1Z－5

ネットモデルを用いた自立分散型ファイルシステムのコンカレント設計

山口　真之介　和崎　克己　師玉　康成
信州大学工学部　情報工学科

1．あらまし

ネットワークによって接続された，計算機間でのファイルシステムの共有を行う際，そのサーバの秘匿性が確実でない場合，データの信頼性に問題が生ずる．本研究では，ファイルサーバの設計にネットモデルによるコンカレント設計を用いて，ファイルシステム上のデータの暗号化，キャッシュ，ファイル管理を並列に実行していくことで，ネットワークファイルシステム全体の信頼性の向上を図ることが目的である．

2．パイプライン化 DES 暗号による設計

ファイルサーバの構成を図 1 に示す．ファイルデータの暗号化には，DES 暗号[1] を使用した．DES 暗号は平文を 64 ビットに分割し，それぞれに 16 回の演算を繰り返し行って暗号化する．そこで，1 回の演算を一段のステージとして排他処理を行えるように，16 段のパイプライン構造として設計した．システム仕様の設計にはペトリネット[2] を用いた．その際に，従来のペトリネットではトークンの持つ情報，トランジションの発火条件などでモデルの記述性に限界があるため，トークンに情報を付加し，発火条件を論理式で表し，更にプレースの出力も演算結果として表現する拡張ペトリネット[3] を採用した．

64 ビットの情報を持つトークンは，暗号化モジュール内で L,R 各々32 ビットに 2 分され，DES 暗号アルゴリズムに基いた暗号化が逐次行われる．DES の暗号化と同時に，ファイルの持つ情報を同期させながら，別の経路を通す．これらファイル情報用トークンには，名前，更新日時，容量，分割ブロック数などを持たせており，暗号化されたデータがどのファイルのデータかを定義している．この経路ではファイル名，更新日時等も換字式暗号によって暗号化される．こうして暗号化されたファイルデータと情報は UNIX File System(UFS) に記録される．一方，データの読み込みには，同じ演算を逆に 16 回行っていくことで，元の平文を得ることができる．こうして，暗号化されたデータを記録することで，共有されたファイルシステム情報の秘匿性を保つことができる．

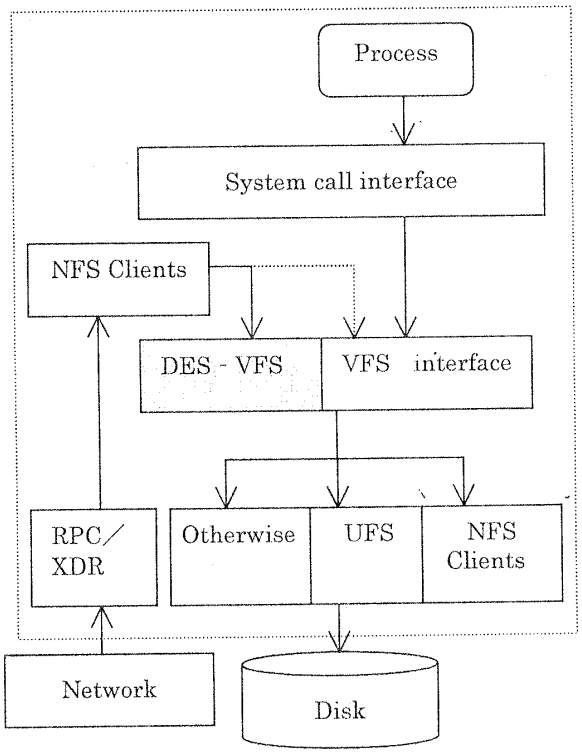

図 1．ファイルサーバーの構成

3．暗号化ファイルシステムの並列化

暗号化システムは，暗号化するデータの容量によって処理時間に大きく影響を及ぼすため，ファイルトランザクションのターン・アラウンド・タイムが劣化する恐れが生じる．よって多くの命令を同時実行していく上で，並列化によって処理時間を減らすことを考えなければならない．暗号化システムを複数並列に実行させ，空いているもの（暗号化を実行していないもの）に順次新しい命令とデータを渡して行くシステムの並列化を考える．

"Concurrent Design of the Autonomous-Distributed File System by using the PNs Modeling"; Shin'nosuke Yamaguchi, Katsumi Wasaki and Yasunari Shidama; Department of Information Engineering, Faculty of Engineering, Shinshu University; 500 Wakasato, Nagano 380-8553, Japan.

この時問題となるのは，暗号化による命令の実行にパイプラインニングの遅延が生じることで起こる，命令同士の依存関係があげられる．例えば，書き込むデータの暗号化と書き込み，読み出しが同時に起こる場合のデータ依存関係についてなどである．この様な問題を解決していくために，暗号化システムに現在実行中の状態を示すステータス用のプレースを用意し，命令と暗号化中のファイル名をトークンとして出力させておき，それらをチェックした上で，次の命令をどの暗号化システムに渡すのか決定させることを考える．同じファイルに関わる命令であれば，現在暗号化中のシステムに次の命令を渡すことで，依存関係を誤ることなく逐次実行させることができる．現在実行中の命令と依存関係が無いのであれば，空いているシステムに命令を渡せばよい．

暗号化システムの前には，それらの命令とデータを溜めておくキャッシュを設置しておく必要がある．また，ファイルの追加書き込みを行う場合，以前暗号化されたファイルは，64 ビット毎に暗号化されているので，64 ビットにするために本来のファイルデータの容量より暗号化データが，増加している場合が考えられる．そのため追加書き込みのためには，まずファイルの最終ブロックを読み出し，復号化した後，新しいデータを付加して再び暗号化する処理を行う．また，書き込みデータの暗号化が行われている時に，ファイルの消去命令が実行された場合は，依存関係を考慮した処理にとどまらず，暗号化中のデータを書き込まずに放棄したり，その後の同ファイルに対する命令を拒否しなければならないなど，より複雑な処理が求められる．このため，制御部分の仕様設計にもペトリネットを用い，動作の正当性の検証を行う．

4．設計例

図 2 にパイプライン化された DES のネットの一部を示す．64 ビットに分割されたデータは，プレース P1に投入される．次に IP により転置が行われ，L0，R0 に分割される．R0 は拡大転置 E や鍵スケジュール K1 等による演算を実行した後，L0 との排他的論理和が行われ R1 に出力される．L1 は一段目では R0 の値がそのまま引き継がれる．これらの操作を一回としてプレース Pr n（n＝1, 2, …, 16）により，排他処理を行うことで 16 段のパイプライン化を図っている．

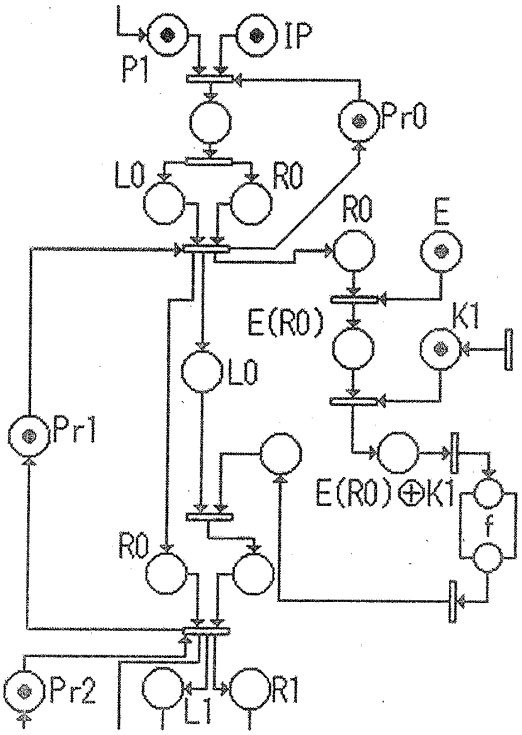

図２　パイプライン化 DES のネット構造

ネットの規模の規模について，今回設計した暗号化ファイルシステムの仕様として，DES パイプライン 16 段を符号，復号各々4 ラインを用意した場合，プレースは 3328，トランジションは 1600，アークは 5696 を要した．

5．まとめ

拡張ペトリネットを用いて暗号化ファイルシステムのモデル化を行った．今後の課題は，システムの実装とそれに伴う評価，また暗号化するファイルの容量に応じた，より柔軟な暗号化を行うための制御機構を考察していくことである．

参考文献

[1]"Data Encryption Standard", National Bureau of Standard., Federal Information Processing Standards Pubilications, 46, 1997

[2]T.Murata : "Petri Nets: Properties, Analysis and Applications",Proc. IEEE, 77, 4, 541-580, 1989.

[3]K.Wasaki, Y.Fuwa, M.Eguchi, Y.Nakamura: "Logical Coloured Petri Net Expanded to be Suitable making the Control System Model", Proc. ICARCV96, TA-2-2, 708-71, 1996.