migration has occurred. Section 5.1.6.

In any case, adjustments will be required to deal with the fact that [RFC5661] currently assumes that a client id can only be confirmed by issuing a CREATE_SESSION. In order to properly deal with the status of migrated client ids, we have to distinguish among:

o The confirmation status as reported by EXCHANGE_ID.

o Whether the client id is considered confirmed as that term is used in the many other cases in which the confirmation status of a client ID affects how requests are handled.

o How the client is to determine the initial sequence id to be used when doing operations such as CREATE_SESSION.

In [RFC5661] as it currently stands all of these are tied together and it is not obvious how migrated client IDs could be accommodated in this structure, and what changes are necessary to make this possible. For more discussion of this issue, see Section 5.2.1.

5.1.5. Addressing Changes in Trunking Configuration

When the client us capable of finding out a set of network addresses to use in accessing a server, it is always possible for that set to change.

This sometimes requires that a network address previously used to access a server becomes invalid for that purpose. This requires a way of notifying the client and a way for the client to adapt to this change by using a new set of network addresses to access the server. his will involve recovery much like hat for migration although the same server and file system is used throughout.

5.1.6. Addressing Session Migration in NFSv4.1

Some issues that need to be addressed regard the migration of sessions, in addition to client IDs and stateids

- It needs to be made clearer how the client can deal with the possibility that sessions might or might not be transferred as part of Transparent State Migration.

- Rules need to be clarified regarding possible transfer of sessions when either the source session is being used to access other file systems on source server or there is already a session connecting the client to the destination server.

- There needs to be more detail regarding how the protocol avoids situations in which the same session is subject to concurrent changes on two different servers at the same time.

5.2. Possible Resolutions for NFSv4.1 Protocol Issues

The subsections below explore some ways of dealing with clarifying the protocol to address issues discussed in Section 5.1

5.2.1. Client ID Confirmation Issues

As mentioned previously [RFC5661], makes no provision for client IDs that are confirmed other than through the use of CREATE_SESSION. For example Section 18.35 of [RFC5661] states:

> The client uses the EXCHANGE_ID operation to register a particular client owner with the server. The client ID returned from this operation will be necessary for requests that create state on the server and will serve as a parent object to sessions created by the client. In order to confirm the client ID it must first be used, along with the returned eir_sequenceid, as arguments to CREATE_SESSION. If the flag EXCHGID4_FLAG_CONFIRMED_R is set in the result, eir_flags, then eir_sequenceid MUST be ignored, as it has no relevancy.

In deciding how to address the status of migrated client IDs in the case of Transparent State Migration, we should avoid giving undue weight to the last sentence of the above simply because it is stated in the form of a normative requirement. We should instead focus on the reasons such terms (i.e. those defined by [RFC2119]) are to be used, to state interoperability constraints. In this case, the "MUST" applies to a conclusion based on the premise that a CREATE_SESSION must have been done to assure that the client ID is reliably known to the server.

In that light, let us consider a possible replacement, that treats confirmation by means of CREATE_SESSION as one of a number of possible means and avoids some the undesirable consequences of

adherence to the current approach, originally conceived without taking state migration into account.

The client uses the EXCHANGE_ID operation to register a particular client_owner with the server. However, when the client_owner has been already been registered by other means (e.g. Transparent State Migration), the client may still use EXCHANGE_ID to obtain the client ID assigned previously.

The client ID returned from this operation will be associated with the connection on which the EXHANGE_ID is received and will serve as a parent object for sessions created by the client on this connection or to which the connection is bound. As a result of using those sessions to make requests involving the creation of state, that state will become associated with the client ID returned.

In situations in which the registration of the client_owner has not occurred previously, the client ID must first be used, along with the returned eir_sequenceid, in creating an associated session using CREATE_SESSION.

If the flag EXCHGID4_FLAG_CONFIRMED_R is set in the result, eir_flags, then it is an indication that the registration of the client_owner has already occurred and that a further CREATE_SESSION is not needed to confirm it. Of course, subsequent CREATE_SESSION operations may be needed for other reasons.

The value eir_seqenceid is used to establish an initial sequence value associate with the client ID returned. In cases in which a CREATE_SESSION has already been done, there is no need for this value, since sequencing of such request has already been established and the client has no need for this value and will ignore it

5.2.2. Dealing with Multiple Location Entries

The possibility that more than one server address may be present in location attributes requires further clarification. This is particularly the case, given the potential role of trunking for NFSv4.1, whose connection to migration needs to be clarified.

The description of the location attributes in [RFC5661], while it indicates that multiple address entries in these attributes may be used to indicate alternate paths to the file system, does so mainly in the context of replication and does so without mentioning trunking. The discussion of migration does not discuss the

possibility of multiple location entries or trunking, which we will explore here.

We will cover cases in which multiple addresses appear directly in the attributes as well as those in which the multiple addresses result because a single location entry is expanded into multiple location elements using addresses provided by DNS.

When the set of valid location elements by which a file system may be accessed changes, migration need not be involved. Some cases to consider:

- When the set of location elements expands, migration is not involved. In the case in which the additional elements are not trunkable with ones previously being used, the new elements serve as additional access locations, available in case of the failure of server addresses being used. When additional elements are trunkable with those currently being used the client may use the additional addresses just as they might have if they had been available when use of the file system began.

  There is no current mechanism by which the client can be notified of a change in the set of available location for an fs. Given the client has at least one IP address available to access the filesystem in question, periodic polling is an adequate mechanism for the client to find additional server addresses to use to access the file system.

- When the set of location elements contracts but none of the elements no longer usable were in fact being used by the client, then no migration is involved and no change in network addresses is needed. Only if the client were to start using one of the unavailable elements would the client be notified (via NFS4ERR_MOVED) of the need to not use those elements and to use others provided by a location attribute.

When a specific server address being used becomes unavailable to service a particular file system, NFS4ERR_MOVED will be returned, and the client will respond based on the available locations. Whether continuity of locking state will be available depends on a number of factors:

- If there are still elements in use trunkable with the element that has become unavailable, there will still be a continuity of locking state, even though Transparent State Migration per se has not occurred. If the in-use addresses are session-trunkable with the address becoming unavailable, only one connection is lost and all existing sessions will remain available. If, on the other

hand, the in-use addresses are only clientid-trunkable with the address becoming unavailable, a session can be lost. However, that session can be made available on those other nodes, just as they it would have been if Transparent State Migration were in effect, even though no migration has occurred.

- Otherwise, if there are available addresses trunkable with the one that has become unavailable, the client has access to existing locking state once it establishes a connection with the new addresses, using a new or existing session depending on the type of trunking in effect. This is also similar to the case in which Transparent State Migration has occurred, even though there is no migration, with the state remaining on the existing server.

  Note that this case, as well as the previous one, can be expected in the case in which the server seeks to direct traffic with regard to particular file systems to choose addresses, in the interest of load balancing, to adjust to hardware availability constraints, or for other reasons.

- In other cases, migration has occurred and the client can determine whether Transparent State Migration occurred and whether any locking state was lost during the transfer.

Whether migration has occurred or not, the client can use the procedure described in Section 5.4.3 to recover access to existing locking state and, in some cases, sessions.

One should note the following differences between migration with Transparent State Migration and the similar cases in which there is a continuity of locking state with no change in the server.

- When locks are lost (as indicated when using them or via the SEQ4_STATUS flags) and migration has not been done, they are not to be reclaimed, except when SEQ4_STATUS_RESTART_RECLAIM_NEEDED is set. Instead such losses are treated as lock revocations and acknowledged using FREE_STATEID.

- When migration has not been done, there is no need for a RECLAIM_COMPLETE (with rca_one_fs set to true).

5.2.3. Migration and pNFS

When pNFS is involved, the protocol is capable of supporting:

- Migration of the MDS, leaving DS's in place.

- Migration of the file system as a whole, including the MDS and associated DS's.

- Replacement of one DS by another.

- Migration of a pNFS file system to one in which pNFS is not used.

- Migration of a file system not using pNFS to one in which layouts are available.

Migration of the MDS function is directly supported by Transparent State Migration. Layout state will normally be transparently transferred, just as other state is. As a result, Transparent State Migration provides a framework in which, given appropriate inter-MDS data transfer, one MDS can be substituted for another.

Migration of the file system function as a whole can be accomplished by recalling all layouts as part of the initial phase of the migration process. As a result, IO will be done through the MDS during the migration process, and new layouts can be granted once the client is interacting with the new MDS. An MDS can also effect this sort of transition by revoking all layouts as part of Transparent State Migration, as long as the client is notified about the loss of state.

In order to allow migration to a file system on which pNFS is not supported, clients need to be prepared for a situation in which layouts are not available or supported on the destination file system and so direct IO requests to the destination server, rather than depending on layouts being available.

Replacement of one DS by another is not addressed by migration as such but can be effected by an MDS recalling layouts for the DS to be replaced and issuing new ones to be served by the successor DS.

Migration may transfer a file system from a server which does not support pNFS to one which does. In order to properly adapt to this situation, clients which support pNFS, but function adequately in its absence, should check for pNFS support when a file system is migrated and be prepared to use pNFS when support is available.

5.3. Defining Server Responsibilities for NFSv4.1

The subsections below discuss the responsibilities of source and destination servers in effecting the necessary transfer of information to support Transparent State Migration.

5.3.1. Server Responsibilities in Effecting Transparent State Migration

The basic responsibility of the source server in effecting Transparent State Migration is to make available to the destination server a description of each piece of locking state associated with the file system being migrated. In addition to client id string and verifier, the source server needs to provide, for each stateid:

- o The stateid including the current sequence value.
- o The associated client ID.
- o The handle of the associated file.
- o The type of the lock, such as open, byte-range lock, delegation, layout.
- o For locks such as opens and byte-range locks, there will be information about the owner(s) of the lock.
- o For recallable/revocable lock types, the current recall status needs to be included.
- o For each lock type there will by type-specific information, such as share and deny modes for opens and type and byte ranges for byte-range locks and layouts.

A further server responsibility concerns locks that are revoked or otherwise lost during the process of file system migration. Because locks that appear to be lost during the process of migration will be reclaimed by the client, the servers have to take steps to ensure that locks revoked soon before or soon after migration are not inadvertently allowed to be reclaimed in situations in which the continuity of lock possession cannot be assured.

- o For locks lost on the source but whose loss has not yet been acknowledged by the client (by using FREE_STATEID), the destination must be aware of this loss so that it can deny a request to reclaim them.
- o For locks lost on the destination after the state transfer but before the client's RECLAIM_COMPLTE is done, the destination server should note these and not allow them to be reclaimed.

An additional responsibility of the cooperating servers concerns situations in which a stateid cannot be transferred transparently because it conflicts with an existing stateid held by the client and

associated with a different file system. In this case there are two valid choices:

o Treat the transfer, as in NFSv4.0, as one without Transparent State Migration. In this case, conflicting locks cannot be granted until the client does a RECLAIM_COMPLETE, after reclaiming the locks it had, with the exception of reclaims denied because they were attempts to reclaim locks that had been lost.

o Implement Transparent State Migration, except for the lock with the conflicting stateid. In this case, the client will be aware of a lost lock (through the SEQ4_STATUS flags) and be allowed to reclaim it.

5.3.2. Synchronizing Session Transfer

When transferring state between the source and destination, the issues discussed in Section 7.2 of [RFC7931] must still be attended to. In this case, the use of NFS4ERR_DELAY is still necessary in NFSv4.1, as it was in NFSv4.0, to prevent locking state changing while it is being transferred.

There are a number of important differences in the NFS4.1 context:

o The absence of RELEASE_LOCKOWNER means that the one case in which an operation could not be deferred by use of NFS4ERR_DELAY no longer exists.

o Sequencing of operations is no longer done using owner-based operation sequences numbers. Instead, sequencing is session-based

As a result, when sessions are not transferred, the techniques discussed in [RFC7931] are adequate and will not be further discussed.

When sessions are transferred, there are a number of issues that pose challenges since,

o A single session may be used to access multiple file systems, not all of which are being transferred.

o Requests made on a session may, even if rejected, affect the state of the session by advancing the sequence number associated with the slot used.

As a result, when the filesystem state might otherwise be considered unmodifiable, the client might have any number of in-flight requests,

each of which is capable of changing session state, which may be of a number of types:

1. Those requests that were processed on the migrating file system, before migration began.

2. Those requests which got the error NFS4ERR_DELAY because the file system being accessed was in the process of being migrated.

3. Those requests which got the error NFS4ERR_MOVED because the file system being accessed had been migrated.

4. Those requests that accessed the migrating file system, in order to obtain location or status information.

5. Those requests that did not reference the migrating file system.

It should be noted that the history of any particular slot is likely to include a number of these request classes. In the case in which a session which is migrated is used by filesystems other than the one migrated, requests of class 5 may be common and be the last request processed, for many slots.

Since session state can change even after the locking state has been fixed as part of the migration process, the session state known to the client could be different from that on the destination server, which necessarily reflects the session state on the source server, at an earlier time. In deciding how to deal with this situation, it is helpful to distinguish between two sorts of behavioral consequences of the choice of initial sequence ID values.

- The error NFS4ERR_SEQ_MISORDERED is returned when the sequence ID in a request is neither equal to the last one seen for the current slot nor the next greater one.

  In view of the difficulty of arriving at a mutually acceptable value for the correct last sequence value at the point of migration, it may be necessary for the server to show some degree of forbearance, when the sequence ID is one that would be considered unacceptable if session migration were not involved.

- Returning the cached reply for a previously executed request when the sequence ID in the request matches the last value recorded for the slot.

  In the cases in which an error is returned and there is no possibility of any non-idempotent operation having been executed, it may not be necessary to adhere to this as strictly as might be

proper if session migration were not involved. For example, the fact that the error NFS4ERR_DELAY was returned may not assist the client in any material way, while the fact that NFS4ERR_MOVED was returned by the source server may not be relevant when the request was reissued, directed to the destination server.

One part of adapting to these sorts of issues would restrict enforcement of normal slot sequence enforcement semantics until the client itself, by issuing a request using a particular slot on the destination server, established the new starting sequence for that slot on the migrated session.

An important issue is that the specification needs to take note of all potential COMPOUNDs, even if they might be unlikely in practice. For example, a COMPOUND is allowed to access multiple file systems and might perform non-idempotent operations in some of them before accessing a file system being migrated. Also, a COMPOUND may return considerable data in the response, before being rejected with NFS4ERR_DELAY or NFS4ERR_MOVED, and may in addition be marked as sa_cachethis.

Some possibilities that need to be considered to address the issues:

o Do not enforce any sequencing semantics for a particular slot until the client has established the starting sequence for that slot on the destination server.

o For each slot, do not return a cached reply returning NFS4ERR_DELAY or NFS4ERR_MOVED until the client has established the starting sequence for that slot on the destination server.

o Until the client has established the starting sequence for a particular slot on the destination server, do not report NFS4ERR_SEQ_MISORDERED or return a cached reply returning NFS4ERR_DELAY or NFS4ERR_MOVED, where the reply consists solely of a series of operations where the response is NFS4_OK until the final error.

5.4. Defining Client Responsibilities for NFSv4.1

The subsections below discuss the responsibilities of the client in dealing with transition to a new server (migration) and to use of new network addresses in accessing existing servers.

5.4.1. Client Recovery from Migration Events

When a file system is migrated, there a number of migration-related status indications with which clients need to deal:

- If an attempt is made to use or return a filehandle within a file system that has been migrated away from the server on which it was previously available, the error NFS4ERR_MOVED is returned.

  This condition continues on subsequent attempts to access the file system in question. The only way the client can avoid the error is to cease accessing the filesystem in question at its old server location and access it instead on the server to which it has been migrated.

- Whenever a SEQUENCE operation is sent by a client to a server which generated state held on that client which is associated with a file system that has been migrated away from the server on which it was previously available, the status bit SEQ4_STATUS_LEASE_MOVED is set in the response.

  This condition continues until the client acknowledges the notification by fetching a location attribute for the migrated file system. When there are multiple migrated file systems, a location attribute for each such migrated file system needs to be fetched, in order to clear the condition. Even after the condition is cleared, the client needs to respond by using the location information to access the destination server to ensure that leases are not needlessly expired.

Unlike the case of NFSv4.0 in which the corresponding conditions are both errors, in NFSv4.1 the client can, and often will, receive both indications on the same request. As a result, implementations need to address the question of how to co-ordinate the necessary recovery actions when both indications arrive simultaneously. It should be noted that when the server decides whether SEQ4_STATUS_LEASE_MOVED is to be set, it has no way of knowing which file system will be referenced or whether NFS4ERR_MOVED will be returned.

While it is true that, when only a single migrated file system is involved, a single set of actions will clear both indications, the possibility of multiple migrated file systems calls for an approach in which there are separate recovery actions for each indication. In general, the response to neither indication can be subsumed within the other since:

- If the client were to respond only to the MOVED indication, there would be no effective client response to a situation in which a

file system was not being actively accessed at the time migration occurred. As a result, leases on the destination server might be needlessly expired.

- If the client were to respond only to the LEASE_MOVED indication, recovery for migrated file systems in active use could be deferred in order to accomplish recovery for others not being actively accessed. The consequences of this choice can pose particular problems when there are a large number of file systems supported by a particular server, or when it happens that some servers, after receiving migrated file systems have periods of unavailability, such as occur as a result of server reboot. This can result in recovery for actively accessed migrated file systems being unnecessarily delayed for long periods of time.

Similar considerations apply to other arrangements in which one of the indications, while not ignored per se, is subsumed within a single recovery process focused on recovery for the other indication.

Although clients are free to decide on their own approaches to recovery, we will explore below an approach with the following characteristics:

- All instances of the MOVED indication, whether they involve migration or not, should be dealt with promptly, either by doing the necessary recovery directly, providing that it be done asynchronously, or ensuring that it is already under way.

- All instances of the LEASE_MOVED indication should be dealt with asynchronously, in a migration discovery thread whose job is to clear that indication by fetching the appropriate location attribute. Because this thread will only be fetching a location attribute and the fs_status attribute for the file systems referenced by the client, it cannot receive MOVED indications. Some useful guidance regarding possible implementation of a migration discovery thread can be found in Section 5.4.2.

- When a migration discovery thread happens upon a migrated file system (i.e. not present and not a referral), the thread is likely to have cleared one (out of an unknown number) of file systems whose migration needs to be responded to. The discovery thread needs to schedule the appropriate migration recovery (as described in Section 5.4.3). This is necessary to ensure that migrated file systems will be referenced on the destination server in order to avoid unnecessary lease expiration.

  For many of the migrated file systems discovered in this way, the client has not received any MOVED indication. In such cases,

lease recovery needs to be scheduled but it should not interfere with continuation of the migration discovery function.

- When a migration discovery thread receives a LEASE_MOVED indication, it takes no special action but continues its normal operation. On the other hand, if a LEASE_MOVED indication is not received, it indicates that the thread has completed its work successfully.

5.4.2. The Migration Discovery Process

As noted above, LEASE_MOVED indications are best dealt with in a migration discovery thread. Because of this structure,

- No action needs to be taken for such indications received by the migration discovery threads, since continuation of that thread's work will address the issue.

- For such indications received in other contexts, the generally appropriate response is to initiate or otherwise provide for the execution of a migration discovery thread for file systems associated with the server IP address returning the indication.

- In all cases in which the appropriate migration discovery thread is running, nothing further needs to be done to respond to LEASE_MOVED indications.

This leaves a potential difficulty in situations in which the migration discovery thread is near to completion but is still operating. One should not ignore a LEASE_MOVED indication if the discovery thread is not able to respond to migrated file system without additional aid. A further difficulty in addressing such situation is that a LEASE_MOVED indication may reflect the server's state at the time the SEQUENCE operation was processed, which may be different from that in effect at the time the response is received.

A useful approach to this issue involves the use of separate externally-visible discovery thread states representing non-operation, normal operation, and completion/verification of migration discovery processing.

Within that framework, discovery thread processing would proceed as follows.

- While in the normal-operation state, the thread would fetch, for successive file systems known to the client on the server being worked on, a location attribute plus the fs_status attribute.

- o If the fs_status attribute indicates that the file system is a migrated one (i.e. fss_absent is true and fss_type != STATUS4_REFERRAL) and thus that it is likely that the fetch of the location attribute has cleared one the file systems contributing to the LEASE_MOVED indication.

- o In cases in which that happened, the thread cannot know whether the LEASE_MOVED indication has been cleared and so it enters the completion/verification state and proceeds to issue a COMPOUND to see if the LEASE_MOVED indication has been cleared.

- o When the discovery thread is in the completion/verification state, if others get a LEASE_MOVED indication they note this fact and it is used when the request completes, as described below.

When the request used in the completion/verification state completes:

- o If a LEASE_MOVED indication is returned, the discovery thread resumes its normal work.

- o Otherwise, if there is any record that other requests saw a LEASE_MOVED indication, that record is cleared and the verification request retried. The discovery thread remains in completion/verification state.

- o If there has been no LEASE_MOVED indication, the work of the discovery thread is considered completed and it enters the non-operating state.

5.4.3. Overview of Client Response to NFS4ERR_MOVED

This section outlines a way in which a client that receives NFS4ERR_MOVED can respond by using a new server or network address if one is available. As part of that process, it will determine:

- o Whether the NFS4ERR_MOVED indicates migration has occurred, or whether it indicates another sort of file system transition as discussed in Section 5.2.2.

- o In the case of migration, whether Transparent State Migration has occurred.

- o Whether any state has been lost during the process of Transparent State Migration.

- o Whether sessions have been transferred as part of Transparent State Migration.

During the first phase of this process, the client proceeds to examine location entries to find the initial network address it will use to continue access to the file system or its replacement. For each location entry that the client examines, the process consists of five steps:

1. Performing an EXCHANGE_ID is directed at the location address. This operation is used to register the client-owner with the server, to obtain a client ID to be use subsequently to communicate with it, to obtain tat client ID's confirmation status and, to determine server_owner and scope for the purpose of determining if the entry is trunkable with that previously being used to access the file system (i.e. that it represents another path to the same file system and can share locking state with it).

2. Making an initial determination of whether migration has occurred. The initial determination will be based on whether the EXCHANGE_ID results indicate that the current location element is server-trunkable with that used to access the file system when access was terminated by receiving NFS4ERR_MOVED. If it is, then migration has not occurred and the transition is dealt with, at least initially, as one involving continued access to the same file system on the same server through a new network address.

3. Obtaining access to existing session state or creating new sessions. How this is done depends on the initial determination of whether migration has occurred and can be done as described in Section 5.4.4 in the case of migration or as described in Section 5.4.5 in the case of a network address transfer without migration.

4. Verification of the trunking relationship assumed in step 2 as discussed in Section 2.10.5.1 of [RFC5661]. Although this step will generally confirm the initial determination, it is possible for verification to fail with the result that an initial determination that a network address shift (without migration) has occurred may be invalidated and migration determined to have occurred. There is no need to redo step 3 above, since it will be possible to continue use of the session established already.

5. Obtaining access to existing locking state and/or reobtaining it. How this is done depends on the final determination of whether migration has occurred and can be done as described in Section 5.4.4 in the case of migration or as described in Section 5.4.5 in the case of a network address transfer without migration.

Once the initial address has been determined, clients are free to apply an abbreviated process to find additional addresses trunkable with it (clients may seek session-trunkable or server trunkable addresses depending on whether they support clientid trunking). During this later phase of the process, further location entries are examined using the abbreviated procedure specified below:

1. Before the EXCHANGE_ID, the fs_name field is examined and if it does not match that currently being used, the entry is ignored. otherwise, one proceeds as specified by step 1 above,.

2. In the case that the network address is session-trunkable with one used previously a BIND_CONN_TO_SESSION is used to access that session using new network address. Otherwise, or if the bind operation fails, a CREATE_SESSION is done.

3. The verification procedure referred to in step 4 above is used. However, if it fails, the entry is ignored and the next available entry is used.

5.4.4. Obtaining Access to Sessions and State after Migration

In the event that migration has occurred, the determination of whether Transparent State Migration has occurred is driven by the client ID returned by the EXCHANGE_ID and the reported confirmation status.

- If the client ID is an unconfirmed client ID not previously known to the client, then Transparent State Migration has not occurred.

- If the client ID is a confirmed client ID previously known to the client, then any transferred state would have been merged with an existing client ID representing the client to the destination server. In this state merger case, Transparent State Migration might or might not have occurred and a determination as to whether it has occurred is deferred until sessions are established and we are ready to begin state recovery.

- If the client ID is a confirmed client ID not previously known to the client, then the client can conclude that the client ID was transferred as part of Transparent State Migration. In this transferred client ID case, Transparent State Migration has occurred although some state may have been lost.

Once the client ID has been obtained, it is necessary to obtain access to sessions to continue communication with the new server. In any of the cases in which Transparent State Migration has occurred, it is possible that a session was transferred as well. To deal with

that possibility, clients can, after doing the EXCHANGE_ID, issue a BIND_CONN_TO_SESSION to connect the transferred session to a connection to the new server. If that fails, it is an indication that the session was not transferred and that a new session needs to be created to take its place.

In some situations, it is possible for a BIND_CONN_TO_SESSION to succeed without session migration having occurred. If state merger has taken place then the associated client ID may have already had a set of existing sessions, with it being possible that the sessionid of a given session is the same as one that might have been migrated. In that event, a BIND_CONN_TO_SESSION might succeed, even though there could have been no migration of the session with that sessionid.

Once the client has determined the initial migration status, and determined that there was a shift to a new server, it needs to re-establish its lock state, if possible. To enable this to happen without loss of the guarantees normally provided by locking, the destination server needs to implement a per-fs grace period in all cases in which lock state was lost, including those in which Transparent State Migration was not implemented.

Clients need to be deal with the following cases:

o In the state merger case, it is possible that the server has not attempted Transparent State Migration, in which case state may have been lost without it being reflected in the SEQ4_STATUS bits. To determine whether this has happened, the client can use TEST_STATEID to check whether the stateids created on the source server are still accessible on the destination server. Once a single stateid is found to have been successfully transferred, the client can conclude that Transparent State Migration was begun and any failure to transport all of the stateids will be reflected in the SEQ4_STATUS bits. Otherwise. Transparent State Migration has not occurred.

o In a case in which Transparent State Migration has not occurred, the client can use the per-fs grace period provided by the destination server to reclaim locks that were held on the source server.

o In a case in which Transparent State Migration has occurred, and no lock state was lost (as shown by SEQ4_STATUS flags), no lock reclaim is necessary.

o In a case in which Transparent State Migration has occurred, and some lock state was lost (as shown by SEQ4_STATUS flags), existing

stateids need to be checked for validity using TEST_STATEID, and reclaim used to re-establish any that were not transferred.

For all of the cases above, RECLAIM_COMPLETE with an rca_one_fs value of true should be done before normal use of the file system including obtaining new locks for the file system. This applies even if no locks were lost and needed to be reclaimed.

5.4.5. Obtaining Access to Sessions and State after Network Address Transfer

The case in which there is a transfer to a new network address without migration is similar to that described in Section 5.4.4 in that there is a need to obtain access to needed sessions and locking state. However, the details are simpler and will vary depending on the type of trunking between the address receiving NFS4ERR_MOVED and that to which the transfer is to be made

To make a session available for use, a BIND_CONN_TO_SESSION should be used to obtain access to the session previously in use. Only if this fails, should a CREATE_SESSION be done. While this procedure mirrors that in Section 5.4.4, there is an important difference in that preservation of the session is not purely optional but depends on the type of trunking.

Access to appropriate locking state should need no actions beyond access to the session. However. the SEQ4_STATUS bits should be checked for lost locking state, including the need to reclaim locks after a server reboot.

5.5. Resolution of NFSv4.1 Issues

One possibility is that addressing all of the NFSv4.1 issues would entail publication of a standards-track document updating [RFC5661].

Such a document would have three major elements:

o A considerable expansion of the existing Section 11.4 explaining the various uses of the location attribute and the possible interactions among these various uses. This, like the corresponding replacement section for NFSv4.0 would be based on our Section 3.2 above. Information regarding the specifics of trunking discovery might appear in this section, in a new sub-section. As part of this revision, the existing Section 11.4.2 would need to be revised to explain all the possible results of NFS4ERR_MOVED including migration and a possible transparent transition in which the network address changes but the server does not.

- A revision of the existing section 18.35 (dealing with EXCHANGE_ID) addressing the issues discussed in Section 5.2.1.

- A major replacement of the existing Section 11.7, entitled "Effecting File System Transitions", as discussed below.

In addition, there is a set of smaller changes necessary

- Update the existing Section 2.10.5 to clarify the proper response to server_owner changes, as described in our Section 5.1.3.

- Replacement of the existing Section 15.1.2.4 to reflect the fact that NFS4ERR_MOVED can occur when a file system is now accessible at a different network address. A possible replacement text might be:

  > The file system that contains the current filehandle object is not accessible using the network address which has been used. It may have been relocated, migrated to another server, be accessible using another network address on the current server, or it may have never been present. The client may obtain the new file system location by obtaining the "fs_locations" or "fs_locations_info" attribute for the current filehandle. For further discussion, refer to Section 11.4.2

The replacement for the existing section 11.7 would maintain most sections essentially as they are, only making minor changes to include server-trunking in the discussion. However, in some cases involving more significant changes to existing sub-sections, and potential new sub-sections are listed below:

- The existing Section 11.7.1 needs to be modified to refer explicitly to the previous discussion of trunking discovery.

  In addition, the term "multi-home single-server namespace", used nowhere else in [RFC5661], poses difficulties. From the description given it appears that the case being referred to in one in which two network addresses return server_owners with the same major_id and different minor_id values, making the network addresses server-trunkable without being session trunkable.

  A better approach would be to refer to "server-trunking" as used elsewhere in this document and use the replacement for the existing Section 18.35 to identify clientid trunking as the means to adapt to network addresses which are server-trunkable without being session-trunkable and session trunking as the means to adapt to network addresses which are session-trunkable.

- The existing Section 11.7.2 needs to be better connected to trunking discovery. By calling these "transparent" transitions, it obscures the fact that some (or all) of the "transitions" it is discussing are not in fact transitions between servers or file systems but merely changes the set of communication paths in use.

- The existing Section 11.7.2.1, needs to address more clearly the case of server-trunkable addresses which are not session-trunkable. As it is, it mentions the related concept of clustering, but only deals explicitly with the case in which two distinct servers share access to one or more file systems and does not mention the case in which the network addresses can be used to access a shared stateid space without being session-trunkable.

- The existing Section 11.7.2.2, while correct, needs to be part of a general re-organization since the characteristics it lists as necessary for a transparent transition will be of use in other contexts, particularly as they apply to Transparent State Migration as well. It make sense to move these to a new sub-section within the equivalent of the Existing Section 11.7.

- The existing Section 11.7.7, needs the a major rework to deal with its basic assumption, that existing state can only be made available on the destination server if the source and destination co-operate in state management and maintain a common client id space. It is not clear how this can be done, other than for servers working together so as to provide clientid trunking, a case that is probably considered as a "transparent transition". The section needs to modified to allow something along the lines of NFSv4.0-style Transparent State Migration with the details provided by a later section (see below).

  A related issues concerns the sentence, "In the case of migration, the servers involved in the migration of a file system SHOULD transfer all server state from the original to the new server. It is unclear why this is a "SHOULD" as the rest of the paragraph essentially tells the client that it needs to be prepared for the server not to do this. The equivalent is a "should" in [RFC7931], and there is no reason to add to confusion by making a "SHOULD" in NFSv4.1. also, there is no mention of the need to provide a fs-specific grace period in the cases in which Transparent State Migration is not made available.

- Adding a new section (at level of the existing Section 11.7.7) about state transfer during migration. Although the phrase "Transparent State Migration" is well established in the context of NFSv4.0, the word "transparent" could cause confusion given the

existing use of the phrase "transparent transitions". A possible title for the new section is "State Transfer during Migration"

The new section would present the NFSv4.1-equivalent of Transparent State Migration as described in [RFC7931]. This would address the issues presented in Section 5.1 along the lines suggested in Sections 5.2, 5.3, and 5.4.

6. Security Considerations

With regard to NFSv4.0, the Security Considerations section of [RFC7530] as modified by that of [RFC7931] takes proper care of migration-related issues. No change is needed.

With regard to NFSv4.1, the Security Considerations section of [RFC5661] takes proper care of migration-related issues. No change is needed.

7. IANA Considerations

This document does not require actions by IANA.

8. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC5661] Shepler, S., Ed., Eisler, M., Ed., and D. Noveck, Ed., "Network File System (NFS) Version 4 Minor Version 1 Protocol", RFC 5661, DOI 10.17487/RFC5661, January 2010, <http://www.rfc-editor.org/info/rfc5661>.

[RFC7530] Haynes, T., Ed. and D. Noveck, Ed., "Network File System (NFS) Version 4 Protocol", RFC 7530, DOI 10.17487/RFC7530, March 2015, <http://www.rfc-editor.org/info/rfc7530>.

[RFC7931] Noveck, D., Ed., Shivam, P., Lever, C., and B. Baker, "NFSv4.0 Migration: Specification Update", RFC 7931, DOI 10.17487/RFC7931, July 2016, <http://www.rfc-editor.org/info/rfc7931>.

Appendix A. Acknowledgements

The editor and authors of this document gratefully acknowledge the contributions of Trond Myklebust of Primary Data, Robert Thurlow of Oracle, and Andy Adamson of NetApp. We also thank Tom Haynes of

Primary Data and Spencer Shepler of Microsoft for their guidance and suggestions.

Special thanks go to members of the Oracle Solaris NFS team, especially Rick Mesta and James Wahlig, for their work implementing an NFSv4.0 migration prototype and identifying many of the issues documented here. Also, the work of Xuan Qi of Oracle with NFSv4.1 client and server prototypes was helpful.

Appendix B. History of this Document

The contents of successive versions of this document have changed because new issues have been discovered, because there here have changes in our understanding of how these features should interact, and because some of the issues have been adequately addressed with regard to certain protocol versions.

As a result, it may be helpful to understand the history of these issues, which is complicated because multiple NFSv4 protocols have been involved.

This history can be summarized as follows

o Initially, the focus was on the difficulties seen in NFSv4.0 implementations of Transparent State Migration, and on identifying possible corrections to [RFC7530] that might address these issues.

  At this point, treatment of NFSv4.1 was minimal.

o As examination of the issues continued, it became clear that the use of the non-uniform client string model was a critical element of the problem and further work proceeded on that basis.

  During the period, treatment of NFSv4.1 was expanded but the fact that NFSv4.1 had existing facilities for trunking detection was taken as an indication that the problems would not be difficult to address..

o As work proceeded on a standards-track document addressing the NFSv4.0 issues, material that proposed changes to address the issues became less relevant, since the effective vehicle for addressing these issues became the standards-track document.

  During this period, and subsequently, treatment of NFSv4.1 remained essentially unchanged.

o With the publication of [RFC7931], material regarding fixes for the NSV4.0 became vestigial but the material was retained for a

while together with a shift from proposing those changes to reporting that they had been made.

- Later, in response to experiences testing existing NFSv4.1 implementations of migration, the focus of the document shifted decisively to NFSv4.1. As part of the analysis of migration within NFSv4.1, it was realized that issues related to the appearance of multiple addresses were fundamental to clearly describing how migration would work and that changes in the set of such addresses might or might not involve migration.

  At this point, discussion of NFSV4.0 issues was further limited. The issues seen were noted but the discussion of the resolution was limited to explaining that they had been addressed by the publication of [RFC7931].

- Finally, based on the results of work to provide NFSv4 with trunking discovery facilities, a decision was made that this work was most appropriately dealt with together with migration, for reasons noted previously.

  Since the trunking discovery facilities apply to all NFSv4 minor versions, work was needed to define those for NFSv4.0as well, together with the necessary interactions with migration.

Authors' Addresses

David Noveck (editor)
NetApp
26 Locust Avenue
Lexington, MA 02421
US

Phone: +1 781 572 8038
Email: davenoveck@gmail.com

Piyush Shivam
Oracle Corporation
5300 Riata Park Ct.
Austin, TX 78727
US

Phone: +1 512 401 1019
Email: piyush.shivam@oracle.com

Charles Lever
Oracle Corporation
1015 Granger Avenue
Ann Arbor, MI 48104
US

Phone: +1 248 614 5091
Email: chuck.lever@oracle.com

Bill Baker
Oracle Corporation
5300 Riata Park Ct.
Austin, TX 78727
US

Phone: +1 512 401 1081
Email: bill.baker@oracle.com