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

2004－OS－97　(11)
2004／8／1

ネットワーク上に動的に分散する多数のデバイスを制御する基盤ソフトウェアの検討

尾崎亮太[†]　丸山勝巳[†,††]
日高宗一郎[†,††]　児玉和也[††]

ネットワークを介してアクセスできる計算機の入出力デバイスを活用するための技術に対する要求が高まっている．またモバイルデバイスの普及により，ネットワーク的な移動に対する透過性が必要になっている．そのような機能は基盤ソフトウェアの基本的なサービスの一つになりつつある．そこで本研究では，遠隔アクセス・コモンインタフェース・位置透過性という 3 つの要件を実現するシステムについて検討を行なう．上記要件から実現すべき機能を抽出し，システムのモデルを基に遠隔デバイスアクセスのためのプロトコルの設計を行なう．本システムを用いることにより，ネットワークを意識しない遠隔デバイスアクセスサービスを利用することが可能となる．またミドルウェアでの実現との比較を行ない，システムの特徴と違いを明らかにした．応用について例をあげ，本システムの発展性について言及した．

Network Extended Device Management System Supporting Common Interface and Location Transparency

Ryota OZAKI,[†] Katsumi MARUYAMA,[†,††]
Soichiro HIDAKA[†,††] and Kazuya KODAMA [††]

There has been inscreasing demand for accessing the I/O device attached to computers on network. Location transparency is also needed for widespread of use of mobile devices. Such a function is becoming fundamental services of infrastructure software. In this study, we have investigated the infrastructure software realizing *remote device accessing* capability with *common interface* and *location transparency* features. We have extracted functions from the above requirements, and have designed the protocol for remote device accessing, based on the model we have defined. The our system enables remote device accessing service with network transparent manner. We have compared a difference between implementation by middleware and infrastructure software, and have mentioned possibilities of our system by mentioning applications using our system.

1. はじめに

ネットワークを介してアクセスできる計算機の資源を有効に活用するための研究は 80 年代にネットワーク接続技術が普及し始めたことから盛んに行なわれてきた．当時は計算機性能が低く入出力デバイスも少なかったため，資源の対象は計算資源（プロセッサやメモリ）であった．しかし計算機性能の向上により，十分な性能を持つ時代となった今では通信を行ないネットワーク上の計算資源を利用することのメリットは小さくなった．膨大な計算性能を必要とするハイパフォーマンスコンピューティング分野ではネットワーク上の計算資源を活用する必要性がなくなることはないが，日常的な用途ではローカルの計算資源で十分であると言える．

一方現在は，ネットワークを介してアクセスできる計算機に接続された入出力デバイス（以下遠隔デバイスと呼ぶ）を活用するための技術に対する要求が高まってきている．人間が日常的空間でいついかなるときでも計算機資源を活用できるような環境を実現するための研究開発が盛んに行なわれている[1),8),12)]．人間は持ち歩けるような小型デバイスを持ち，周りの環境に配置されたより便利なデバイスを利用するような状況が一つのシナリオとして考えられている[12)]．そのような状況では，遠隔デバイスを利用できる仕組みが当然備わっていることが求められる．本研究ではこのような要求に応えるような基盤ソフトウェアについての

† 総合研究大学院大学 情報学専攻, Department of Informatics, The Graduate University of Advanced Studies.
†† 国立情報学研究所, National Institute of Informatics.

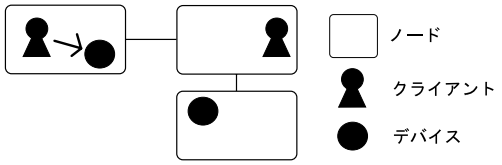

図 1 システムモデル

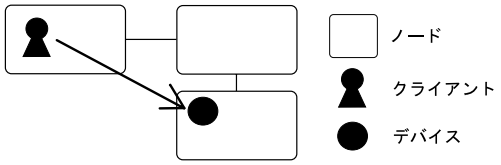

図 2 遠隔アクセス

検討を行なう．

ネットワーク上の計算機の資源を活用するための分散処理技術は数多く研究がなされてきた[1),3),4),8),12)]．また UPnP[11)] などネットワーク接続されたスキャナやプリンタなどの周辺デバイスを計算機に直接接続されているように扱うための技術仕様が策定されている．しかし UPnP では遠隔の計算機のデバイスは対象としておらず，ミドルウェアによる分散処理技術では新たなインタフェースによるプログラミングを要求し敷居が高い，などの問題点が存在する．広く普及し容易に利用でできるインタフェースは使いやすいシステムにとって重要な要素である[2),3)]．

本研究では遠隔の計算機の入出力デバイスをローカルのデバイスとして扱えるようにするためのシステムを構築することを目的とする．またそのような利用法を維持しつつ計算機資源のネットワーク上の移動にも対応する手法を提案する．本システムを利用すると，デスクトップ PC に接続された入出力デバイスを近くにあるノート PC から利用することができるようになる．

本稿では以上のような特徴を持つシステムを実現するための要件を述べ，それを基にシステムの設計を行なう．次章でシステムのモデル化とシステム構成を定義し，ネットワーク接続を維持するプロトコルを示す．3 章ではミドルウェアでの実現方法との比較とシステムの応用について考察を行なう．4 章で関連研究と比較を行ない本研究の位置付けを明らかにする．そして 5 章で本稿をまとめる．

2. システムの設計

この章では，遠隔デバイス管理機構の設計を行なう．最初に諸要素をモデル化する．その後システムの要件を基に必要な機能を明らかにし，システム構成を設計する．その後要件を実現するプロトコルを設計する．

2.1 システムモデル

まずシステムのモデルを示す．以下の章ではこのモデルを基に説明を行なう．

本モデルはノード・デバイス・クライアントの 3 つのオブジェクトで構成されている（**図 1**）．**ノード**とはネットワーク接続機能をもつ計算機である．ネットワークを構成する要素であり，それ自身が移動しネットワークの構成を変化させる．**クライアント**はノードで実行されるプログラムであり，**デバイス**はノードに接続された入出力デバイスである．クライアントはデバイスからの入力を利用し，デバイスへの出力を生成する．ノードはクライアントとデバイス間のデータの橋渡しを行なう．クライアントとデバイスはノードに所属しており他のノードへ移動することはない．

本システムのモデルは特定のハードウェアやネットワークに依存しない．クライアントがノードを介してデバイスへアクセスするというモデルのみを規定する．例えばノードはデスクトップ PC やモバイル端末などになる．その場合デバイスは PC やモバイル端末へ接続された入出力デバイスとなる．またノードを結合しているネットワークは Ethernet を通信媒体とする LAN やアドホック無線ネットワークなどである．

2.2 システムの要件

ネットワークという隔たりを意識せず，ネットワーク上の計算機にある入出力デバイスを扱うことができる環境が求められている．特に日常的な状況ではその要求は大きく，目の前にある計算機の入出力デバイスを簡便に扱えないことは人々の大きな不満となっている．またあらゆるアプリケーションプログラムがその環境を利用できる必要がある．特定のプログラムのみの対応では利用者の利便性はあまり高くならない．独自のインタフェースでなく広く普及している手法が良い．

本研究では**図 2** のような遠隔アクセスサービスを基礎とし以下の 2 つの特徴を有するシステムを設計する．

コモンインタフェース 現在広く普及している手法でデバイスにアクセスできる．**図 3** のようにローカルのノード内のデバイスに対してアクセスするように遠隔デバイスをアクセス可能になる．また遠隔アクセスのサービスを OS の基本サービスとして実現することで，OS を利用するすべてのクライアントがサービスを受けることが可能となる．

位置透過性 ネットワークの構成の変化に対して透過的にサービスを提供し続けることができる．**図 4**

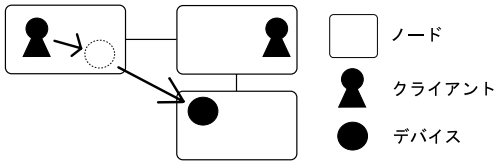

図 3 コモンインタフェース

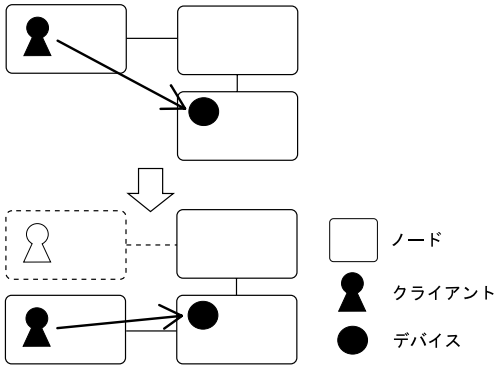

図 4 位置透過性

のようにクライアントがいるノードやデバイスが接続されているノードが移動しても，クライアントやデバイスはサービスを継続することが可能となる．

我々は遠隔デバイスへのアクセスをファイルインタフェースとして実現する．近代の OS はデバイスにアクセスするサービスをファイルを扱うインタフェースと同じようなインタフェースを提供している[5),10)]．

本モデルではネットワークの移動にだけ焦点を当てると，クライアントとデバイスは等価なものとして扱われる．クライアントがいるノードとデバイスが接続しているノードのどちらが移動してもサービスの透過性は維持される．

2.3 本研究の扱う範囲

ここで本研究で扱う範囲を明確にする．

2.3.1 ネットワーク

本研究ではネットワークプロトコルに関しては既存のものを利用する．既存のネットワークの上位層としてシステムを構築する．下位のネットワークとしてはポイントツーポイントのデータ転送機能を提供しており，データが確実に送信先へ届けられることを保証しているものであるならばどのようなものでもよい．例えば TCP/IP などが考えられる．

2.3.2 デバイスの種類

本研究ではマウスやキーボードなどデバイスが外部事象により新たなデータを生成する（またはデータを外部事象に変える）ようなデバイスのみを対象とする．ハードディスクのようなストレージデバイスは対象としない．なぜならば，そのようなデバイスで扱われる入出力データは永続的でローカルのストレージにコピーすることにより遠隔デバイスへの継続的なアクセスを必要としないからである．

2.4 実現すべき機能

ここでは 2.2 節で述べた要件を実現するためのシステムに必要な諸機能について述べる．

（1）遠隔デバイスの存在を知る．
　どのノードにどんなデバイスが存在するかが判らなければならない．

（2）遠隔デバイスへアクセスするための仕組み．
　（2-1）クライアントとデバイス間でやり取りされるデータをネットワークを介して転送するための仕組みが必要となる．（2-2）また外部のデバイスを取得利用するための手続きを規定しなければならない．

（3）遠隔デバイスをローカルにあるデバイスに見せる仕組み．

（4）ノード移動時の一時的なネットワーク切断とシステムからの完全な離脱への対応．
　（4-1）ノードの移動が起きてもクライアントとデバイスの接続が途切れないための仕組みが必要である．（4-2）また一時的なネットワーク断線時におけるデータバッファリングが必要である．（4-3）デバイスの入出力がないときとノードのシステムから離脱していることを区別できなければならない．（4-4）一時的な断線と離脱を扱えるように（2-2）の手続きの拡張が必要である．

2.5 システム構成

以上を考慮して我々は**図 5** のようなシステム構成を提案する．本システムはリレー・バッファ・情報管理・状態管理という 4 つのサブシステムで構成されている．本システムは OS の内部で実現されている．本システムのサービスは OS の API でクライアントへ提供される．デバイスの直接の制御は基本的に OS のデバイスドライバが行なう．また下位ネットワークプロトコルを利用して他のノードと通信を行なう．

これ以降は説明を簡単にするために一つのクライアントが存在するノードと一つのデバイスが接続されているノードに注目する．クライアントが存在するノードをクライアントノード，デバイスが接続されているノードをデバイスノードと呼ぶことにする．またクライアントがあるデバイスを使用中であるとき，クライアントとデバイスは**接続している**ということにする．

2.5.1 デ ー タ

サブシステムの役割を説明する前に本システムが扱

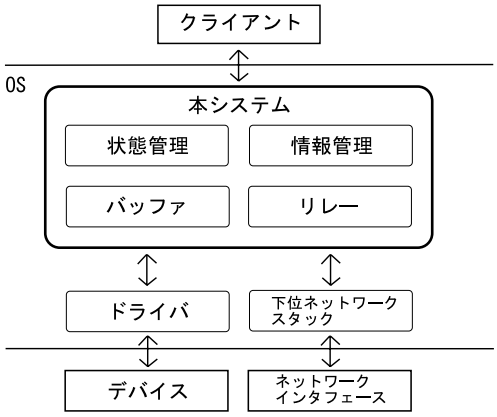

図 5 システム構成

うデータについて説明する．

クライアントハンドル/デバイスハンドル クライアント/デバイスの識別子．ネットワークカードの MAC アドレスと日時を組合せ生成される UUID（Universally Unique Identifier）を使う．デバイスハンドルは本システムの内部で利用されるもので，クライアントが直接扱うものではない．

ネットワーク ID ノードの識別子．下位層のネットワークの識別子が用いられる．TCP/IP ならば IP アドレスである．

デバイス属性 入力デバイスか出力デバイスか，複数のクライアントを許すか，使用許可情報，などの情報がある．使用許可情報はネットワーク ID などをもとにつくられる ACL（Access Control List）で管理される．

デバイスタイプ 具体的なデバイスの種類に関する情報である．これはクライアントにデバイス情報を掲示するときに利用される．

2.5.2 サブシステムの詳細

ここでは個々のサブシステムの詳細について述べる．

リレーサブシステム 下位層のネットワーク機構を利用し，デバイスとクライアント間でやり取りされるデータを転送する．デバイスハンドルとクライアントハンドル，およびそれらが所属するノードに対応するネットワーク ID を管理している．ハンドルからネットワーク ID を割り出し下位層へ通信要求を行なう．他のサブシステムはデバイス/クライアントハンドルを指定するだけで通信を行なうことができる．また定期的にビーコンを送受信し通信相手の存在を確認する．そして通信のタイムアウトを判定する役割を持つ．ビーコンとタイムアウトに関する詳しい説明は 2.6.1 節で行なう．

バッファサブシステム クライアントとデバイスの入出力データをバッファリングする役割を持つ．デバイスハンドルとクライアントハンドルとそれぞれに対応するデータバッファを管理する．

情報管理サブシステム ローカルにあるデバイスと遠隔デバイスの情報を管理する．デバイスハンドルとそのデバイス属性とタイプを管理する．

状態管理サブシステム クライアントとデバイスの状態を管理し，プロトコル（詳細は 2.7 節で説明する）に従い適切な処理を行なう．また接続しているクライアントとデバイスも管理する．情報管理サブシステムから情報を得て，デバイスへの接続要求の判定なども行なう．デバイスハンドルとクライアントハンドル，およびそれぞれの接続情報，またデバイスとクライアントの状態などを管理する．

2.6 機能の実現方法

2.4 節で述べた機能の実現方法を示す．

（1）本システムでは基本的な機能だけを提供する．詳しい説明は 2.6.3 節で示す．

（2-1）クライアントとデバイスの入出力データを管理し，下位ネットワークで転送することで実現する．

（2-2）PnP（Plug and Play）[5] の手続きをネットワークに拡張したプロトコルを用いる．詳しくは 2.7 節で示す．

（3）システムを OS 内部で実現しローカルのデバイスと同じように振る舞わせることで実現する．既存の着脱可能デバイスと利用方法を合わせることで，クライアントはコモンインタフェースを利用することができる．

（4-1）状態管理サブシステムがクライアントとデバイスの接続を管理することで実現する．

（4-2）バッファサブシステムでデータをバッファリングすることで，上位層にネットワークの一時的断線を隠蔽する．

（4-3）ビーコンを用いてノードの存在を定期的に確認することで，ノードが存在するかしないかを判別する．詳しい説明を 2.6.1 節で行なう．

（4-4）2.7 節で示す．

以下ではプロトコル処理の説明の前に必要な機能と処理内容に関して述べる．

2.6.1 ビーコンとタイムアウト

ビーコンは接続しているデバイスノードやクライアントノードがシステム内に存在するか否かを判断するためのものである．一定間隔で自分の存在を通知するデータを接続相手に送信する．ビーコンを送信してこ

ないノードはシステム内に存在しないと判断される．（以降ノードがシステムから離れたことを**離脱**と呼ぶ．）この機構はリレーサブシステム内で完結しているため 2.7 節で述べるプロトコルが簡潔なものになっている．

本システムで扱うタイムアウトには 2 種類あり，一つは上記ビーコンのタイムアウト，もう一つは下位ネットワークプロトコルの通信の失敗を示すタイムアウトである．後者のタイムアウトはリレーサブシステムが検出し，再送などを行なう．前者のタイムアウトは本システムで指定する期間を越えた場合にリレーサブシステムが状態管理サブシステムに対し発生させる．以降の説明では特に断りがない限り，タイムアウトといえばビーコンのタイムアウトのことを指すこととする．

2.6.2 ノードの移動に関する処理

デバイスノードやクライアントノードの移動には以下の手順で対応する．

ノードが移動するとネットワーク ID が変わるため，そのノードへの下位ネットワークの通信は失敗する．リレーサブシステムは失敗を検出すると，定められたタイムアウト期間の間待つ．システムに再接続した場合ノードは自身の存在を全ノードに通知する．リレーサブシステムはその通知を検出すると，ネットワーク ID を更新して通信を再開する．この処理はリレーサブシステム内で完結しているため，他のサブシステムはノードが移動したことを考慮する必要はない．

しかしタイムアウト期間を過ぎてもノードからの通知がなかった場合は，リレーサブシステムはノードが離脱したとみなし，状態管理サブシステムへその旨を通知する．

2.6.3 デバイス情報の広告と収集

この処理は WS-Discovery[13] プロトコルの Hello や Probe メッセージのような働きをする．

- デバイス情報の広告
  デバイスノードがシステムのネットワークに接続するときに自身のデバイス情報を全ノードへ通知し，システムに参加しているノードにデバイスの存在を通知する．
- デバイス情報の収集
  クライアントノードはシステムに参加するときにデバイスの情報を収集する．参加ノードにデバイス情報を教えてもらうためのリクエストを全ノードへ通知する．
- デバイス情報の公開
  クライアントノードでは取得したデバイス情報とともにデバイスの存在をクライアントに公開する．デバイスタイプを公開することでクライアントはデバイスの種類から使いたいデバイスを選ぶことができる．この機能は非常に簡単なサービス選択インタフェースである．高度なサービス探索については本研究では扱わない．

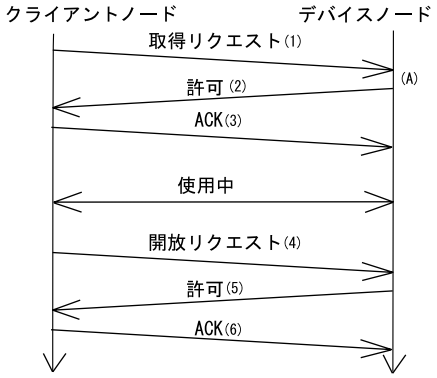

図 6 デバイス取得・使用・開放手続き（理想的な状況）

2.7 本システムで扱うプロトコル

ここではデバイス取得処理とデバイス開放処理に分けて説明する．このプロトコルは状態管理サブシステムで扱われるものである．

デバイス取得・使用・開放手続きの最も理想的な（つまり例外が起きない）手順を**図 6** に示す．また離脱を含めたノードの状態遷移図を**図 7**, **図 8** に示す．

本システムのプロトコルは PnP の手法を参考にしている．

2.7.1 プロトコルの概説

図 6 のシーケンスにしたがって説明する．クライアントはデバイス情報をもとにコモンインタフェースを利用して，デバイスの使用許可を取得するための要求を出す．

（1）デバイス取得要求
クライアントが指定したデバイスは内部でデバイスハンドルに変換される．そのデバイスハンドルを用いてデバイスノードへ取得要求を送信する．ネットワーク ID はリレーサブシステムが割り出す．

（A）,（2）デバイス使用許可判定
デバイスノードではデバイスの使用状態やクライアントノードのネットワーク ID などをもとに使用を許可するか否かを判定する．使用可能であるならば使用要求を受理した旨を，使用不可であるならば要求を拒否した旨をクライアントノードへ返信する．

（3）ACK の返却と接続の確立
デバイスノードからの返信に対しクライアントノードは ACK を返す．これによってデバイス使

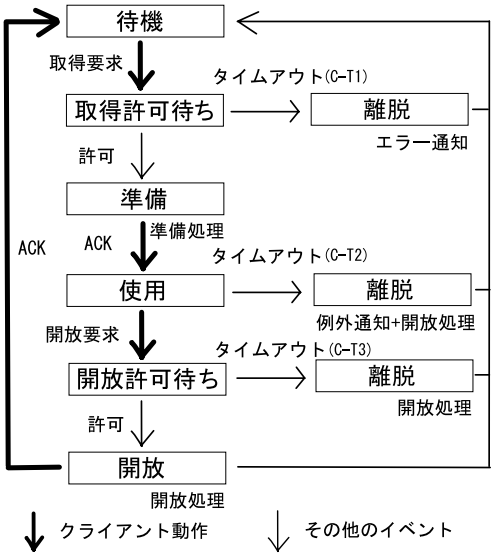

図 7 クライアントノード状態遷移図

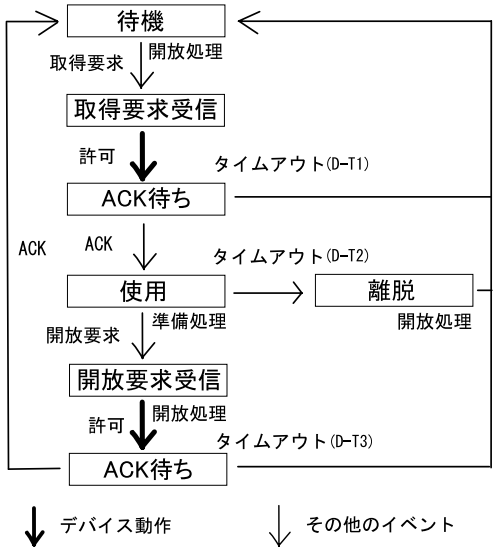

図 8 デバイスノード状態遷移図

用要求は完了する．デバイスノードは ACK の受信をもって接続の確立を確定する．

（4）～（6）開放処理

開放処理は取得処理と同様の処理が行なわれる．

2.7.2 ノードの離脱に関する処理

図 7, 図 8 を用いて説明する．

- デバイスノードの離脱
  - （C-T1）クライアントに取得失敗のエラーを返して待機状態へ戻る．
  - （C-T2）開放処理を行ない，クライアントに突然デバイスが抜き取られたという例外を返却して待機状態へ戻る．
  - （C-T3）開放処理を行ない待機状態へ戻る．
- クライアントノードの離脱
  - （D-T1）単に待機状態へ戻る．
  - （D-T2）開放処理を行ない待機状態へ戻る．
  - （D-T3）同上．

2.8 ま と め

本章ではモデルを立て，現在の基盤ソフトウェアに求められる要件をもとに遠隔デバイスアクセスのためのシステムの設計を行なった．

以下では本節で扱わなかった諸問題について取り上げる．

2.8.1 デバイスの同時利用

本研究の設計では一つのデバイスに対し一つのクライアントしか接続しないという前提になっている．デバイスによっては一つのデバイスに対して複数のクライアントを持つことができる．例えばキーボード入力は複数のクライアントが受け取っても不都合はない．このような状況を考慮したプロトコルの改編が必要となる．

2.8.2 デバイス探索

本システムではデバイスの探索については基本的な機能のみ提供する．これは複雑なデバイス探索機能を持つことはシステムを過度に複雑にしてしまうからである．高度な探索機能はミドルウェアなどの上位のシステムが提供すべきである．例えば STONE[7] などのシステムを利用することが考えられる．

3. 議　　論

3.1 ミドルウェアでの実現との比較

ミドルウェアでの実現では OS の上に新たなレイヤをつくる形で分散システム環境をアプリケーションプログラムに提供する．OS が提供するデバイスアクセスサービスを用いデバイスの入出力を得る．また OS が提供するネットワークサービスを用い他の計算機のミドルウェアと通信を行なう．

遠隔計算機の資源を活用するような研究の中で，ミドルウェアで実現する研究は数多く存在する[1)~4),8),12)]．そこでミドルウェアにおける実現との比較を行なう．ここでは以下の 3 つの要素について考察を行なう．

（1）インタフェース

（2）通信機構

（3）デバイス制御

（1）ミドルウェアで提供するインタフェースとはアプリケーションプログラムに対するインタフェースである．アプリケーションプログラムが最初から抽象

化されたインタフェースを用いて作られているならば，既存のアプリケーションプログラムを利用できる．しかしそうでなければアプリケーションプログラムを書き換える必要が生じる．またシステムによっては独自プログラミング環境を提供するものもあり，その場合インタフェースも独自のものとなることが多い．それに対し OS での実現方法におけるインタフェースはミドルウェアやアプリケーションプログラムに対するものである．インタフェースとその使用方法を変えなければ上位のプログラムの変更無しにサービスを利用することが可能である．

（2）ミドルウェアにおける実現方法では通信機構はオペレーティングシステムや他のミドルウェアが提供するものを利用できるため，柔軟で実現は容易である．例えば UNIX ソケットや Java RMI や CORBA ORB などが利用される．しかしデータが一度ミドルウェアのレイヤにコピーされるため処理効率はよくない．対して OS の場合は，通信は OS それ自身のものを利用される．アプリケーションプログラムが利用する OS のインタフェースからの呼出しと異なるため，利用するのはやや難しい．L4[6] や Mach などのマイクロカーネルを利用する OS でネットワークコンポーネントが分離されて実現されているならば，ネットワーク機能を IPC で利用できるため実現は難しくない．また高レベルな通信ライブラリを使うことができない．

（3）ミドルウェアではデバイスを直接制御することはできない．OS が提供するある程度加工されたデータを利用する．そのために入出力デバイスに合わせた細かいチューニングなどは難しい．OS の場合，既存のデバイスドライバをそのまま利用することもできるし，システムに合わせて改良を加えることができ柔軟性は高い．

以上のように両手法にはそれぞれ一長一短がある．実現の容易性はミドルウェアによる実現方法が優れており，対応できるアプリケーションプログラムの範囲や処理効率においては OS による実現方法が優れているといえる．

3.2 応　　用

ここでは本システムの応用について述べる．

3.2.1 プロセスマイグレーション補助

プロセスマイグレーションとはある計算機のプロセスを実行状態を保持したまま別の計算機に移送する技術のことである．マイグレーション元のデバイスがマイグレーション先の計算機に存在するように見せることが可能となる．それによりハードウェア資源を直接操作するようなプロセスのマイグレーションが容易になる．プロセスマイグレーション技術ではプロセスの環境を移送先に再現することが重要である．ハードウェア環境を移送先に再現することは難しいが，本システムを用いることでそれが可能となる．

3.2.2 ソフトウェア PC スイッチ

本システムを導入するとネットワークで結合された複数の計算機はその中の計算機がもつ入出力デバイスを共有することが可能となる．そのため現在のハードウェア PC スイッチの代わりの役割を果たすことが可能となる．そしてソフトウェア的な実現であるため柔軟性が高くスケーラビリティも期待できる．クラスタシステムなどのフロントエンドマシンのキーボード入力を計算ノードにブロードキャストすることですべての計算ノードをキーボードで操作することが可能になる．

4. 関 連 研 究

STONE[7] はサービスの表現方法や合成などに関する研究であり，ハードウェア操作には踏み込んでいない．本システムとは相補関係にあると言える．

UPnP[11] は Microsoft をはじめとする企業間で規定した，家電製品などのデバイスを相互に機能を提供しあうための技術仕様である．しかし個々のコンピュータ内部のデバイスを共有することは想定されていない．また DHCP，HTTP，SOAP などの技術を必要とする．

Personal Server[12] はクライアントが持ち歩くことを前提とした個人データが入った小型デバイスであり，身近にあるリッチなデバイスを活用するようなコンセプトハードウェアである．UPnP 技術を前提としており同技術と同様の制約がある．

ゆかりプロジェクト[14] は個々の家電の機能を抽出・合成し利用することを可能とする基盤システムである．デバイスの対象が家電であり，本システムのように計算機内のデバイスを対象とはしていない．

Solelc[9] は分散 OS であり，割込みを抽象化したイベントとして別ホストへ転送することが可能であるため，遠隔ホストのデバイスを操作できる．しかしネットワーク上のノードの移動には対応していない．

5. お わ り に

遠隔デバイス操作技術は今後ますますその必要性を増やしていくだろう．特に計算機がより身近になるに従い入出力デバイスの多様さが増してくると思われる．またそれらの入出力デバイスを使いやすいものにするためには無線によるアドホックなネットワーク接続は

必要不可欠である．

本研究では要件を基にコモンインタフェース・位置透過性という特徴を持つ遠隔デバイスアクセスを実現するシステムの検討を行なった．モデルを構築し，実現のための機能を明らかにし遠隔デバイスアクセスのためのプロトコルの設計を行なった．またミドルウェアでの実現との比較を行ない，システムの特徴と違いを明らかにした．応用について例をあげ，本システムの発展性について言及した．本システムを利用することで遠隔の入出力デバイスの管理が容易になり，アプリケーションプログラムの開発は促進されるであろう．

現在はプロトタイプの実装を行なっている．今後は，本稿では扱わなかった問題について検討を行なう．デバイス一つに対して複数のクライアントを持つデバイスの扱うことができるようにプロトコルを改良する，クライアントとデバイスのノード間の移動を許すモデルとシステムの設計を行なう，などが挙げられる．

参 考 文 献

1) Brumitt, B., Meyers, B., Krumm, J., Kern, A. and Shafer, S.: EasyLiving: Technologies for Intelligent Environments, *Proc. 2nd International Symposium on Handheld and Ubiquitous Computing*, pp.12–27, September (2000).
2) Dasgupta, P., Itzkovitz, A. and Karamcheti V.: Active Files: A Mechanism for Integrating Legacy Applications into Distributed Systems, *Proc. 20th International Conference on Distributed Computing Systems (ICDCS 2000)*, pp.680–690, April (2000).
3) Goldman, K.J., Anderson, M.D. and Swaminathan, B.: The Programmers' Playground: I/O Abstraction for Heterogeneous Distributed Systems, *Proc. 27th Hawaii International Conference on System Sciences (HICSS-27)*, pp.363–372, January (1994).
4) Johanson, B. and Fox, A.: The Event Heap: A Coordination Infrastructure for Interactive Workspaces, *Proc. 4th IEEE Workshop on Mobile Computing Systems and Applications*, pp.83–93, June (2002).
5) クリス カント: WDM デバイスドライバ Windows98/2000 のための新しいドライバモデル, 翔泳社 (2000).
6) Liedtke, J.: On μ-Kernel Construction, *Proc. 15th ACM Symposium on Operating System Principles*, pp.237–250, December (1995).
7) 南 正輝, 杉田 馨, 森川博之, 青山友紀: ユビキタス環境に向けたインターネットアプリケーションプラットフォーム, 電子情報通信学会論文誌, Vol.J85-B, No.12, pp.2313–2330, December (2002).
8) Roman, M. and Campbell, R.H.: A User-Centric, Resource-Aware, Multi-Device Application Framework for Ubiquitous Computing Environments, *Technical Report: UIUCDCS-R-2002-2284 UILU-ENG-2002-1728*, July (2002).
9) 芝 公仁, 大久保英嗣: 分散オペレーティングシステム Solelc の設計と実装, 電気情報通信学会論文誌, Vol.J84-D1, No.6, pp.617–626, June (2001).
10) Tsegaye, M. and Foss, R.: A comparison of the Linux and Windows device driver architectures, *ACM SIGOPS Operating Systems Review*, Vol.38, pp.8–33, April (2004).
11) Universal Plug and Play. `http://www.upnp.org/`
12) Want, R., Pering, T., Danneels, G., Kumar, M., Sundar, M. and Light, J.: The Personal Server: Changing the Way We Think About Ubiquitous Computing, *Proc. 4th International Conference on Ubiquitous Computing (UbiComp 2002)*, LNCS 2498, pp.194–209, September/October (2002).
13) Web Services Dynamic Discovery. `http://xml.coverpages.org/WS-Discovery20040217.pdf`
14) 山崎達也, 沢田篤史, 多鹿陽介, 大倉計美, 中尾敏康, Shirazi, M.N., 佐野睦夫, 金田重郎: ゆかりプロジェクトにおける分散協調基盤ミドルウェア — YUKARIプロジェクト報告 No2—, 情報処理学会第 66 回全国大会, Vol.5, No.3TTC-3, pp.9–12, March (2004).