NFSの動向

2017月４月

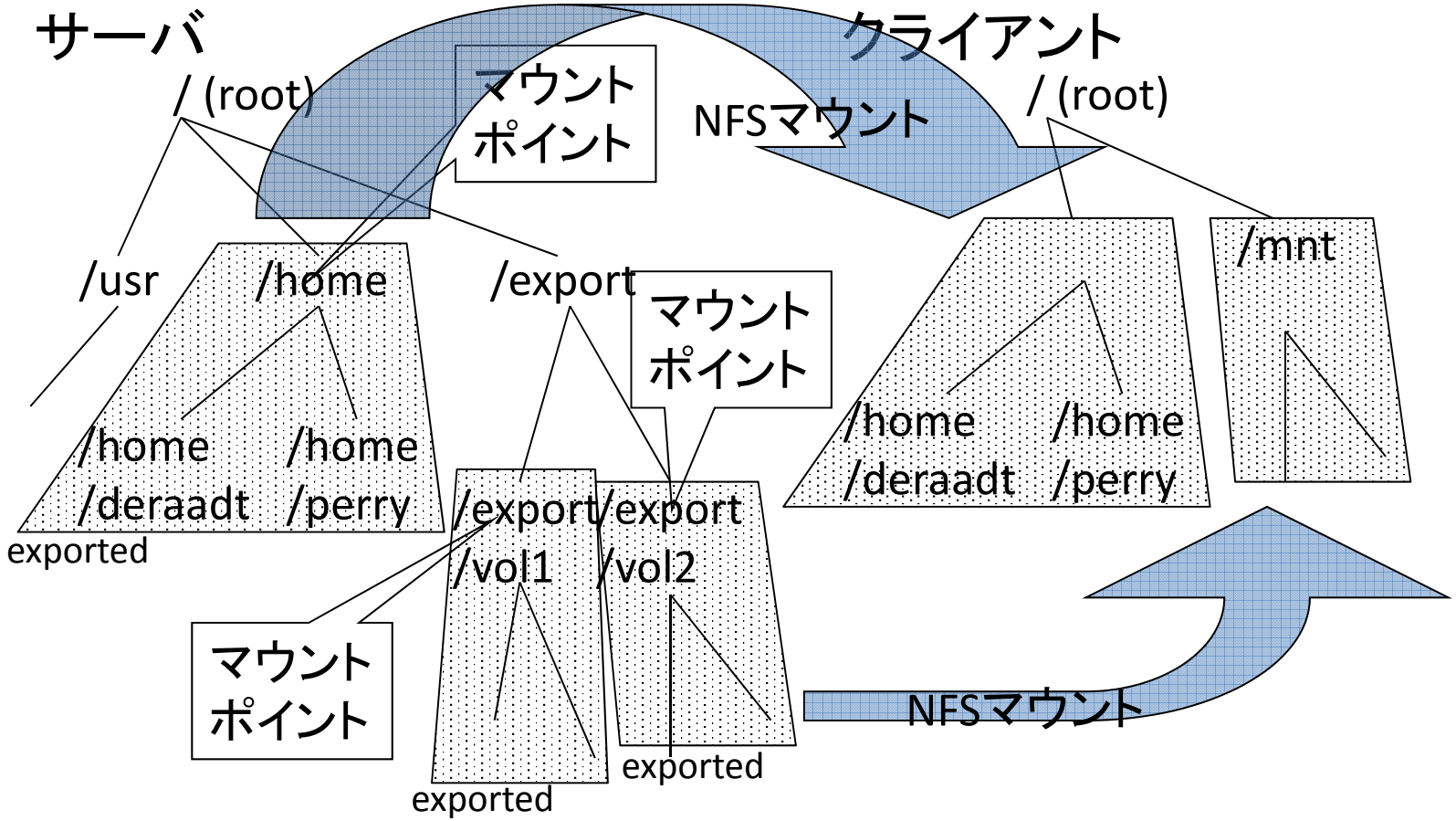
SCSK株式会社
R＆Dセンター　技術戦略部

NFSの動向

- NFSv3/v4プロトコル概要
- NFSv4.1 (RFC 5661)
  - セッションの導入
  - pNFS (Parallel NFS)
  - 委譲の強化
- NFSv4.2 (RFC 7862)
  - サーバサイドコピー
  - I/Oアドバイス
  - 穴あきファイル
- Linuxでの実装
  - カーネル
  - Red Hat Enterprise Linux

NFSv3ファイルシステムモデル (1/3)

- Unixのファイルシステムモデル



NFSv3ファイルシステムモデル (2/3)

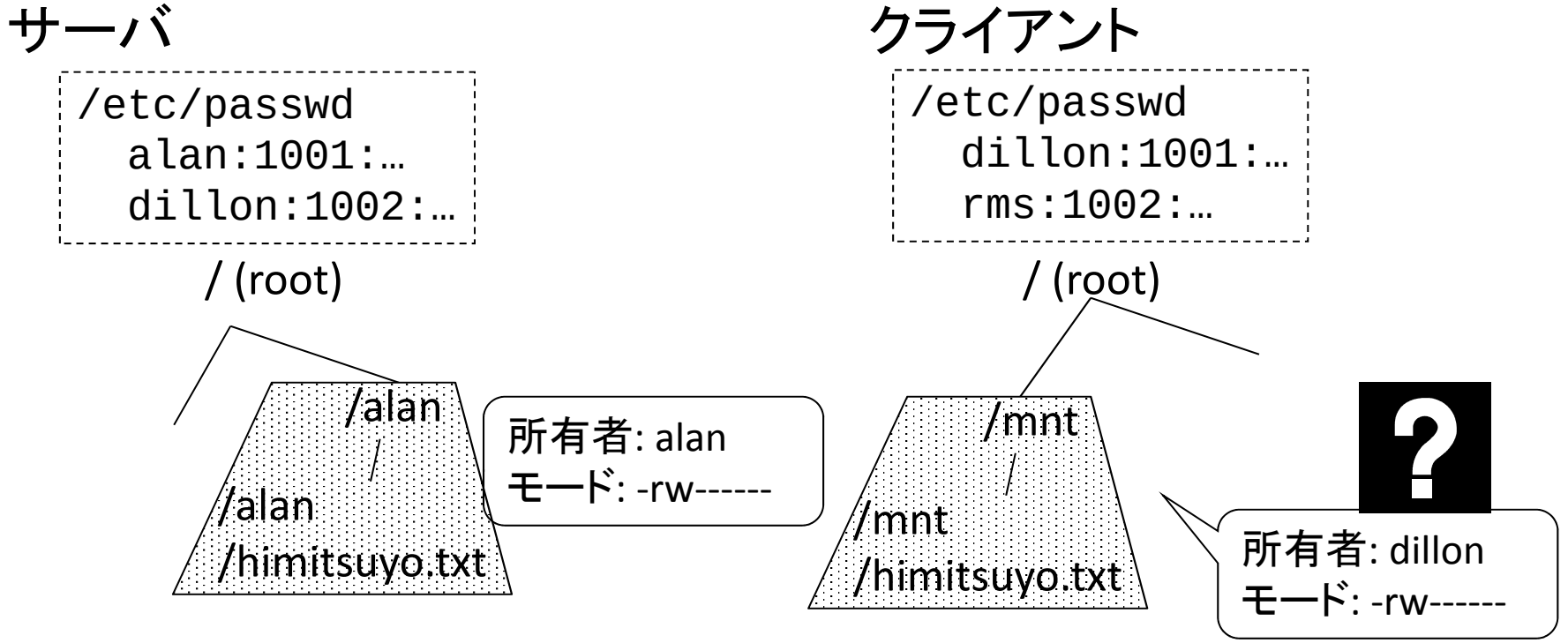
- ファイルハンドル: サーバ内のファイルを一意に識別するID
- NFSのすべてのオペレーションの基本
- 永続的: 通常、マウントポイントのID (デバイス番号など)、inode番号、世代番号などから合成
  - 世代番号: ファイルが削除され、inodeが再利用されたら?
- サーバの再起動などでも不変
- クライアントからは透過、サーバがすべて解釈

NFSv3ファイルシステムモデル (3/3)

- 特殊ファイル – デバイスノード、名前つきパイプ、シンボリックリンク
  - デバイスそのものがexportされるわけではない (サーバ側のデバイスの操作はできない)
  - パイプの通信内容が転送されるわけではない (サーバ側のプロセスとの通信はできない)
  - シンボリックリンクの解釈はクライアントが行う (readlink(2)相当READLINKオペレーション)
- 参考: ディスクレスクライアント

NFSv2/v3認証・アクセス制御モデル

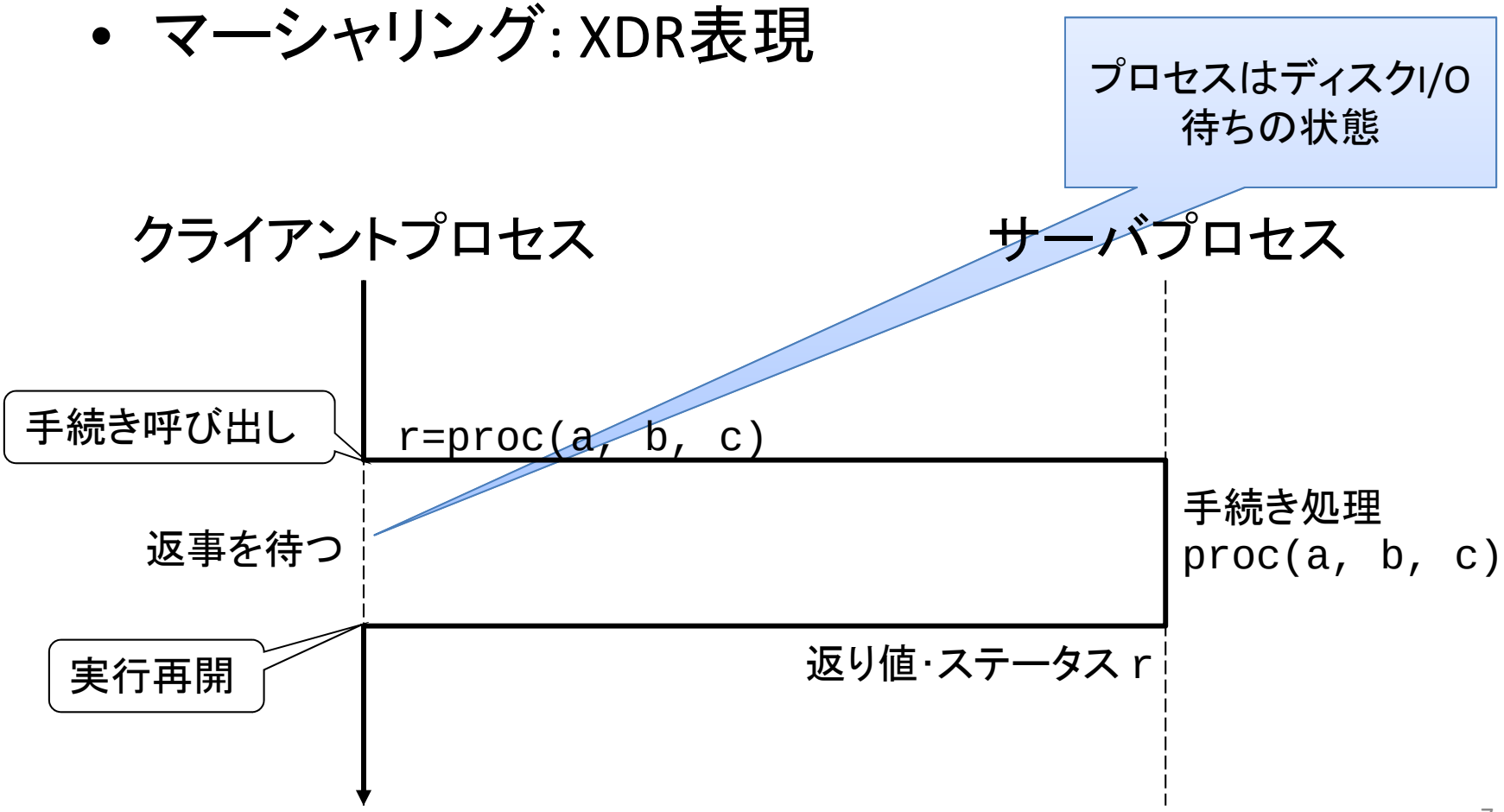
- AUTH_SYS認証～UnixのUID、GIDを利用
  - UID、GIDの一貫性が必須



■Kerberos認証、GSS-APIによる認証

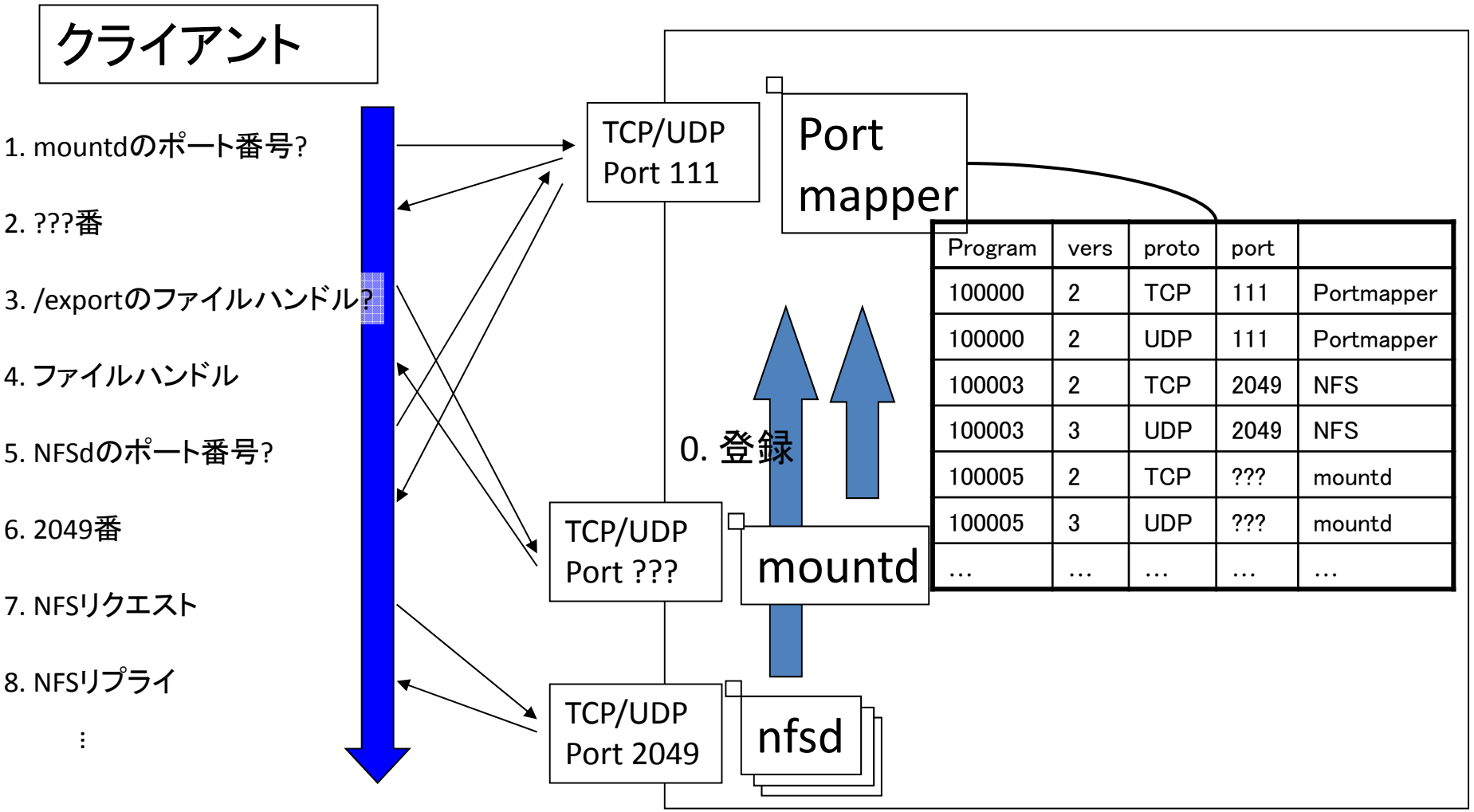
ONC RPC (Sun RPC)

- Remote Procedure Call – リモートホストの手続きを呼び出す
- マーシャリング: XDR表現



Port mapper

(NFSv4では1~6を飛ばす)



Program	vers	proto	port	
100000	2	TCP	111	Portmapper
100000	2	UDP	111	Portmapper
100003	2	TCP	2049	NFS
100003	3	UDP	2049	NFS
100005	2	TCP	???	mountd
100005	3	UDP	???	mountd
…	…	…	…	…

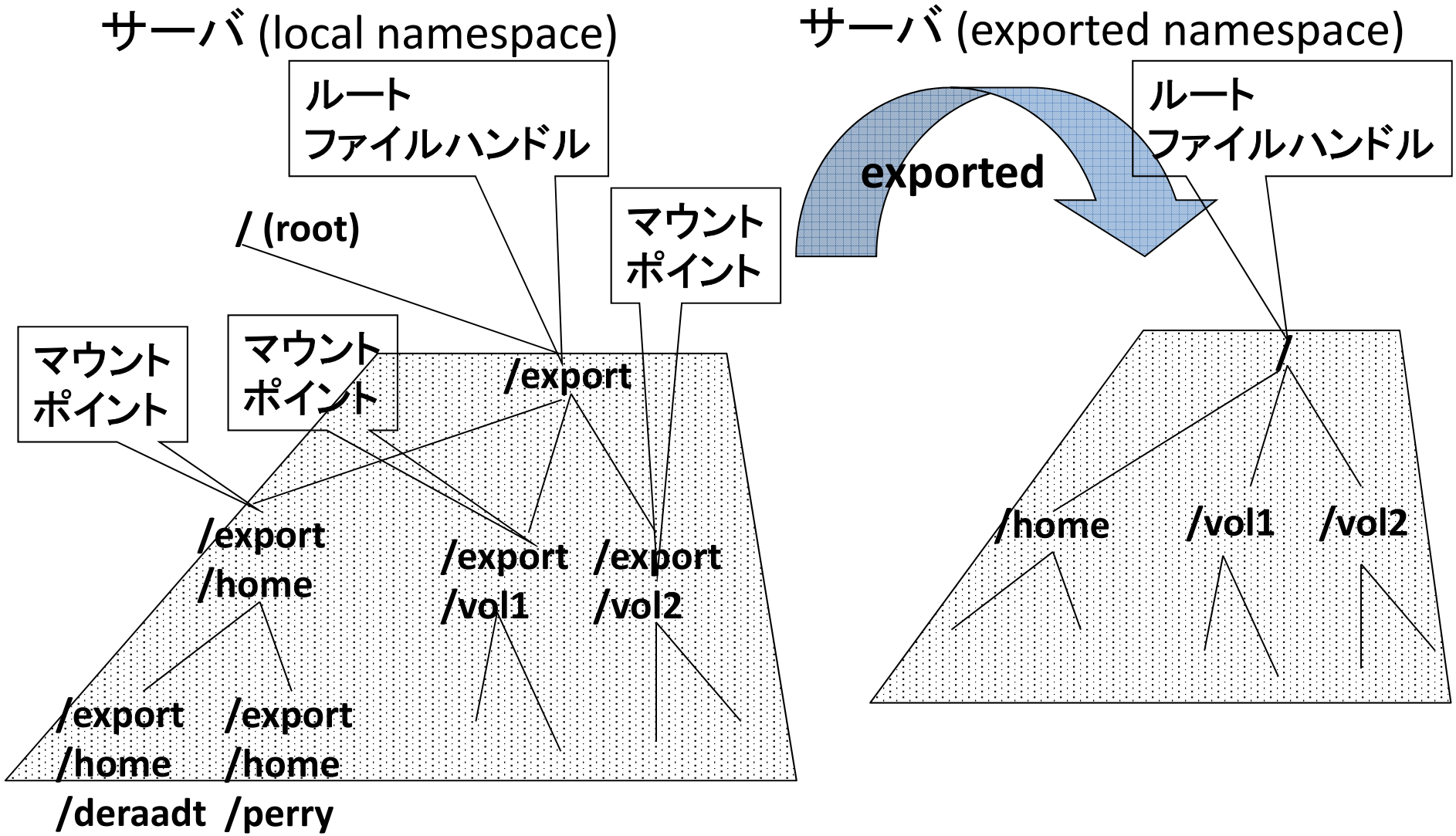
NFSv4概要

- NFS v3までの課題を解決
  - 広域での利用に適したメッセージ構造
  - Windowsとの相互運用性
  - 強力なセキュリティ機能
- 既存実装の再利用は考えず、根本から設計を改訂

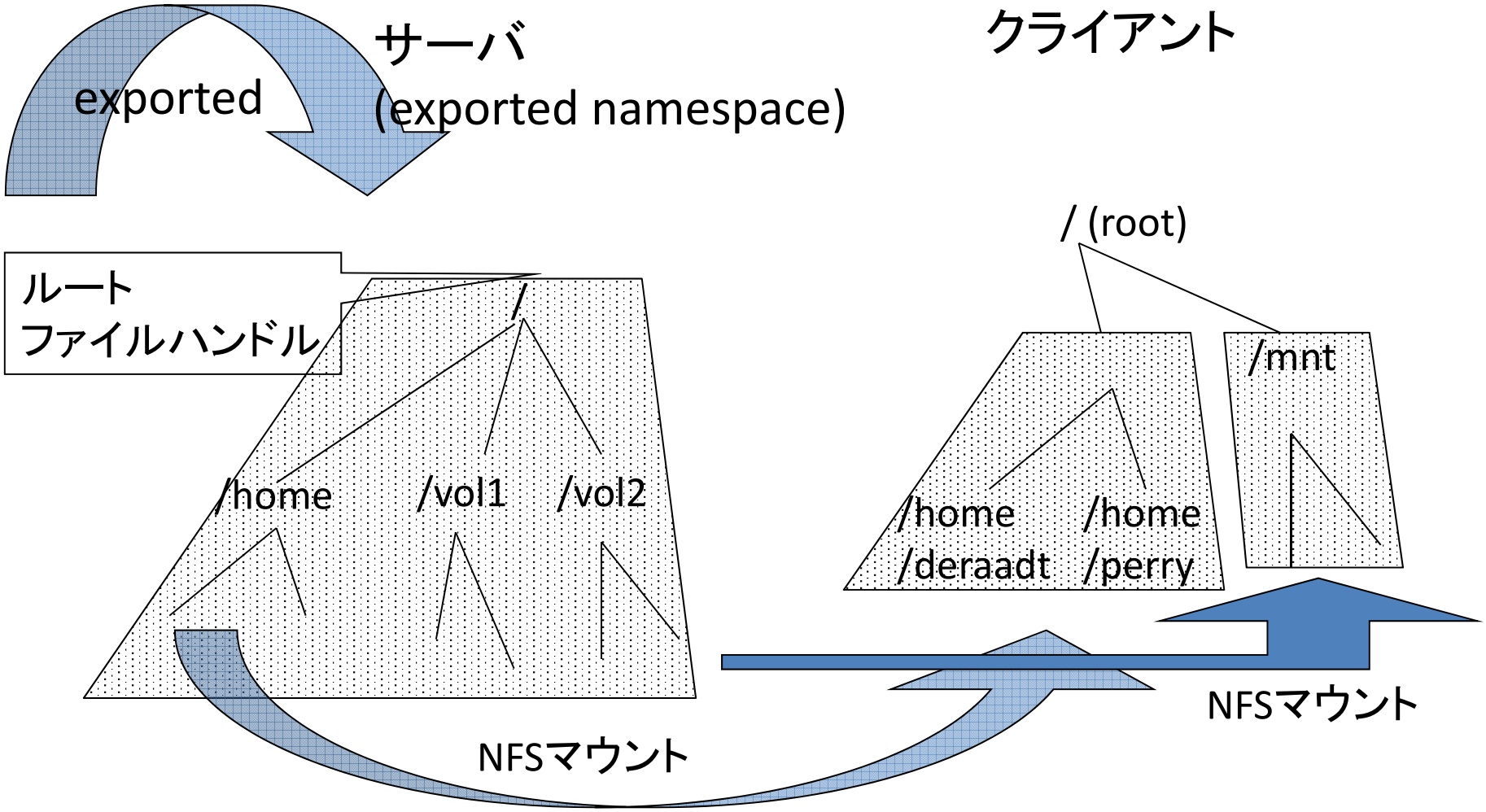
NFSv4ファイルシステムモデル (1/4)

- 基本は同じ
- ユーザID/グループID (v3) → UTF8文字列によるユーザ名/グループ名 (v4)
- ファイル名は0で終端するASCII文字列 (v3) → 0で終端するUTF8文字列
- ルートファイルハンドルの概念: サーバのファイルシステム構造のエクスポート (v3) → サーバのファイルシステム構造の隠蔽 (v4)

NFSv4ファイルシステムモデル (2/4)



NFSv4ファイルシステムモデル (3/4)



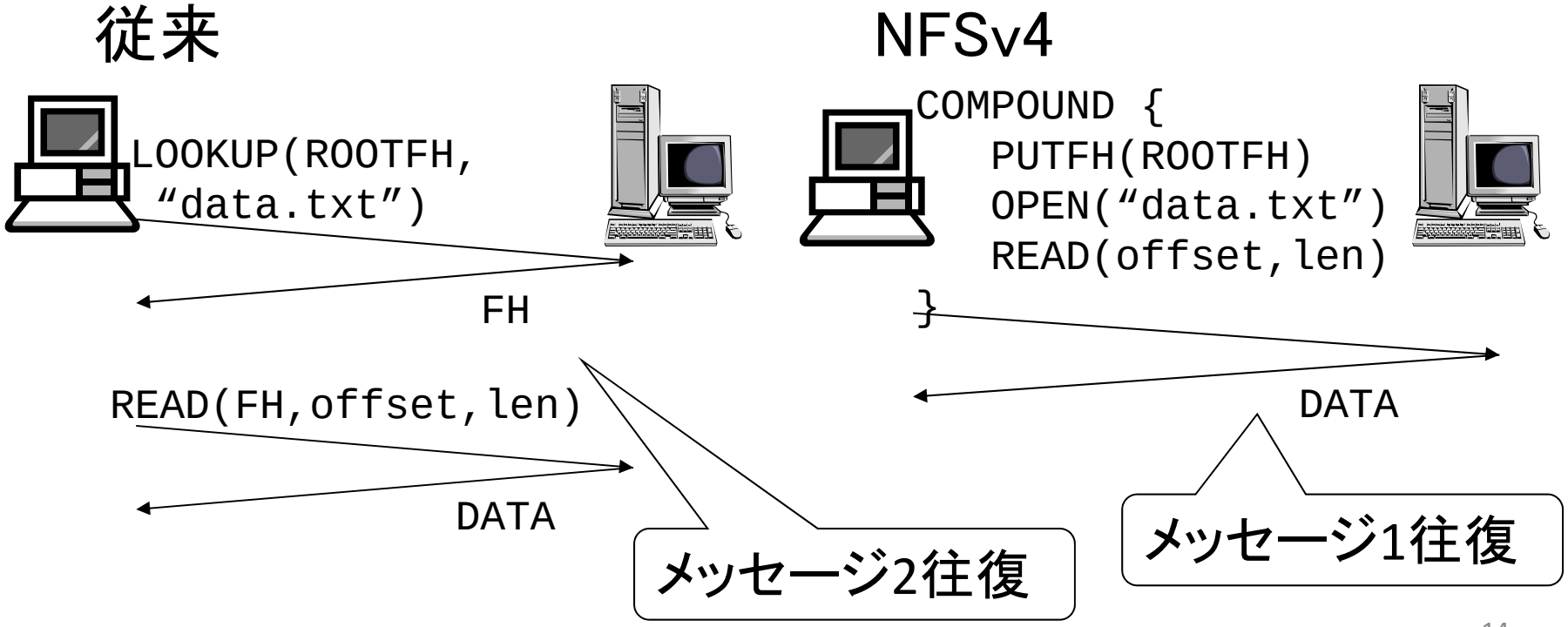
NFSv4ファイルシステムモデル (4/4)

- ファイルハンドルは永続的とは限らない
- ファイル属性fh_expire_type
  - FH4_PERSISTENT、FH4_NOEXPIRE_WITH_OPEN、FH4_VOLATILE_ANY、FH4_VOL_MIGRATION、FH4_VOL_RENAME
- NFSエラーコードNFS4ERR_FHEXPIRED

COMPOUND処理

- 複数のオペレーションを1つのメッセージに
- カレントファイルハンドルの導入

  例: data.txtの読み込み



クライアント側のキャッシュ処理

- Delegationによるキャッシュ
  - ファイルを占有できる場合:
    - OPENしているクライアントが1つである場合
    - 書き込みOPENしているクライアントがない場合
  - Delegationの解除→クライアントへのコールバック

Windowsとの相互運用性

- OPEN/CLOSEの導入 → stateless性の放棄: client IDにより再起動時の状態復旧を実現
- ロックプロトコルの統合
- 揮発性ファイルハンドル

  例: ファイルのリネーム Unixではi-node番号が変わらないためファイルハンドルも不変だった
- Windowsのアクセス制御モデルのサポート
- Unicode (UTF-8) ファイル名

Windowsとの相互運用性

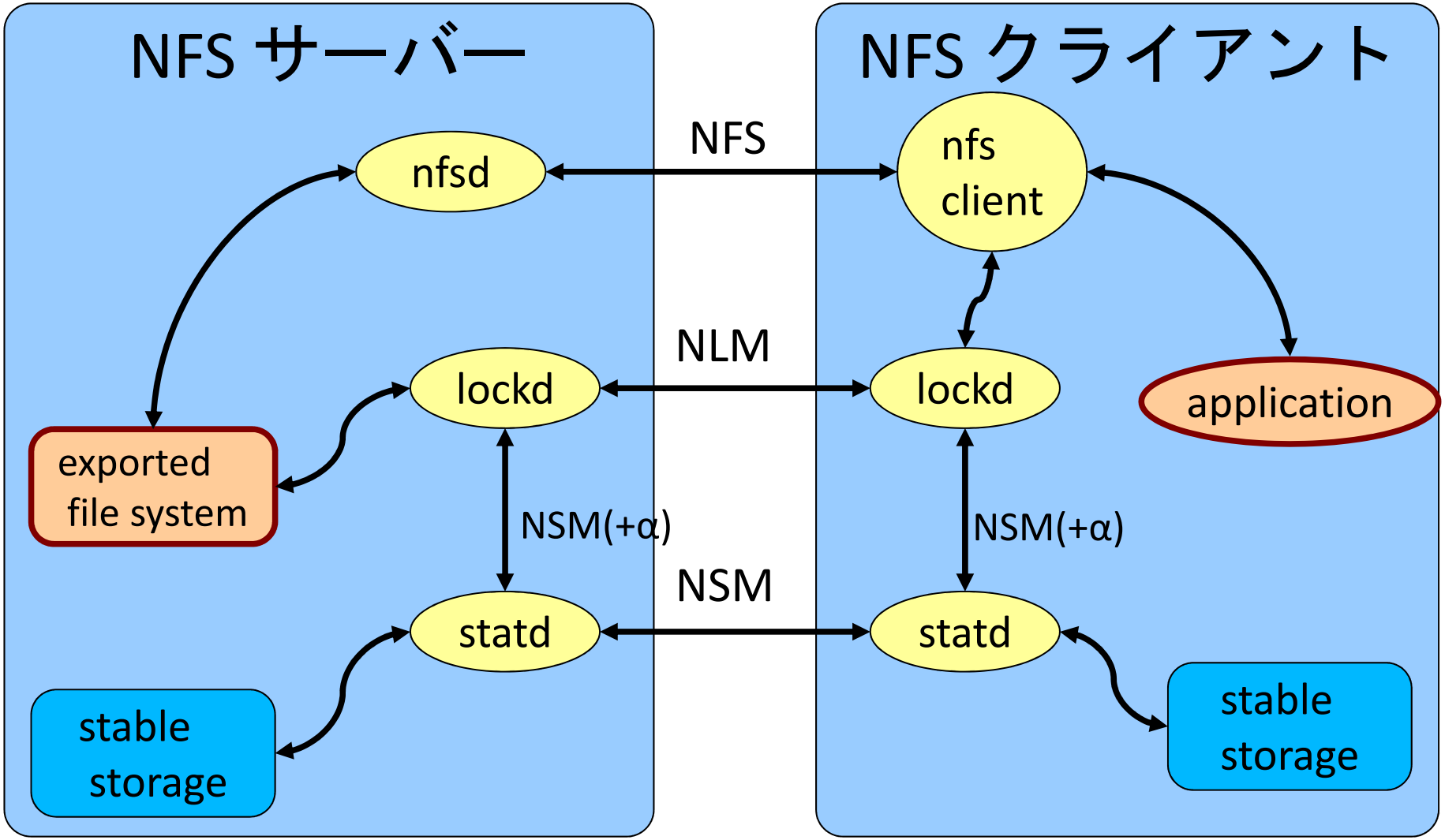
- OPEN/CLOSEの導入 → stateless性の放棄: client IDにより再起動時の状態復旧を実現
- ロックプロトコルの統合
- 揮発性ファイルハンドル

  例: ファイルのリネーム Unixではi-node番号が変わらないためファイルハンドルも不変だった
- Windowsのアクセス制御モデルのサポート
- Unicode (UTF-8) ファイル名

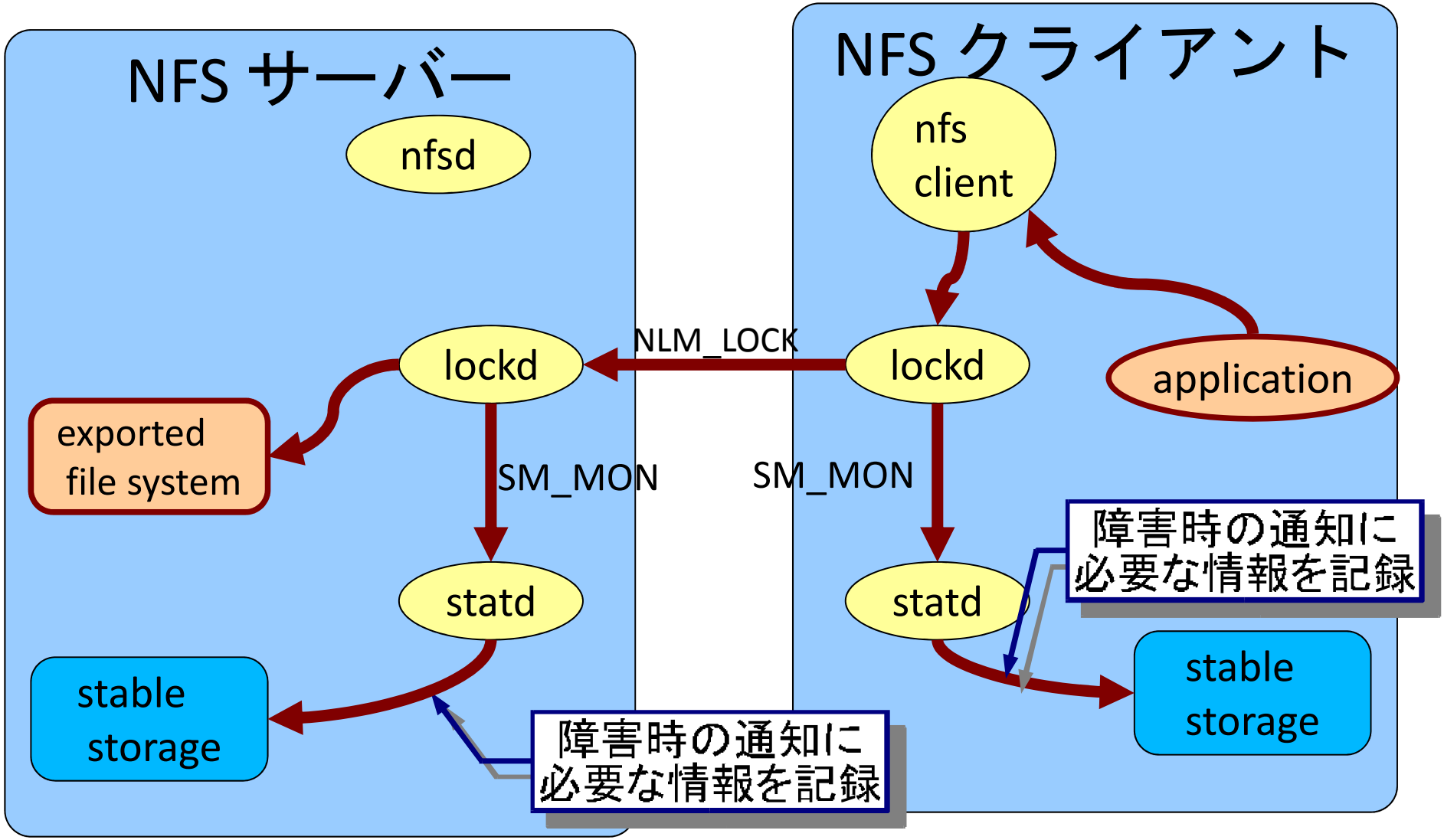
NFS file locking

- NFSv3
  - optional (NLM)
  - callback based sleep lock
  - fragile,racy
  - crash recovery by notification (callback)
- NFSv4
  - polling based sleep lock
  - seqid4: at-most-onc e semantics by more reliable DRC (Duplicate Request Cache)
  - crash recovery by lease expiration
  - if a client holds a write delegation, locking can be performed locally

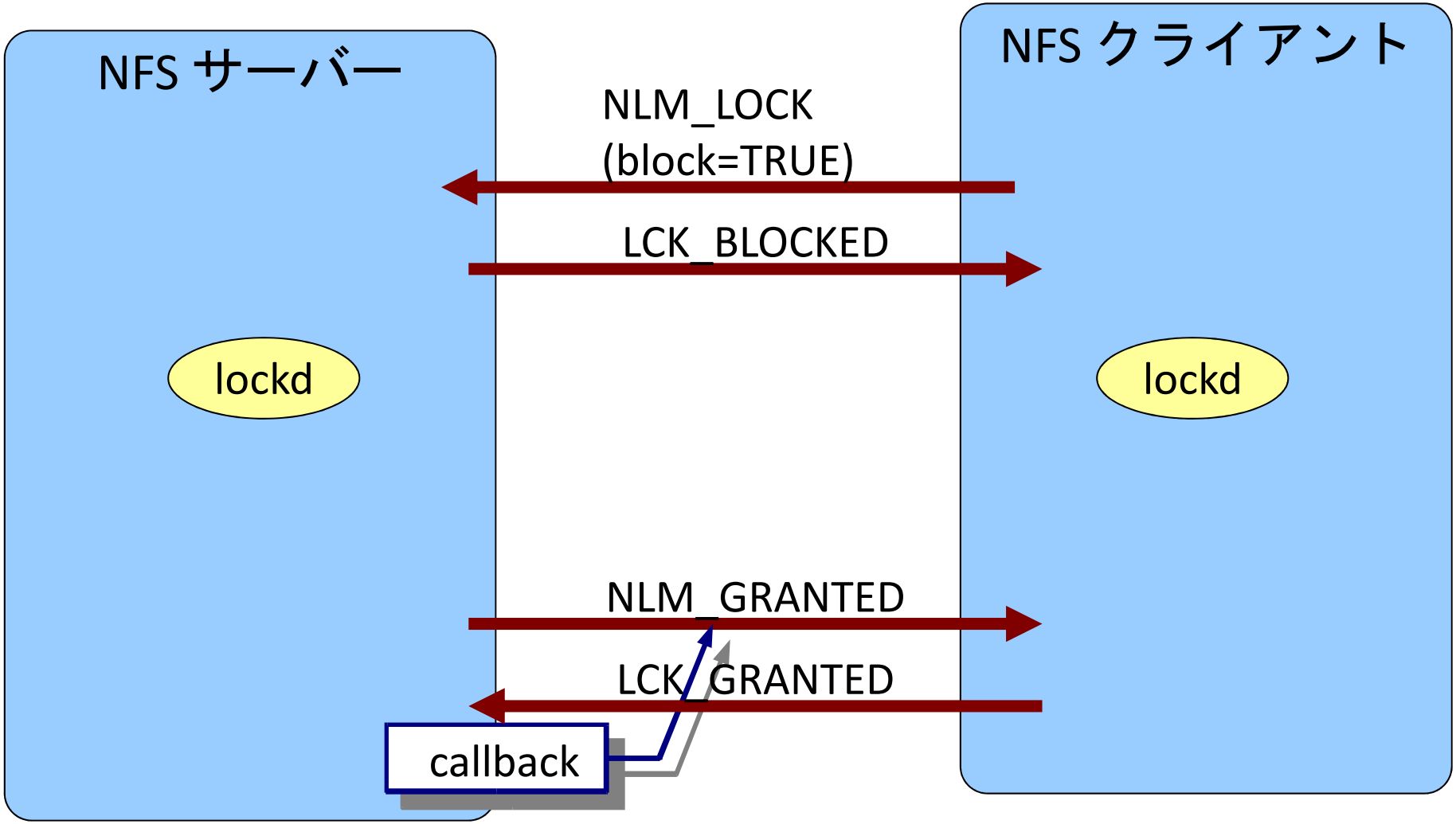
NFSv3+NLM


NFS サーバー
NFS クライアント
nfsd
NFS
nfs client
NLM
lockd
lockd
application
exported file system
NSM(+α)
NSM(+α)
NSM
statd
statd
stable storage
stable storage

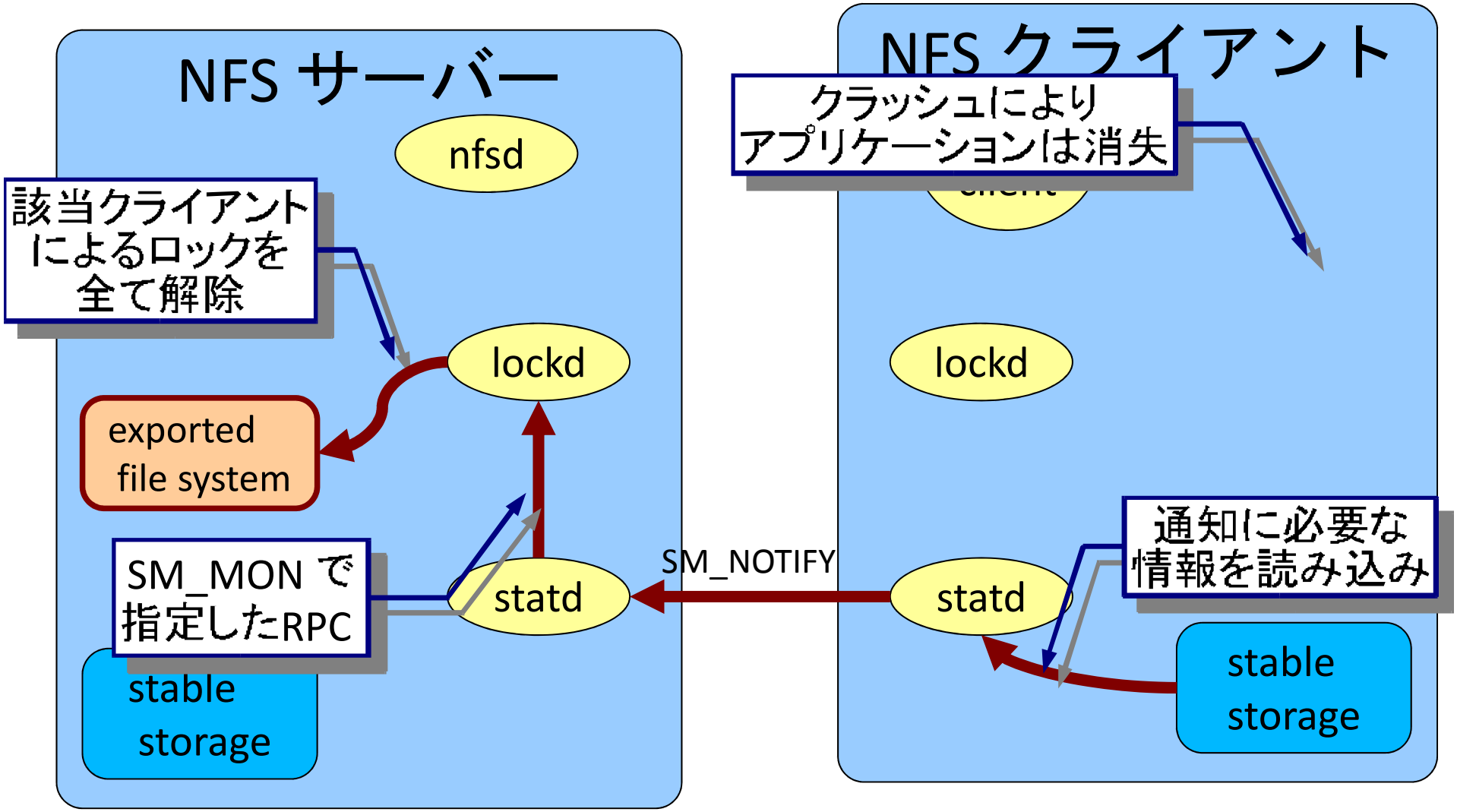
lockf 処理概要(NFSv3+NLM)



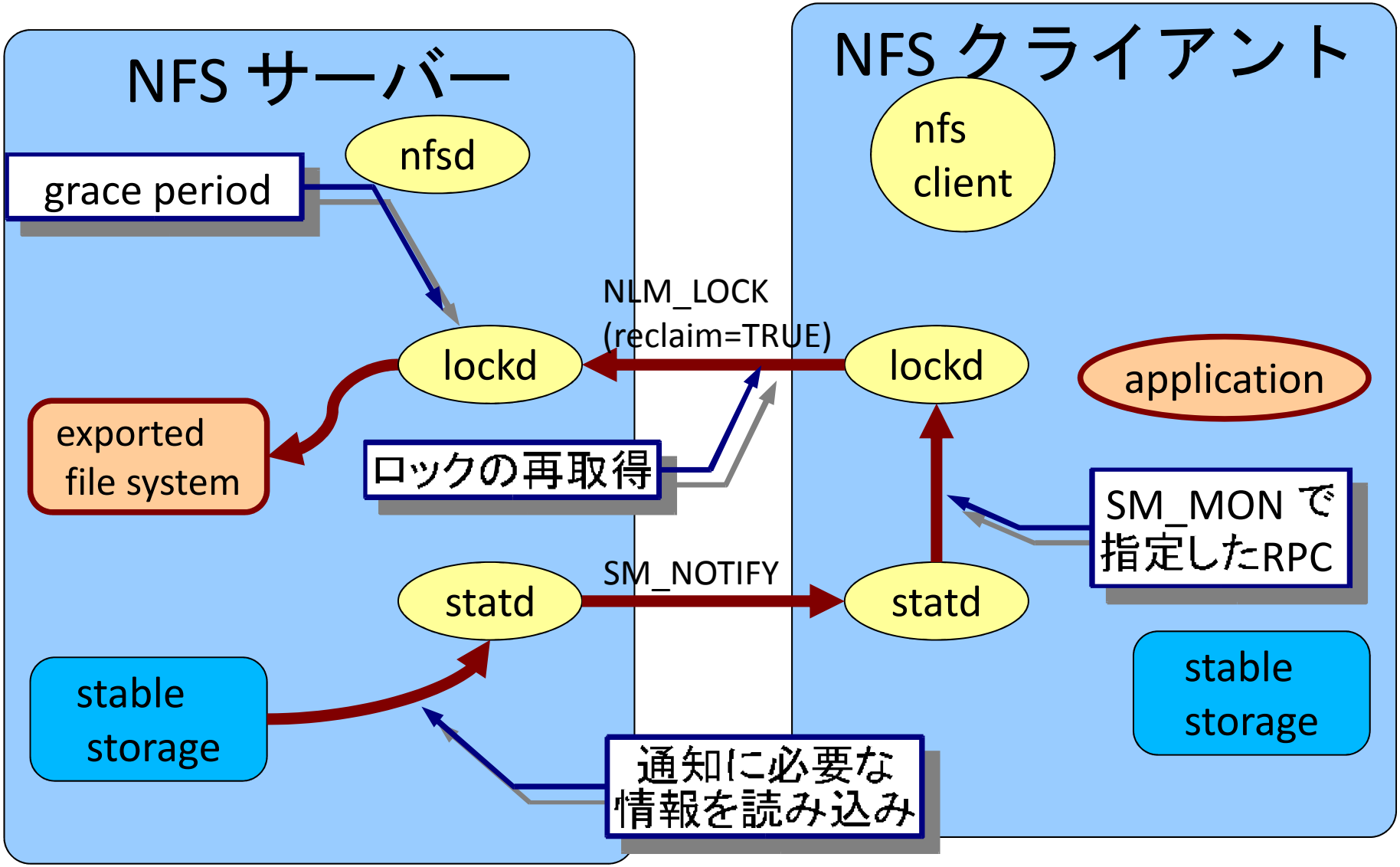
blocking lock(NFSv3+NLM)



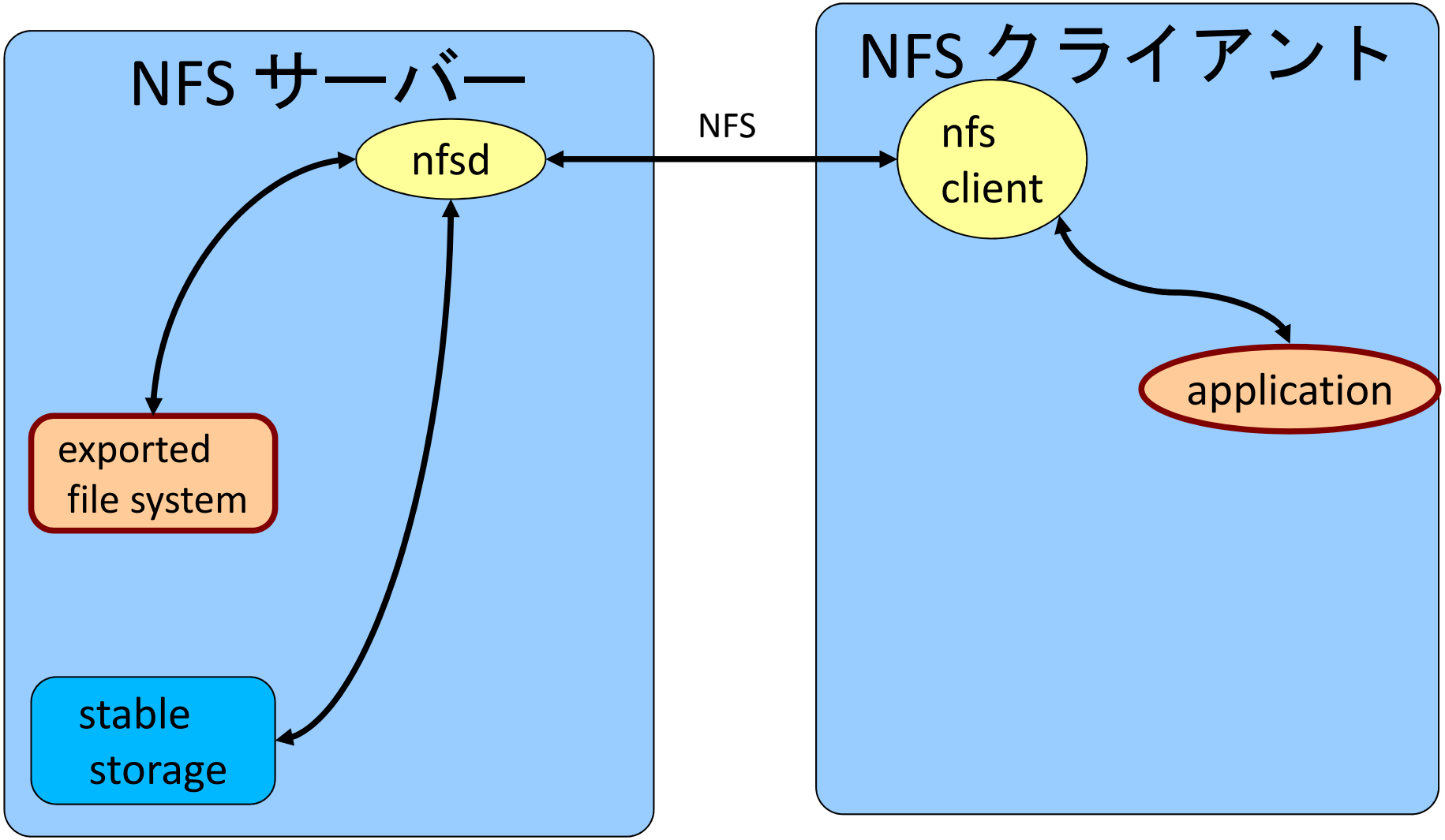
client crash recovery(NFSv3+NLM)



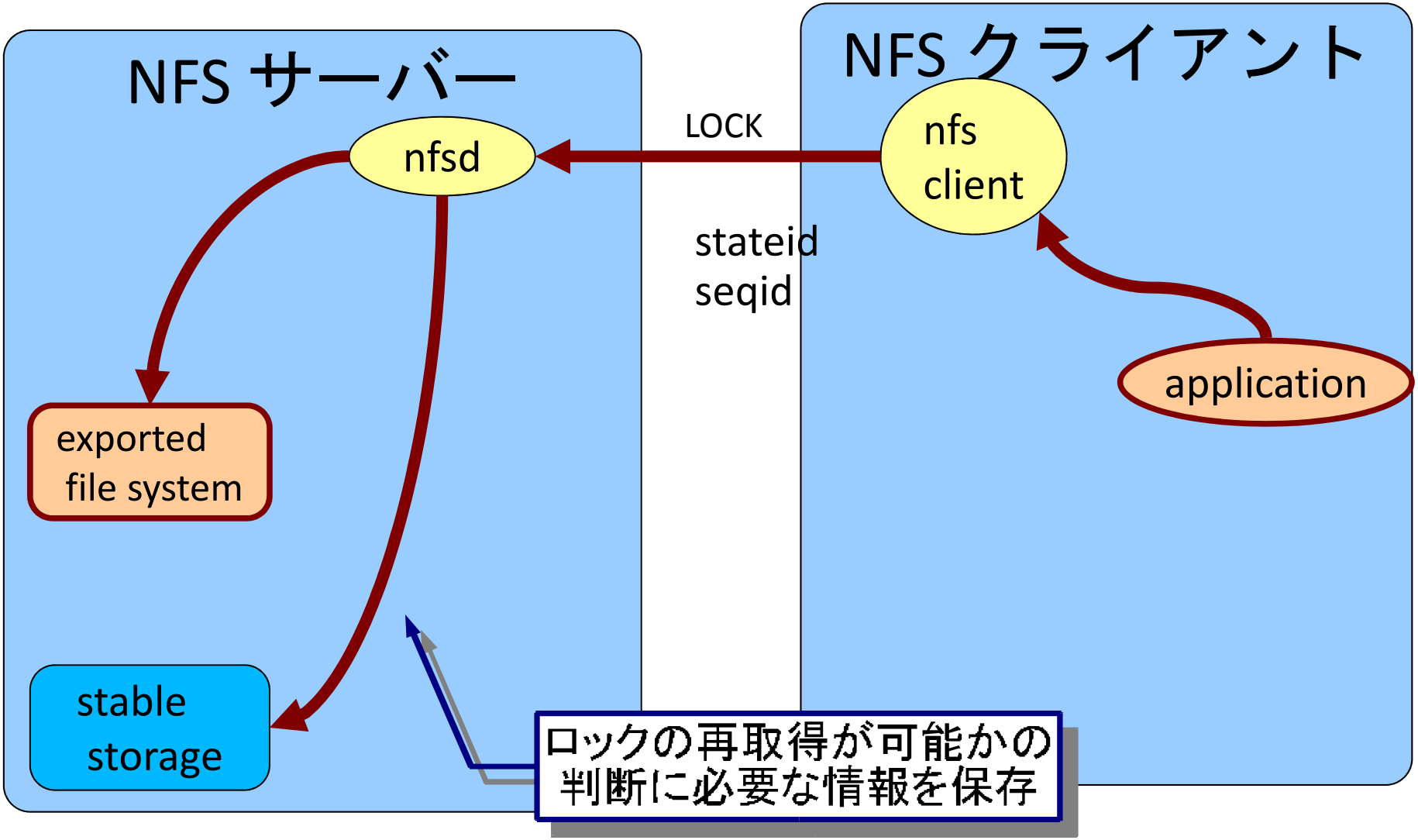
server crash recovery(NFSv3+NLM)



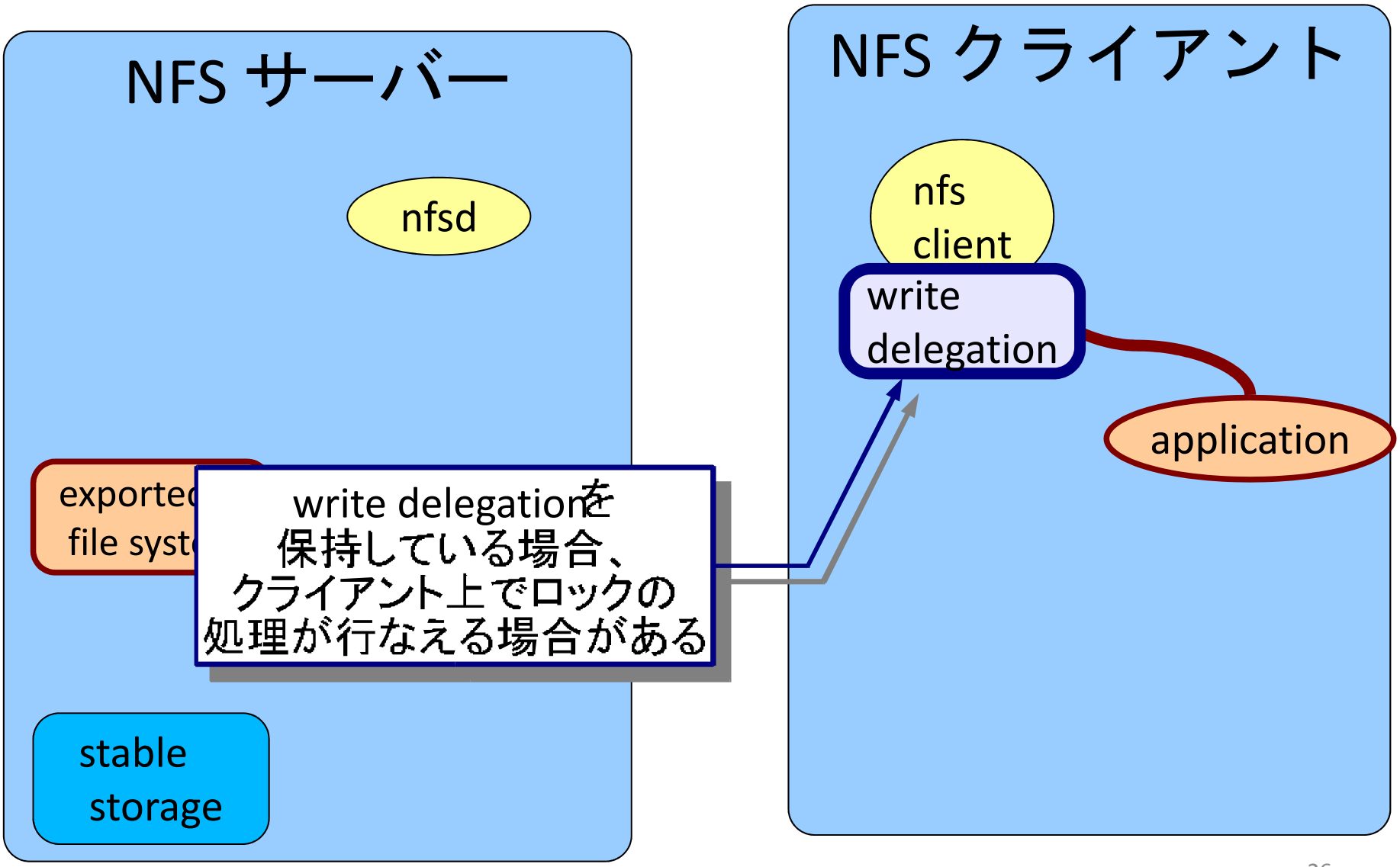
NFSv4


NFS サーバー
nfsd
exported
file system
stable
storage
NFS
NFS クライアント
nfs
client
application

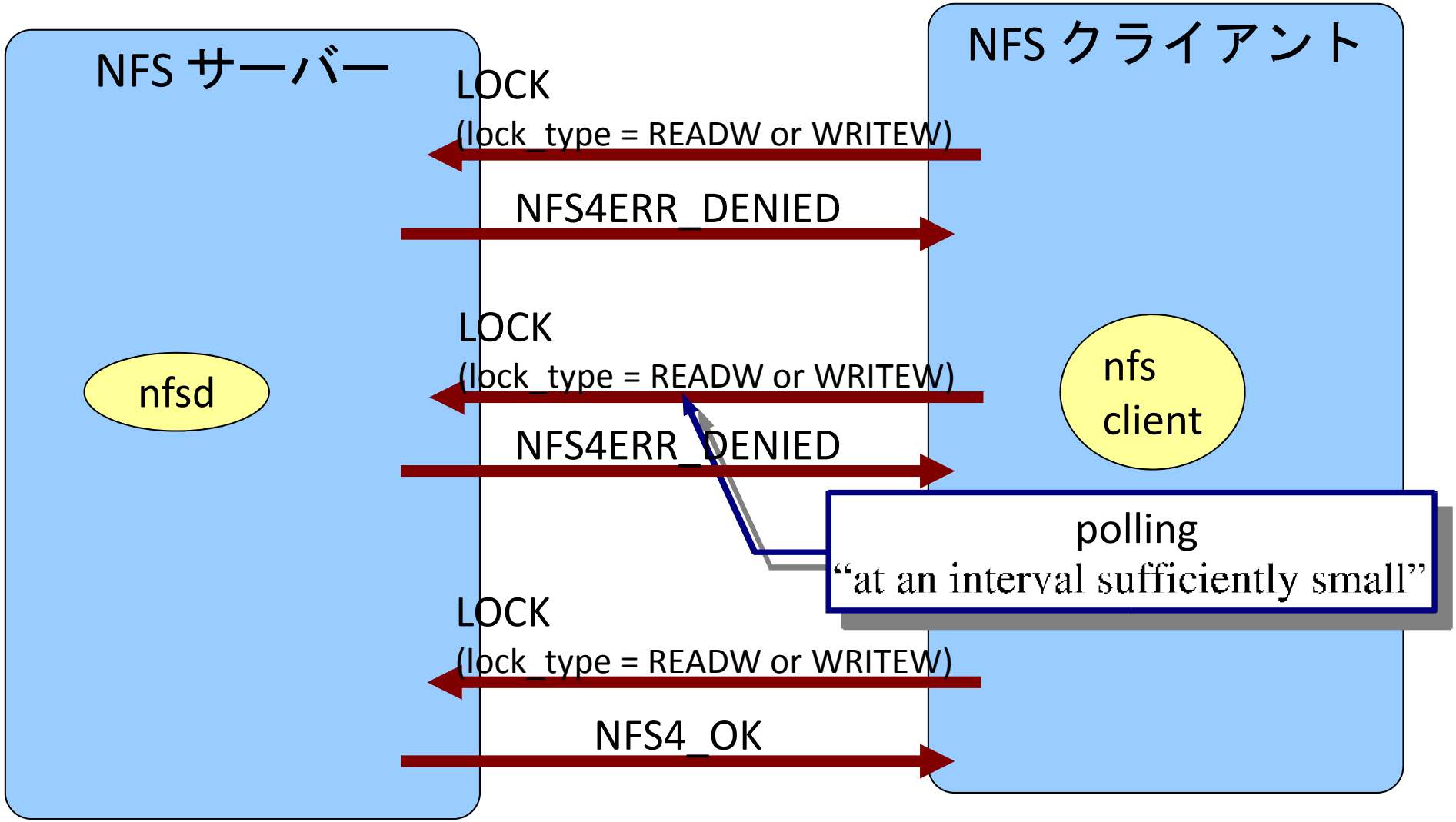
lockf 処理概要(NFSv4)



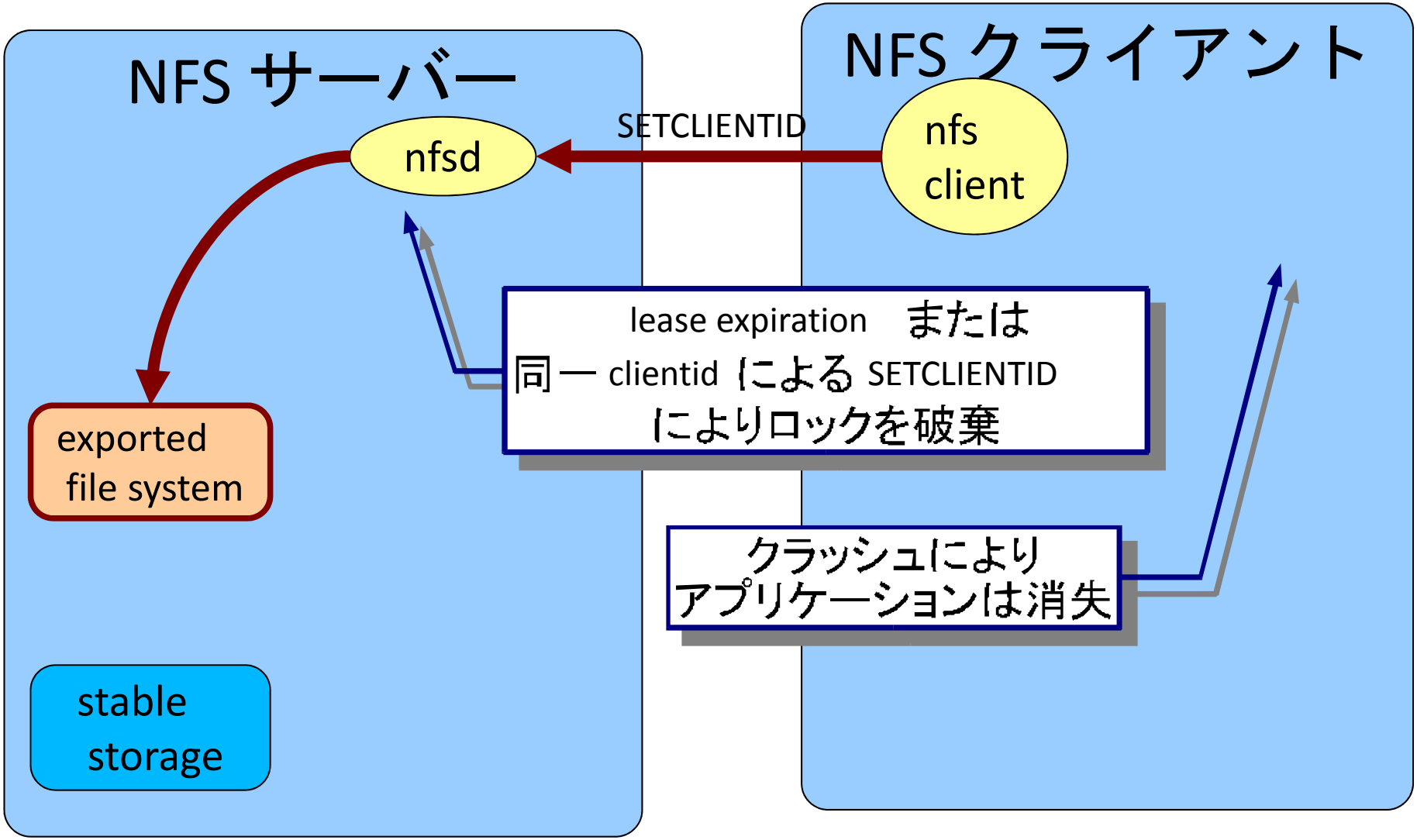
lockf 処理概要(NFSv4+ delegation)



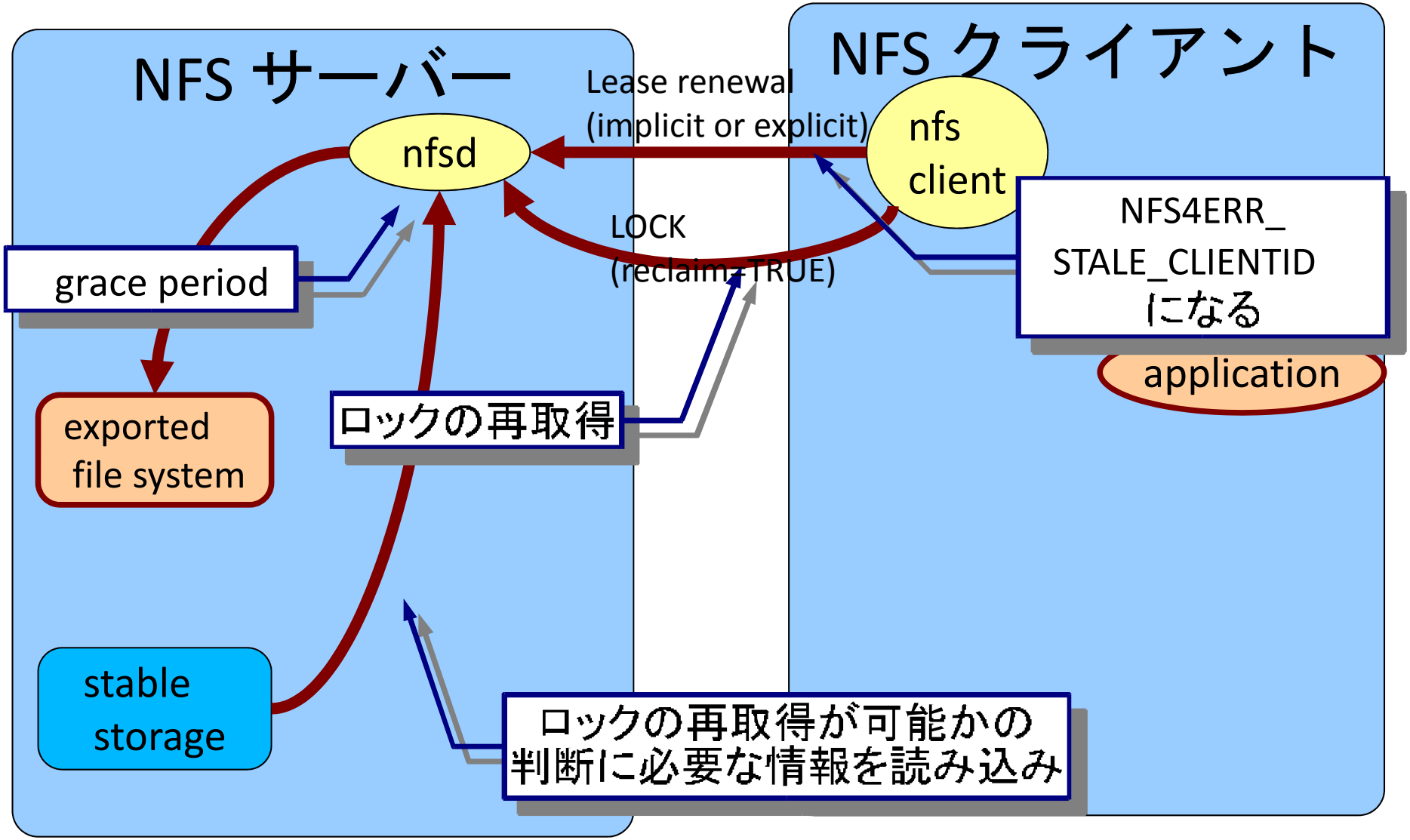
blocking lock(NFSv4)



client crash recovery(NFSv4)



server crash recovery(NFSv4)



NFSv4.1概要

NFSv4.1の主な機能

- セッションの導入
- pNFS
- 委譲の強化
  - ディレクトリなどファイル以外の委譲
    - ディレクトリの変更をコールバック可能
  - WANT_DELEGATION (委譲が可能になったらコールバック)　など

セッション

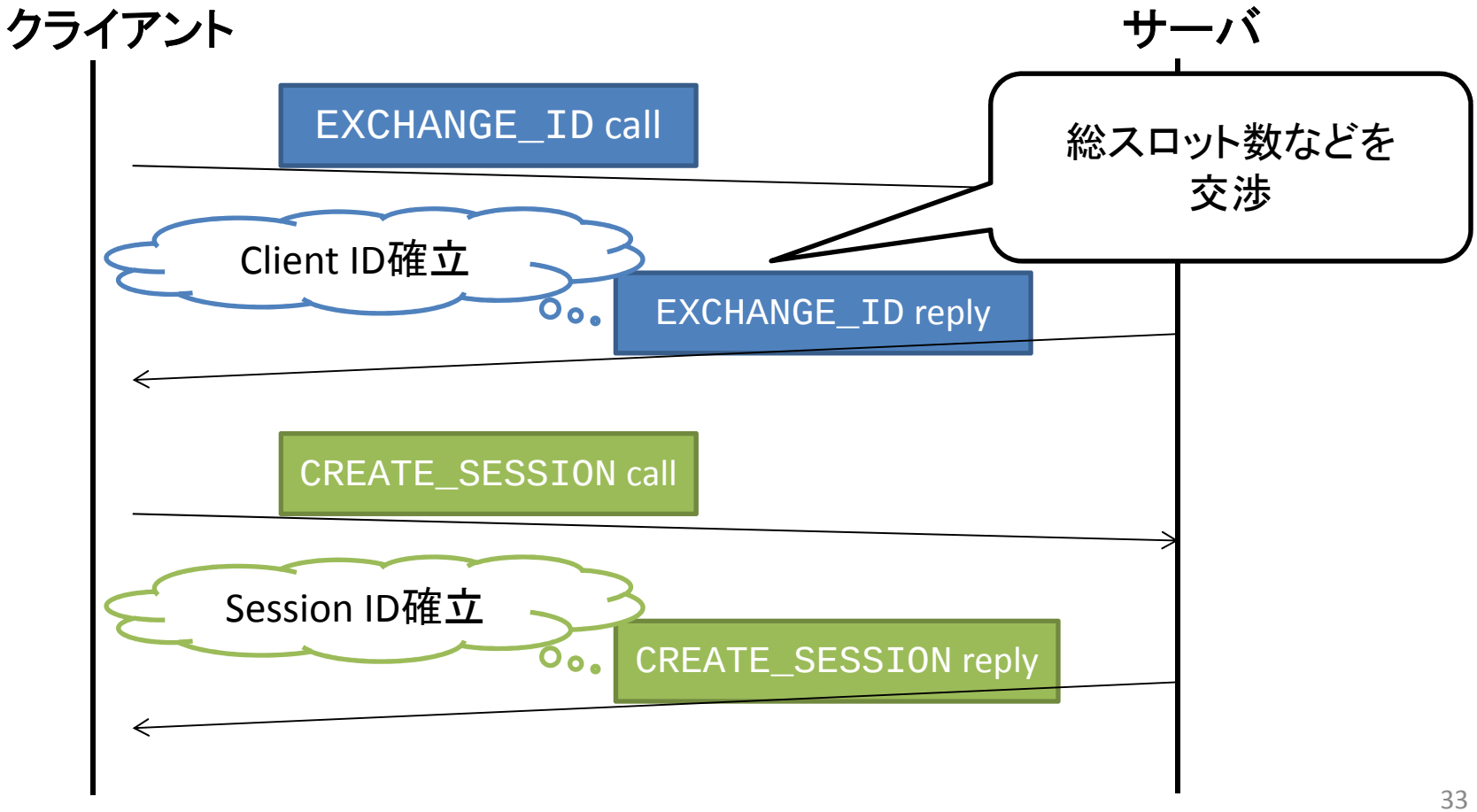
NFSv2/v3	NFSv4	NFSv4.1
ステートレス ロックは外出し (NLM)	ステートフル	ステートフル
	OPEN/CLOSE: 委譲 Client ID: ファイルロック、委譲	OPEN/CLOSE: 委譲 Client ID: ファイルロック、委譲
		Session ID: EOS、トランキング

EOS: Exact Once Semantics

セッション

- mount時に確立

コネクションを既存のセッションに
関連付けることも可能
(BIND_CONN_TO_SESSION)



EXCHANGE_ID

他にセッション乗っ取り防止策など

- 主な引数
  - client_owner: verifierとowneridの組
    - verifierはクライアントの再起動ごとに変わる整数
    - owneridはクライアントを識別する文字列
  - flags: pNFSやマイグレーションなどの受け入れ状況をサーバに知らせる
- 主な返り値
  - clientid: サーバがクライアントごとに振るID番号
  - sequenceid: CREATE_SESSIONのsequence
  - flags: 引数のflagsとのANDを取る
  - server_owner: minor_idとmajor_idの組
    - major_idはサーバを識別する文字列。一致すれば同じサーバ
    - minor_idは数値で、これも一致すればセッションを共有できる

Linuxでは時刻

Linuxではホスト名やIPアドレスを組み合わせる

Linuxでは時刻と通し番号から生成

Linuxではホスト名

Linuxでは常に0

CREATE_SESSION

- 主な引数
  - clientid: EXCHANGE_IDで得たもの
  - sequenceid: 同上
  - flags 以下のほか、RDMA関連
    - FLAG_PERSIST: reply cacheをpersistentにしてほしい
    - FLAG_CONN_BACK_CHAN: このコネクションを、セッションのfore channelかつback channelにしてほしい
  - fore_chan_attrs/back_chann_attrs: 要求/応答の最大バイト数、同時に発行できるCOMPOUND数など
- 主な返り値
  - sessionid: 以後使うセッションのID
  - sequenceid: 引数と同じ値を返す
  - flags: 引数のflagsとANDを取る
  - fore_chan_attrs/back_chan_attrs: 引数の値に対して減らしていい属性は定められている

複数のEXCHANGE_ID/CREATE_SESSIONが、単一コネクションに同時に流れた時に、混ざらないようにする

Linuxではこの2つを必ずセットする

LinuxではFLAG_CONN_BACK_CHANのみサポート

チャネル

- 各セッションは1または2つのコネクションを持つ
  - Fore channel: クライアント⇒サーバ方向のRPC呼び出しと応答
  - Back channel: サーバ⇒クライアント方向のRPC呼び出しと応答 (コールバック)
  - 両者は同じコネクションでもよい
    - LinuxのNFSクライアントでは今のところいつも同じコネクション

セッション

- 以後COMPOUNDの最初のオペレーションはSEQUENCE
  - 引数: sessionid、sequenceid、slotid
  - sessionid: Session ID
  - sequenceid: 各スロットごとに1から始まる番号。スロットの再利用時に1増える
  - slotid: スロットの番号。EXCHANGE_IDで交渉された数のスロットが確保され、それ以上の要求を同時に送ることはできない

これが必ず付くので、パケットダンプの様相はNFSv4とはだいぶ違う

COMPOUND

- NFS v4のCOMPOUND

COMPOUND	tag	minor-version	# ops	PUTFH	OPEN	READ
1	(unused)	0	3	22 xx-xx	18 file.txt	25 …

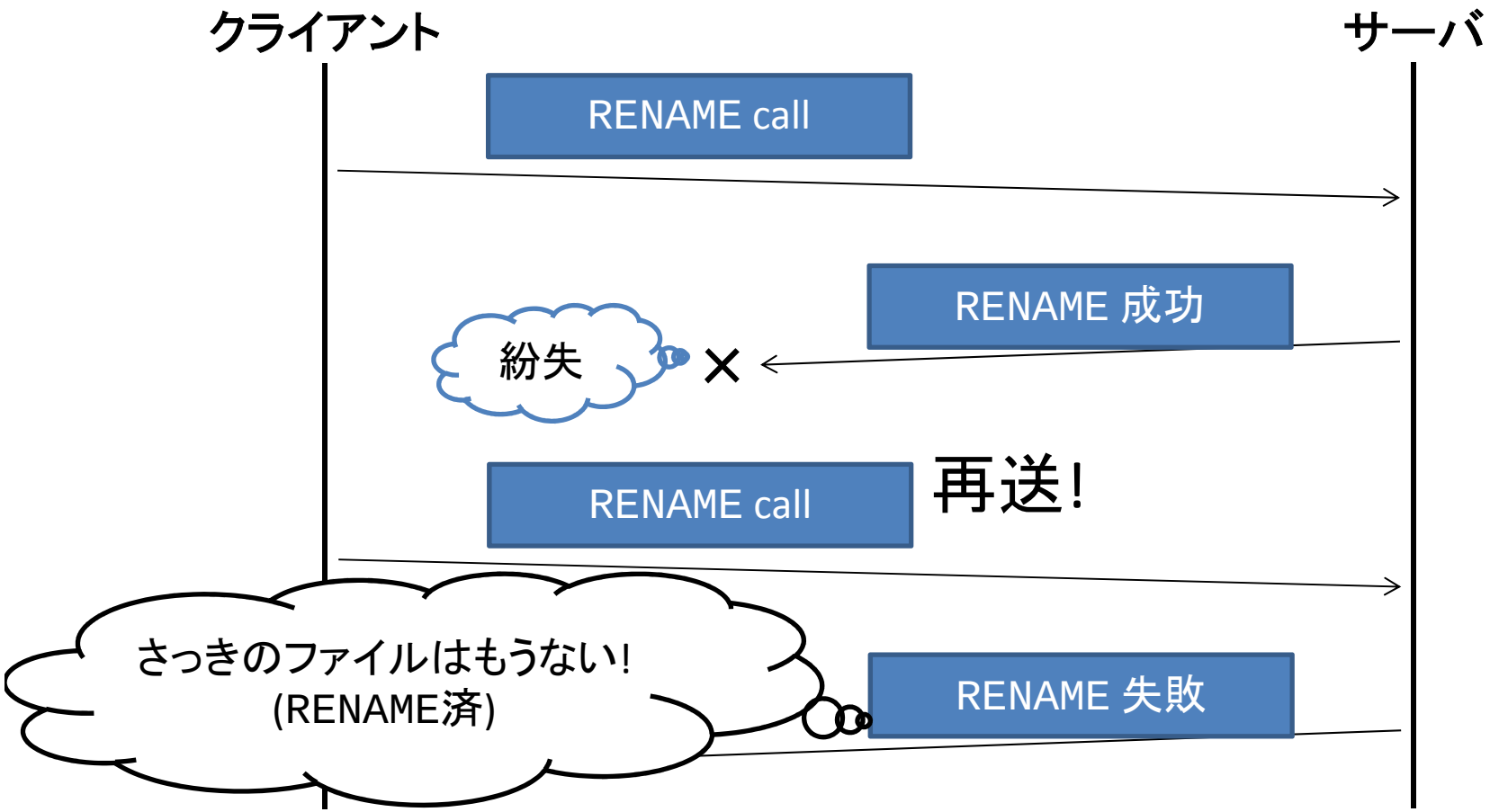
- NFS v4.1のCOMPOUND

COMPOUND	tag	minor-version	# ops	SEQUENCE	PUTFH	…
1	(unused)	1	4	53 …	22 xx-xx	…

SEQUENCE	sessionid	sequenceid	slotid	highest slotid
53	yy-yy-yy	3387	1	3

EOS (Exactly Once Semantics)

- 例: RENAME



EOS (Exactly Once Semantics)

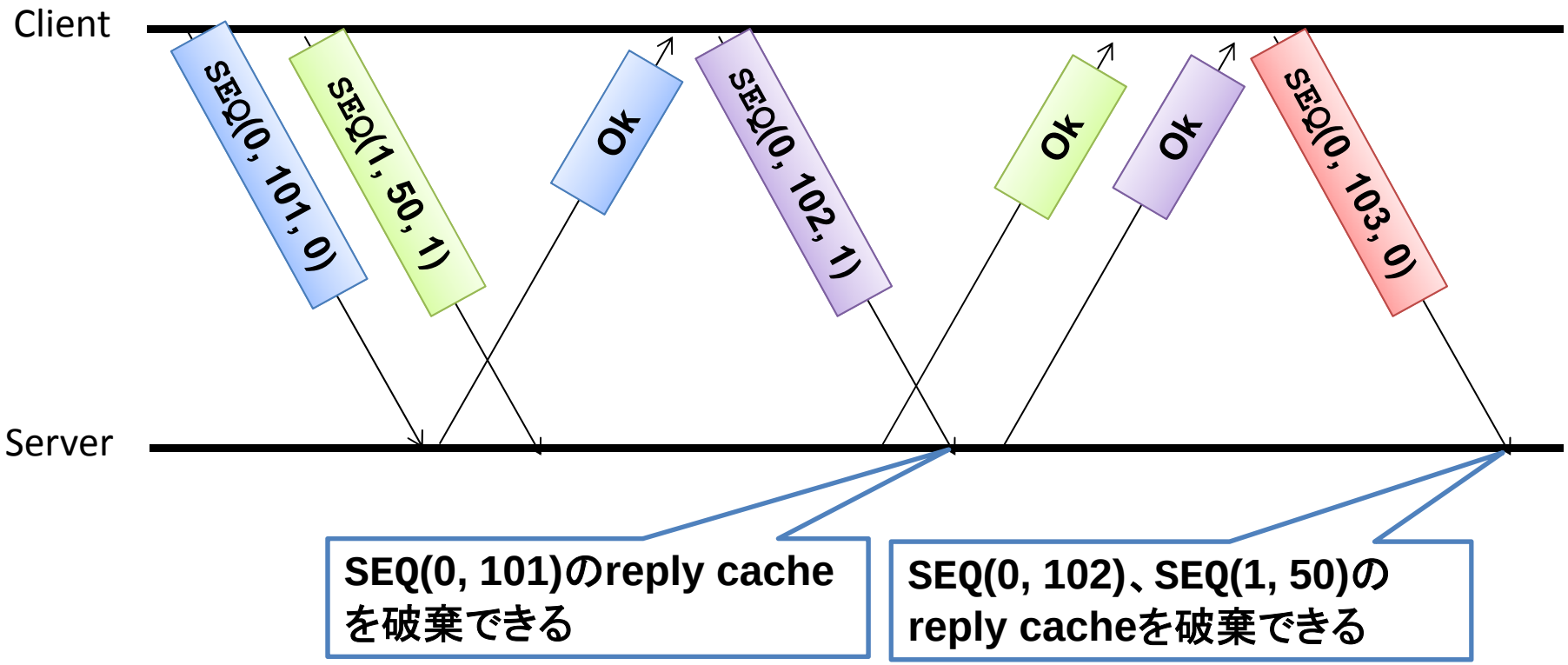
- NFSv3は32bitのRPC XID (Transaction ID)で対応
  - XID (とクライアントのアドレス)が同じであれば、同じ要求の再送とみなす
  - このため、送信した応答をしばらく保存しておく必要がある (リプライキャッシュ)
  - いつまで保存しておけばよい??
  - 32bitで (将来的に) 足りる?
- NFSv4はclient IDごとのsequence IDを利用
  - OPEN、LOCKなどのstateを変えるオペレーションにクライアントが付与
  - client IDと組で利用するため、再起動が検出可能
  - 依然としてリプライキャッシュの保存期間の問題

EOS (Exactly Once Semantics)

- NFSv4.1では、セッションごとのリプライキャッシュに対応
  - セッションごとに、最初の交渉で決定されたスロット数が上限 (Linuxではクライアントの値はsysctl変数で設定、サーバは160が上限。ただしback channelはクライアント側の制限で1)
- スロットが再利用されたことはシーケンスIDで検出可能
  - シーケンスIDが同じ: 再送
  - シーケンスIDが戻った: 古い要求がなぜか来た
  - シーケンスIDが飛んだ: 何かがおかしい

SEQUENCE

- 要求をSEQ(slotid, sequenceid, highest_slotid) として、



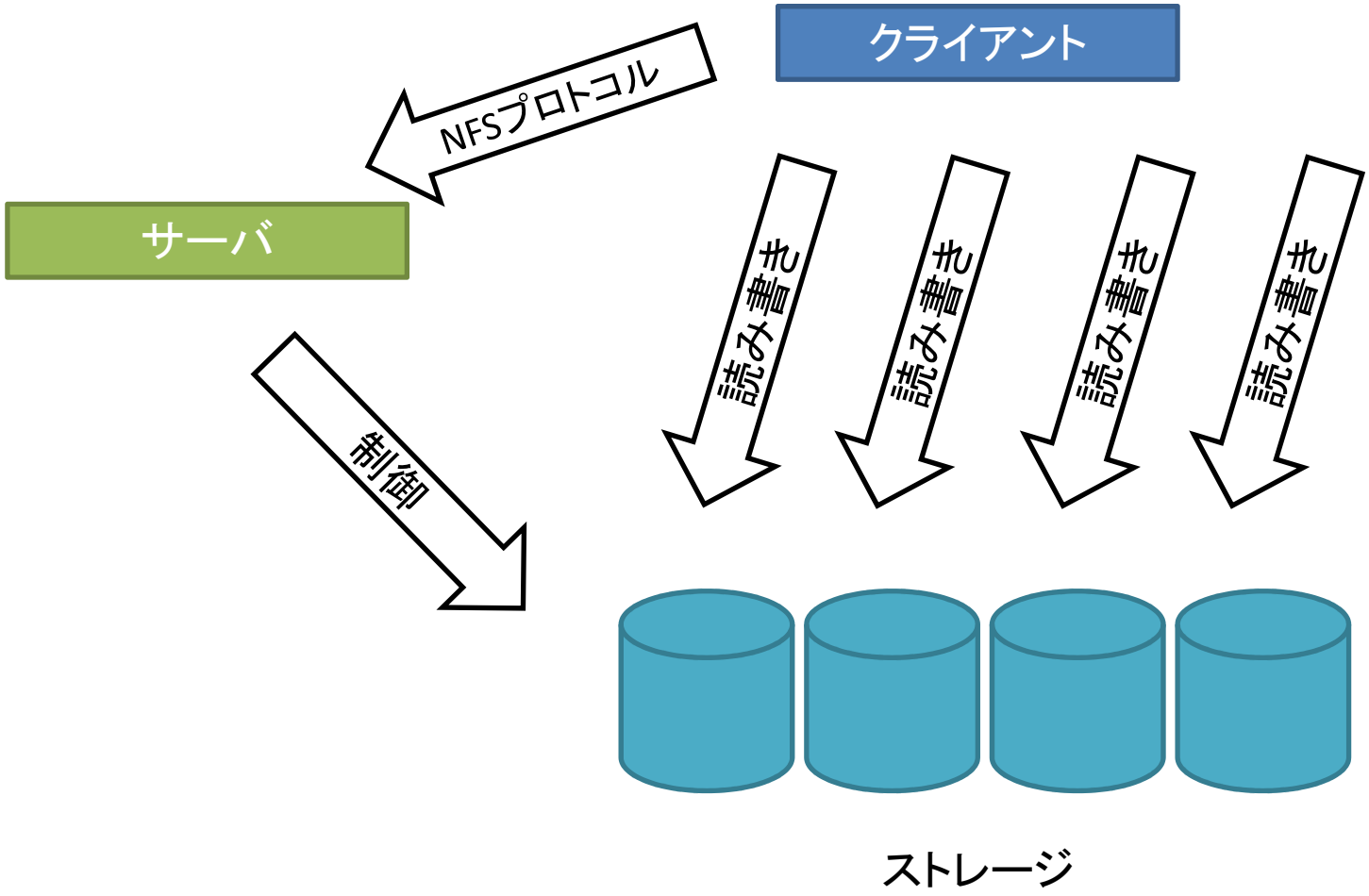
トランキング

- セッショントランキング: Session IDを同じくする複数のコネクションよるトランキング
- Client IDトランキング: Client IDを同じくする複数のコネクションによるトランキング
  - つまり、複数のセッションを用いたトランキング

NFSv4のSETCLIENTID

- Sessionの概念がなかったため、CREATE_SESSIONに相当するものはない。Fore/back channelは分離できない。
- クライアント (client idで識別)、セッション、コネクションの関係が整理されておらず、client idは状態を作るOPENなどのオペレーションが実行されるまで不要。SETCLIENTIDの実行もそれまで遅延できる。

pNFS



「データサーバ」

pNFS

- クライアントはサーバから「レイアウト」を取得
  - 通常のLOOKUP/OPENに次いで、READ/WRITEの代わりに「LAYOUTGET」
- クライアントはストレージと直接やり取りして読み書き
  - ファイルレイアウト
  - オブジェクトレイアウト
  - ブロックレイアウト
- クライアントはサーバに読み書きの終了を宣言
  - LAYOUTCOMMIT (書き込みのみ)、LAYOUTRETURN

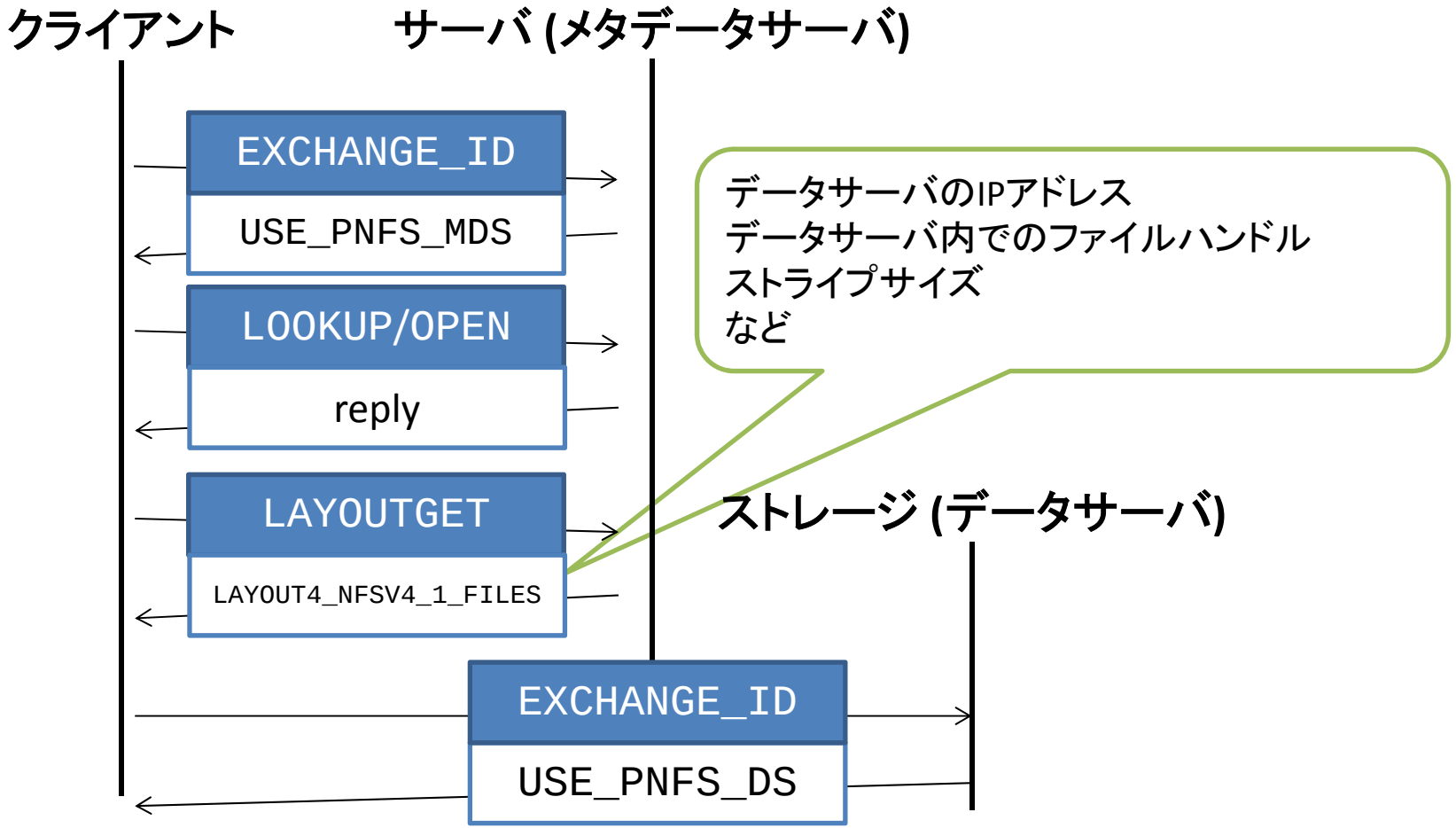
pNFS

- 最初のEXCHANGE_IDで交渉。5通り
  - EXCHGID4_FLAG_USE_PNFS_MDS
  - EXCHGID4_FLAG_USE_PNFS_MDS|EXCHGID4_FLAG_USE_PNFS_DS
  - EXCHGID4_FLAG_USE_PNFS_DS
  - EXCHGID4_FLAG_NON_PNFS
  - EXCHGID4_FLAG_USE_PNFS_DS|EXCHGID4_FLAG_NON_PNFS

MDS: Metadata Server, DS: Data Server

pNFS

- シナリオ例


クライアント
サーバ (メタデータサーバ)
EXCHANGE_ID
USE_PNFS_MDS
LOOKUP/OPEN
reply
LAYOUTGET
LAYOUT4_NFSV4_1_FILES
データサーバのIPアドレス
データサーバ内でのファイルハンドル
ストライプサイズ
など
ストレージ (データサーバ)
EXCHANGE_ID
USE_PNFS_DS

pNFS

	ファイルレイアウト	オブジェクトレイアウト	ブロックレイアウト
ストレージ(例)	(p)NFSサーバ	オブジェクトストレージ	SAN
データパス・プロトコル	NFSv4.1 (以降)	SCSI Object-Based Storage Device Commands (OSD) ANSI INCITS 400-2004	iSCSI、FCP、FCoEなど
レイアウト規定	RFC 5661 (NFSv4.1自体と同じ)	RFC 5665	RFC 5663
制御プロトコル	規定なし		

ファイルの委譲の強化

- NFSv4: OPEN時に委譲できた場合のみ委譲
- NFSv4.1: 委譲できるようになったら知らせられる
  - OPEN時に委譲できなくても、委譲できるようになったらコールバックするよう要求
  - OPEN後にWANT_DELEGATIONを発行し、委譲できるようになったらコールバックするよう要求
- Linuxでは (サーバ、クライアントとも) 未実装

マルチサーバ構成

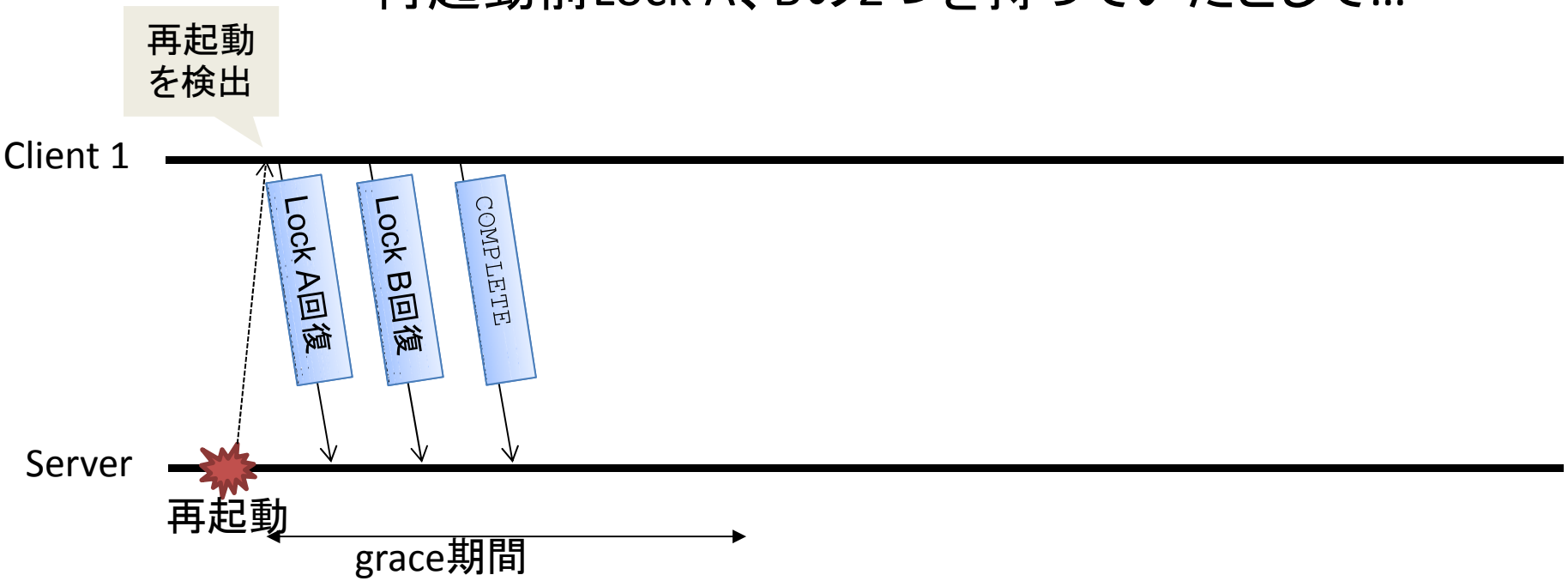
- NFS v4でも、referralにより他のサーバにリダイレクトすることができた (レプリケーション、マイグレーション)
  - `fs_locations`属性: サーバのアドレス (DNS名またはIPアドレス) とルートパスの組を複数並べられる
- ⇒FedFS (Federated FS)
- NFS v4.1のreferralではサーバのアドレスのほか、各サーバの優先順位 (読み書き別々に)、有効期限などが指定できる (`fs_locations_info`属性)
- その他: サーバのminor id、サーバスコープ…

ロックの拡張

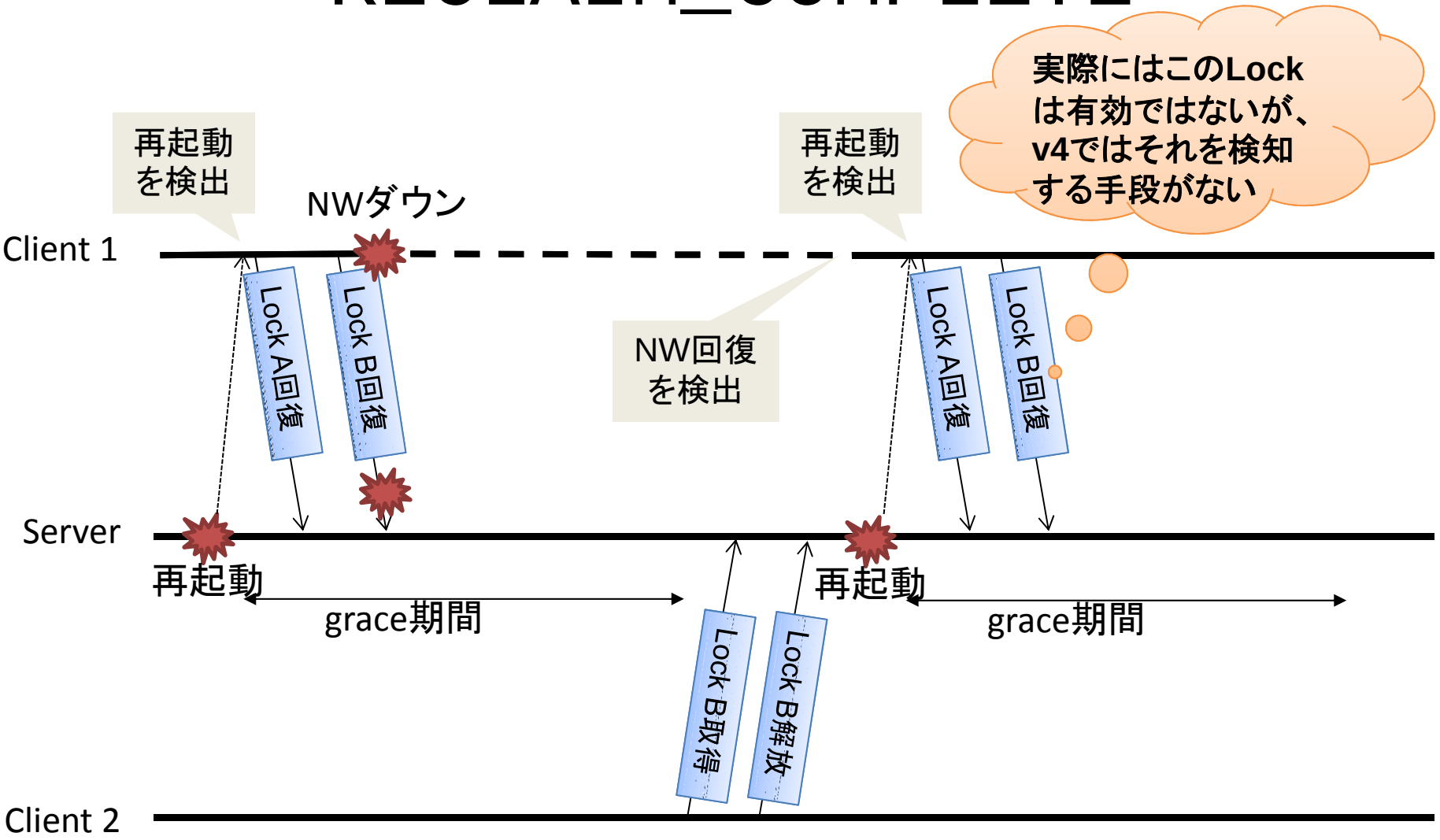
- RECLAIM_COMPLETEの導入
  - RHEL6でも実装済
  - (次ページ)
- CB_NOTIFY_LOCKの導入
  - RHEL7でも未実装と思われる
  - ロック要求時に衝突のため拒否された場合、ロックできるようになったらコールバックする機能
    - NFS v4ではポーリングする必要があった

RECLAIM_COMPLETE

- サーバ再起動後のgrace期間におけるロック回復がすべて終わったことを明示する
- 回復すべきロックを持たないクライアントは、セッション回復に続いて即座に発行
- 再起動前Lock A、Bの2つを持っていたとして…



RECLAIM_COMPLETE



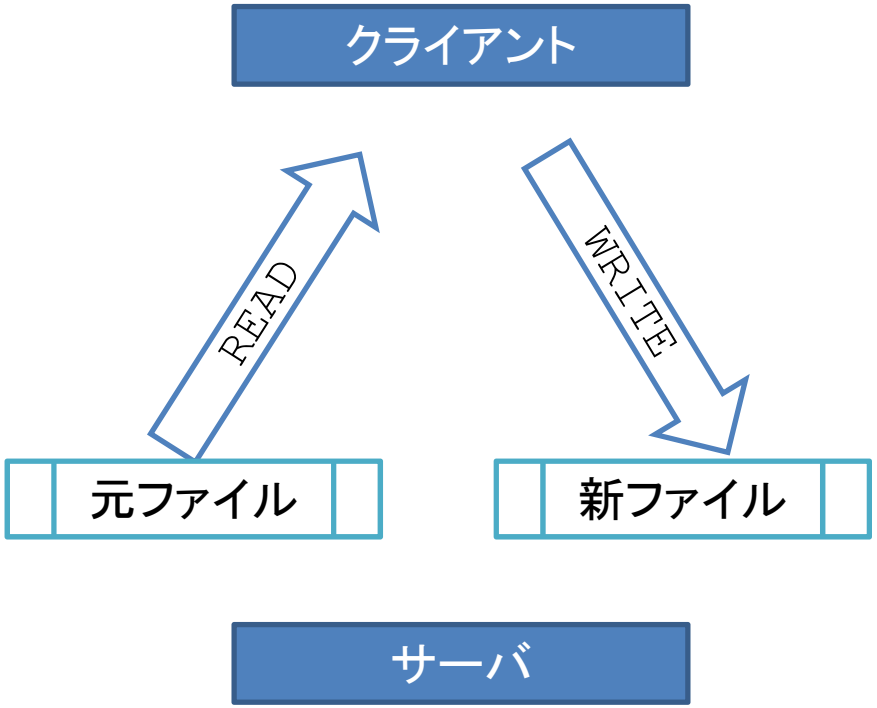
- サーバが、回復中 (RECLAIM_COMPLETE前) のクライアントをディスクに覚えておけば解決できる

NFSv4.2

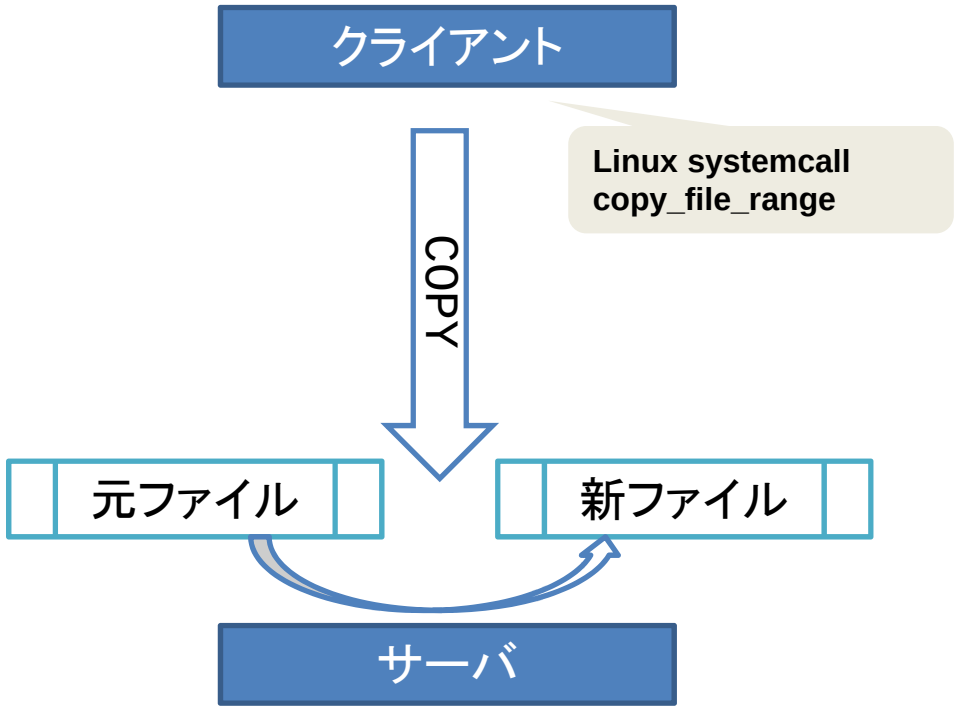
- RFC 7862 (2016年11月)
- 主な新機能
  - サーバサイドコピー (Linux copy_range(2))
  - I/Oアドバイス
  - 穴あきファイル
  - Labeled NFS (SELinuxラベルのサポート)
- NFS v4.1と互換性あり (すべての新機能はオプション)

サーバサイドコピー

- 従来

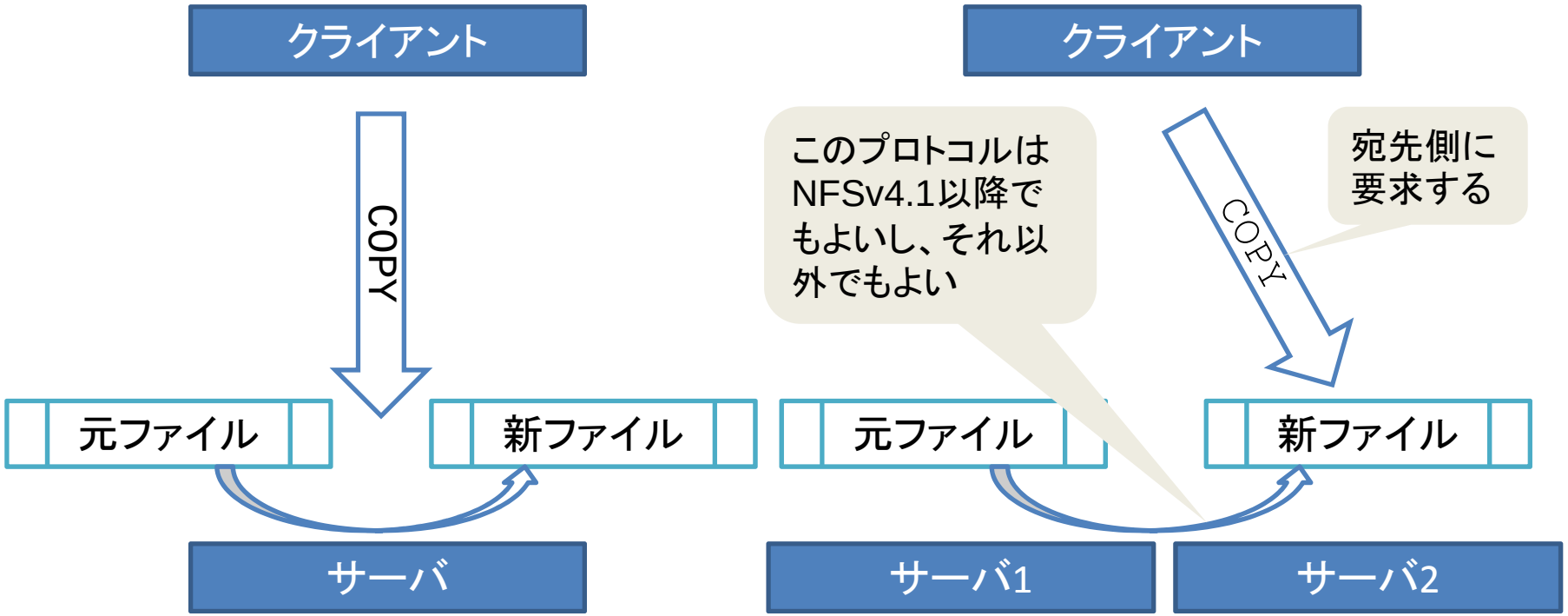


- サーバサイドコピー
  - トラヒックの削減に効果大



サーバサイドコピー

- サーバサイドコピー
- サーバをまたいだコピーにも対応



I/Oアドバイス

- POSIX fadvise(2)相当
  - アプリケーションがfadvise(2)を発行、クライアントがサーバにIO_ADVISEを発行
  - SEQUENTIAL、SEQUENTIAL_BACKWORDS、RANDOM、WILLNEED、DONTNEED、NOREUSEなど

穴あきファイル

- 連続した0と「穴」を区別できるREAD_PLUSとWRITE_PLUSを追加
  - 区別できなくてもよい (READ、WRITEと同じ)
  - 特定の (アプリケーションが指定した) パターンで初期化することも可能 (Application Data Hole)
- 領域の予約が可能

Linuxでの実装状況 (クライアント)

2.6.30	NFSv4.1のコード統合
2.6.36	NFSv4.1のコードにあった「開発者限定」のコメント削除
3.7	NFSv4.1が「実験的」扱いではなくなる
3.10	NFSv4.2のコード統合 (実際にはLabeled NFSのみ)

Linuxでの実装状況 (サーバ)

2.6.26	NFSv4のminorversionが0でない要求をはじくようになる
2.6.30	NFSv4.1のコード統合
3.8	NFSv4.1を含むNFSv4が「実験的」扱いではなくなる
3.10	NFSv4.2のコード統合 (実際には新機能は追加されていない)
3.11	NFSv4.1のサポートが既定で有効に
3.19	NFSv4.2のサポートが既定で有効に

使い方 (クライアント)

- 機能が有効かどうかを直接知る方法はないので…

```
# grep NFS_V4_1 /boot/config-`uname –r`
CONFIG_NFS_V4_1=y
# mount –t nfs –o vers=4,minorversion=1 sv:/fs
/fs
# mount | tail -1
sv:/fs on /fs type nfs
(rw,vers=4,minorversion=1,
addr=xxx.xxx.xxx.xxx,clientaddr=xxx.xxx.xxx.xx
x)
#
```

使い方 (サーバ)

- 既定では無効かもしれない

```
# cat /proc/fs/nfsd/versions
+2 +3 +4 -4.1
# echo +4.1 >/proc/fs/nfsd/versions
-bash: echo: write error: Device or resource busy
# service nfs stop
(略)
# echo +4.1 >/proc/fs/nfsd/versions
# cat /proc/fs/nfsd/versions
+2 +3 +4 +4.1
# service nfs start
(略)
```

Ubuntuではサービス名はnfs-kernel-server
stopで/proc/fs/nfsdがumountされてしまうので手でmountする

Red Hat Enterprise Linuxでのサポート状況

6.0	`mount -o vers=4,minorversion=1`ができる (サポート状況は不明) このときセッションが使われる
	`/proc/fs/nfsd/versions`に「`-4.1`」の文字 有効にすることも可能 (サポート状況は不明)
6.2	クライアントのpNFSのうち、ファイル・レイアウトが technology previewに
6.4	クライアントのファイル・レイアウトのpNFSをサポート
7	クライアントのpNFSのうち、ブロック・レイアウトをサポート サーバの4.1サポートが既定で有効に
7.3	クライアントのpNFSのうち、SCSI (オブジェクト) レイアウトがtechnology previewに サーバの4.2サポートが既定で有効に