社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2003－CSEC－23 (10)
2003／12／19

冗長度とアクセス制御を考慮した分散ストレージシステムの一方式

平野 仁之† 中村 康弘‡

防衛大学校情報工学科 〒239-8686 神奈川県横須賀市走水 1-10-20

E-mail: †g42034@nda.ac.jp, ‡yas@nda.ac.jp

あらまし 比較的小規模な組織内で共有ファイルを分散保管する一方式として，分散ストレージシステム[3]が提案されている．この方式は，共有ファイルを複数のセグメントに分割し，各セグメントにヘッダ情報を付加してネットワーク内の複数端末に分散保管する手法である．本稿ではさらに，各セグメントに付加するヘッダ情報を利用して，共有ファイルに公開・非公開の属性を与え，これを用いたアクセス制御の方法を提案する．また，稼動していない端末が存在するなどの理由でいくつかのセグメントが消失しても，元ファイルを復元可能とするため，ファイル分割時に冗長符号を利用する手法を提案し，冗長性の点で特に考慮すべき事項について示す．本手法を用いれば，ネットワーク内の各端末のストレージ使用率が均等化され，ストレージの有効活用が図れるとともに，個別端末の稼動状況によらず，高い復元性を保った分散ストレージシステムを構築できる．

キーワード アクセス制御，分散ストレージ，冗長符号

A study of a Distributed Storage System using redundancy storage for the access control

Hitoshi HIRANO† and Yasuhiro NAKAMURA‡

Dept. of Computer Science, National Defense Academy

1-10-20 Hashirimizu, Yokosuka-shi, Kanagawa,

239-8686 Japan

E-mail: †g42034@nda.ac.jp, ‡yas@nda.ac.jp

Abstract Distributed Storage System [3] is proposed as a one of network file sharing system for a small peer to peer network environment. The method divides a target file into small segments, add some header information to each segment and distribute them to the network. This paper proposes additional scheme that enables to control the accessibility by expanding the header attribute of segments. To prevent elimination of a segment, error correction code is applied to data segments. Moreover, this paper indicates a special point that must be considered when implementing a fault tolerance capability to this system.

This method equalizes the deviation of each storage usage rate on the network, and as results, implementation of Distributed Storage System with fault tolerance and publicity control will be achieved independent of other storages.

Keyword Access Control, Distributed Storage System, Redundancy code

1. はじめに

近年，FTTH や ADSL などの高速常時接続環境の普及により，ネットワークを経由したシームレスな情報交換の自由度が増大している．また，大容量ストレージの低価格化により，音楽や映像などの比較的容量の大きいコンテンツがエンドユーザ間で共有・交換されるようになってきた．同時に，様々な規模の組織内ネットワークでファイル共有が活用されており，その需要は高まっている．

ネットワークを利用したファイル共有システムについては既に数多くの方法が知られており，それぞれの特性を考慮して活用されている．オペレーティングシステムが統一されている環境下では NFS, Netware や LAN Manager などの既存のファイル共有システムを利用すれば，利用者の一元管理と様々なアクセス制御が可能となる．逆に，オペレーティングシステムに依存しない方法は，サーバクライアント方式による FTP サーバや HTTP サーバなどを利用する方法と，Napster, Gnutella[1]や Freenet[2]に代表される P2P プロトコルを利用する方法に大別できる．前者はサーバ側での利用者の一元管理が必要なのに対し，後者は元々利用者管理という概念が存在せず，ネットワーク上で自由なファイル交換が可能な環境を提供している．

ここで，マルチプラットフォームから構成される比較的小規模な組織内におけるネットワークファイル共有システムの導入について考える．オペレーティングシステムの統一が難しい状況下で，ある程度のアクセス制御を考慮しつつファイル共有機能を実現しようとするとサーバクライアント方式を選択せざるを得ない．しかしながら，サーバクライアント方式によるファイル共有ではネットワーク負荷の集中，利用者管理の複雑化とその労力，サーバ管理などの負担が大きい．これに対し，P2P プロトコルによる方法ではこれらの問題点が発生しない反面，利用者管理という概念がないため，アクセス制御を行うことができない．また，いずれの方式においてもファイルがネットワーク内に偏在するため，ネットワーク全体のストレージの使用効率という面では，得策とは言えない．

そこで本稿では，P2P 方式のファイル共有機能を基本として，ネットワーク上で提供されたすべてのストレージ装置の集合を仮想的な単一のファイルシステムとして扱う分散ストレージシステムの一方式を提案する．本システムではひとつのファイルを複数のストレージ装置に分散保管するため，ネットワーク内におけるファイルの偏在化の問題を解決しつつ，空きリソースを有効活用することができる．また，ストレージを提供する個々のコンピュータは P2P プロトコルにおけるサーバントとして機能する．このため，分割単位に冗長度を持たせることによりファイルの復元性を高めるとともに，アクセス制御の自由度と情報の秘匿性を確保することができる．

2. 既存のネットワークファイル共有方式

2.1. ファイル共有方式の分類

ネットワーク上でファイル共有を行う方式は，大きく次の２つに分類できる．

分類１：オペレーティングシステムやネットワークレイヤで提供されるプラットフォーム依存のストレージ共有機能．それぞれのオペレーティングシステムの利用者管理機能に依存したアクセス制御が可能であるが，マルチプラットフォーム環境では相互のプロトコルのエミュレーションなどが必要であり，基本的にプラットフォーム依存となる．UNIX 上の NFS，Microsoft の LAN Manager などがこの例である．

分類２：マルチプラットフォーム向けのファイル共有では，既存のオペレーティングシステムのユーザ管理を超えたファイル共有が可能である．既存ネットワークに対するオーバレイ・ネットワークとして実現されるが，統一されたアクセス制御の方式を持たない．P2P プロトコルがこれにあたる．

2.2. ネットワークファイルシステム

ネットワークファイルシステム(NFS)は，複数の UNIX ワークステーションでディスク資源を共有するための機能である．NFS は Sun RPC により機能し，データの転送には UDP/IP が使用されている．NFS のファイル共有は，ローカルディスク上にあるファイルシステムと同様に，自身のファイルシステムに NFS ボリュームをマウントすることで実現され，これにより資源の共有が実現する．

NFS は，それ自身にアクセス制御の機構を持っておらず，オペレーティングシステムのユーザ管理に依存している．このため，同一のオペレーティングシステムの管理範囲を超えてアクセス制御を行うことができず，データの秘匿性もオペレーティングシステムのユーザ管理に依存する．また，ネットワーク上の複数のクライアントに対して共有ファイルを提供することになるため，共有ファイルを多数擁するサーバと，ほとんど持たないサーバの間で，ストレージ使用率の偏りが生ずる．

2.3. P2P プロトコル

Napster, Gnutella, Freenet 等の P2P オーバレイ・ネットワークは，エンドユーザ間におけるデータ共有，交換を，アップロード・ダウンロード志向で行うものである．P2P システム自体は自律分散システムであり，

耐故障性やスケーラビリティに優れている．P2P システムは，一般に Hybrid 型と Pure 型に分類できる．

Napster は Hybrid 型に分類される．各サーバントは，アプリケーションの起動と同時に Napster Server に接続し，所有コンテンツ・リストを送信する．コンテンツの検索は，検索 Query を Napster Server に送信することによって行われる．Napster Server は，検索結果(所有者，コンテンツの質)をサーバントに返し，サーバントはコンテンツを所有するサーバントと TCP セッションを確立し，コンテンツの転送を実行する．

一方 Gnutella，Freenet は Pure 型に分類される．Pure 型では，各サーバントが TCP セッションを確立し，その状態を維持することで，オーバレイ・ネットワークを形成する．Pure 型には，Hybrid 型のようなコンテンツ情報を管理する専用 Server は存在せず，コンテンツの検索・発見はネットワークの機能として実現される．

一般に，P2P によるファイル共有は，各サーバントの保有するデータリストを介して実現する．共有の対象となるデータの保持は各サーバントが個別に行い，コンテンツの検索は，サーバントの保有するデータリストをもとに行われる．

共有に設定されたファイルへのアクセス制御は，サーバント間のメッセージの相互交換により実現するため，NFS におけるファイル共有の方法とは大きく異なる．共有ファイルに対するアクセス制限は，メッセージ交換によって実現しているため，NFS よりも柔軟なファイル共有が可能であるが，共有の対象となるファイルは，各サーバントの記憶装置に保存される．

2.4. 既存システムの問題点

以上に述べた既存のネットワークファイル共有方式の比較を表 1に示す．いずれの方式もファイル共有機能それ自身に利用者のアクセス制御の機構を持たない．また共有ファイルは，そのファイルを提供するサーバ自身の記憶装置に保存され，必要のつど転送する方式をとるため，共有ファイルはネットワーク内で偏在化する．したがって，ネットワーク全体に空きリソースが発生する原因となっている．

表 1：ネットワークファイル共有の比較

分類	例	特徴
1	NFS (Sun) LAN Manager(MS)	ネットワークにファイルが偏在 アクセス制御なし →OS(プラットフォーム)依存 共有ファイルの秘匿性が低い
2	Napster Gnutella Freenet	ネットワークにファイル偏在 アクセス制御なし →メッセージ依存 共有ファイルの秘匿性が低い

3. 提案方式

3.1. 前提条件

ここでは，比較的小規模な組織内ネットワークを仮定し，各端末は十分なストレージを保持しているものとする．一般に個々記憶装置の使用率には偏りがあり，ネットワーク全体では多大な空きリソースが発生している．そこで共有ファイルを分割保管することにより，空きリソースを解消し，記憶装置の有効活用を図るシステムを構築する．この際，共有ファイルの閲覧可能範囲について考慮する．

3.2. 分散ストレージシステムの概念

本論文で提案するネットワークファイル共有システムをここでは「分散ストレージシステム」と呼ぶ．本システムでは，共有ファイルの保存要求の発生時に当該ファイルを分割・暗号化してヘッダを付したセグメントを生成し，ネットワーク上に分散した記憶装置に分割保管する(図 1)．復元時には，分散保管された各セグメントを回収して復号化・結合する(図 2)．

3.3. ファイルの分割・配布方法

分散ストレージシステムでの分割・配布の概念を図3に示す．ネットワーク上の各端末を，分散ストレージシステムの配布先 S とする．配布元 C は配布先 C にもなり得る．配布元 C は，セグメントから元ファイルが復元されないように，元ファイルをスペクトラム拡散し，それぞれに対応したヘッダを付してセグメントを形成，全ての配布先 S に分散保管する．

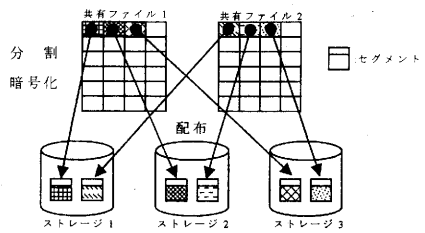

図 1：保存時の概念図

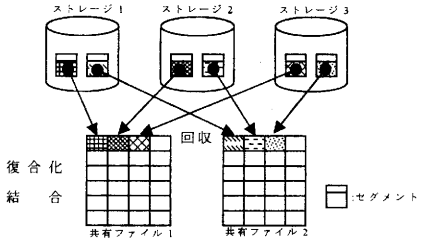

図 2：復元時の概念図

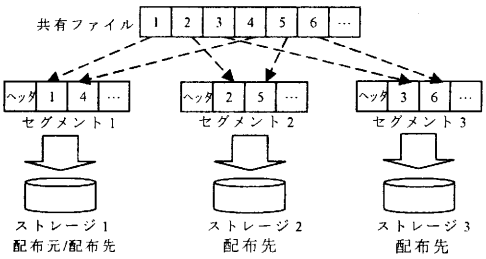

図 3：ファイル分割・配布方法

表 2：ヘッダ情報

保持区分	ヘッダの項目	非公開	公開
配布元C（保存者）	IP：配布元IPアドレス	○	○
	A：元ファイル名	○	○
	P：元ファイルサイズ	○	○
	L：配布先IPアドレスリスト	○	○
配布先S（協力者）	Ks：各協力者にユニークな鍵	○	×

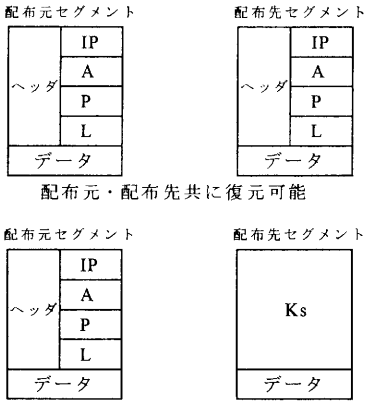

図 4：アクセス制御のためのセグメント構造

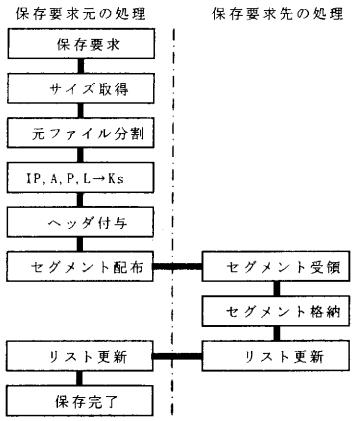

図 5：保存の処理手順

3.4. アクセス制御

各端末は，それぞれに分散保管されたセグメントのヘッダから，保持するセグメントのリストを生成する．このため，共有ファイルへのアクセス制御は，ヘッダに含まれる情報を制限することで実現できる．ここで，共有ファイルのアクセス制御を，公開(配布元・配布先共に復元可能な場合)と，非公開(配布元のみに復元可能な場合)に分けて考えると，表 2のヘッダ情報を定義することで，アクセス管理を行うことができる．

共有ファイルを公開したい場合は，配布元セグメントに付加するヘッダ情報と同じものを，配布先セグメントにも付加する(図 4)．また，共有ファイルを非公開にしたい場合は，配布元セグメントに付加するヘッダ情報と，配布先セグメントに付加するヘッダ情報を異なるものとする(図 4)．

各端末が共有ファイルにアクセスする場合には，セグメントのヘッダ情報から得られたリストを用いてファイルを復元する．配布元のみ復元可能なファイルの場合は，暗号化された鍵 Ks を用いてセグメントを復元する．

3.5. 鍵 Ks の構成

鍵 Ks は，送信元 IP アドレス，ファイル名，送信先 IP アドレスの情報から HASH 関数を用いて作成された，ネットワーク内でユニークな値である．

保存者は鍵 Ks を作成し，配布先 S に対するセグメントのヘッダとして付加する．復元者も同様に鍵 Ks を作成し，配布先でセグメントを探索する際のキーとして使用する．

3.6. 保存の処理手順

配布元 C は，共有ファイルを分割し，アクセス制御の分類に応じたヘッダを付加し，セグメントを配布する．保存の処理手順を，図 5に示す．

3.7. 復元の処理手順

配布元 C がデータを取り出す際は，復元を要求するセグメントに付加されたヘッダ情報から復元可否を判断する．以後は各配布先 S に対して，暗号化された Ks に対応するセグメントを要求し，セグメントを回収，復号化し，結合する．復元の処理手順を，図 6に示す．

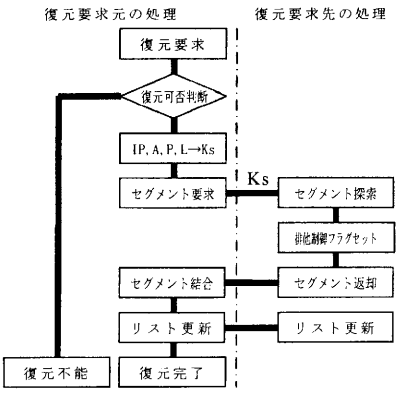

図 6：復元の処理手順

表 3：既存方式との比較

	NFS	P2P	提案手法
ネットワーク上のトラフィック	○	×	○
ストレージ使用率の平滑化	×	×	○
I/O のオーバヘッド	○	○	×
アクセス制御の制約	○	×	○
データ亡失時の復元性	×	×	○

4. 提案手法の検討

提案手法はストライピング方式によりファイルを分散させるため，各端末間の使用率の偏りが解消されることが期待できる．すなわち，各端末の空きリソースを統合し，ネットワーク上のストレージを有効活用することができる(表 3)．

また共有ファイルは，保存時にスペクトラム拡散した後にヘッダを付加する．このヘッダによりアクセス制御が行われる．たとえば非公開として保存したセグメントを保持する端末においてヘッダ情報中の Ks を解読したとしても，配布先 IP アドレスリストが得られないため，非公開のセグメントを保持する端末上で許可されない復元操作を完了することは極めて困難であり，各ストレージに保存されているセグメントの秘匿性を高く維持できるものと考えられる．

しかし，分散ストレージシステムでは，各端末の状況により分割保存したセグメントを，必ずしも復元時に回収できる保証がないという問題が発生する．この問題に対しては，共有ファイルの分割を冗長構成化し，適切な符号化行うことにより，セグメントの部分消失に対する耐久性を確保することができると考える．

ここで，S Byte の共有ファイルを i 個に分割し，それぞれに付加されるヘッダを H Byte とすれば，単位セグメントは $(S/i)+H$ Byte となる．ここではじめに，K を情報点数，n を符号長とし，符号化率 $r=k/n$ ($0<r<1$)を定義すれば，共有ファイルは分散保管後に $(iH/S)+(1/r)$ 倍のサイズとなり得るため，分散保管の対象とするネットワークへの冗長性は，iH と r により決定できる．

H は配布先の IP アドレスのリストを保持しているため i に比例し，したがって iH は i^2 に比例することが自明である．例として符号化方式に，誤り訂正数 t の原始 BCH 符号(t/n は一定，i は自然数)を適用すると，$n=2^i-1$ であり，$n-k \leqq it$ なので，符号語を小さくしなければネットワーク全体に高い冗長性を与え，必要以上の冗長性が付加されることとなるため，符号化手法は特に考慮しなくてはならない．共有ファイルに高い復元柔軟性を与えるほど，ネットワーク上のストレージ全体を圧迫する特徴があるが，このような冗長度に伴う問題は，既存方式で生ずることはない．

5. まとめ

本稿では，共有ファイルを分割，秘匿化，分散保存する新しいネットワークストレージシステムとして，分散ストレージシステムを提案した．また，アクセス制御を考慮したセグメント構造及び保存・復元処理の手順を示し，更に既存方式と比較・検討することにより，その有効性と考慮すべき事項を示した．

提案システムは，ファイルを分散保管し，誤り訂正符号により復元性能を設定することができるため，単にファイルを保存する目的に限定されたものとは異なる．したがって，暗号化に使用される鍵の分散保管と多数決的復元等に応用の場面が考えられる．

今後の課題として，システムの実装と導入に伴う運用上の効果について検討する必要がある．

文　献

[1] Gnutella, http://www.gnutelliusms.com/
[2] I. Clarke, et al. : Proc. ICSI Workshop on Design Issues in Anonymity and Unobservability, June 2000.
[3] 平野仁之，岩切宗利，中村康弘，“冗長度を付加する分散ストレージシステムの一実装方式”，電子情報通信学会ソサイエティ大会，pp.15，Sep.2003.