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

2005－MBL－35（25）
2005－ITS －23（25）
2005／11／18

携帯端末を想定したUSB機器のリモートプラグ＆プレイの検討

桑原純吾[†]　峰野博史[††]　田中希世子[‡]
鈴木偉元[‡]　石川憲洋[‡]　水野忠則[††]

携帯端末の高性能化と短距離無線通信技術の発展により，携帯端末が周辺機器とのダイレクトな通信端末となる要求が高まっている．しかし，既存の周辺機器は計算機に直接接続された形での利用しか想定されていない．そこで我々は，様々な種類の周辺機器のインタフェースとして利用されているUSB に着目し，携帯端末に直接 USB 機器接続することなく，ネットワークを介して USB 機器を利用する方法 (intelligent USB）を提案する．また，USB 機器のネットワーク越しのプラグ&プレイ（リモートプラグ&プレイ) についても検討し評価した．実験より十分な帯域と低遅延の環境であればリモートプラグ&プレイに要する処理時間は短く，有効であることを確認した．

A Study on Remote Plug and Play of USB Equipment that Assumes Portable Terminal

Jungo Kuwahara,[†] Hiroshi Mineno,[††] Kiyoko Tanaka,[‡] Hideharu Suzuki,[‡]
Norihiro Ishikawa[‡] and Tadanori Mizuno[††]

The demand that the portable terminal becomes a direct communication terminal with peripherals has risen by making of the portable terminal and efficient the development of the short distance wireless communication technology. However, only the shape connected directly with the computer use is assumed as for existing peripherals. Then, we propose method (intelligent USB) of using the USB equipment without carefully to USB used as an interface of a variety of kinds of peripherals, and connecting the USB equipment directly with the portable terminal through the network. Moreover, plug & play (remote plug & play) on the network of the USB equipment was examined and evaluated. It was confirmed to remote plug & play that it was short, and effective at the required processing time if it was an environment of a band and low latency that was more enough than the experiment.

1. はじめに

近年，Windows Mobile OS や Symbian OS を搭載した携帯端末の登場に見られるように，携帯端末の高機能化が進んでいる．また，IrDA，Bluetooth，ZigBee 等の短距離無線通信技術も進歩してきた．このように携帯端末と無線通信技術の発展に伴い，携帯端末を使った情報家電の遠隔操作やカメラ画像の印刷など，携帯端末が周辺機器とのダイレクトな通信端末となることが現実味を帯びてきた．そこで筆者らは，新たな PAN 利用形態として mobile Personal Area Network（mPAN）を提案している[1]．mPAN とは，携帯端末が周りに存在する周辺機器と PAN を構成し，移動先でもサービス起動可能なサービスモビリティを実現するためのコントロールポイントとなるネットワークサービスである．

mPAN では，インタフェース数の制限がある小型な携帯端末からでも PAN を構成し，ネットワークを介してテレビやスピーカーといった周辺機器を利用することで，様々な高品質な入出力機器を利用可能となる．一方，現在では様々なインタフェース（USB，SCSI，IEEE1394 など）の周辺機器が登場している．一般にこれらの周辺機器は計算機に直接接続した形で利用される．そのため，携帯端末がこれらの周辺機器を利用する場合，各々のインタフェースを持つ必要がある．しかし，小型な携帯端末に全てのインタフェースを持たせるのは困難である．そこで，我々は UPnP[4] や iSCSI[5] に代表されるようにネットワークを介して周辺機器を制御する技術に注目している．ただし，これまでの既存技術では専用アプリケーションに特化していたり，特定のデバイスの機能，対応デバイ

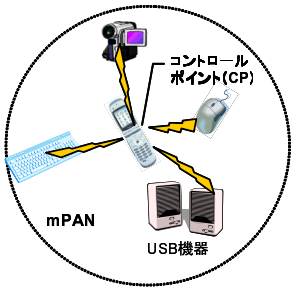

図 1　mPAN 内の USB 機器透過的利用

† 静岡大学大学院情報学研究科
Graduate School of Infomatics, Shizuoka University
†† 静岡大学情報学部
Faculty of Infomatics, Shizuoka University
‡ 株式会社 NTT ドコモ ネットワークマネジメント開発部
Network Management Development Department，NTT DoCoMo，Inc.

スでしか利用できないといった問題がある．

本稿では，キーボード，スピーカーといった主要な周辺機器全てに対応しつつある USB に注目し，携帯端末が PAN 内に存在する USB 機器（リモート USB デバイス）を透過的に利用可能にする iUSB（intelligent USB）を提案する（**図**1）．iUSB はカーネルモードドライバを導入することで，任意のアプリケーションでリモート USB デバイスの利用を可能とする．さらに，USB 機器のネットワーク越しのプラグ&プレイ（リモートプラグ&プレイ）も実現する．

以下，本稿の構成について述べる．2 節ではリモートデバイス利用技術を紹介する．3 節では iUSB を提案し，課題を述べる．4 節では iUSB のアーキテクチャを設計する．5 節では iUSB におけるリモートプラグ&プレイの実現方法を述べる．6 節ではリモートプラグ&プレイの評価実験を行い，7 節でまとめとする．

2. 関 連 技 術

2.1 iSCSI

iSCSI は Internet Engineering Task Force（IETF）の IP Storage Working Group で標準化されている．ストレージの世界で標準となっている SCSI コマンドやデータを TCP/IP パケットの伝送フレーム中に包み込み，SCSI コマンド体系を外から見えなくすることで，ストレージ製品の IP ネットワークへの直接接続を可能にする．現在すでに実用化されており，ネットワーク・ストレージで利用されている．

iSCSI の利点として，OS や計算機のアーキテクチャに依存しないため，異種 OS 間での接続が可能であることが挙げられる．しかし，SCSI コマンドの特性上ストレージデバイスしか扱うことができず，iSCSI では様々な種類のデバイスをリモートデバイスとして利用することが困難である．

2.2 USB/IP

USB/IP[2] は，Linux のカーネル内でネットワーク上の他の計算機に接続された USB デバイスを認識し，利用することを実現している．USB 機器を利用する側のクライアントには仮想ホストコントローラ（VHCI) を用いて，USB デバイスの検出，トランザクション管理を行う．利用される側のサーバには，カーネルスレッドとしてスタブドライバ[6] を導入し，USB リクエストの送受信，接続されている USB 機器へのアクセスを実現している．USB/IP はクライアント側の VHCI とサーバ側のスタブドライバが 1 組となり，その間で USB リクエストをやり取りすることでリモートの USB 機器を動作させている．

この手法の特徴は，クライアントが USB リクエストをそのまま IP パケットにカプセル化し，サーバ側に送信していることである．そのため，ほぼ全ての USB 機器をネットワークを介して直接利用可能となる．しかし現状の課題点として，Linux 端末同士でしか利用することが難しい挙げられる．さらに，スタブドライバを利用し，独自のデバイスドライバ呼び出し関数を設定している．そのため，デバイスドライバの各関数は今後変更される可能性が高く，呼び出しを転送する関数として不適切である．

2.3 AnywhereUSB

AnywhereUSB[8] は Inside Out 社が開発したイーサネット経由で接続できる USB ハブである．USB over IP 技術を採用し，有線と無線のいずれのネットワークを利用した場合も USB ポートの配置を可能とする．AnywhereUSB ではプリンタや USB カメラを含む USB デバイスをネットワーク上のどこにでも配置することができる．

しかし，AnywhereUSB ではオーディオデバイスのようなアイソクロナス転送を要求する USB デバイスを扱うことができない．また AnywhereUSB の USB ハブを用意する必要があり，一般の計算機に備わっている USB ポートをそのまま利用することができずコストがかかる．

3. iUSB の提案

3.1 iUSB

本稿で提案する iUSB は，実行環境として Windows を想定し，リモート USB デバイスの透過的な利用を実現する．また，ネットワーク越しのリモートプラグ&プレイを可能とする．ここでリモート USB デバイスを利用し制御するユーザ端末をコントロールポイントと定義する．また，利用される USB 機器が接続されている端末をサーバと定義する．iUSB ではコントロールポイントである携帯端末が中心となり，周辺に存在するサーバ端末に接続されたリモート USB デバイスを制御する集中処理システムの形態をとる．

本手法では，USB 通信で用いられる USB リクエストを IP パケットでカプセル化し，サーバに送信することでリモート USB デバイスを利用する．USB リクエストに処理を加えることがないため，ほぼ全ての USB 機器が利用可能となる．また，通常の USB 機器は 1.5Mbps 及び 12Mbps の転送モードで制御される．iUSB では mPAN 内の USB 機器操作を想定しているため，ネットワークを介しても通信帯域の点では問題がなく，ネットワーク遅延もそれほど大きくないと考える．

iUSB はカーネルドライバを導入し，リモート USB デバイスを利用可能とする．つまり，アプリケーションレベルではなく，カーネルレベルでリモート USB デバイスが操作可能となる．ユーザはデバイスマネージャ越しに USB 機器の接続を確認でき，USB 機器がリモートにあると意識することなく通常のローカルに接続された USB 機器と同様に操作できる．

3.2 課　　題

以上のような iUSB の処理を実現するためには，以下の 3 つの課題が挙げられる．

(1)　全 USB リクエストの処理
(2)　USB 機器のプラグ&プレイ
(3)　デバイスドライバの取得

1 つ目の課題は様々な種類の USB 機器に対応するために，ほぼ全ての USB リクエストを処理する必要がある．一般に USB は 11 個の標準リクエストとベンダ，クラス特有のリクエストが用意されている．そのため，USB リクエストを一つ一つ個別に処理することは困難である．そこで USB ドライバ間通信で用いられる URB（USB Request Block）に注目し，URB を IP パケットにカプセル化しコントロールポイントとサーバ間でやり取りする．URB をそのまま処理することで，全 USB リクエストの処理が可能であると考える．

2 つ目の課題はコントロールポイントとサーバ間の USB 機器のプラグ&プレイを実現することである．通常のプラグ&プレイは，USB 機器が計算機に接続されると自動的に行われる．そのため，iUSB ではネットワーク越しのプラグ&プレイ（リモートプラグ&プレイ）の方法を考える必要がある．

3 つ目の課題は携帯端末のような小型な端末からでも，USB 機器を操作可能にすることである．通常 USB 機器を利用する場合，利用する端末はほぼ全ての USB デバイスドライバを持っている．しかし，携帯端末のような小型な端末に全ての USB ドライバを標準に持たせておくことは難しい．そこで，必要なデバイスドライバをサーバより受け取ることで，リモート USB デバイスを扱えるようにする．

以下に，これら 3 つの課題を解決した iUSB アーキテクチャを記す．

4. iUSB の設計

4.1 USB ドライバスタック

Windows OS において，USB の実装は大きく分けて 3 層の構成になっている．最上位層にアプリケーションがあり，USB ファンクションドライバ，USB バスドライバの順に構成されている．この階層を USB リクエストが昇降することで，USB 機器との通信を行う．

USB ファンクションドライバは，ハードウェアを制御する低レベルのバスドライバと通信ができる．ファンクションドライバはデバイスドライバとも呼ばれ，ハードウェアに対するユーザインタフェースを定義する．これによりアプリケーションは API 関数を使って USB デバイスと通信が可能となる．

USB バスドライバは，USB ハブドライバ，USB バスクラスドライバ，ホストコントローラドライバの 3 つのドライバで構成されている．USB ハブドライバはポートの初期化を管理し，USB バスドライバは電力管理やトランザクションの管理を行う．ホストコントローラドライバは最下位に位置しており，計算機上のホストコントローラの状態を監視する．また，上位層から託された USB リクエストの USB 機器への送信をホストコントローラに対して命令することを担う．Windows は UHCI，OHCI，EHCI などのホストコントローラをサポートしており，それぞれ独自のホストコントローラドライバを用意している．そのため，上位のドライバに対してホストコントローラごとの違いを意識させないようになっている．

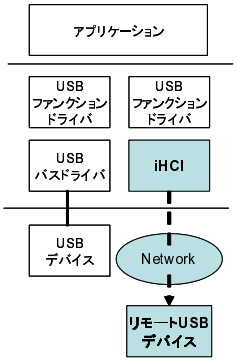

図 2　iHCI の位置づけ

4.2 仮想ホストコントローラインタフェース

iUSB では，コントロールポイントがリモート USB デバイスを制御できなければならない．リモート USB デバイスがあたかも手元の計算機に直接接続され，利用可能な状態（仮想接続）にする．しかし，従来の OS のデバイスアクセス機構はネットワーク上のデバイスについて何ら考慮されていない．そのため，USB 機器は直接計算機に接続された利用しかできないのが現状である．そこで，既存 OS の枠内でネットワーク上に存在する USB デバイスを仮想接続する機構を検討する．この機構は OS デバイスアクセス部分以外への変更を最小限にすることが求められる．既存 OS の枠内でリモート USB デバイスを仮想接続する機構により，従来のファイルアクセス，デバイスアクセスが適用でき，既存のソフトウェア資源を最大限に利用することが可能となる．つまり，ユーザは慣れ親しんだ既存のアプリケーションを利用しながら，リモート USB デバイス使うことができる．

USB 通信ではホストとデバイスが USB リクエストをやり取りする．Windows OS において USB リクエストを取り出すことは容易である．そこで，ホストコントローラドライバの一つとして iHCI（iUSB Host Controller Interface）を作成する．**図**2 に計算機内における iHCI の位置づけを示す．iHCI は USB ファンクションドライバから USB リクエストを得て，IP パケットカプセル化し，ネットワーク越しの USB 機器に対してリクエストを送信す

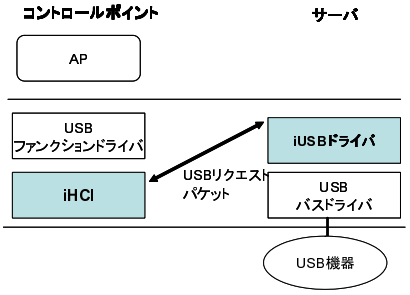

図 3 iUSB アーキテクチャ

る．リモート USB デバイスのトランザクション処理は全て iHCI が担うことになる．一方，各デバイスごとの固有の処理は専用の USB ファンクションドライバが対応する．USB ドライバスタックの最下位に位置する iHCI は USB ファンクションドライバからの USB リクエストを得るだけのため，個々の USB ファンクションドライバは改変する必要がない．これにより既存のデバイスドライバを利用することができ，多数の USB 機器に対応することができる．

また，iHCI はソフトウェアで構成された仮想的なホストコントローラのため，利用側は USB インタフェースを持たなくてもよい．つまり，iHCI がリモート USB デバイスとの通信を行うことで，インタフェース数に制限がある携帯端末からでもリモート USB デバイスが制御可能となる．これにより，我々が想定している mPAN 内の USB 機器をコントロールポイントである携帯端末から操作できる．

4.3 iUSB アーキテクチャ

iUSB のアーキテクチャを**図**3 に示す．コントロールポイントへ iHCI を導入し，仮想的なホストコントローラとして動作させる．iHCI の上には USB ファンクションドライバがあり，これがサーバに接続されている USB 機器を制御するデバイスドライバにあたる．コントロールポイントの最上位にある AP は，サーバに接続された USB 機器の検出を行う．一方，サーバには iUSB ドライバを導入する．iUSB ドライバはファンクションドライバであり，サーバは iUSB ドライバを利用して接続されている USB 機器の制御を行う．iUSB ではコントロールポイントで発生した USB リクエストが，サーバに接続された USB 機器のリクエストにあたる．そのため，iUSB ドライバの仕事はコントロールポイントから送信された USB リクエストを受信し，接続されている USB 機器へ渡すことである．iUSB はサーバの利用される USB 機器一つ一つに iUSB ドライバを設定し，コントロールポイントの iHCI が一括で管理し各々の制御を行う．

次に，コントロールポイントの処理について説明する．コントロールポイントで行う処理は 2 つあり，1 つ目はサーバに接続された USB 機器を検出し，サーバの USB

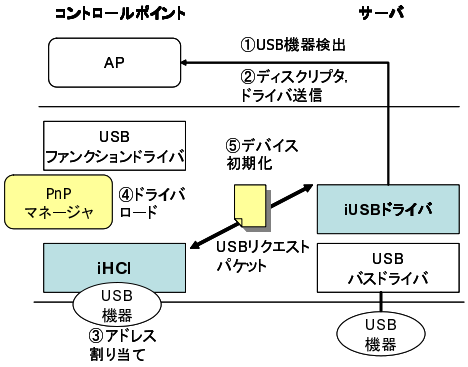

図 4 iUSB プラグ&プレイ処理フロー

機器をコントロールポイントから利用可能な状態にすることである．これらの詳細な処理に関しては次節で述べる．2 つ目は iHCI による USB リクエストの処理である．通常ホストコントローラドライバは，計算機上のホストコントローラに対して USB 機器への USB リクエスト発行を命令する．しかし，iUSB では利用する USB 機器はサーバに接続されているため，USB リクエスト発行の命令をサーバのホストコントローラに対して行う必要がある．Windows OS において，USB リクエストは USB ドライバ間通信で用いられる URB（USB Request Block）によって処理される．そこで，iHCI は上位ドライバより URB を受け取り，UDP/IP パケットでカプセル化する．UDP/IP パケットでカプセル化された URB（USB リクエストパケット）はサーバへ送信される．USB リクエストパケットはサーバで処理され，処理結果がコントロールポイントへ返される．iHCI は処理結果を受信し，上位ドライバに結果を渡す．この一連の動作により，コントロールポイントで発生した USB リクエストの処理が完了する．

最後に，サーバの処理について述べる．サーバに期待される動作は 2 つあり，1 つ目は USB 機器の接続をコントロールポイントへ通知することである．この動作の詳細は次節で説明する．2 つ目はコントロールポイントから送られてきたリクエストを元にリモートデバイスとして振る舞い，その結果を返すことである．そこで，iUSB ドライバは USB リクエストパケットを受信し，ホストコントローラドライバに URB を渡す．その後，処理結果を受け取りコントロールポイントへ返す．この動作により，サーバの USB 機器はリモートデバイスとしてコントロールポイントから制御される．

5. リモートプラグ&プレイ

5.1 iUSB のエニュメレーション

iUSB はコントロールポイントとサーバ間でプラグ&プレイ処理を行う．そのためにコントロールポイントの iHCI は，サーバに接続された USB 機器のエニュメレーションを行う．エニュメレーションとは接続された USB 機器を

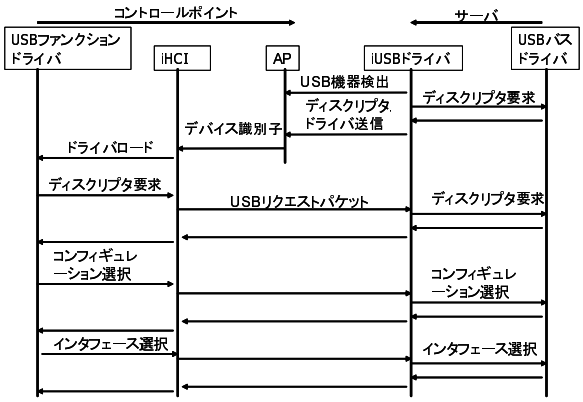

図 5　iUSB のプラグ&プレイ処理シーケンス

検出し，ホストが USB 機器と通信できるようにすることである．つまり，エニュメレーションが USB 機器のプラグ&プレイ処理に相当する．通常，一般の USB ホストコントローラのエニュメレーション処理は以下の手順で行われる．

(1)　ホストがデバイスを検出
(2)　ホストがデバイスのディスクリプタを取得
(3)　ホストがデバイスにユニークなアドレスを設定
(4)　ホストがデバイスドライバをロード
(5)　ホストがデバイスを初期化

図4 と**図**5 に iUSB におけるプラグ&プレイ処理フローとシーケンスを示す．iHCI は通常の USB ホストコントローラと同様にエニュメレーションをし，リモートプラグ&プレイを実現する．以下，それぞれの処理について説明する．

5.1.1　デバイス検出

サーバは USB 機器が接続されると，コントロールポイントに対してデバイス検出メッセージを送信する．このデバイス検出メッセージは，USB 機器のファンクションドライバに当たる iUSB ドライバが生成する．コントロールポイントの AP はデバイス検出メッセージを受け取ると，サーバに USB 機器が接続されたと認識する．

5.1.2　ディスクリプタ，ドライバ送信

コントロールポイントは接続された USB 機器の情報を知る必要ある．デバイスの特性や属性などは USB 機器が持つディスクリプタに定義されている．そのため，サーバはディスクリプタを取得しコントロールポイントに送信する必要がある．そこで，iUSB ドライバは接続された USB 機器よりディスクリプタを取得し，コントロールポイントの AP に送信する．コントロールポイントは受信したディスクリプタを見ることにより，どのような種類のデバイスが接続されたかを確認する．

また，iUSB では携帯端末をコントロールポイントとして想定しているため，接続した USB 機器用のドライバも送信する．これにより携帯端末は USB 機器のドライバを持つ必要がない．

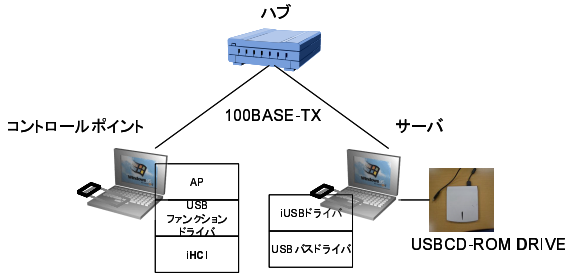

図 6　実験環境

5.1.3　アドレス割り当て

iHCI は検出した USB 機器に対して一つ一つオブジェクトを作成し管理する．iHCI がオブジェクトを作成する際には，シリアル番号を一つオブジェクトに対して割り振る．これにより，サーバに接続された USB 機器をユニークに管理することができる．

5.1.4　ドライバロード

iHCI は与えられたディスクリプタを基に，検出した USB 機器に対して適切なデバイスドライバをロードする．Windows OS においてドライバロードをする場合，ディスクリプタに記述されている USB デバイスのベンダーコード，製品コード，リビジョンコードよりデバイス識別子のリストを作成する．作成したリストを PnP マネージャに渡すことで，作成したデバイス識別子と一致するデバイスドライバが自動的にロードされる．

5.1.5　USB 機器の初期化

コントロールポイントは USB 機器のドライバをロードすると，サーバの USB 機器の初期化をし，利用可能な状態にしなければならない．そのためにコントロールポイントとサーバ間で，USB 機器の初期化に必要な USB リクエスト処理を行う．デバイス初期に必要な USB リクエストはディスクリプタ要求，コンフィギュレーション選択，インタフェース選択の 3 つのリクエストである．よって，iHCI はこれらのリクエストを受け取ると，USB リクエストパケットとしてサーバに送信する．サーバの iUSB ドライバは USB リクエストパケットを受信し，USB リクエストを取り出し下位ドライバに渡すことで USB 機器の初期化を行う．3 つのリクエストの処理し USB 機器の初期化が完了すると，コントロールポイントは正常にサーバに接続された USB 機器を認識する．以上により，iUSB におけるプラグ&プレイ処理が完了する．

6.　実　　験

6.1　実 験 環 境

本稿で提案した iUSB の環境を構築するために，iHCI と iUSB ドライバを実装しコントロールポイントとサーバに導入した．本実験の環境は**図**6 に示す．干渉のない

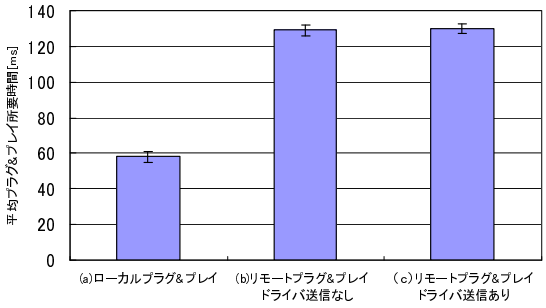

図 7　プラグ&プレイ所要時間

理想的な mPAN 環境で iUSB のリモートプラグ&プレイの基本処理時間を測定するために，コントロールポイントとサーバをハブを介して繋ぎ，コントロールポイントとサーバがダイレクトに通信する環境を構築した．この実験環境において USB 機器として USBCD-ROM DRIVE を利用し，リモートプラグ&プレイ処理時間を測定した．また，ローカルのプラグ&プレイ処理時間も測定し比較した．

6.2　プラグ&プレイ処理時間の測定

図6 の環境において，次のような 3 種類の場合のプラグ&プレイ処理時間の測定を行った．**図**7(a) はサーバに USB 機器を接続し，サーバがプラグ&プレイ処理に要する時間を測定した．つまり，通常通り USB 機器がローカルの計算機に直接接続された場合，プラグ&プレイ処理に要する時間である．**図**7(b) は，コントロールポイントが USB 機器のドライバを持つ場合に要するリモートプラグ&プレイ処理時間を測定した．**図**7(c) は，コントロールポイントが USB 機器のドライバを持たず，サーバから USB ドライバを送信した場合に要するリモートプラグ&プレイ処理時間を測定した．USB ドライバは一般に数 10kbyte から数 100kbyte の大きさであり，本実験では USB ストレージ系デバイスで利用される USBSTOR.sys ドライバ（26kbyte）を送信した．以上の 3 種類の場合においてそれぞれ 10 回プラグ&プレイを試行し，平均プラグ&プレイ所要時間を計測した．

図7 に結果を示す．(b) と (c) は，(a) に比べ平均プラグ&プレイ所要時間が 2 倍である．これはコントロールポイントとサーバ間でリモートプラグ&プレイを行うために，USB リクエストパケットを送受信しているからである．しかし，平均プラグ&プレイ所要時間は 130ms 前後であり，ごく僅かな時間で USB 機器のリモートプラグ&プレイ処理が完了している．この程度の処理時間であれば，ユーザはローカルに USB 機器を接続した場合との違いを感じることなく，USB 機器を利用できると考えられる．また，(b) と (c) は，平均プラグ&プレイ所要時間がほぼ同じである．つまり一般的な USB ドライバであれば，コントロールポイントは USB ドライバを持つ必要がなく，携帯端末のように USB ドライバを持たない端末でもサーバから USB ドライバを受信することで，サーバの USB 機器を利用することが可能であると言える．

USB 通信において，ホストとデバイス間のデータロスや遅延はほとんどない．そのため，コントロールポイントとサーバ間の通信接続性が不安定な環境においては，USB リクエストのパケットロスが頻発してしまいプラグ&プレイ処理やデバイス制御が難しいと考えられる．iUSB は，十分な帯域と低遅延である mPAN 環境ならば有効であると考える．

7. おわりに

本稿では，リモート USB デバイスを透過的に利用する方式 iUSB を提案し，設計を行った．iUSB は iHCI を用いることで USB 機器のリモートプラグ&プレイを実現した．また実験より，リモートプラグ&プレイ処理は USB 機器を接続すると直ちに完了することを確認し，リモートプラグ&プレイの有効性を示した．

今後，iUSB で 4 種類の USB 転送方式を扱えるようにし，様々な USB 機器を利用できるようにする．実装後，コントロールポイントとリモート USB デバイス間のスループットを計測し，USB のデータ転送速度と比較することで評価を行う．また，USB リクエストのパケットロスやデバイス制御データの QoS の問題[7] についても検討していく必要がある．

参　考　文　献

1) 田中希世子他,“ モバイルパーソナルエリアネットワークの提案，”情報学ワークショップ，pp. 241-245, 2004.
2) Takahiro Hirofuchi，Eiji Kawai，Kazutoshi Fujikawa, and Hideki Sunahara,“ USB/IP - a Peripheral Bus Extension for Device Sharing over IP Network，” USENIX，2005.
3) K. Koyama, Y. Ito, S. Ishihara and H. Mineno: Performance evaluation of TCP on Mobile IP SHAKE, Proc. of 1st International conference on Mobile Computing and Ubiquitous Networking（ICMU2004), pp.8-13 (2004).
4) UPnP．http://www.upnp.org.
5) Julian Satran，etc．Internet Small Computer Systems Interface（iSCSI）．RFC3720，Apr 2004.
6) 佐藤友隆他,“ カーネルレベルで実装したネットワーク透過な周辺機器制御の枠組み，”システムソフトウェアと OS，No.92-16，2003.
7) Chin-Yuan Huang，etc.“ A Cyclic-Executive-Based QoS Guarantee over USB，”RTAS，2003
8) Inside Out Networks．AnywhereUSB．http://www.ionetworks.com/.