分散ハイパーメディア OS Net-BTRON におけるハイパーメディアサービス管理機構

重定如彦[†] 越塚登[††] 坂村健[†††]

本論文では，分散ハイパーメディア OS“Net-BTRON” におけるハイパーメディア資源管理の機構について述べる．近年，World Wide Web（WWW）を基盤とした全世界規模のハイパーメディア環境がインターネット上に構築され，情報発信システムとして大きな成功を収めている．一方，ネットワーク上の他のハイパーメディアコンテンツを参照・引用をしながら，頻繁なデータ編集による試行錯誤を繰り返して，人間の思考を支援することも，ハイパーメディアシステムの当初の用途の 1 つである．ところが，現在の WWW システムをこうした用途に用いると，リンクの接続性が脆弱であるために，リンク切れや Editing Problem といった問題が頻繁に発生する．そこで我々は，頑強なリンク接続性を提供する分散型ハイパーメディア資源管理機構を持った新しいオペレーティングシステム，Net-BTRON を構築している．Net-BTRON のハイパーメディア資源管理機構は次の特徴を持つ．まず，ハイパーメディアノードやリンクをファイルシステムで実現する．Net-BTRON 以外の他のハイパーメディアとの相互接続性を実現する柔軟なファイルシステム拡張機能を持つ．ハイパーメディアアプリケーションの操作性を標準化するためのツールキットを提供する．我々は Net-BTRON のパイロットシステムを BTRON3 仕様 OS を拡張して実装し，その上に多くの分散ハイパーメディアアプリケーションを構築することによって，これらの技術的特徴の有効性を確認した．

Hypermedia Resource Management Mechanism of the Distributed Hypermedia Operating System “Net-BTRON”

Yukihiko Shigesada,[†] Noboru Koshizuka[††] and Ken Sakamura[†††]

This paper proposes, the hyper-media resource management mechanism of distributed hyper-media operating system “Net-BTRON”. Recently, worldwide hyper-media environments based on the World Wide Web (WWW) have been constructed on the Internet. These systems achieved a great success as systems for information distribution. However, they are inconvenient for frequent editing purpose and can not serve as tools for thinking, which is one of the original purposes of the hyper-media system, because of the weakness of the link. If we use the current WWW systems for the purpose of tools for thinking, problems such as broken link or editing problem will occur frequently. In this paper, we constructed a new operating system called “Net-BTRON” which can be used as a tool for thinking by its robust distributed hyper-media resource management mechanism. The features of the Net-BTRON are as follows. First, the Net-BTRON realizes the hyper-media nodes and links by the file system. Second, the Net-BTRON provides a flexible file system extension mechanism to integrate with the other hyper-media systems. Third, the Net-BTRON provides a toolkit for standardization of the user interface of hyper-media applications. We implemented the Net-BTRON pilot system by extending the BTRON3 OS, and proved the technical advantage of the Net-BTRON by implementing a lot of distributed hyper-media application on it.

† 東京大学大学院理学系研究科情報科学専攻
Department of Information Science, Interfaculty Initiative in Information Science, Graduate School of Science, The University of Tokyo
†† 東京大学情報基盤センター
Information Technology Center, The University of Tokyo
††† 東京大学大学院情報学環学際情報学府
Interfaculty Initiative in Information Studies, Graduate School of the University of Tokyo

1. はじめに

現在 World Wide Web（WWW）を基盤とした全世界規模のハイパーメディア環境がインターネット上に構築されている．WWW はオーサリングツール等で作りこんだコンテンツや，データベースに格納された定型データを配信するシステムとしては大きな成功を収めている．しかし本来のハイパーメディアは，Engelbert や Nelson ら[1)]によって思考のための道具

として提案されており，頻繁なデータ編集による試行錯誤を繰り返しながら知識を表現する多くの情報コンテンツの間で密な関係性を構築することで，人間の思考を支援することが指向されている．現在の WWW システムをこうした思考のための道具として利用する際のシステム上の最大の問題点は頻繁にデータ編集を繰り返す際に細心の注意を施さない限り，リンク切れや Editing Problem[2)] と呼ばれるコンテンツの不整合が容易に生じることである．これは，ブラウザ等のハイパーメディアソフトウェアがすべてアプリケーションで実装され，ハイパーメディアコンテンツの整合性を保つような，ハイパーメディア資源の保護管理機構がまったく提供されていないことに起因する．

従来よりこの問題に対して，2 つのアプローチが知られている．オーサリングツールによるアプローチと，ミドルウェアのような何らかの実行時常駐システムでハイパーメディア資源を管理するアプローチ[3)]である．

(1) オーサリングツールアプローチ

現在商用のソフトウエアを中心として，HTML（Hyper-Text Markup Language）や XML（eXtensible Markup Language）を出力するオーサリングツールが数多く開発されている．これらを使ってハイパーメディアコンテンツの整合性を保ちながら編集するためには，ハイパーメディアシステム（HMS）が 1 つのシステム，たとえば HTML によって記述される WWW システムだけである場合には有効である．ところが，近年 WWW を中心としたハイパーメディア環境は拡大し，マルチメディアアプリケーションからワードプロセッサや表計算に至るまで，多種多様なアプリケーションがそれらの扱うデータの中にも WWW のリンクを埋め込めるようになった．したがって，ハイパーメディアの編集にはどうしても複数のオーサリングツールを用いることになる．そこでコンテンツの整合性を保つためには，ツール間で整合性の調整する基盤システムが必要となる．

(2) ミドルウェアアプローチ

従来より，ハイパーテキストの分野において，ハイパーメディアサービスを提供するミドルウェアの研究が行われてきた．これらの研究によるシステムは，以下の 2 つの問題点を持つ．第 1 に，ハイパーメディア資源の保護，管理に必要な機能には，アクセス制御，認証，キャッシュ，転送プロトコル，リンクユーザインタフェース等があり，これらの多くは OS（operating system）が提供するサービスと共通する．これらのサービスをミドルウェアで実現する場合，OS との二重に実装になり，計算機資源を浪費，もしくは OS による実現とミドルウェアによる実現との間での不整合が生じる．第 2 に，OS の生のシステムコール等，ミドルウェアが提供する API（Application Programming Interface）よりも低レベルの API がアプリケーションに開放されているため，アプリケーションがハイパーメディア構造に不整合を起こす動作を行うことが可能である．このため，これらのシステムではアプリケーションにバグが含まれている場合や，コンピュータウィルス等の悪意あるプログラムが侵入した場合等に，コンテンツの整合性がきわめて脆弱になる．

そこで，我々は，強固な分散型ハイパーメディア資源管理機能を持った新しいオペレーティングシステム，Net-BTRON を構築している．我々は Net-BTRON に対して以下の技術的な目標を達成することを掲げている．

- 整合性のある頑強なハイパーメディアのリンク
- すべてのアプリケーションがハイパーメディア機能を持つ
- アプリケーション間での一貫したハイパーメディアの操作性
- Net-BTRON 以外のハイパーメディアシステムとの相互接続性

我々はこれらの目標を実現するために，以下のアプローチをとった．

- 分散ハイパーメディア型記憶サービスのファイルシステムによる実現
- キャッシュや転送プロトコルモジュール，ユーザ設定等，ハイパーメディア関連資源の一括管理
- ハイパーメディアユーザインタフェースの OS レベルツールキットでの実現
- 他のハイパーメディアシステムを接続するためのファイルシステムの拡張性

以下，本論文では，Net-BTRON が提供するこれらの特徴を持った分散型ハイパーメディアサービスについて焦点を絞って述べる．

本論文の構成は次のとおりである．2 章では Net-BTRON の基本ハイパーメディアモデルである実身/仮身モデルを紹介する．3 章では Net-BTRON のハイパーメディアアーキテクチャを述べる．4 章では Net-BTRON と WWW の統合を例にあげて Net-BTRON の拡張性について述べる．5 章では Net-BTRON のパイロットシステムの実装について述べ，6 章で関連研究との比較を行う．最後に 7 章で本研究のまとめを述べる．

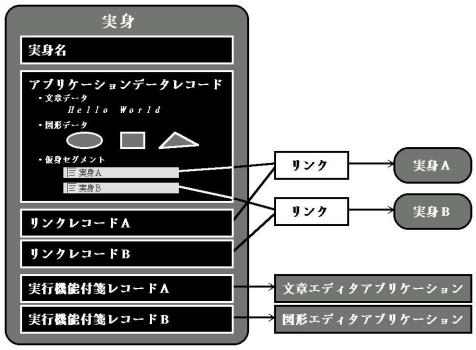

図 1 実身/仮身モデル概念図
Fig. 1 Basic hypermedia model of Net-BTRON.

2. Net-BTRON 基本ハイパーメディアモデル[4)~6)]

Net-BTRON のハイパーメディアモデルは，従来より我々が提案してきた “実身/仮身モデル”[4)~6)]に基づいている．本章では Net-BTRON の実身/仮身モデルの記憶モデルとユーザインタフェースモデルについて簡単に紹介する．

2.1 記憶モデル

Net-BTRON ハイパーメディアの記憶モデルはノードである “実身” とそれらの間を結ぶリンクである “仮身” から構成される．仮身は実身中の点を始点とし，実身または実身中の領域を終点とする片方向型リンクである．

実身は複数の “レコード” から構成され，レコードにはアプリケーションが扱う実データを格納する “アプリケーションデータレコード” のほかにその実身のデータを扱うアプリケーションプログラムを指定する “実行機能付箋レコード” を複数格納することができる．これにより，ハイパーテキストの重要な性質の 1 つである単一ノードのマルチビューを扱うことができる（**図 1**）．

2.2 ユーザインタフェースモデル

画面上では実身は 1 つのウィンドウを通じて表示され，仮身はその仮身の始点である実身のウィンドウ中に矩形状のグラフィカルオブジェクトとして表示される．仮身にはピクトグラムや実身名，実身のデータを扱うデフォルトアプリケーション名，その他最終更新日時等の実身に関する情報を表示することができる（**図 2**）．これらの情報は，あるアプリケーションが仮身を表示しているときに，他のアプリケーションがこの仮身の指す実身を更新したことによって変更が生じた場合，速やかに仮身の表示も更新される．

図 2 仮身
Fig. 2 Virtual object.

仮身上へのユーザのダブルクリック操作によって，その仮身が指す実身を新規ウィンドウ中に表示する．実身内に複数の実行機能付箋レコードが存在するマルチビュー状態の場合，クリック操作によって仮身を選択状態にしたうえで，メニューで複数のアプリケーションから 1 つを選択して起動する．

3. Net-BTRON ファイルシステムアーキテクチャ

Net-BTRON は，分散型ハイパーメディアを支援するためのネットワーク透過なファイル共有サービスをファイルシステムによって実現する分散ハイパーメディア型 OS である．既存のネットワーク透過なファイル共有サービスを提供する分散ファイルシステムとしては NFS（network file system）[7)]等がある．これらは，基本的に LAN 上で互いに信頼しているマシン間のみでファイルを共有することが想定されている．したがって，たとえば，地球の裏側にある見知らぬマシンからインターネットを介したファイル共有は扱わない．これに対し Net-BTRON は，WAN 上の信頼していない不特定多数のマシン上からのネットワーク透過な匿名アクセスが主な想定利用である．こうしたハイパーメディアを指向した新しい分散型ファイルシステムを我々は “ハイパーメディア型ファイルシステム” と呼んでいる．

Net-BTRON ではファイルシステムレベルでハイパーメディア型記憶サービスを実現することにより，ローカルなファイルに対して頑強なシステムを実現した．Net-BTRON はローカルな資源に対しては，ユーザの操作ミスだけでなくアプリケーションのバグや悪意のあるアプリケーションによってリンクの整合性が破壊されない頑強なハイパーメディアリンクを提供する．それだけでなくリモートな Net-BTRON ファイルに対してもファイルの削除を除くファイルの移動，ファイル名の変更等の操作に対して頑強なリンクを提供する．

さらに，Net-BTRON はハイパーメディア型のユーザインタフェースの構築のためにツールキットを提供している．これを使うことによって，ハイパーメディアアプリケーションを容易に構築すること，ハイパー

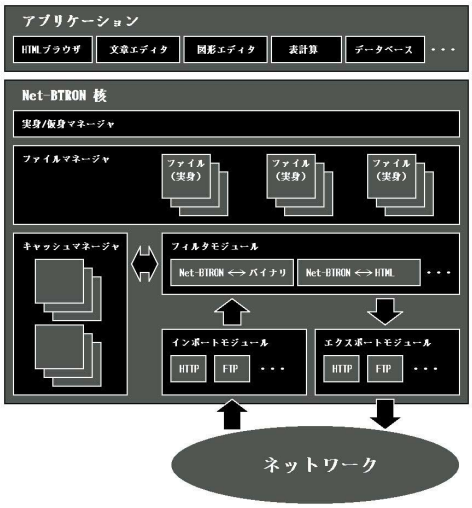

図 3 Net-BTRON ハイパーメディアアーキテクチャ
Fig. 3 Net-BTRON hypermedia architecture.

メディアアプリケーション間で共通のユーザインタフェースを実現できる．

ここでは Net-BTRON のファイルシステムとユーザインタフェースツールキット部分をあわせて，“Net-BTRON ハイパーメディアアーキテクチャ” と呼び，本章ではこの Net-BTRON ハイパーメディアアーキテクチャについて述べる．Net-BTRON ハイパーメディアアーキテクチャは**図 3** のように主に 6 つのモジュールから構成される．

- ファイルマネージャ
  ハイパーメディア構造のファイルシステムを実現する．
- インポートモジュール
  コンピュータネットワークを通じてリモートなホスト上に存在するオブジェクトを取得するデータ転送プロトコルを実装する．
- エクスポートモジュール
  Net-BTRON 上のオブジェクトをコンピュータネットワークを通じてリモートなホストへ転送するデータ転送プロトコルを実装する．
- フィルタモジュール
  Net-BTRON 形式のローカルでマルチレコードな実身のデータをネットワーク転送可能な形式のデータに変換，またはその逆を行うモジュール．また，フィルタモジュールは他のハイパーメディアシステムとの間のデータコンパチビリティを確保するために，Net-BTRON 形式のデータを HTML 等の他のデータフォーマットに変換する機能を提供する．
- キャッシュマネージャ
  インポートモジュールが取得したオブジェクトデータのキャッシュを行う．
- 実身/仮身マネージャ
  実身/仮身モデルのユーザインタフェースを管理する．

以下本章では各マネージャの詳細について述べる．

3.1 ファイルマネージャ

ファイルマネージャは Net-BTRON の実身を管理するファイルシステム本体である．Net-BTRON ファイルシステムは 2 章で述べたように実身/仮身モデルに基づいた片方向リンク式のハイパーテキスト構造を持つファイルシステムである．実身には大別して，Net-BTRON ユーザと同じマシン上に存在する “ローカル実身”，ネットワークで接続された遠隔のマシン上に存在する “リモート実身” がある．Net-BTRON では，ローカル実身とリモート実身の差異から透明な実身操作 API を提供する．これによりアプリケーションプログラマは通常のファイル操作 API を用いるだけで，分散型のハイパーメディアアプリケーションを容易に構築できる．

ネットワーク透明な分散ファイル API を提供する既存のシステムには NFS[7] 等があるが，これらと Net-BTRON とは次の点が異なる．NFS では相互に信頼しあったマシンどうしでファイル操作を行うが，Net-BTRON では基本的に匿名アクセスによる遠隔ノード操作を許している．ただし，セキュリティの観点から，リモート実身への匿名アクセスに対しては基本的に読込操作のみに限定している．

3.1.1 実　　身

実身は，主としてアプリケーションが扱う実データを格納するハイパーテキストノードであり，そのほかにも “仮身” と呼ばれる他のノードへのリンクや “実行機能付箋” と呼ばれるそのデータを扱うアプリケーションへのリンクを，任意個格納することができる．そのために，実身は次のような構造をしている．実身は実身名を含む制御情報と複数のレコードから構成され，このレコードにはアプリケーションが扱うデータ本体が格納される “アプリケーションデータレコード”，他の実身を指す “リンクレコード”，その実身のデータを扱うアプリケーションプログラムを指す “実行機能付箋レコード” 等が存在する．これらのレコードは任意の数ずつ含むことができる（図 1）．

ローカル実身の識別子は “ファイルシステム名” と “ファイル ID” である．ファイル ID はその実身が生成されてから削除されるまで同一の値をとる．リモー

ト実身の識別子は“アクセスプロトコル”，“マシン IP アドレス”，“ファイルシステム名”，“ファイル ID”の組である．

ファイル ID はシステム内部値であり，ユーザがその値を直接扱うことはない．そこで Net-BTRON はユーザまたはプログラマがファイルを識別するための表現形式として，ローカル実身に対しては“パス名”，リモート実身に対しては現在 Web 環境上の資源の識別子として広く使用されている“URL”（Uniform Resource Locator）を採用した．

3.1.2 仮　　身

仮身は 2 つの実身の間を接続するリンクである．仮身には“リンクレコード”と呼ばれる実身間の接続を表すリンク部と，“仮身セグメント”と呼ばれる仮身が画面上でどのように見えるかを表すアンカー部から成り立つ．リンクレコードはファイルマネージャが管理し，仮身セグメントは後述する実身/仮身マネージャが管理する．リンクレコードは実身のレコードとして実現され，仮身セグメントはアプリケーションデータレコードの中にアプリケーションデータと一緒に格納される（図 1）．

リンクレコードはローカル実身およびリモート実身を参照するポインタであり，そのデータ構造を**図 4** に示す．Net-BTRON アプリケーションからの実身への参照はこの LINK データ構造を用いて行う．この LINK はローカル実身の識別子をすべて含み，それ単独で直接実身を指定できる．リモート実身の場合は“リンク実身”と呼ばれる LINK が指す特殊なローカル実身の中に識別子を含ませて間接的に指定する．**図 5** にそのデータ構造を示す．リンク実身はリモート実身識別子と拡張用の予約パラメータから構成されており，これらの予約パラメータは他の HMS のハイパーメディアノードを統合するためにファイルシステムを拡張する際に使用される．この拡張については 4 章で述べる．

このように，リモート実身を間接参照で設計した理由は次のとおりである．第 1 に，ローカル実身の識別子に比べて，リモート実身の識別子が長いため，すべての LINK にリモート実身の識別子のための領域を確保することは記憶利用効率が著しく悪い．第 2 に，Net-BTRON では他のハイパーメディアと Net-BTRON を統合する際に，他のハイパーメディアノードへのリンクも仮身で表す．その際ハイパーメディアの種類によって，そのノードの識別子のデータ構造が大きく異なることが予想されるため，より柔軟にデータ構造を替えられる間接指定の方法を採用した．

```
typedef struct {
   char   fs_name[20];  /* ファイルシステム名 */
   int    f_id;         /* ファイル ID */
   int    attr;         /* 仮身タイプ，属性 */
   int    rel;          /* 仮身の続柄 */
   int    appl[3];
/* 標準アプリケーション ID */
} LINK;
```

図 4　LINK のデータ構造
Fig. 4　Data structure of LINK.

```
typedef struct {
     int type;
 /* 1: Net-BTRON */
     int protocol;              /* 1: http */
     struct {
          int type;
  /* 0:IP アドレス，1:ドメイン名 */
          union {
               char ipadr[4];
               char domainname[256];
          } id;
     } host;
     int port;                  /* 拡張用予約 */
     struct {
          int type;
  /* 0: パス名 1 :オブジェクト ID */
          union {
               int  object ID;
               char path[256];
          } id;
     } obj;
     char objectname[256];    /* 拡張用予約 */
} RN_REDORD;
```

図 5　リモート実身を指すリンク実身のデータ構造
Fig. 5　Data structure of LINK to a remote file in the LINK file.

3.1.3 ファイルマネージャ API

ファイルマネージャの主要なシステムコールは以下の 6 つである．

- *get_lnk()*
  パス名または URL で指定した実身を指す LINK データを作成する．ユーザが直接パス名や URL を指定するようなユーザインタフェースを実現するときにのみ用いられる．
- *opn_fil()*
  LINK データによって指定された実身をオープンし，そのオープンしているセッションの間だけ有効なテンポラリな実身識別子を関数値として返す．リモート実身をオープンする場合はこの関数内で実身内のデータをローカルマシン上に転送しキャッシュする．
- *cls_fil()*

オープンした実身をクローズする．

- *fnd_rec()*，*rea_rec()*，*wri_rec()*
  一般的なファイルアクセス関数として上記の 3 つの関数が用意されている．それぞれ実身内のシーク，データの読み込み，データの書き込みを行う．

3.2 インポートモジュール

インポートモジュールは，Net-BTRON のネットワークファイルシステムのクライアントがリモート実身にアクセスするときに用いる転送プロトコル（Transfer Protocol）を実現する．Net-BTRON では標準の転送プロトコルとして HTTP（Hyper-Text Transfer Protocol）を用いるが，他の複数の転送プロトコルをインポートモジュールに登録し，動的に切り替えることができる．これによって，Net-BTRON は多様な転送プロトコル上に分散ファイルシステムを構築することが可能となる．さらに，この転送プロトコルを動的に切り替える機能と，3.5 節で述べるフィルタモジュールのデータ変換機能を用いることで，他のハイパーメディアシステムのサーバと接続して，そのノードを Net-BTRON のノードとして組み込むことが可能となる．

インポートマネージャーが呼び出されるタイミングは，通常 *opn_fil()* によってファイルマネージャがリモート実身をオープンするときである．このときインポートモジュールはリモート実身データをダウンロードし，必要に応じてフィルタモジュールによってデータ形式の変換を行ったうえで，キャッシュマネージャにそのデータを転送する．

3.3 エクスポートモジュール

エクスポートモジュールはインポートモジュールとは逆に，Net-BTRON 上のローカル実身をリモートの Net-BTRON 上でリモート実身として見せるための，サーバ側の転送プロトコルを実現する．つまり，外部からのファイルデータ転送要求に応じて Net-BTRON 内の実身データを転送するネットワークファイルサーバである．3.2 節で述べたように，Net-BTRON は標準プロトコルとして，HTTP を実装している．エクスポートモジュールもインポートモジュールと同様に，複数のモジュールを登録し，動的に呼び出すモジュールを切り替えることができる．これによって，たとえば，ftp クライアントや gopher ブラウザといった，Net-BTRON 以外のシステム上のブラウザに対してもデータを配信することができる．

エクスポートモジュールはネットワークからファイル転送要求が到着した時点で呼び出される．要求されたモジュールが起動し，送られてきたファイル識別子で指定された実身情報を獲得する．さらに必要に応じてデータをフィルタモジュールを通して Net-BTRON のマルチレコードデータを文字ストリーム形式等に変換してからクライアントにデータを転送する．

3.4 キャッシュマネージャ

Net-BTRON では，ファイルシステムの中でインポートモジュールが取得したリモート実身のキャッシュ管理を行う．キャッシュマネージャはインポートモジュールから独立しており，任意の転送プロトコルによって得られたデータを共通に扱う．どのハイパーメディアアプリケーションも，このキャッシュ管理サービスを前提としてプログラムを記述することができる．このサービスの利点は，アプリケーション独立にキャッシュ機能を実現できるため，アプリケーションごとにキャッシュの重複実装するという無駄を省けること，また各アプリケーション間で統一したキャッシュ置換が適用でき，キャッシュのヒット率の向上が期待できることである．

3.5 フィルタモジュール

フィルタモジュールはファイル転送の前後で，必要に応じてデータ形式の変換を行うモジュールで，基本的にはインポートおよびエクスポートモジュールから呼び出される．フィルタモジュールには目的に応じて 2 種類が存在する．1 つは Net-BTRON 実身のようなマルチレコードで表現されるデータをキャラクタストリームのようなネットワーク転送可能な形式に変換するモジュールである．もう 1 つは他の Net-BTRON 以外のハイパーメディアノードにアクセスする際にそのノードのデータフォーマットを Net-BTRON 形式のデータ形式に変換するモジュールである．たとえば HTML から Net-BTRON 形式に変換するフィルタ等が存在する．フィルタモジュールもインポートモジュールと同様ユーザが複数登録することが可能である．

3.6 実身/仮身マネージャ

実身/仮身マネージャはリンクを表す仮身を管理するツールキットであり，通常のユーザインタフェースツールキットと同様に以下の 2 点の実現を目的としている．第 1 に，ハイパーメディアアプリケーションのリンク操作部の実装を支援する高抽象度の API を提供すること，第 2 にその操作部分に標準的なユーザインタフェースを提供することである．

仮身セグメントは画面上での表示に必要なデータ構造で構成され，アプリケーションデータレコードの中にアプリケーションデータと混在して保持される．

実身/仮身マネージャは仮身に対するユーザ操作にそれぞれ対応した豊富な API を提供している．アプリ

ケーションは *oreg_vobj()* を用いてアプリケーションが扱う仮身をウィンドウマネージャに登録し，登録した仮身を扱うための仮身 id を取得する．以後アプリケーションは仮身 id を指定して次のような操作を加えることができる．たとえば，仮身を移動 *omov_vob()*，変形 *orsz_vob()*，各種属性変更 *ochg_***()*（***は属性の種類によって異なる），仮身が指す実身をオープンする *oopn_obj()* といった関数等がある．

アプリケーションプログラマは，ユーザの仮身に対する操作に応じてこれらの関数を呼び出すコードを記述することで簡単にハイパーメディアプログラミングを行うことができる．たとえば，仮身にダブルクリック操作を加え，その仮身が指す実身をオープンする場合，対応する仮身の id を指定して，*oopn_obj()* を呼べばよい．

3.7 Net-BTRON のハイパーメディアアーキテクチャの動作例

本節では，リモート実身からデータを読み出す動作を例として，Net-BTRON ファイルシステムの動作を説明する．S_Aはリモートホストの実身を提供するサーバ，C_Bはハイパーメディアアプリケーション AP_Bが動作しているクライアントを表す．この例では AP_B が S_A内の実身 N_Aにアクセスし，そのデータを読み込む．

3.7.1 LINK の取得

まず，AP_Bが *get_lnk()* を用いて N_Aを表す URL を LINK に変換する．*get_lnk()* が実行されるとファイルマネージャはキャッシュマネージャに N_Aのキャッシュの有無を問い合わせる（**図 6** (1)）．キャッシュが存在した場合，キャッシュの実身を表す LINK を返し動作を終了する．キャッシュが存在しない場合，ファイルマネージャは HTTP インポートモジュールに N_Aの制御データの転送を要求する（図 6 (2)）．インポートモジュールは HTTP を用いて S_Aから要求されたデータを取得する（図 6 (3)，(4)）．ファイルマネージャは N_Aを表すリンク実身 LF_Bを作成し，リモート実身情報を LF_Bに格納する（図 6 (5)）．*get_lnk()* はリンク実身を表す LINK を返し動作を終了する．

3.7.2 実身のオープン

次に AP_Bは *opn_fil()* を用いてリンクをたどって実身をオープンする．*opn_fil()* はまずキャッシュマネージャに N_Aのキャッシュデータの有無を問い合わせる（**図 7** (1)）．キャッシュが存在する場合は，そのキャッシュの実身をオープンしその実身記述子を返す．もしキャッシュが存在しない場合，ファイルマネージャは HTTP インポートモジュールに N_A の全データの

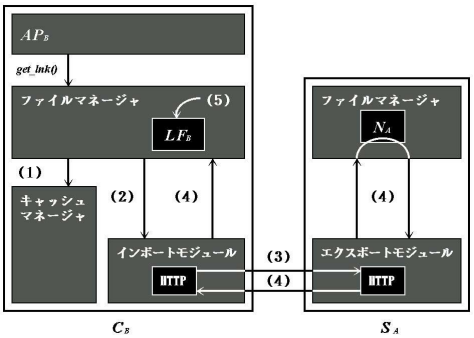

図 6 リモート実身のリンクのデータ構造取得時の Net-BTRON の内部動作
Fig. 6 An example of an internal action “Get LINK” in Net-BTRON.

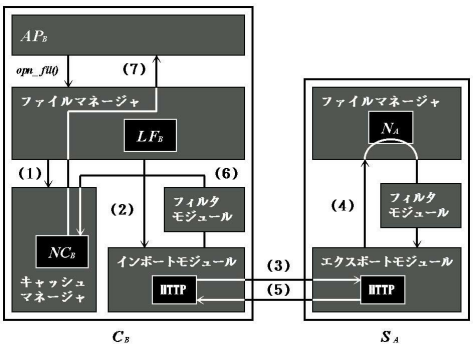

図 7 リモート実身のオープン時の Net-BTRON の内部動作
Fig. 7 An example of an internal action “Opening Node” in Net-BTRON.

転送を要求する（図 7 (2)）．インポートモジュールは HTTP を用いて S_Aへデータ要求を送る（図 7 (3)）．S_Aは N_Aのデータを読み込む（図 7 (4)）．次にフィルタモジュールはネットワーク転送用データ形式に変換し転送する（図 7 (5)）．C_Bはデータを受け取ると，受け取ったデータをフィルタモジュールを通して Net-BTRON 形式に変換しキャッシュマネージャに転送する．キャッシュマネージャは転送されたデータをキャッシュ実身 NC_Bとして保存してオープンし（図 7 (6)），NC_Bの実身記述子を返す（図 7 (7)）．キャッシュファイルのサイズが最大値を超えた場合はキャッシュ置換が行われる．

3.7.3 実身データの読み込み

次に AP_Bは *rea_rec()* を用いて実身データを読み込む．このとき AP_Bは直接 NC_Bをアクセスする．

3.7.4 実身のクローズ

最後に AP_Bは *cls_fil()* を実行し実身をクローズする．キャッシュマネージャは次のキャッシュ置換が行

```
LINK   *lnk;
int    fd;
char   buf[100];
ERR    err;

 /* ローカル実身の場合の LINK データ構造を取得 */
err    = get_lnk("/USR/test.txt",
                 lnk, F_NORM);
 /* リモート実身の場合の LINK データ構造を取得 */
err    = get_lnk("http://www.tron.org/",
                 lnk, F_NORM);
 /* リードオンリー属性で実身をオープン */
fd     = opn_fil(lnk, F_READ);
 /* アプリケーションデータレコードをサーチ */
err    = fnd_rec(fd, F_TOPEND,
                 MAIN_DATA_RECORD);
 /* データの読み込み */
err    = rea_rec(fd, 0, buf, 100);
 /* 実身のクローズ */
err    = cls_fil(fd);
```

図 8 ローカルおよびリモート実身を読み込む場合のサンプルコード
Fig. 8 Sample code handling a local node and a remote node.

われるまで NC_B をキャッシュデータとして保存する．

以上の流れをたどったリモート実身を読み込むクライアントのサンプルコードを**図 8** に示す．

4. WWW と Net-BTRON の統合

ハイパーメディアシステムの有用性を高めるためには，参照可能なコンテンツの量が多いことがきわめて重要である．そこで Net-BTRON では WWW をはじめとして，他のハイパーメディアシステムとの間と Net-BTRON ハイパーメディアアーキテクチャを統合するための拡張機能を有している．具体的には 3 章で述べたように，インポート/エクスポートモジュールによって多様な転送プロトコルに対応する機能，フィルタモジュールによって多様なコンテンツ記述形式との間での相互変換を行う機能がこれにあたる．本章では，この機能を利用した統合の例として Net-BTRON と WWW の統合について述べる．

4.1 Net-BTRON ノードから WWW ノードへのリンク

Net-BTRON のリンク実身には，拡張可能な領域があり，インポート/エキスポート/フィルタ各モジュールごとに定義可能である．WWW コンテンツへのリンクは，**図 9** に示すように，リンク実身の拡張領域を利用して表現している．こうした拡張利用領域を使って様々なハイパーメディアノードとリンクするためのリンク実身を “拡張リンク実身” と呼ぶ．WWW ノードを指す拡張リンク実身の実行機能付箋レコードは通常 WWW ブラウザを指定する．WWW ブラウザはファイルマネージャがダウンロードした HTML ページを読み込んで表示する．

```
typedef struct {
     int  type;          /* 2:WWW */
     int  protocol;      /* 1:http */
     struct {
          int type;
/* 0:IP アドレス，1:ドメイン名 */
          union {
               char ipadr[4];
               char domainname[256];
          } id;
     } host;
     int  port;
/* ポート番号 (デフォルトは 80) */
     int  type;
/* 0:パス名*/
     char pathname[256];
/* HTTP サーバのパス名 */
     char objectname[256];    /* 拡張用予約 */
} RN_WWW_REDORD;
```

図 9 WWW ノードを表すリンク実身のデータ構造
Fig. 9 Data structure of the LINK to a WWW node in a LINK file.

4.2 HTML から Net-BTRON 形式へのデータフォーマットの変換

WWW ではハイパーメディアノードのほとんどは HTML によって記述されている．Net-BTRON 上で HTML ノードを扱うアプリケーションは，HTML に対応した WWW ブラウザによって処理されるため，通常の文書部分が HTML によって表現されていても問題なく表示することができる．しかし，HTML のハイパーリンクに関する表現は A タグや IMG タグ等によって表現されており，Net-BTRON でのハイパーリンク表現である仮身と大きく異なっている．

この相違を解消するため，Net-BTRON ではフィルタモジュールとして HTML-BTRON 変換フィルタモジュールを実装した．このフィルタモジュールは HTML 形式の文字ストリーム型データを Net-BTRON 形式のマルチレコードデータ形式に変換する．そのために，第 1 に HTML ノードの中で他のノードを参照している部分を WWW 用の拡張リンク実身表現に変換する．第 2 に主データレコードには，HTML テキストデータ内で A タグや IMG タグ等によって他ノードを参照している部分を仮身セグメント形式に置換する．第 3 にリンクレコードには，参照しているノードの数だけ作成した拡張リンク実身を指すリンクレコードが格納される．実行機能付箋レコードには，WWW ブラウザを指定する．この変換処理を施した後で，キャッ

シュマネージャにデータが渡され，ローカル実身として保存される．保存されたデータは他のリモート実身と同様に扱われる．

Net-BTRON は HTML 以外の任意の形式で記述されたデータを Net-BTRON のマルチレコード形式に変換するためにユーザ定義のフィルタモジュール複数登録する機能を有している．インポートモジュールが呼び出すフィルタモジュールの種類の指定は URL による実身指定の拡張子部分を用いて行う．たとえば，拡張子が html の場合は HTML-Net-BTRON フィルタモジュールが呼び出される．もしも Net-BTRON 側で，必要なフィルタモジュールが実装されていないときは，その URL へのアクセスに対して，“File Not Found” のエラーが返る．

4.3 WWW からの Net-BTRON ノードへの参照

Net-BTRON ノードは Net-BTRON クライアントだけではなく，WWW ブラウザ等の他の OS 上で動作するハイパーメディアクライアントからのアクセス要求に応えるための統合機能を提供している．Net-BTRON の標準エクスポートモジュールは HTTP サーバとして動作し，一般的な WWW ブラウザに対してサーバの役割を果たす．

HTML ページから参照される Net-BTRON 上のノードの URL の表現は Net-BTRON ノードのパス表記で行う．たとえば，ホスト “h1.tron.org” 上の実身 “/SYS/sample” は “http://h1.tron.org/SYS/sample.html” と表記される（すなわち Net-BTRON 実身 ID の URL 表現である）．

最後の拡張子はクライアント側がどのデータ形式に変換して渡してほしいかを指定するための指定子で，Net-BTRON がこれに応じて呼び出すフィルタモジュールを決定する．この例では拡張子が html となっており，これにより Net-BTRON データを HTML 形式に変換するフィルタモジュールが呼び出される．この変換を施した後 WWW クライアントにデータを転送する．もしも Net-BTRON 側で，この HTML 変換のフィルタモジュールが実装されていないときは，この URL へのアクセスに対して，“File Not Found” のエラーを返す．

5. 実　　装

本章では Net-BTRON カーネルとその上に構築されているいくつかの分散型ハイパーメディアアプリケーションの実装について述べる．

5.1 実装環境

我々は Net-BTRON を BTRON3 仕様 OS を拡張して実装した．BTRON3 仕様 OS では，ローカル実身に対するハイパーメディアサービスを提供している．この上に，本論文で主に述べてきた，リモート実身へのアクセスサービスを含んだ分散ファイルシステムおよび WWW 等他のハイパーメディアシステムとの統合メカニズムを実装した．

Net-BTRON のファイルマネージャ，実身/仮身マネージャはマイクロカーネル部分に相当する内核に実装した．インポートマネージャ，エキスポートマネージャ，フィルタマネージャ，キャッシュマネージャはカーネル外核として実装した．

5.2 Net-BTRON 核

本節では Net-BTRON 核の実装の詳細について述べる．BTRON カーネルは全部で 345 のシステムコールから構成されており，その内訳としてファイルマネージャは 57，実身/仮身マネージャは 41 のシステムコールから構成される．Net-BTRON ではこのうちファイルマネージャと実身/仮身マネージャのシステムコールの拡張を設計し，実装した．実装した Net-BTRON のカーネルのバイナリサイズは約 2MB である[6]．

他のハイパーメディアとの統合機構として，具体的に以下のモジュールを実装した．“HTTP インポートモジュール”，“HTTP エクスポートモジュール”，“Net-BTRON-HTML フィルタモジュール”，“Net-BTRON エンコードフィルタモジュール”，“Net-BTRON デコードフィルタモジュール”，そして “キャッシュマネージャ” である．インポートモジュールおよびエクスポートモジュールは Net-BTRON の標準データ転送プロトコルである HTTP1.1 に基づいている．キャッシュマネージャキャッシュリプレースアルゴリズムとして Least-Recently-Used アルゴリズムを採用した．

5.3 性能評価

Net-BTRON 分散ファイルシステムの性能を評価するために，実装したシステムでの実身のアクセスに要する時間を計測した．

Net-BTRON では実身のアクセスは *opn_fil* による実身オープンと *rea_rec* による実身データの読み込みの 2 つの API によって実行する．*opn_fil* は以下のような手順で行われる．

(1) オープンする実身の種類がローカル実身であるかリモート実身であるかを判定する．
(2) リモート実身の場合，3.7.2 項で述べた手順によって他のマネージャに実身の取得を依頼し，取得したデータを保存したローカル実身に対し

表 1 Net-BTRON ファイルシステムの性能計測結果
Table 1 Result of the performance experimentation of the Net-BTRON file system.

	opn_fil			*rea_rec*
	(1)	(2)	(3)	
ローカル実身	0	-	0.15(ms)	0.27(ms) + 0.0053(μs/byte)
リモート実身	0	8000(ms)	0.15(ms)	2.4(ms)

て *opn_fil* を実行する．

(3) ローカル実身の場合，実身をオープンし，*rea_rec* 等の実身アクセスを行うために必要な実身識別子を返す．

計測実験ではローカル実身とリモート実身に対して上記の *opn_fil* の各手順に要する時間と *rea_rec* に要する時間を計測しその平均値を求めた．実験は 128 MB のメモリと 400 MHz の PentiumII プロセッサを搭載したマシン上で行った．ローカル実身の計測では様々なサイズのローカル実身に対して複数回実験を行い得られた結果の平均値を求めた．リモート実身に対しては LAN 上に存在する 40 KB のリモート実身に対して毎回キャッシュをクリアして複数回計測を行いその平均値を求めた．**表 1** はその結果を表す．

Net-BTRON では *opn_fil* の手順 (1) は実身の種類を表すパラメータを判定することにより条件文 1 行によって瞬時に行うことができ，所用時間が計測不能であったため表中では 0 と記した．このことからリモートアクセスとローカルアクセスを統一的に扱うことによってローカルなアクセスに及ぼすオーバヘッドはほぼ無視することができることが分かる．

リモート実身では *opn_fil* の手順 (2) に要する時間はネットワーク回線の混雑状況等によって可変であり，同じリモート実身の取得に要する時間は一定ではない．しかし，手順 (2) では実際のデータをネットワークを通じて転送するため，最低でもデータ転送速度に転送するデータ量を乗じた時間が必要である．実際にはこれに加え，データ転送プロトコルで必要なデータ転送以外のオーバヘッドや得られたデータをローカルファイルに保存するための時間が必要となる．仮にデータ転送に 10 Mbps のパフォーマンスを得られた場合，1 バイトのデータを転送するためには約 0.76 μs/byte の時間が必要である．これに対し，表 1 が示すように *rea_rec* はデータ読み込み以外の処理に必要な 0.27 ms のオーバヘッドを除くと 0.0053 μs/byte という約 100 倍の速度でデータの読み込みが可能である．実際の計測データにおいても *opn_fil* の手順 (2) に要する時間と *rea_rec* によるデータ読み込みの時間を比較すると約 4000 倍の差があることが確認できた．このようにリモート実身の読み込みに必要な時間の大部分は Net-BTRON ファイルシステムとは直接関係のないネットワーク上のデータ転送によって消費される．このことから Net-BTRON ファイルシステムはリモートアクセスに対して十分なパフォーマンスを持つことが分かった．

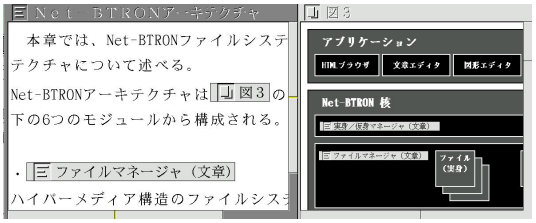

図 10 文章エディタ（左），図形エディタ（右）の画面例
Fig. 10 Sample screen shots of a wordprocessor (left) and a graphic editor (right) applications.

5.4 Net-BTRON ハイパーメディアアプリケーション

編集を支援するために，Net-BTRON の基本アプリケーションは，ブラウザではなくエディタとなる．エディタは，ほぼ WYSIWYG（what you see is what you get）に近いビューで，実身（ノード）を編集することができる．リンクの挿入も，仮身セグメントをマウスでドラッグアンドドロップによる複製操作によって行え，また削除も仮身セグメントを選択してから削除コマンドによって削除することができる．こうした簡単な GUI 操作によって気軽にリンクを編集しても，リンクの結合を Net-BTRON ファイルマネージャが管理しているため，リンクが切れるような不整合は生じない．ユーザは安心してリンクの付け替えやファイルの移動といった試行錯誤を行うことができる．

図 10 はそれぞれ Net-BTRON の文章エディタと図形エディタを表している．これらのアプリケーションはともに Net-BTRON の仮身を他の文字や図形オブジェクトと同様に扱うことが可能であり，ドキュメントの任意の場所にリンクを埋め込むことが可能となっている．同様に表計算やデータベース（**図 11**）に Net-BTRON のローカル/リモート実身また，WWW ノードを指す仮身を埋め込むことが可能である．さらに我々は，WWW と Net-BTRON を統合するために，Net-BTRON 上に WWW ブラウザを実装した

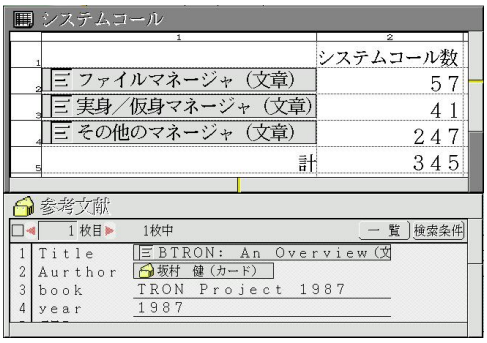

図 11 表計算（上），データベース（下）の画面例
Fig. 11 Sample screen shots of a spreadsheet (above) and a database (below) applications.

図 12 WWW ブラウザの画面例
Fig. 12 Sample screen shot of a WWW browser application.

（**図 12**）．Net-BTRON のハイパーメディア支援 API により，我々は基本ブラウザをも容易に実装することができた．

6. 関連研究と評価

これまで HMS の研究はミドルウェアアプローチを用いて実現する研究と OS によって実現する研究の 2 種類に分類することができる．ミドルウェアアプローチを用いた HMS としては Microcosm[8]，Chimera[9]，HyperDisco[10]，Hyper-G[11]，OS を用いて実現する HMS としては HOSS[12]や我々の Net-BTRON があげられる．

HOSS は構造コンピューティングという概念に重点を置いたシステムである．HOSS では OS がハイパーメディアデータの構造を定義し，直接管理することによりデータのプリフェッチ等のハイパーメディア構造に特化した様々なサービスを OS が行うことができる．Net-BTRON と HOSS の最大の違いはリンクサービスの実現方法である．HOSS は Illustra や Postgres 等のデータベースを用いたミドルウェアレベルアプローチを用いてリンクサービスを実現している．このため，ファイルシステムレベルでハイパーメディア構造を実現した Net-BTRON と比べ，低レベルなファイル操作に対する頑強性や本章で後述する効率の良いリンクのトラバースを持たない．

以下本章ではミドルウェアアプローチを用いた HMS と Net-BTRON との比較評価をリンクの頑強性，機能の比較，リンクのトラバースの性能の点から行う．

6.1 リンクの頑強性

本節では Net-BTRON のリンクの頑強性について述べる．Net-BTRON のリンクが指すノードに対して“ファイル名の変更”，“ファイルの移動”，“ファイルの消去”という 3 つのファイル操作が行われた場合についてそれぞれ述べ，他の HMS との比較を行う．

Net-BTRON ではリンクの頑強性はリンクが指す実身がローカルファイルであるか，リモートファイルであるかによって異なる．

6.1.1 ローカルファイルを指すリンクの頑強性

- ファイル名の変更
  Net-BTRON のリンクは，ファイルシステム名とファイル ID のペアで識別され，ファイル ID の値は実身が生成されてから削除されるまでの間は一定の値に保たれる．したがって，ファイル名やパス名が変更されても，リンクが切れることはない．この点において，UNIX 等におけるシンボリックリンクや，WWW システムの A タグにおけるアプリケーションレベルのリンクより頑強である．一般的な HMS では，リンクをデータモデルマネージャ等と呼ばれる独自のミドルウェアレベルのマネージャによって管理する．この方法では，データモデルマネージャを使ったファイル操作しか行わない場合にはリンクの頑強性が保証される．しかし，OS の生のシステムコール等，ミドルウェアが提供する API よりも低レベルの API がアプリケーションに開放されているため，ユーザの誤操作またはアプリケーションソフトウェアのバグ等によって，低レベルのファイル操作が行われたときに，一貫性が保たれないという問題が起こり，ファイル名の変更等についても頑強性を得ることができない場合もある．
- ファイルの移動
  Net-BTRON では，ファイル（実身）を移動する場合にも，そのファイルのファイル ID は変化することがないので，そのファイルを参照している

表 2 ハイパーメディアサービスシステム間のリンクの頑強性の比較
Table 2 Comparison of the robustness of the LINK between HMSs.

ノード操作	WWW	Microcosm	HyperDisco	Chimera	Hyper-G	Net-BTRON
名前変更	×	○	×	×	○	○
同一マシン内移動	×	○	×	×	○	○
他マシンへの移動	×	×	×	×	○	×
ローカルマシンでの削除	×	×	×	×	○	○
リモートマシンでの削除	×	×	×	×	○	×
低レベルな操作に対する頑強性	×	×	×	×	×	○

表 3 ハイパーメディアサービスシステム間の機能比較
Table 3 Comparison of the functions between HMSs.

機能	WWW	Microcosm	HyperDisco	Chimera	Hyper-G	Net-BTRON
実現箇所	AP	MW	MW	MW	MW	OS
転送プロトコル	固定	複数可	固定	固定	複数可	複数可
キャッシュ管理	AP	MW	MW	MW	MW	OS
UI 標準化	×	×	×	×	×	○
他 HMS との相互運用	×	○	×	×	○	○

リンクの接続性はそのまま継続される．したがって，Net-BTRON のリンクは，ファイルの移動に対して頑強である．

- ファイルの削除
  Net-BTRON 上のファイルはリンクの参照数を保持しており，そのファイルを参照するリンク数が 1 以上であれば，ファイル本体を削除することはない．参照数が 0 となったファイルはガーベジとして集められ，実装に依存した GC のアルゴリズムに従って削除される．したがって，Net-BTRON のローカルファイルシステムでは安全にファイルを削除するため，削除によって，リンクの頑強性が損なわれることはない．
  これに対し，他の HMS では一般にリンクはファイルシステムではなく，アプリケーションやミドルウェアシステムが管理するため，ハイパーメディアデータ内にリンクが存在している場合でも，そのリンクが指すファイルを直接削除することが可能であり，その場合リンクは切れてしまう．

6.1.2 リモートファイルを指すリンクの頑強性

Net-BTRON のリモートファイルはリモートな Net-BTRON 上のファイルの場合と他の Net-BTRON に統合された HMS 上のノード（たとえば WWW）がある．まず，リモートファイルが Net-BTRON のファイルを指す場合，リンク実身はリモートファイルが存在するホストの IP アドレスとファイル ID を用いてファイルを参照している．これにより，ファイル名の変更および，同一ホスト内でのファイルの移動に関してはローカルファイルと同様に Net-BTRON は頑強なリンクを提供することができる．また，今回のバージョンでは，リンク実身によるリモートファイルの参照はその実身のリンクの参照数に入れなかった．したがって，リモートマシンからリンクが張られている場合でも，ローカル参照数が 0 となれば，ファイルが削除される可能性があり，その場合，リモートからのリンクは切れてしまう．このため，ホスト間をまたぐファイルの移動はファイルの削除とファイルの移動の複合操作であるため頑強性を保持しない．

他の HMS のファイルへのリンクの頑強性は，その HMS に依存する．たとえば現在の実装では WWW 上のファイルに対してリンクをはることが可能であるが，WWW はリンクの頑強性を保証しないシステムであるため，Net-BTRON からの WWW 上のリモートノードへのリンクの頑強性は保証しない．

表 2 にハイパーメディアサービスシステム間のリンクの頑強性の比較をまとめた．名前変更，移動，削除の項目はそれぞれのノードに対する操作をシステムがサポートしているかどうかを表す．

6.2 ハイパーメディアサービス機能

本節では Net-BTRON のハイパーメディアサービス機能を他の HMS のそれと比較する（**表 3**）．

- 実現箇所：システムの実現箇所．AP はアプリケーション，MW はミドルウェアを表す．
- 転送プロトコル：転送プロトコルの種類．
- キャッシュ管理：リモートノードデータに対するキャッシュの管理の実現箇所．
- UI の標準化：ハイパーメディアユーザインタフェースの標準化に対するサポートの有無．
- 他 HMS との相互運用：他の HMS との相互運用の有無．

6.3 リンクのトラバースの性能

Net-BTRON はリンクの情報を実身内のリンクレコードと仮身セグメントの 2 つに分割して保存する．リンクレコードは実身間の接続を表すレコードであり，各リンクに対してそれぞれ 1 つのリンクレコードが実身内に割り当てられる．仮身セグメントはリンクの表示属性等，アプリケーションがリンクを表現するために必要なデータが格納され，アプリケーションデータレコード内に他のアプリケーションデータと混在して格納される．このように実身間の接続を表すデータをリンクレコードとして，アプリケーションデータから分離して扱うことにより，Net-BTRON ではリンクのトラバースを高速に行うことができる．具体的には *get_lnk* によってパス名からリンクを取得する際に，リンクレコードが指す実身を順にたどることによって高速にリンクをトラバースすることが可能である．

一方，HMS ではリンクはミドルウェアによって管理されているため，リンクのトラバースはミドルウェアにリンクのトラバースを依頼し，その結果を待つというアプリケーション間通信が必要となる．また，一般にリンクの情報はデータベースによって管理されるため，リンクのトラバースを行うためには，HMS がデータベースからデータを検索する必要がある．リンクのトラバースに必要な処理を比較した場合，Net-BTRON はミドルウェアアプローチで実現された HMS と比べ，プロセス間通信やデータベースの検索というオーバヘッドの高い処理を必要としないため，より効率的なリンクのトラバースを行うことができる．

7. ま と め

本論文は強固な分散ハイパーメディア資源管理機構を持つ OS である Net-BTRON の分散ハイパーメディアサービス機能について論じた．Net-BTRON は以下の 3 つの特徴を持っている．第 1 に頻繁なコンテンツ編集に対してコンテンツの整合性を強固に保つ資源保護管理機能を提供する．そのためにハイパーメディア的記憶管理をファイルシステムによって提供した．第 2 にハイパーメディアユーザインタフェースを実現するユーザインタフェースツールキットを OS で提供することで，ユーザインタフェースおよびシステムインタフェースをアプリケーション間で標準化することが可能となった．第 3 に，他のハイパーメディアシステムのコンテンツと Net-BTRON コンテンツを統合し，双方から参照しあうことを可能にした．これによってインターネット上の既存の HTML コンテンツをすべて Net-BTRON のコンテンツとして扱うことが可能となった．

本論文では我々の Net-BTRON が 3 つの目的を実現することができることを示した．また，我々は Net-BTRON のパイロットシステムを BTRON3 仕様 OS を拡張して実装し，その上に多くの分散ハイパーメディアアプリケーションを構築することによって，技術的特徴の有効性を示した．将来的にはノード管理機能部分の機能をより強化し，バージョン管理やリアルタイムなマルチユーザアクセス機能を加えることにより，CSCW（computer supported corporative work）ツールの基盤となるように発展させたいと考えている．

参 考 文 献

1) Nelson, T.H.: *Literary Machines 93.1*, Eastgate Systems (1993).
2) Davis, H., Hall, W., Heath, I., Hill, G. and Wilkins, R.: MICROCOSM: An Open Hypermedia Environment for Information Integration, Technical Report, University of Southampton. CSTR 92-1 (1992).
3) Nürnberg, P.J. and Aarhus, U.: Open Hypermedia Systems Working Group, http://bush.cs.tamu.edu/ohs/.
4) Sakamura, K.: *BTRON: An Overview*, TRON Project 1987, pp.75–82, Springer-Verlag (1987).
5) Sakamura, K.: *BTRON: Human-Machine Interface*, TRON Project 1987, pp.83–96, Springer-Verlag (1987).
6) Sakamura, K.: BTRON1 標準プログラミングハンドブック，パーソナルメディア (1992).
7) Callaghan, B., Pawlowski, B. and Staubach, P.: NFS Version 3 Protocol Specification (1995).
8) Hill, G. and Hall, W.: Extending the Microcosm Model to a Distributed Environment, *Proc. 1994 ACM European Conference on Hypermedia Technology*, p.32, ACM (1994).
9) Whitehead, E.: An Architectural Model for Application Integration in Open Hypermedia Environments, *Proc. 8th ACM Conference on Hypertext, 1997*, pp.1–12, ACM (1997).
10) Wiil, U. and Leggett, J.: WorkSpaces: The HyperDisco Approach to Internet Distribution, *Proc. 8th ACM Conference on Hypertext*, pp.13–23, ACM (1997).
11) Andrews, K., Kappe, F. and Maurer, H.: The Hyper-G Network Information System, *Universal Computer Science*, Vol.1, No.4, pp.206–220 (1985).
12) Nürnberg, P., Leggett, J. and Schnase, J.: Hypermedia Operating Systems: A New Paradigm for Computing, *Proc. 7th ACM Conference on*

HYPERTEXT '96, pp.194–202, ACM (1996).

(平成 12 年 8 月 22 日受付)
(平成 13 年 3 月 9 日採録)

重定 如彦（学生会員）

昭和 46 年生．東京大学大学院理学系研究科情報科学専攻．ユーザインタフェースソフトウェア，オペレーティングシステム等の研究に従事．IEEE，ACM 各会員．

越塚 登（正会員）

東京大学情報基盤センター助教授．1994 年東京大学大学院理学系研究科情報科学専攻博士課程修了．博士（理学）．ユーザインタフェースソフトウェア，オペレーティングシステム，デジタルミュージアム等の研究に従事．IEEE，ACM 各会員．

坂村 健（正会員）

東京大学大学院情報学環教授．1984 年より TRON プロジェクトリーダーとして新しい概念に基づくコンピュータ体系の構築に精力を注ぐ．TRON は現在，携帯電話，デジタルビデオ，エンジン制御等，組み込みシステムの世界で最も使われている OS である．さらに最近はプロジェクトの最終目的である超機能分散システムの研究をすすめている．IEEE Micro の EIC（Editor-in-Chief）．