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

2007－MBL－40 (12)
2007－UBI－13 (12)
2007／2／22

iUSB：カーネル空間からのネットワーク透過なUSBデバイス制御機構

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

近年，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．本稿では，携帯端末がネットワーク透過に遠隔USB機器を利用する方法iUSB (intelligent USB) を提案する．iUSBでは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）も実現している．本稿ではiUSBバルク転送の性能評価実験によって，iUSBの入出力実行速度を評価した．また，リモートプラグ&プレイの評価実験から，リモートUSBデバイスが即座に利用可能になることを確認した．

iUSB：Network-Transparent USB Device Control in Kernel Space

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

Advances in high performance mobile terminals and short distance wireless communication technology have been achieved. There has been renewed of interest in controlling peripheral devices with mobile terminals. However, in general peripheral devices are used in the form directly connected to a computer. We propose an intelligent USB (iUSB) where mobile terminals can remotely use USB devices for network-transparency. We achieved Plug and Play on the network (remote Plug and Play) with a USB devices. In this paper, we have experiment performance evaluation of iUSB bulk transfer. Our experiments show iUSB has sufficient I/O performance. We also experimet remote Plug and Play in many USB devices. We confirmed that remote USB devices use immediately.

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開発されてきた．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いる．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，携帯端末が周囲に存在する周辺機器とPANを形成し，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を形成し，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．

我々はUPnPやiSCSI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[3)4)9)]．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，特定のデバイスの機能，対応デバイスでしか利用できないといった制約がある．そこで，最近ではキーボード，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機器を透過的に利用可能にするintelligent USB（iUSB）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している．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）を実現し，遠隔自動設定も可能である．本稿ではiUSBのI/Oパフォーマンスを評価するために，バルク転送の性能とリード/ライト性能の評価結果をまとめる．また，様々なUSB機器で実施したリモートプラグ&プレイの評価結

† 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 サービス&ソリューション開発部
Service & Solution Development Department, NTT DoCoMo, Inc.

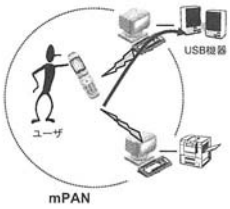

図 1　mPAN 内における iUSB 利用イメージ

果もまとめる．以下，本稿の構成を説明する．第 2 章では現在の遠隔デバイスアクセスの関連研究について説明する．第 3 章では，iUSB のアーキテクチャとリモートプラグ&プレイ機能について説明する．第 4 章では iUSB の I/O パフォーマンスの評価結果とリモートプラグ&プレイの評価結果を示す．第 5 章では，本稿を総括し，今後の課題を示す．

2. 関連技術

本章では，ネットワーク上のデバイスを遠隔制御する関連技術として，iSCSI，USB/IP，AnywhereUSB について説明する．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．

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．クライアントに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．サーバにはカーネルスレッドとしてスタブドライバ[5] を導入し，USB リクエストの送受信を行い，USB デバイスへのアクセスを実現している．この手法の利点は，ほぼ全ての USB デバイスをネットワークを介して直接利用可能なことである．しかし現状では，Linux 端末同士でしか利用することが難しいことが挙げられる．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，今後各関数が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．

AnywhereUSB 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というような柔軟な使い方ができない．

3. iUSB：intelligent USB

3.1 iUSB 概要

我々は，ネットワーク上の USB 機器をリモート制御する方式として iUSB を提案している．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 USB 機器を制御する集中処理形態をとる．

iUSB の 3 つの特徴を以下にまとめる．

- USB デバイスの全機能を利用可能
  旧来の遠隔デバイスアクセス技術は専用のアプリケーションを必要とし，アプリケーションの機能に依存する利用しかできないという制約があった．この制約を解決するために，iUSB はカーネル空間からデバイス制御を行う手法をとる．iUSB は既存の USB ドライバスタックを拡張することで，従来のファイルアクセス機構を用いて USB 機器を制御することを可能にしている．つまり，ユーザはネットワーク越しの USB 機器をデバイスレベルで認識し，USB の全機能を利用可能となる．
- USB デバイスのネットワーク透過な利用
  iUSB はネットワーク越しに存在する USB 機器を手元の端末に直接接続しているかのように制御可能とし，デバイスのネットワーク透過性を実現している．ソフトフェアで構成された仮想的な USB ホストコントローラを用いて，ネットワーク越しの USB 機器と仮想的に接続する．そのため，ユーザはローカルに USB 機器を接続した場合と全く同じようにリモート USB デバイスを制御できるようになる．
- USB デバイスの遠隔自動接続
  iUSB は USB のプラグ&プレイ機能をネットワーク越しに拡張したリモートプラグ&プレイを実現している．従来のプラグ&プレイと同様にネットワーク

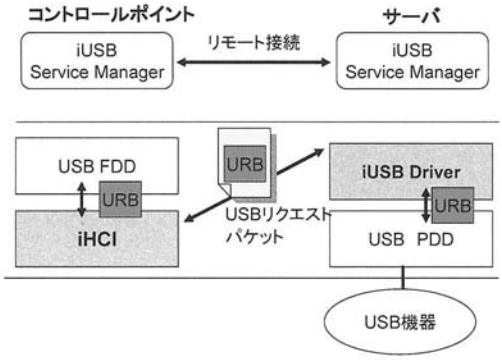

図 2　iUSB アーキテクチャ

越しにデバイスが接続されても自動的に認識し，利用可能な状態に設定される．

3.2　iUSB アーキテクチャ

iUSB のアーキテクチャを図2に示す．我々は既存の USB ドライバスタックにリモート USB デバイスを制御するドライバを組み込むことで iUSB を実現している．コントロールポイントは，ソフトウェアで構成した仮想的なホストコントローラである iUSB Host Controller Interface（iHCI）を導入する．iHCI はネットワーク越しの USB 機器の接続を制御するドライバであり，既存の USB ホストコントローラドライバと同じように USB 機器へのリクエストを処理する．サーバには iUSB Driver を導入する．iUSB Driver はファンクションドライバであり，サーバは iUSB Driver を利用して，接続されている USB 機器の制御を行う．コントロールポイントとサーバにある iUSB Service Manager はコントロールポイントとサーバの接続を管理し，TCP/IP 通信を制御している．iUSB ではアプリケーションである iUSB Service Manager がコントロールポイントとサーバのネットワーク間の通信を保障し，ドライバである iHCI と iUSB Driver がコントロールポイントと USB 機器のデバイス間通信を保障している．

iUSB はリモート USB デバイスを利用するために，ドライバ間の USB リクエストパケットの制御を行う．iHCI と iUSB Driver が対となり，USB リクエストパケットを相互で送受信することでリモート USB デバイスの制御を可能としている．Windows OS において，USB 機器へのリクエスト信号は USB ドライバ間通信で用いられる URB（USB Request Block）から作成される．USB 機器が端末に直接接続されている場合は，USB ホストコントローラは上位ドライバの USB FDD（USB Functional Device Driver）から渡される URB を処理し，端末に接続されている USB 機器へリクエスト信号を送信する．しかし，iUSB ではコントロールポイントで発生した URB を処理し，別の端末であるサーバに接続されている USB 機器へリクエスト信号を送信しなければならない．この問題を解決するために，TCP/IP パケットで URB をカプセル化し，コントロールポイントとサーバ間で USB リクエストパケットの送受信を行っている．まず，iHCI はコントロールポイントで発生した URB をインターセプトし，TCP/IP パケットでカプセル化し，USB リクエストパケットをサーバに送信する．サーバは USB リクエストパケットを受信すると，URB を取り出し iUSB Driver へ渡す．iUSB Driver は URB を下位の USB PDD（USB Physical Device Driver）へ渡し，USB PDD は接続されている USB 機器に USB リクエストを発行する．USB 機器が処理を終えると，iUSB Driver へ結果の URB が返される．サーバは URB を TCP/IP パケットでカプセル化しコントロールポイントへ USB リクエストパケットを送信する．コントロールポイントは USB リクエストパケットを受信し，URB を取り出し iHCI へ渡す．iHCI は URB の完了ルーチンを行い，処理を完了する．

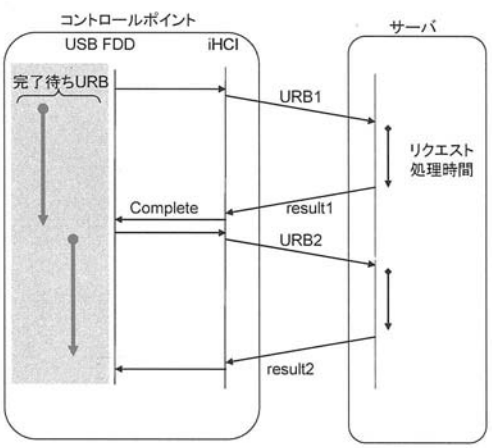

図 3　従来のままの URB 処理フロー

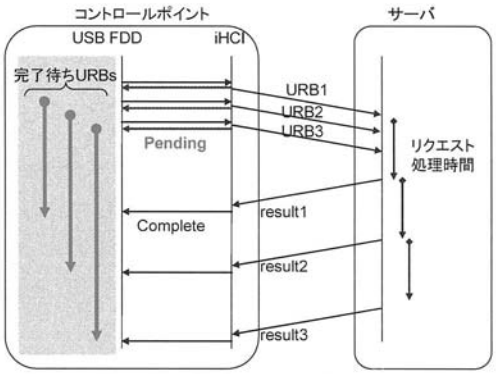

図 4　保留状態を利用した URB 処理フロー

3.3　URB の保留

従来の USB ドライバは，USB FDD が URB を下位ドラ

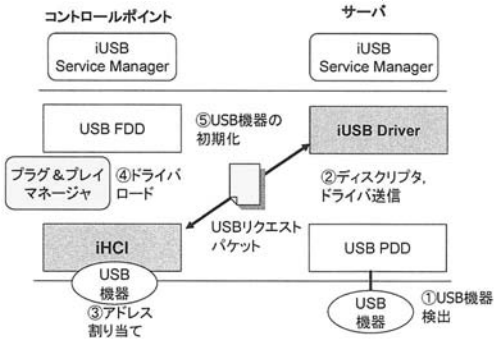

図 5 iUSB リモートプラグ&プレイ処理フロー

イバの USB ホストコントローラドライバに渡し，USB 機器が処理完了するまで次の URB の処理を待たせておく．そのため，**図3** に示すように iUSB で従来のままの URB 処理を行うと，次の URB 処理が開始されるまで長い時間待つこととなりリモート USB デバイスの I/O パフォーマンスが低下してしまう．そのため，iUSB では URB の完了を一時的に保留させる方法をとっている．**図4** に示すように，iHCI は上位ドライバである USB FDD から URB を受け取ると，URB の保留を行い URB の処理を一時的に完了させる．これにより，iHCI は USB FDD から次の URB の処理を促すことが可能となる．iHCI はサーバから URB の処理結果を受信すると一時的に保留していた URB を完全に完了させる．以上の処理を行うことで，iUSB の I/O パフォーマンスを向上させている．

3.4 リモートプラグ&プレイ

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．**図5** に iUSB におけるプラグ&プレイ処理フローを示す．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行い，リモートプラグ&プレイを実現する．以下，順にリモートプラグ&プレイの処理について説明する．

(1) USB 機器検出
サーバに USB 機器が接続されると，USB PDD が USB 機器を検出する．USB PDD は接続された USB 機器に対してファンクションドライバとして iUSB Driver を割り当て，iUSB の機器として利用可能状態にする．

(2) ディスクリプタ，ドライバ送信
サーバは接続された USB 機器の情報を得るために USB 機器が持つディスクリプタを取得する．ディスクリプタにはデバイスの特性，クラス，属性などが記述されている．サーバはディスクリプタを取得し接続された USB 機器がどのクラスに属しているか判定し，各クラスに適した USB 機器のファンクションドライバを決定する．その後サーバはコントロールポイントに対して USB 機器の情報を与えるために，ディスクリプタとドライバを送信する．コントロールポイントはサーバから利用する USB 機器のディスクリプタとドライバを受信し，サーバにどのような種類のデバイスが接続されたかを確認する．
ここで，コントロールポイントとして想定している携帯端末は全ての USB ドライバを持っているとは限らないため，サーバは利用する USB 機器用のドライバを送信する．しかし，コントロールポイントが利用する USB 機器のドライバを持つ端末であればドライバを送信する必要はない．

(3) アドレス割り当て
コントロールポイントは検出した USB 機器に対して固有のオブジェクトを作成し管理する．コントロールポイントは iHCI にユニークな識別子を与え，USB 機器オブジェクトを作成する．これにより iHCI はサーバに接続された USB 機器をユニークに管理する．したがって，コントロールポイントは複数のリモート USB デバイスも個別に利用することができる．

(4) ドライバロード
コントロールポイントは作成した USB 機器オブジェクトに対してサーバから受信したドライバをロードする．サーバからドライバが送信されない場合は受信したディスクリプタを参照し，検出した USB 機器に対して適切なドライバをロードする．Windows OS においてドライバを決定するためにはデバイス識別子を作成する必要がある．USB のデバイス識別子はディスクリプタに記述されている USB 機器のベンダーコード，製品コード，リビジョンコードより作成することができる．作成したデバイス識別子をプラグ&プレイ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ドライバが自動的にロードされる．

(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 機器の初期化を行うために USB 機器の初期化に必要な USB リクエスト処理を行う．デバイス初期化に必要な USB リクエストは 2 種類ある．まず，ディスクリプタ要求を行い，USB 機器の持つ様々なディスクリプタを取得する必要がある．取得したディスクリプタに

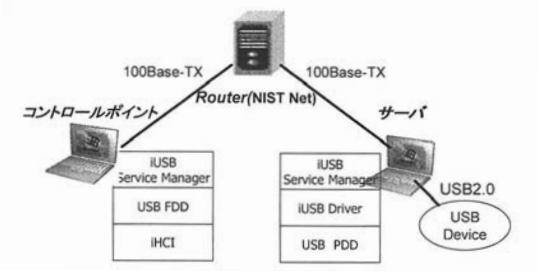

図 6　実験環境

表 1　実験端末のスペック

CPU	Interl Pentium M
Memory	512 MB
OS	Windows XP Professional
USB 2.0 Interface	Intel 82801DB/DBM USB2.0 EHCI

表 2　実験で利用した USB 機器の仕様

製品名	Seagate BARRACUDA ATA
Interface	USB 2.0
容量	80GB
キャッシュ	2MB
スピンドル速度	7200rpm

はUSB機器の持つインタフェースの種類が記述されている．次に，コンフィギュレーション選択を行う．コンフィギュレーション選択ではUSB機器で利用するインタフェースを決定し，初期化を行う．これらのリクエストを行うことでUSB機器の初期化が完了し，コントロールポイントはサーバのUSB機器が利用可能となる．以上により，iUSBにおけるリモートプラグ&プレイ処理が完了する．

4. 実　　験

4.1 実験環境

我々はiUSBの性能評価をするために実験環境を構築した．**表1**に示すように，同じ性能を持つWindows OSの端末を2台用意した．**図6**に示すように2台の端末にコントロールポイントとサーバを構築した．コントロールポイントにはiHCIを実装し，サーバにはiUSB Driverを実装し，iUSB Service Managerを双方の端末に設置することでiUSBの環境を構築した．各端末とRouterは100BASE-TXにより接続した．Router上でNIST Netを動作させ各経路に0 msから5 msの片道遅延を発生させ，干渉の少ない理想的なmPAN環境を実現した．実験では**表2**に示すUSB機器をサーバに接続し，コントロールポイントから制御することでiUSBの性能評価を行った．

4.2 バルク転送の性能評価

図6の実験環境において，コントロールポイントがiUSBバルク転送で1つのURBを処理完了するまでの実行時間を測定し，実行時間の結果よりiUSBバルク転送のスループットを算出した．バルク転送の実行時間の測定実験では，コントロールポイントから0から100 KBまでURBを変化させて送信し，コントロールポイントがサーバから処理結果のURBを受信し完了するまでの応答時間を測定した．NIST Net上で片道遅延を発生させ，往復遅延時間0 ms，1 ms，2 ms，4 msにおけるiUSBバルク転送の実行時間を測定した．実験におけるコントロールポイントとサーバ間の通信は全てTCP/IPを用いている．サイズが小さいデータを即座に送出するために，TCPのNagleアルゴリズムは無効にし，TCP/IPのバッファリングによる実験結果への影響をなくした．また，我々はiUSBにおけるバルク転送と通常のUSBのバルク転送の実行時間の違いを比較するために，端末にUSB機器を直接接続した場合のバルク転送の実行時間も測定した．

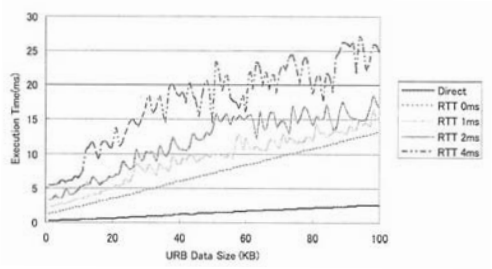

図 7　iUSB バルク転送の実行時間

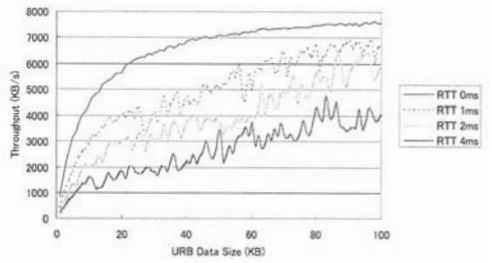

図 8　iUSB バルク転送のスループット

図7，**図8**にiUSBバルク転送の実行時間とスループットの測定結果を示す．**図7**より，往復遅延時間が0 msの場合URBデータサイズと実行時間は比例関係にあることが読み取れる．したがって，実行時間 T_{iusb}[ms]は，URBデータサイズを S[KB] とすると式1で表すことができる．また，バルク転送のスループット $Thpt$[KB/s] は式2で算出することができる．

$$T_{iusb} = A_{iusb} \times S + (RTT + D_{direct}) \quad (1)$$

$$Thpt = \frac{S}{T_{iusb}} = \frac{S}{A_{iusb} \times S + (RTT + D_{direct})} \quad (2)$$

式1において，A_{iusb} は**図7**から算出される傾きである．この値は各往復遅延時間ごとに異なる値を持つ．RTT は

往復遅延時間の実時間である．D_{direct} は直接 USB 機器を接続した場合におけるバルク転送の転送遅延である．バルク転送は他の形態の転送よりタイムシェアリングの優先度が低いため僅かな遅延が発生する．

図7 の往復遅延時間 0 ms の場合と直接接続の場合において回帰分析を行うと，式 3 と式 4 を算出することができる．

$$T_{iusb} = 0.12 \times S + 1.03 \quad (3)$$

$$T_{direct} = 0.02 \times S + 0.17 \quad (4)$$

式 4 より D_{direct} は 0.17 ms と求められる．したがって，式 1 と式 3 より往復遅延時間 0 ms における RTT は 0.86 ms となる．この値は図6 の実験環境において，NIST Net 上で往復遅延時間 0 ms と設定した場合のコントロールポイントとサーバ間の往復遅延時間の実時間に近い値であり，式 1 の妥当性を証明している．

しかし，往復遅延時間が大きくなるにしたがって，iUSB バルク転送の実行時間の揺らぎが増している．そのため，往復遅延時間が大きくなればなるほど式 1 から外れた実行時間となってしまう．この原因は USB のトランザクションが 1 ms 毎に行われることに起因すると考えられる．USB の転送において，1 トランザクションは 1 ms のフレームに納めるように計画をする．そのため，iUSB において URB の到着時間間隔が大きくなると 1 トランザクションの実行される間隔の変動が大きくなり，URB の処理時間がばらつくことに繋がる．したがって往復遅延時間が大きくなるにつれて実行時間の揺らぎが大きくなり，URB データサイズと実行時間が比例関係から外れていくことになると考えられる．

図8 より，iUSB バルク転送のスループットは往復遅延時間 0 ms の時に最大であり，約 7.5 MB/s である．直接 USB 機器を接続した場合のバルク転送の最大スループットは，図7 と式 2 より約 38 MB/s と求められる．したがって，最大で iUSB バルク転送の転送速度は通常の USB バルク転送の約 20%の性能を得ていることになる．また，往復遅延時間 0 ms の時コントロールポイントとサーバ間の TCP/IP のスループットは約 11.5 MB/s であった．よって iUSB バルク転送の転送速度は TCP/IP の 65%の性能に達していることになる．TCP/IP との性能差の原因の一つには，USB 機器でのリクエスト処理時間が影響している考えられる．また，別の要因としてはカーネル空間とアプリケーション空間でのデータのやり取りの時間が影響していると考えられる．iUSB ではドライバである iHCI と iUSB Driver は URB を送受信するために，アプリケーションである iUSB Service Manager とデータ交換を頻繁に行う．Windows OS において相互のデータ交換の待ち時間は 1 ms よりかなり短い．しかし，場合によってはマルチタスクの影響受け 100 ms 以上待たせる場合もある．したがって，これらの影響により TCP/IP より性能が落ちていると考えられる．しかしながら，iUSB バルク転送はマウスやキーボードといった少量のデータを扱う HID デバイスやストレージデバイスの制御としては十分な転送速度であると言える．

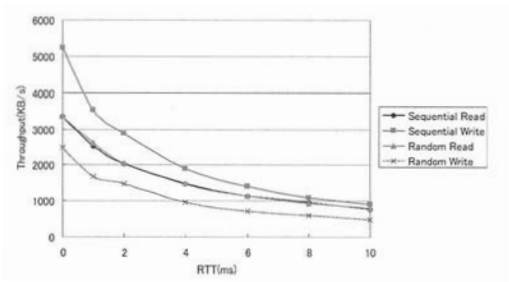

図 9 iUSB のシーケンシャル，ランダムリード/ライトのスループット

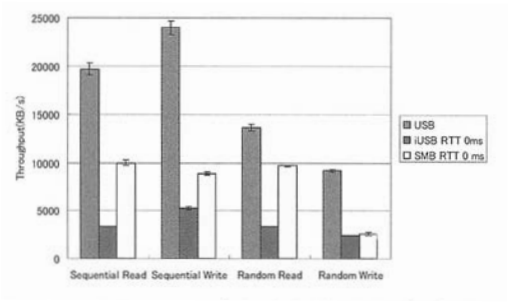

図 10 シーケンシャル，ランダムリード/ライトのスループット (USB, iUSB，SMB)

4.3 リード/ライト性能の評価

同様の実験環境 (図6) において，表2 の USB 機器を用いて iUSB によるデータ読み書きのスループットを計測した．実験に用いた USB 機器はマスストレージクラスのデバイスであり，USB マスストレージクラスデバイスは全てのデータ転送がバルク転送で行われる．スループットの計測には HDBENCH[10] を用い，シーケンシャルリード/ライト，ランダムリード/ライトの 4 種類のリード/ライトのスループットを計測した．各処理において 10 MB のデータ読み書きを行った．NIST Net 上では往復遅延時間を 0 ms から 10 ms まで変化させ，それぞれの場合のスループットについても計測を行った．

また比較として，端末に直接 USB 機器を接続した場合と SMB（Server Message Block）を利用した場合におけるデータ読み書きのスループットを計測した．SMB とは Windows OS 上でネットワークを通じてファイル共有を実現するプロトコルである．

図9 に iUSB のシーケンシャルリード/ライト，ランダムリード/ライトのスループットの結果を示す．図9 において，往復遅延時間 0 ms から 1 ms に変化した時における各リード/ライト処理の性能低下が最も大きくなってい

る．これは往復遅延時間が 1 ms 以上になると実行時間の大きな揺らぎが発生するためだと考えられる．前節の図7 の結果より，往復遅延時間が 0 ms の場合は実行時間の大きな揺らぎは発生していないが，往復遅延時間が 1 ms 以上になると実行時間の大きな揺らぎが発生している．この実行時間の揺らぎの影響を受け，著しく各リード/ライト処理の性能が低下したと考えられる．2 ms 以降の各リード/ライト処理では，スループットの低下は見られるが緩やかな低下になっている．つまり，iUSB においてリード/ライトの性能は実行時間の長さよりも実行時間の揺らぎ発生に大きな影響を受けると考えられる．

図10 に USB，iUSB，SMB におけるシーケンシャルリード/ライト，ランダムリード/ライトのスループットの結果を示す．図10 より iUSB のスループットは USB のスループットと比べ，シーケンシャルリードでは 17%，シーケンシャルライトでは 22%，ランダムリードでは 24%，ランダムライトでは 27%の性能を得ている．これは前節において，図7，図8 の実行時間とスループットから算出した USB バルク転送に対する iUSB バルク転送スループット性能とほぼ同等である．また，iUSB は SMB に対してシーケンシャルリードでは 33%，シーケンシャルライトでは 60%，ランダムリードでは 35%，ランダムライトでは 94%の性能を得ている．本実験において，iUSB と SMB はどちらも同じリモートのリード/ライトにもかかわらず iUSB の性能が低い理由には，USB プロトコルと SMB プロトコルの違いによる影響が大きいと考えられる．第一の原因は，データアクセスに対するリクエスト回数の違いである．USB は 1 回のデータアクセスをするために 3 回のリクエストを送信する．一方，SMB は 1 回のデータアクセスは 1 回のリクエストの送信で完了する．そのため，USB プロトコルは SMB プロトコルよりもリクエスト回数のオーバーヘッドが大きい．第二の原因は，USB プロトコルが定期的に送信する制御パケットである．USB は絶えず正しくデータが送られているか制御パケットを送信して確認する．SMB プロトコルは定期的に送信される制御パケットはないため，USB プロトコルの方が多くのデータ交換をしていることになる．これら二つの影響を受け,iUSB は SMB より低い性能になっていると考えられる．前節の図8 の結果より，iUSB の 1 回のリクエストに対するスループットは約 7.5MB/s であるため，iUSB の実性能としては SMB のリード/ライトのスループットに近い性能を得ることが可能である．また，SMB はストレージデバイスのリード/ライトのみしか行えないが，iUSB はストレージデバイス以外にも HID デバイスやオーディオデバイスも利用できる点で SMB より優位性があると考える．

4.4 プラグ&プレイ処理時間の測定

図6 の実験環境において，ローカルプラグ&プレイ，リモートプラグ&プレイ（ドライバ送信なし），リモートプラグ&プレイ（ドライバ送信あり）の 3 種類のプラグ&プレイ処理時間測定を行った．ローカルプラグ&プレイはサーバに 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 機器がローカル端末に直接接続された場合のプラグ&プレイ処理に要する時間である．リモートプラグ&プレイ（ドライバ送信なし）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．リモートプラグ&プレイ（ドライバ送信あり）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 10 KB から数 100 KB の大きさであり，本実験では USB ストレージデバイスで利用される USBSTOR.sys ドライバ（26 KB）を送信した．2 種類のリモートプラグ&プレイでは，NIST Net の往復遅延時間を 0 ms に設定し測定した．測定には USB CD-ROM ドライブ，USB フラッシュメモリ，USB フロッピーディスクドライブの 3 種類の USB 機器を利用した．

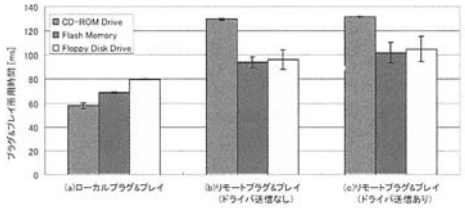

図 11 プラグ&プレイ所要時間

図11 に結果を示す．図11 よりドライバ送信有無にかかわらずリモートプラグ&プレイの所用時間は 100 ms から 130 ms の間であり，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．反応時間が 100 ms 以下であることが，人間が即座に応答していると感じる一つの基準であるため[8]，ユーザは USB フラッシュメモリと USB フロッピーディスクドライブのリモートプラグ&プレイについては即座に完了したと感じることができる．USBCD-ROM ドライブについてもリモートプラグ&プレイ所用時間が約 130 ms と短いため，ほぼ同等のことが言える．また，ローカルプラグ&プレイとリモートプラグ&プレイの所用時間差は数 10 ms 程度であるため，ユーザはリモート接続した場合でも USB 機器をローカル接続した場合との違いを感じることなく，即座に USB 機器の利用できると考えられる．

リモートプラグ&プレイ（ドライバ送信なし）とリモートプラグ&プレイ（ドライバ送信あり）のプラグ&プレイ所用時間はほぼ等しくなっている．このことより，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 機器を即座に利用可能であると考えられる．

5. 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，設計，評価を行った．iUSB バルク転送の性能評価の結果より，iUSB バルク転送速度は TCP/IP の約 65 %の性能であるため，更なる転送速度の向上をする必要がある．ただし，少量のデータを扱う HID デバイスやストレージデバイスの制御としては十分な転送速度を得ていることも確認した．また，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，ユーザはローカルと同様に即座に利用できることを示した．

本提案方式では，往復遅延時間の増加に伴いスループットの著しい低下が起こっている．今後，この問題に対する改善策を検討し，更なるスループット向上を目指す．

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) UPnP. http://www.upnp.org.
4) Julian Satran，etc. Internet Small Computer Systems Interface（iSCSI）. RFC3720，Apr 2004.
5) 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.
6) Chin-Yuan Huang，etc. “A Cyclic-Executive-Based QoS Guarantee over USB，”RTAS，2003
7) Inside Out Networks. AnywhereUSB. http://www.ionetworks.com/.
8) R.B. Miller，“Response time in man-computer conversational transactions，”Proc. AFIPS Fall Joint Computer Conference，vol.33，pp.267-277，1968.
9) 尾崎亮太他，“計算機やデバイス移動に対してもサービスが継続可能な遠隔デバイスアクセス機構，”電子情報通信学会論文誌 B，vol.J189-B，No.8，pp.1357-1366，2006.
10) HDBENCH. http://www.hdbench.net/