ソフトウェアアーキテクチャ
第11回　分散ファイルシステム

環境情報学部
萩野　達也

スライドURL

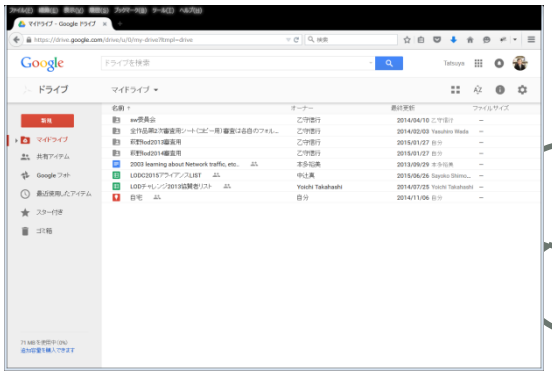
https://vu5.sfc.keio.ac.jp/slide/

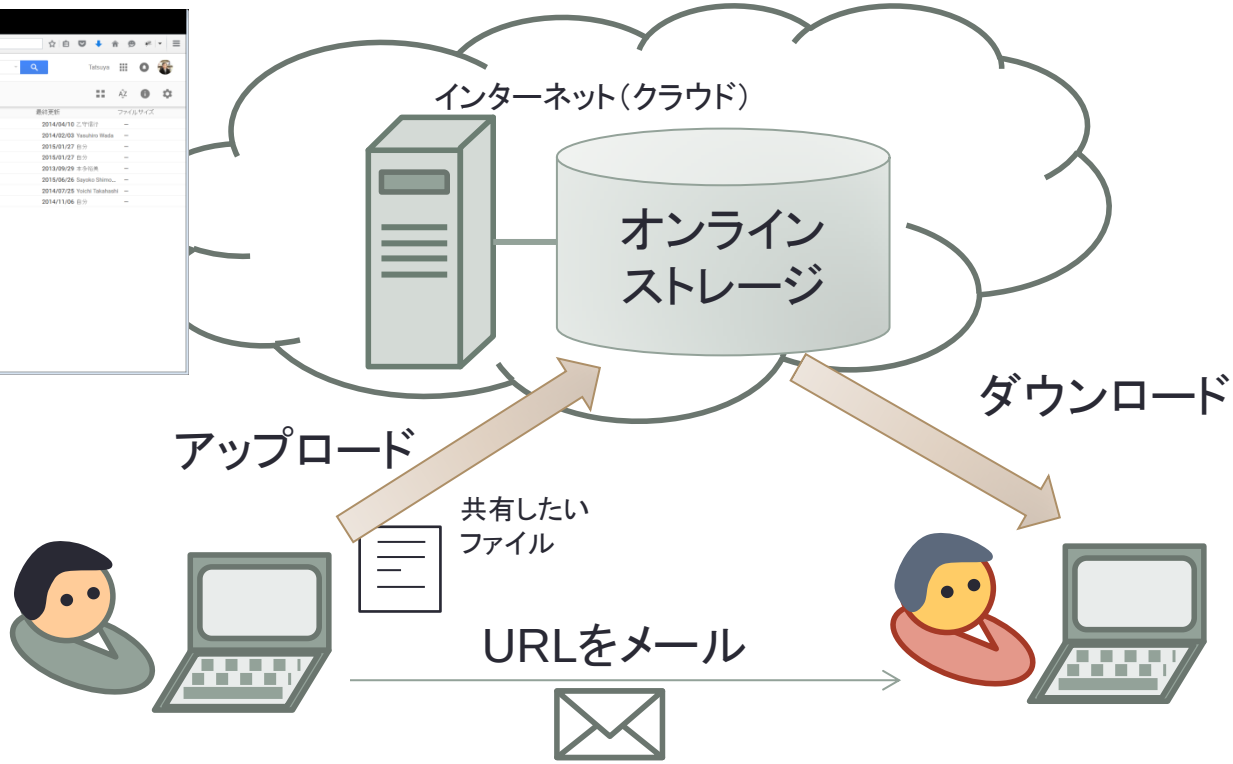
ファイル共有

- オンラインストレージによるファイル共有
  - Webサイトを利用してファイルのダウンロード・アップロードを行う
  - 専用のクライアントソフトがある場合もある
  - Dropbox, Googleドライブ, Skyドライブなど
- アプリケーションによるファイル共有
  - FTPなどでファイルをダウンロード・アップロードする
  - P2Pファイル共有ソフトの利用
  - グループウェア内に組み込まれることもある
  - バージョン管理を行う場合もある
- OSによるファイル共有
  - ローカルファイルと同じように共有ファイルを操作できる
  - アクセス透過性

オンラインストレージ

- ネット（クラウド）上にファイルを置いておく
  - Webインターフェイスを使って操作
  - 専用アプリによりローカルフォルダの自動保存が可能
  - アカウントあるいはURLによって他の人とファイルを共有可能

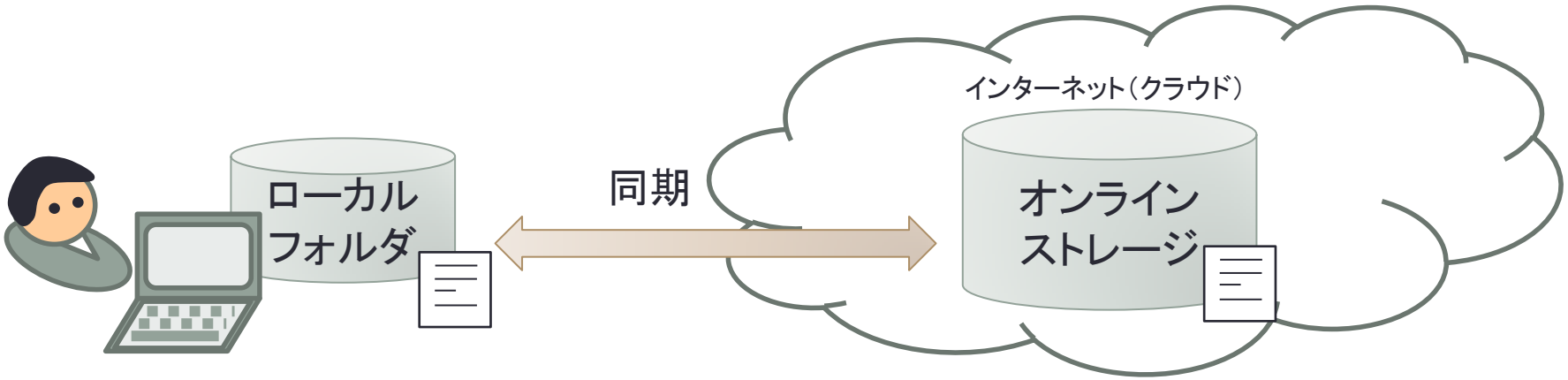




フォルダの自動同期

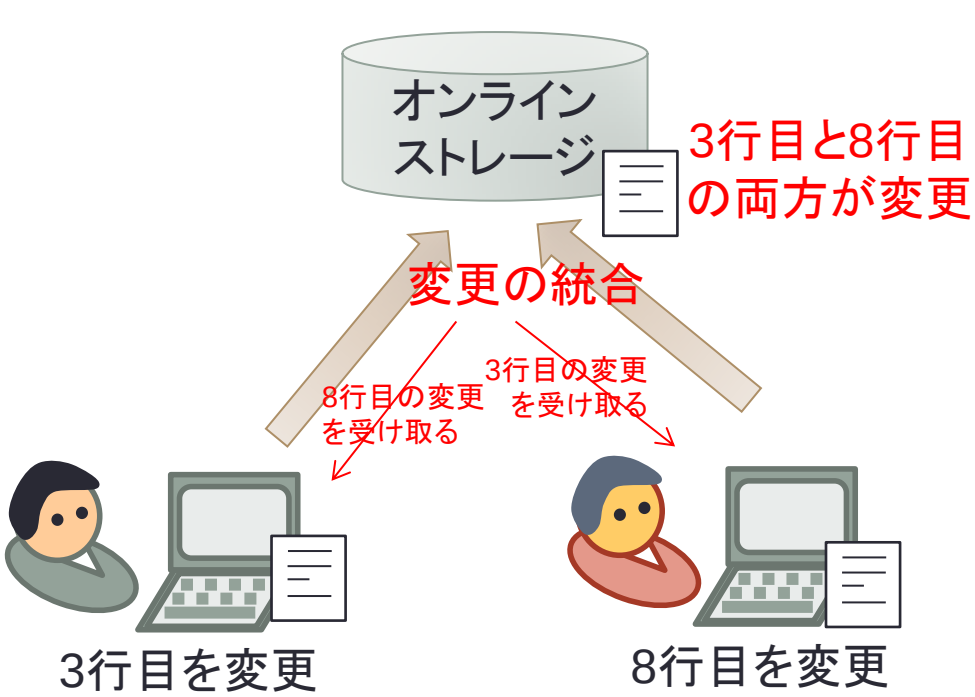
- オンラインストレージのアプリによる自動同期
  - フォルダの内容を自動的にオンラインストレージにアップロード
  - 変更があった場合には，自動的にダウンロード

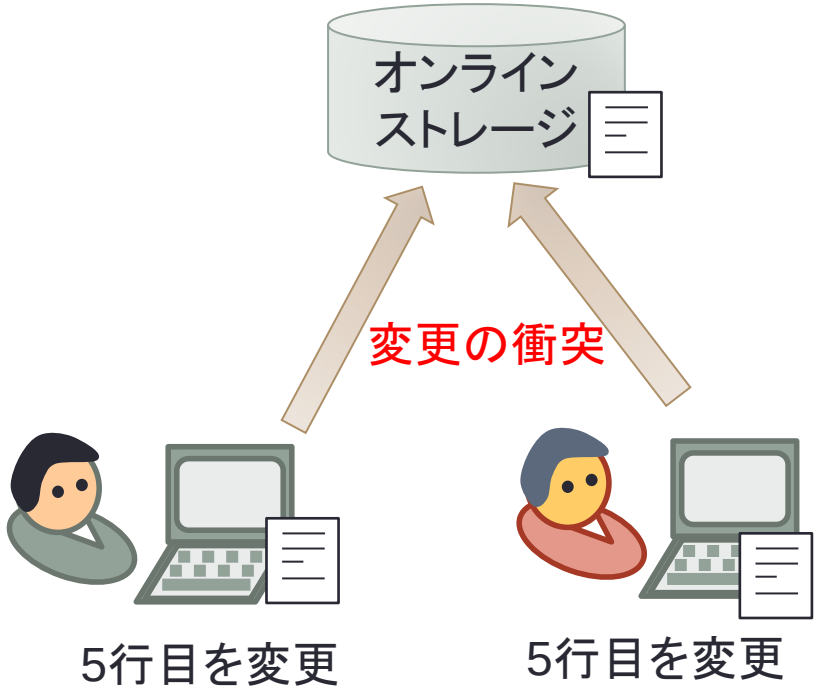
- 仕組み
  - 定期的にフォルダの内容とオンラインストレージの内容を比較
  - 更新があれば，それを反映させる
  - インターネットに接続されていないときには変更は反映されない
  - インターネットに接続した時にまとめて同期を取る



編集内容の統合

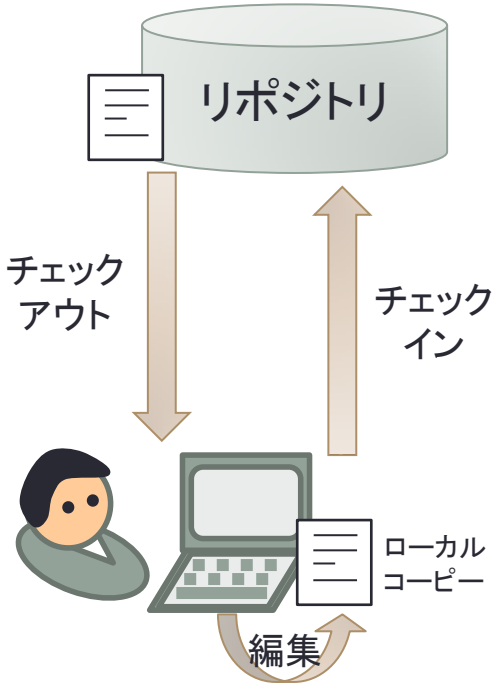
- 複数人でファイルを共有するときの問題
  - 複数人が同時に同じファイルを書き換えるかもしれない
  - 別々の行を書き換えた場合
    - 書き換え内容を統合する必要がある
    - 自動的に行うことが可能
  - 同じ行を書き換えた場合
    - 変更内容が同じなら問題ない
    - 異なる場合は衝突となり手動で結合する必要がある

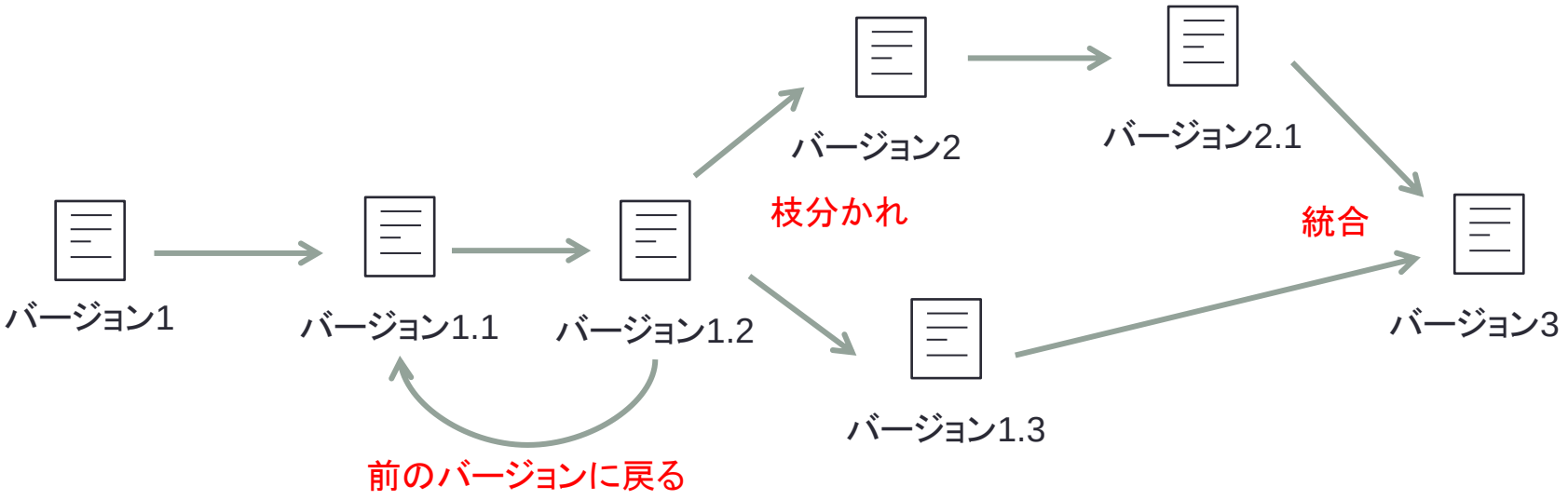




バージョン管理システム

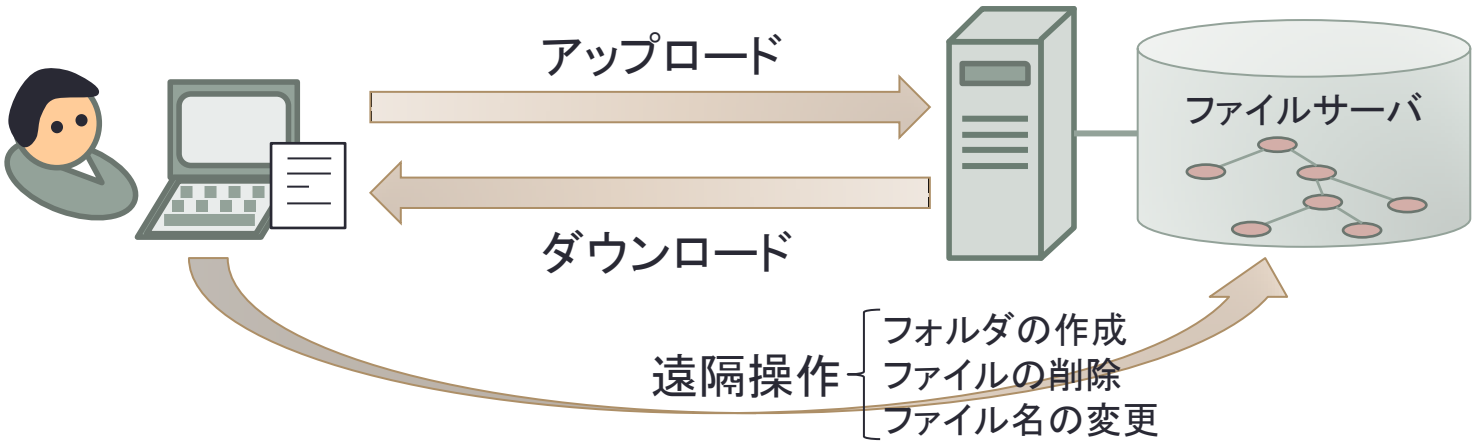
- ファイルの更新をバージョン管理する
  - 更新を戻したり，枝分かれを作ることもできる
  - 大規模なプログラムの開発の時に特に便利
  - バージョン管理システム
    - RCS (Revision Control System)
    - CVS (Concurrent Versions System)
    - Subversion (svn)
    - git





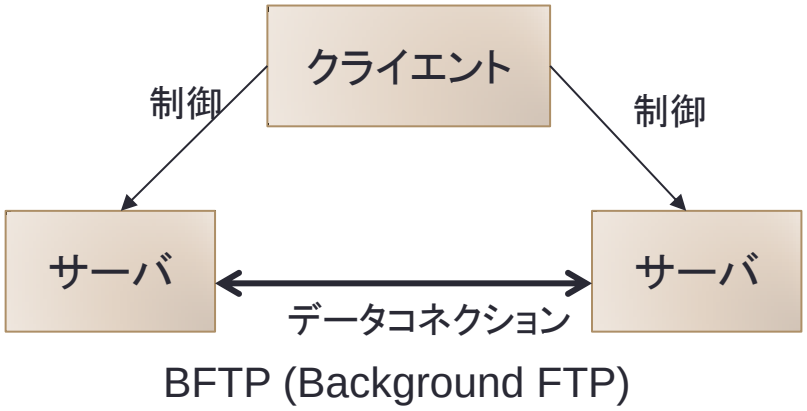
アプリケーションによるファイル共有

- ファイル転送プログラムを使う
  - サーバからファイルを取ってきて，編集し，戻す
  - ファイル転送プロトコル
    - FTP (File Transfer Protocol)
      - 遠隔ファイル操作も可能
    - HTTP (Hyper Text Transfer Protocol)
      - 本来はWebページを取得するためのプロトコル
      - WebDAVによって遠隔ファイル操作も可能

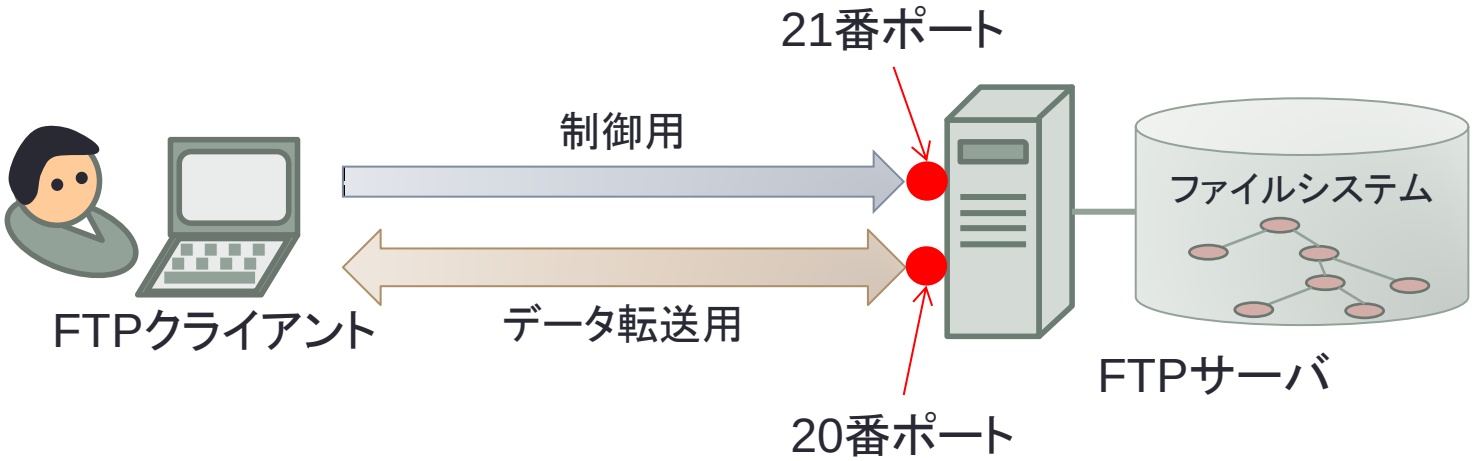


FTP

- File Transfer Protocol
  - インターネットの初期から存在する
  - TCPポート20と21
- クライアントサーバモデル
- 2本のコネクションを使用
  - FTPコマンドと応答用 （制御コネクション）
  - データ転送用 （データコネクション）



BFTP (Background FTP)



データ転送機能

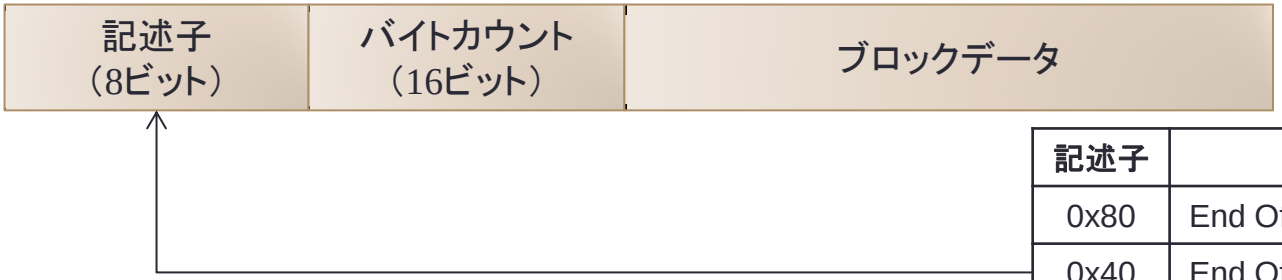
- データタイプ
  - ASCII
    - <cr><lf>により行の終わりを表す
  - IMAGEまたはBINARY
    - レコードのパディング（通常は0）を指定
- ファイルフォーマット
  - ノンプリント
  - Telnetフォーマット制御
    - <CR>, <LF>, <NL>, <FF>
  - キャリッジ制御（ASA）
    - 各行の1文字で制御
      - 空白　1行
      - 0　2行の用紙送り
      - 1　改ページ
      - +　重ね書き
- ファイル構造
  - バイト構造
    - 内部構造が存在せず
    - 連続したデータオクテットがファイルになる
  - レコード構造
    - 連続したいくつかのレコードでファイルが構成される
  - ページ構造
    - ランダムアクセスのファイル

FTP転送モード

- ストリームモード
  - ストリームをそのままデータとして扱う
  - 0xffをESC文字として扱う

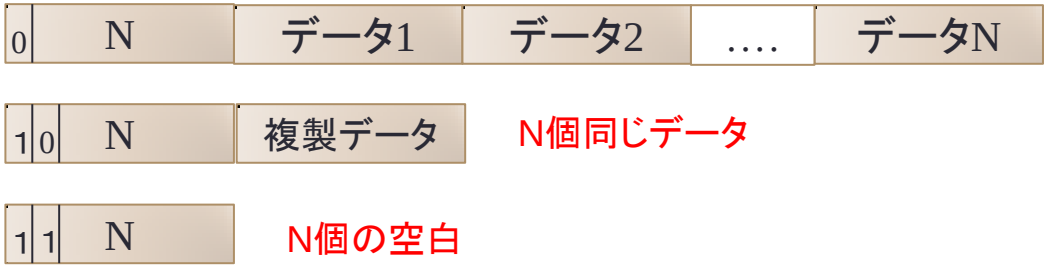
0xff 0x01	End Of Record
0xff 0x02	End Of File
0xff 0x03	EORおよびEOF
0xff 0xff	0xff自身

- ブロックモード



記述子	意味
0x80	End Of Record
0x40	End Of File
0x20	データ中にエラーがある可能性がある
0x10	再スタートマーカー

- 圧縮モード



FTP制御コマンド（1）

- アクセス関連

コマンド	意味
USER	ユーザ名入力
PASS	パスワード入力
ACCT	アカウント情報の入力
CWD	作業ディレクトリの変更
CDUP	親ディレクトリへの移動
SMNT	マウント
REIN	再初期化
LOGOUT	ログアウト

- 転送パラメータ関連

コマンド	意味
PORT	データコネクション用のホストポート
PASV	パッシブ・リスナーのポートを定義
TYPE	データの表現方法を指定 • ASCII • BINARY
STRU	データのファイル構造を示す • F（バイト構造） • R（レコード構造） • P（ページ構造）
MODE	転送モードを示す • S（ストリームモード） • B（ブロックモード） • C（圧縮モード）

FTP制御コマンド(2)

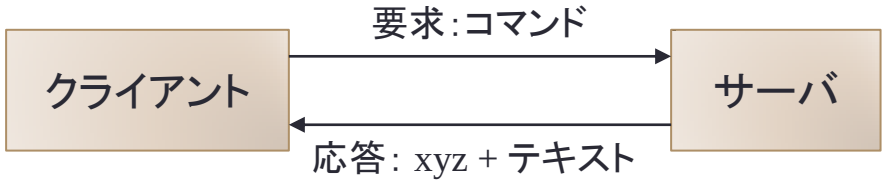
- サービス関連

コマンド	意味
RETR	ファイルを取り出すGET
STOR	ファイルをサーバに送るPUT
STOU	STORだが固有の名前をつける
APPE	ファイルを追加
ALLO	新規ファイルの領域確保
REST	マーカーまでスキップ
RNFR	名前の変更するファイル名
RNTO	変更先のファイル名
ABOR	FTPの中止

コマンド	意味
DELE	ファイルの削除
RMD	ディレクトリの削除
MKD	ディレクトリの作成
PWD	作業ディレクトリの表示
LIST	作業ディレクトリの一覧を表示
NLST	ディレクトリの一覧を転送
SITE	サイト固有のコマンド
SYST	サーバのOSを知る
STAT	サーバの状態
HELP	FTPコマンドの一覧を表示
NOOP	何もしない

FTP応答

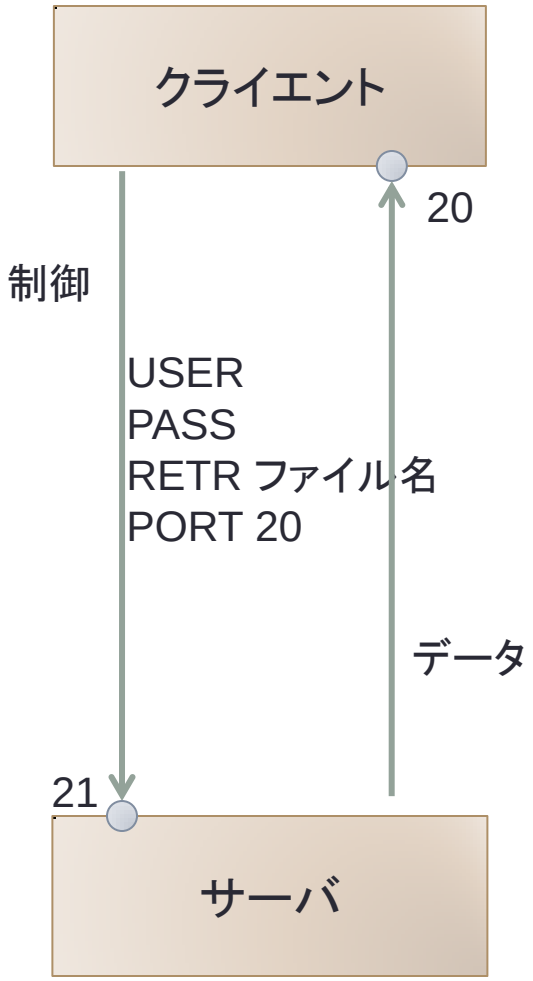
- 形式
  - 3桁のコード+テキスト



- 1桁目
  - 1yz　肯定先行応答
  - 2yz　肯定完了応答
  - 3yz　肯定中間応答
  - 4yz　一時否定完了応答
  - 5yz　永久否定完了応答
- 2桁目
  - x0z　構文
  - x1z　情報
  - x2z　コネクション
  - x3z　認証とアカウント
  - x5z　ファイルシステム

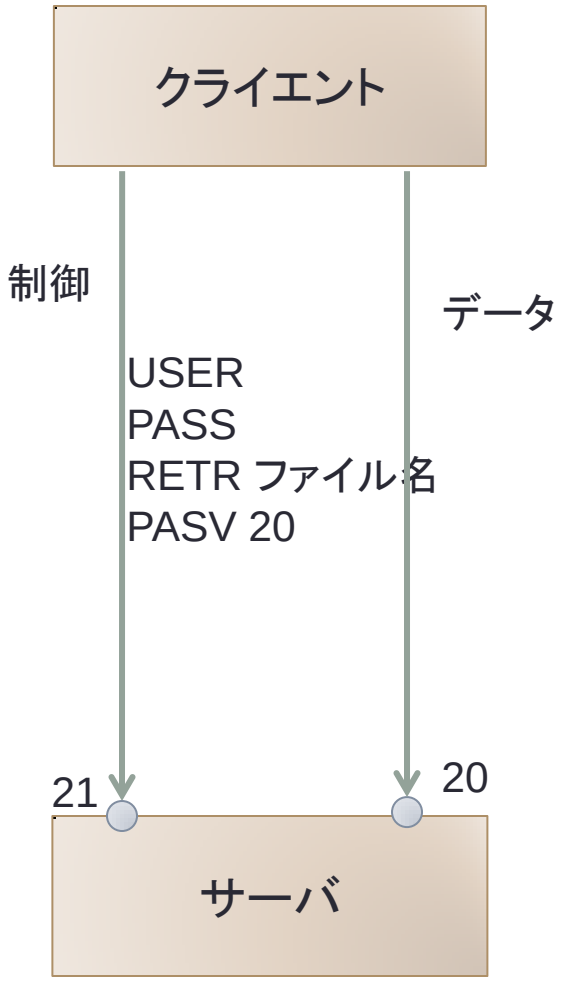
FTPセッションの例

- TCP制御コネクション（ポート21）を開く
- USER，PASSによりログイン
- RETRにより取り出すファイルを指定
- PORTによりデータコネクションポートの指定
- サーバがデータコネクションを開く
- データを転送する
- QUITで終了する



FTPパッシブモードの例

- TCP制御コネクション（ポート21）を開く
- USER，PASSによりログイン
- RETRにより取り出すファイルを指定
- PASVによりデータコネクションポートの指定
- サーバは指定されたポートで待つ
- クライアントがデータコネクションを開く
- データを転送する
- QUITで終了する



FTPについて

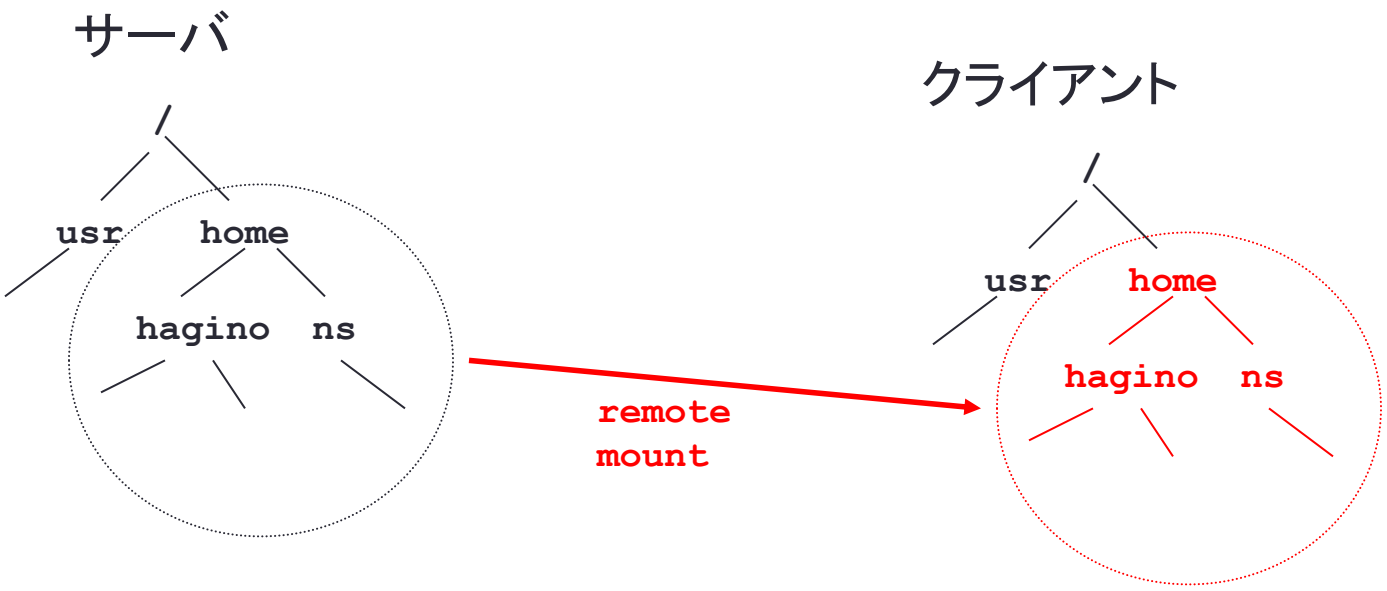
- セキュリティ
  - パスワードが生のまま流れるのでインターネット上での利用には注意が必要
- Anonymous FTP
  - ユーザ名をanonymousあるいはftpとする
  - パスワードは用いず，メールアドレスを入力する（認証されるわけではない）
  - フリーソフトの配布に用いられる
  - HTTPはAnonymous FTPの改良版だともいわれる
  - WebブラウザがFTPクライアントになっていることが多い

OSによるファイル共有システム

- UNIX
  - NFS
    - Network File System
    - Sun Microsystems（現Oracle）が開発
  - AFS
    - Andrew File System
    - CMUが開発
- Windows
  - SMBプロトコル（その拡張CIFS）
    - Server Message Block, Common Internet File System
    - IBMがNetBIOS用に設計
    - マイクロソフトがCIFSとして拡張
- Mac OS
  - AFPプロトコル
    - Apple Filing Protocol
    - AppleTalk上のプロトコルの一つ
    - TCP/IP上でも動く（AFP over TCP）

NFS (Network File System)

- UNIXにおける分散ファイル共有のプロトコル
  - Sun Microsystemsが開発
  - AT&Tが開発したRFS（Remote File Sharing）と一時期張り合うもNFSが生き残る
- 遠隔サーバのファイル木をローカルディスクのようにマウントし利用する
  - サーバはファイル木をexportする
- もともとはUDP上のプロトコル，現在はTCP上でも利用可能
  - ポート2049

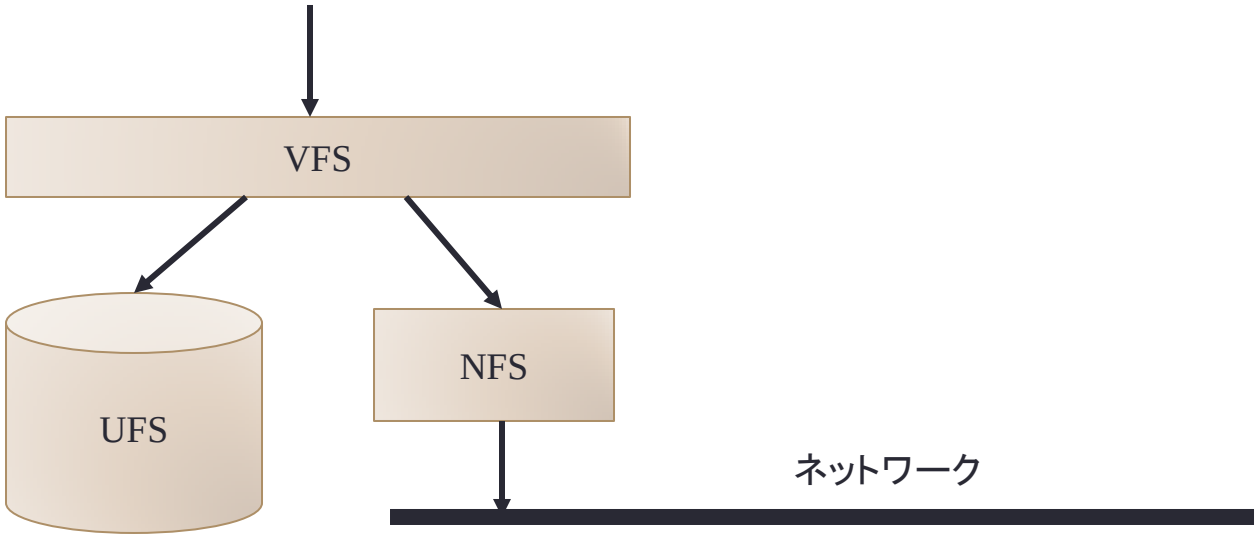


NFSの特徴と問題点

- アクセス透過性を持つ
  - 分散ファイルシステムだから当然
- 位置透過性を持つ
  - 名前空間はローカル
  - 任意の場所にマウント
- 障害透過性を持つ
  - 状態を保持しない
  - ほとんどの処理は再試行可能
- 性能透過性の確保
  - クライアントはデータをキャッシュする
- 複製透過性がない
  - サーバの複製は作ることができない
- 並行透過性がない
  - ファイルをロックすることができない
- 規模透過性はあまり考えない
  - 組織内での共有が主
  - ユーザを統一する必要がある
  - ユーザ名の共有はNIS(YP)などを利用

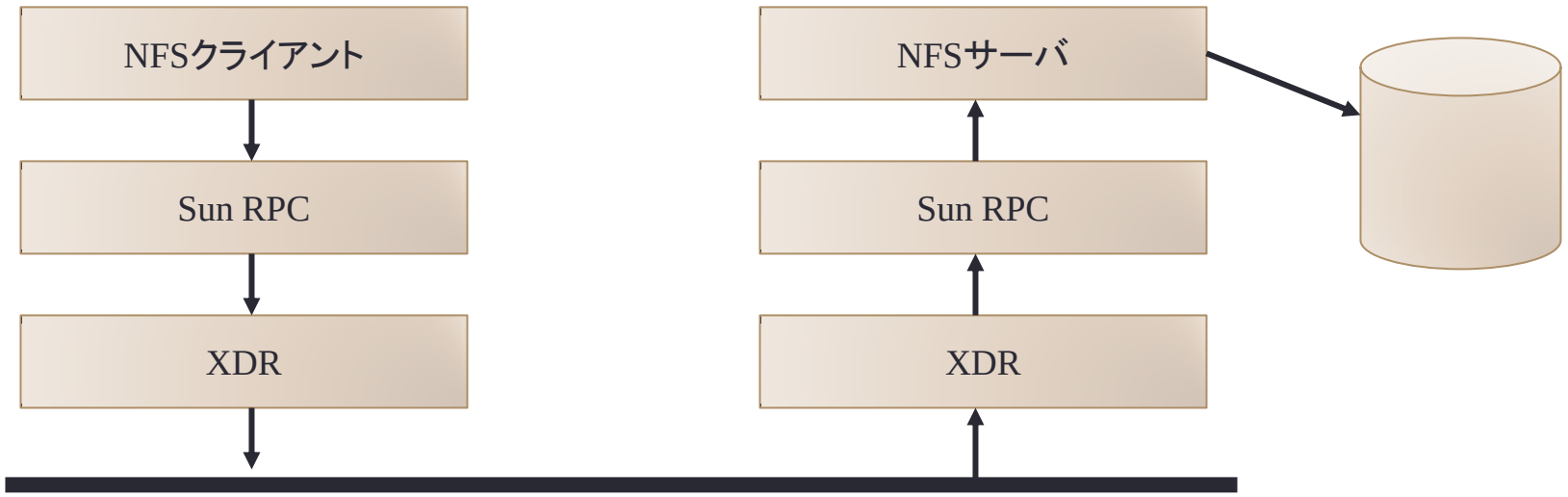
NFSの実装 (1)

- 仮想ファイルシステム(VFS)によりローカルと遠隔を切り分ける



NFSの実装 (2)

- Sun RPCを利用
  - XDR (External Data Representation)を使いメッセージを表現



NFSプロトコル・インターフェイス

- lookup(dirfh, name)
  → fh, attr
- create(dirfh, name, attr)
  → newfh, attr
- remove(dirfh, name)
  → status
- getattr(fh)
  → attr
- setattr(fh, attr)
  → attr
- read(fh, offset, count)
  → attr,data
- write(fh, offset, count, data)
  → attr
- rename(dirfh, name, todirfh, toname)
  → status
- link(newdirfh, newname, dirfh, name)
  → status
- symlink(newdrfh, newname, string)
  → status
- readlink(fh)
  → string
- mkdir(dirfh, name, attr)
  → newfh, attr
- readdir(dirfh, cookie, count)
  → entries
- rmdir(dirfh, name)
  → status
- statfs(fh)
  → fsstatus

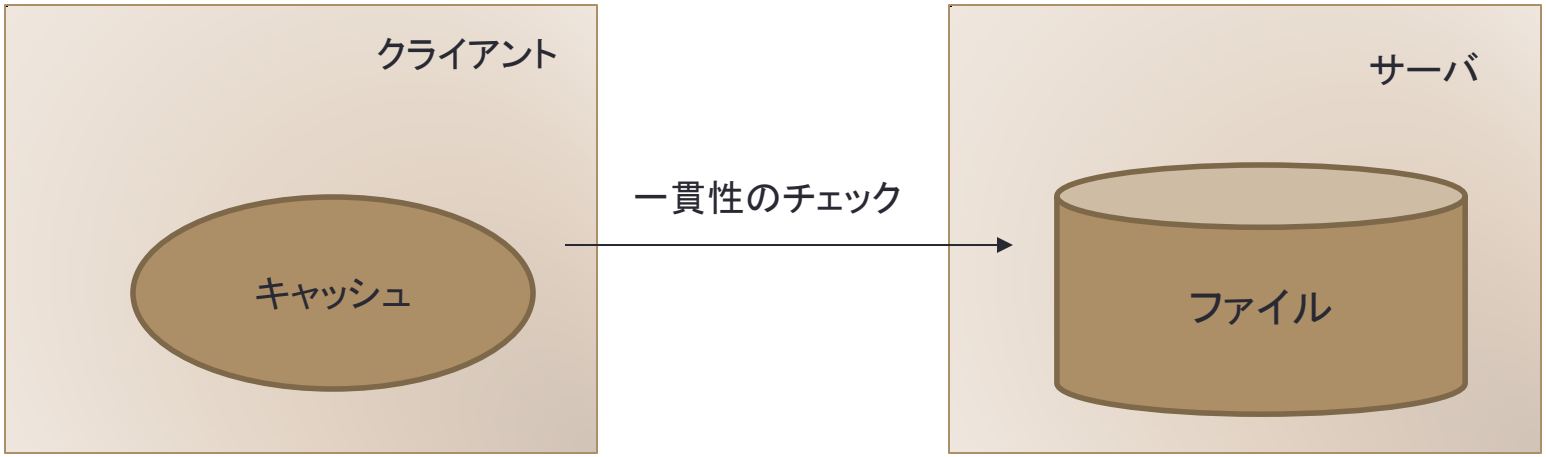
VFS

- Virtual File System
- 従来まではi-nodeによりファイルを識別してきた
  - i-node = index node
- NFSではファイルハンドルにより識別する
  - ファイルシステム識別子
  - ファイルのi-node番号
  - i-node世代番号
- ファイルハンドルは不透明
  - クライアントは中身を見て解釈してはいけない
  - ケイパビリティのようなもの

ファイルシステム識別子	i-node番号	i-node世代番号

NFSキャッシュ

- NFSクライアントとはファイルブロック単位でキャッシュする
  - キャッシュの一貫性は頻繁にチェックする必要がある
  - サーバからは教えてくれない（サーバは状態を持たない）
  - 通常ファイルでは3秒ごとにチェック
  - ディレクトリでは30秒ごとにチェック



ステートレス vs ステートフル

- NFSサーバはクライアントのステート(状態)を知らない
  - 故障透過性が高い
  - キャッシュを活用することにより性能を確保
  - NFS上のファイルをlockできない
  - 遠隔ではUNIXセマンティックを保障しない

- ステートフル
  - サーバが故障のためリスタートするとクライアントのすべての状態を把握しなおす必要がある
  - lockをサポート可能
  - 複製などにも対応可能
  - RFSはUNIXセマンティックを遠隔でも保障しようとした

AFS（Andrew File System）

- CMUで開発
  - キャンパス内の5000台以上のワークステーションによる共有を想定

- 特徴
  - ファイル全体をキャッシュする
  - kerberosによる認証
  - ACL (Access Control List)の設定
  - 組織を越えた共有が可能

AFSのシナリオ

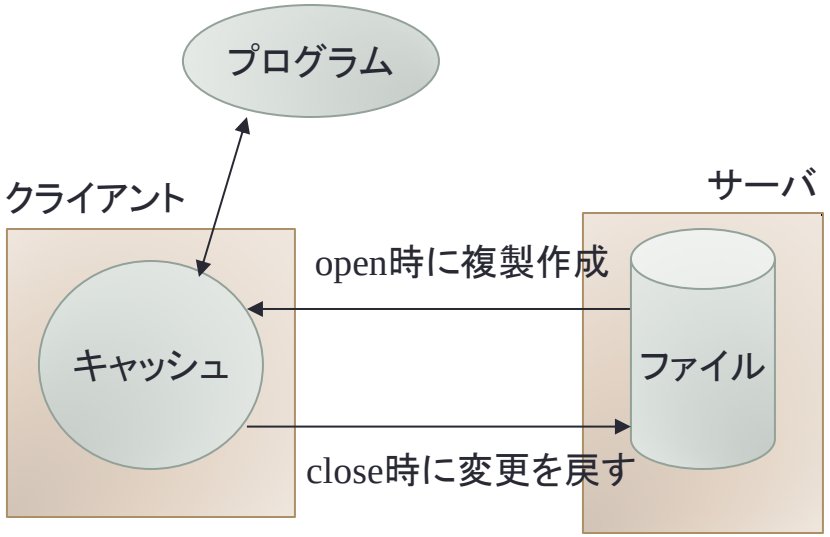
1. クライアントがファイルをopen

2. サーバから最新の複製をコピー
   - 自分が最新の複製を持っていればコピーしない
   - サーバは複製を管理

3. ローカルに変更を行う

4. クライアントがclose

5. サーバに変更部分を送り返す
   - 複製を保持しておく
   - 最後に変更したファイルで上書き



NFS vs AFS

	NFS	AFS
特徴	• ステートレスによって耐故障性が高い • UNIX以外でも広く実装されている	• サーバは複製を管理する • 複製を作ることができる • 組織を超えた共有が可能 • Kerberosによる認証 • ファイルはディレクトリ単位で管理 • ディレクトリ単位でACLを細かく設定可能 • モバイル環境に適用するCodaに発展
問題点	• 複製を作ることができない • 組織内での共有に限定 • ユーザ認証はOSに任せる • ファイルはアクセス制御はグループ1つだけ	• 耐故障性はNFSに比べて低い • OSによらない認証のため，独自のコマンドでACLは管理しなくてはならない

まとめ

- オンラインストレージによるファイル共有
  - Webインターフェイス
  - Dropbox, Google Drive, Sky Drive, iCloud , ownCloudなど
  - 自動同期アプリケーションあり
- アプリケーションによるファイル共有
  - FTP
- OSによるファイル共有
  - 分散ファイルシステム
  - NFS
  - AFS