デバイス携帯利用を支援する
アクセス継続技術に関する研究

尾崎 亮太

博士（情報学）

総合研究大学院大学
複合科学研究科
情報学専攻

平成18年度
（2006）

2006年9月

本論文は総合研究大学院大学複合科学研究科情報学専攻に
博士（情報学）授与の要件として提出した博士論文である.

博士論文審査委員会

主査: 丸山 勝巳
委員: 橋爪 宏達
委員: 多田 好克
委員: 佐藤 一郎
委員: 児玉 和也

A Study on Continuous Access Technologies to Support Device Mobile Usage

Ryota Ozaki

DOCTOR OF
PHILOSOPHY

Department of Informatics,
School of Multidisciplinary Sciences,
The Graduate University for Advanced Studies (SOKENDAI)

September, 2006

A dissertation submitted to
the Department of Informatics,
School of Multidisciplinary Sciences,
The Graduate University for Advanced Studies (SOKENDAI)
in partial fulfillment of the requirements for
the degree of Doctor of Philosophy

Advisory Committee

Katsumi Maruyama (Chair)
Hiromichi Hashizume
Yoshikatsu Tada
Ichiro Satoh
Kazuya Kodama

デバイス携帯利用を支援するアクセス継続技術に関する研究

論文要旨

本論文は，計算機に接続し利用者が直接扱うデバイス（以下デバイス）の携帯利用を支援する技術およびシステムを提案するものである．

利用者が持ち歩くことが可能なノート PC や PDA などのモバイル計算機が普及し，利用範囲が広がっている．モバイル計算機を用いて，重要な情報を扱ったり，複雑な作業に従事することが増えた．そのようなモバイル計算機の利用環境では次の要件を満たすことが望まれる．(i) 高い安全性：データを持ち歩かず，持ち歩くデバイスそのもののが故障・紛失しても損害が少ないこと．(ii) 広範囲の移動での継続利用：部屋や建物の階を跨いだ広範囲の移動を伴っても，継続利用が可能なこと．(iii) 小型軽量でシンプル：デバイスは持ち歩き易く，複雑でないこと．

本研究では，このような利用環境に適合する，新たな計算機とデバイス利用形態を提案する．提案する利用形態は，デバイスを利用者が持ち歩き，手近にある計算機を経由して，必要なデータへのアクセスや遠隔地との相手と会話する．持ち歩くデバイスは安価で，紛失や盗難にあっても被害が小さい．また本研究は，提案する利用形態で利用者が広範囲の移動をしてもデバイスの継続利用を可能とするシステムを実現する．提案するデバイス利用形態では，利用者の移動に伴いデバイスの接続先計算機が切替わる．これを本研究ではデバイス移動と呼ぶ．本システムはデバイス移動が起きても利用者がデバイス利用を継続可能とすることが目標である．

第 1 章では本研究の背景と目的，第 2 章ではデバイス携帯利用を支援する技術および関連研究について述べる．既存の可搬計算機（PDA 等）や Bluetooth と比較し，デバイス携帯利用法の必要性について述べる．さらにデバイス携帯利用を支援するシステムを実現するソフトウェア技術について分類・比較する．またそれらを踏まえ，本研究の特色について述べる．

本システムは，目的を達成するために，複数の機能を組み合わせて実現されている．それぞれの機能の設計と実現方法について 3, 4 章で述べる．

第 3 章では，デバイス移動が起きても利用者がデバイス利用を継続可能とするための機能を遠隔デバイスアクセス機構上に設計する．本機構は遠隔デバイスを

デバイスファイルとしてアプリケーションに提供する．そのため利用者は既存アプリケーションを書き換えることなく利用できる．本機構はサブネットワーク内の計算機上にサービスグループを形成し，アプリケーションはグループ内の計算機に接続されたデバイスにアクセス可能となる．また本機構はサービスグループ内で起きたデバイス移動を検出し，アプリケーションから透過的に遠隔デバイスアクセス先を切り替える．これらは自動的に処理され，利用者に手を煩わせることはない．なお開発の容易化と拡張性を高めるため，機能の大部分はユーザレベルプログラムとして実現した．

第 4 章では，前章で実現したシステムを拡張し，利用者の移動範囲を拡大する支援システムを設計する．一つはファイアウォールによって通信が制限されたサブネットワーク間で，遠隔デバイスアクセスを可能とする支援システムである．これにより利用者は遠い部屋や別階などへの広範囲の移動が可能となる．もう一方は，利用者の移動先に本機構がなく無線アクセスポイントのみ存在する環境において，利用者のデバイス利用継続のための機能である．そのような環境でのアクセス継続は，デバイスの接続したモバイル計算機を利用者が携帯することで可能となる．アプリケーションからデバイスへのアクセスは，無線アクセスポイントに接続したモバイル計算機を経由して行なわれる．本機構には，利用者の移動にともないモバイル計算機の IP アドレスが切り替わっても，遠隔デバイスアクセスを継続することができる機能を実現した．

第 5 章で，本システムの評価実験結果について述べる．デバイス特性を基にした評価実験により，本システムが遅延への要求が高いデバイスの性能要件を満たしていることを確認した．遠隔デバイスに対するデータ書込み処理時間は，データサイズが 4KB の場合，同一サブネットワーク内では約 0.8ms、サブネットワークを超えた場合（Meidator による中継が 2 回の入る）で 1.8ms であった．これは人間が直接使うデバイスとしては，十分実用的な性能である．またデバイスが移動して接続先の計算機が切り替わる際のサービス中断時間は 3~5 秒であり、実用に耐えるといえる．また提案するデバイス利用形態の応用やセキュリティなどの観点で議論する．

最後に第 6 章で本論文の成果をまとめる．

本研究の成果は，新たなデバイス利用形態を提案し，重要な情報を扱い複雑な作業に従事する環境において，利用者に計算機利用法の代替手段をもたらしたこと，支援システムを実現し評価実験によりその有効性を実証したことにある．

A Study on Continuous Access Technologies to Support Device Mobile Usage

Abstract

This thesis describes a new kind of device mobile usage in local mobility situations and its support system.

Mobile computers, such as laptop PCs and Personal Digital Assistants, are widely used. Users carry confidential information in those computers and engage complex work with those computers. In those situations, it is desired to satisfy following requirements; (i) Secure: even if the device breaks or is stolen, the user's loss is minimal. (ii) Continuous access in a wide area: even if a user who has a device walks around between rooms or buildings, he/she continues his/her work with the device. (iii) Easy to use: devices are small and light-weight.

We propose a new kind of device mobile usage to satisfy those requirements in local mobility situations. The idea is that users should be able to easily leave their desks and continue their work by using their devices and other computers in the surrounding area. Network Extended Device Management System (NextD) is a support system for the proposed usage, and it copes with device migration which is a change of the computer to which the device connects and host migration. NextD provides nearly continuous service with short unusable periods while the users are away from their desk.

In Chapter 1, we describe the background and summarize this thesis and in Chapter 2, we discuss related work.

In Chapter 3, we describe the design and implementation of NextD, based on implementation of a prototype system on the Linux OS. NextD allows applications to access remote devices in exactly the same way as they access local devices, through device files. It provides device files that serve as access points toward remote devices.

In Chapter 4, we describe about Mediator which is a support system enables NextD to communicate between subnetworks even if the networks are divided with

a firewall. We also describe about how NextD enables applications to continue accessing to remote devices when a host migration occurs.

In Chapter 5, we evaluated NextD's basic performance and NextD processing time on a device migration. For 4 Kbytes, the latency of NextD's remote device access was 0.80 ms, which is sufficiently lower than the audio requirement of 200 ms. The other experimental result shows that the device-unusable time becomes from 3 to 5 seconds when the radio ranges of the receivers on the source and destination hosts are very near each other or overlap. We also discuss about our proposed usage in terms of applicable scope, security, applications and capability for the future.

In Chapter 6, we conclude the thesis with a summary of the significant findings.

目 次

図 目 次

表 目 次

第1章

序論

1.1 はじめに

計算機を構成するハードウェアは大きく分けて2つに分類される．人間が直接扱うデバイスとそうでないものである．前者は，HID（Human Interface Device）であるマウスやキーボード，ディスプレイ，スピーカ，ヘッドセットなどである．後者は，CPU，メモリ，HDDおよびそれに保存されるデータ，ネットワークである．

分散システムは，後者を利用者に透過的に提供するために研究され発展してきた [2]．複数の計算機のCPU資源をまとめあげタスクを分散処理するAmeba OS，ネットワーク接続された計算機のデータに簡単にアクセスするためのネットワークファイルシステム（NFS）やSMB，ネットワーク接続された計算機のメモリを統一管理する分散共有メモリシステムなどが研究開発されてきた．しかし，前者のデバイスを人間が直接扱わなければならないことは現在でも変わっていない．

また利用者が持ち歩くことが可能なノートPCやPDA（モバイル計算機）が普及し，利用範囲が広がっている．モバイル計算機を用いて，重要な情報を扱ったり，複雑な作業に従事することが増えた．

計算機そのものの価値に加え，その中に保存されているデータの価値が大きくなる一方である．セキュリティ企業Symantec社は，ノートPCの中には100万ドルにおよぶ価値のデータが保存されていると試算している[1]．盗難や機器の故障，ファイル共有ソフトによる情報流出の被害は社会問題にもなっている．日立製作所は，報漏洩防止を目的としたストレージレスノートPCを企業向けに売り出し

[1] http://japan.cnet.com/news/sec/story/0,2000056024,20095429,00.htm

ている[2].

モバイル計算機の利用者の行動範囲は広がっており，移動先でも作業を継続して行なうことが求められる [4]．無線 LAN や Bluetooth は電波到達範囲が 10m〜100m であるが，室内では壁などに遮蔽され実際の利用可能範囲は狭く，離れた部屋やビルの別の階などの間では，それぞれの場所にアクセスポイントや受信機を備えた計算機を設置しなければならない．継続利用を可能する方法として，無線 LAN のローミング機能や Bluetooth のスキャッタネットなどのデータ転送機能を利用することも考えられる．しかし転送用のアクセスポイントや受信機を相互に電波到達範囲内に設置しなければならず，設置コストが大きい．また実際には移動元と移動先の間に通路など，アクセスポイントや受信機を設置しにくい場所があることが多い [4]．そのため，継続利用を可能にするためにはハードウェア的なサポートの他にソフトウェア的なサポートが必須となる．

また計算機は，遠隔地との連携を伴う製品開発所 [4] や，工事現場などでも利用されるようになった [22]．工事現場などの利用環境では手作業を伴うため，計算機の扱いが難しく，破損の危険も高い．そのため計算機は小型で単純なものが求められる [22].

以上のような状況により，以下の要求を満たすシステムが求められる．

- 高い安全性

  データを持ち歩かず，持ち歩くデバイスそのもののが故障・紛失しても損害が少ないこと．

- 広範囲の移動での継続利用

  部屋や建物の階を跨いだ広範囲の移動を伴っても，継続利用が可能なこと．

- 小型軽量でシンプル

  デバイスは持ち歩き易く，複雑でないこと．

1.2　本研究の目的と手法

本研究の目的は，1.1 で述べた利用環境に適合する，新たな計算機とデバイス利用形態の提案とその支援システムについて議論することにある．

[2] http://www.hitachi.co.jp/New/cnews/month/2005/02/0215.html

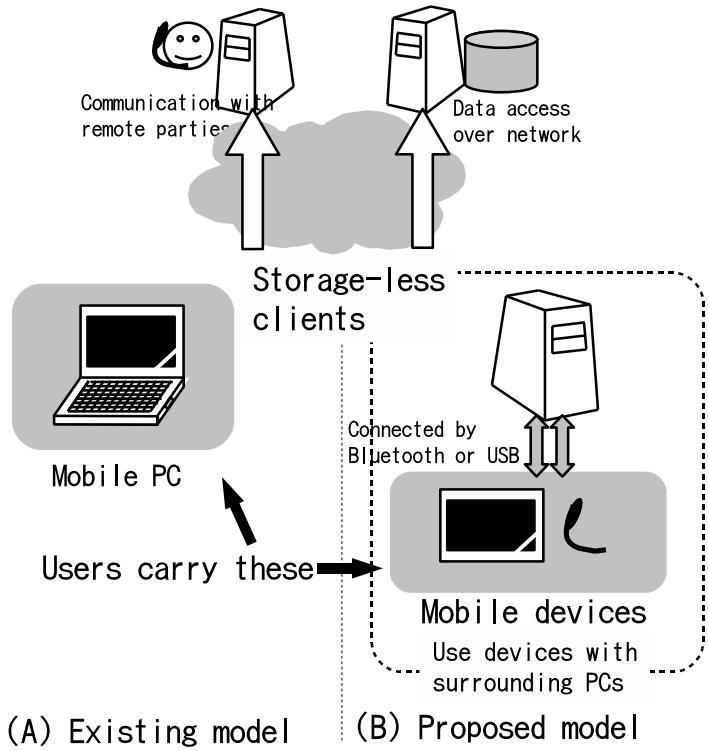


図 1.1: 提案するデバイス利用形態

1.2.1　デバイス携帯利用法

提案する利用形態では，人間が直接扱うデバイス（以下デバイス）を利用者が持ち歩き，手近にある計算機を経由して，必要なデータへのアクセスや遠隔地との相手と会話する（図 1.1）．図 1.1（A）はストレージレスのノート PC の構成である．ノート PC で扱うデータはすべてネットワークの先にあるストレージサーバに蓄えられるため，ノート PC が盗難されても機密情報が漏洩しない．図 1.1（A）ではユーザはノート PC を持ち歩くが，図 1.1（B）ではユーザが持ち歩くのはデバイスのみとなる．デバイスを手近な PC と接続し，ユーザは仕事を行なう．デバイスへは PC と接続するための必要最低限の機能のみ搭載すれば良く，より小型軽量化できる．またデバイスそのものの価値は低く，紛失・故障しても被害は最小限に抑えられる（図 1.2）．

この利用形態で，利用者が広範囲の移動をしてもデバイスの継続利用するためにはソフトウェアによる支援が必要となる．本研究では支援ソフトウェア Network Extended Device Management System（NextD）を設計・実現した．

提案するデバイス利用形態では，利用者の移動に伴いデバイスの接続先計算機

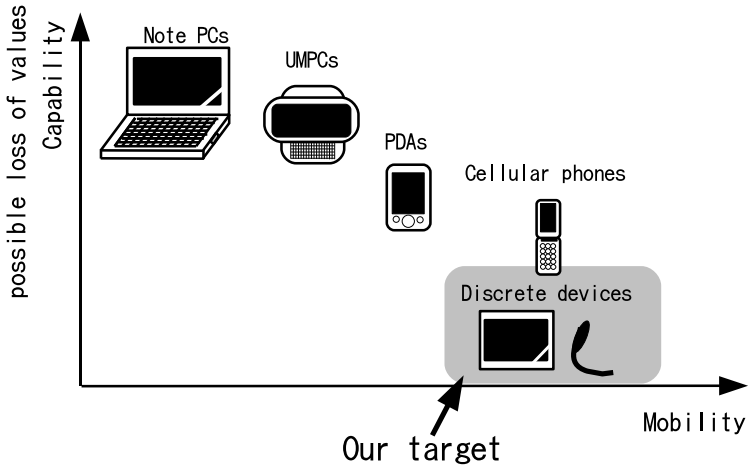


図 1.2: 可搬性 vs. 機能性・紛失や故障した際の損失の大きさ

が切替わる．これを本研究ではデバイス移動と呼ぶ．本システムはデバイス移動が起きても利用者がデバイス利用を継続可能とすることが目標である．

これを実現する支援技術は大きく分けて 2 つから成る．

- デバイス移動が起きても利用者がデバイス利用を継続可能とする支援技術
- 同技術を拡張し，利用者の移動範囲を拡大する支援技術

1.2.2　継続利用

本研究でいう継続利用は 2 つに分けられる．一つは移動先で移動元の仕事を継続することであり，もう一つは移動しながら仕事を行なうことである．提案するモデルでは，利用者が持ち歩くデバイスは単に計算機に接続する機能だけしかもたないため，移動先でも継続利用するために計算機間の連携によるソフトウェアサポートを行なう．デバイス接続先計算機の切り替わり（デバイス移動）を検出し，遠隔計算機へのデータ転送を行なうことで継続利用を実現する．

デバイス移動が起きても利用者がデバイス利用を継続可能とするためには，デバイス移動を検出し，別の計算機に移動したデバイスへアプリケーションがアクセス可能とすることが必要である．そのためネットワーク内の計算機上にサービスグループを形成し，デバイスの接続・切断しを監視する．またこれらの処理を自動化し利用者に手を煩わせない．

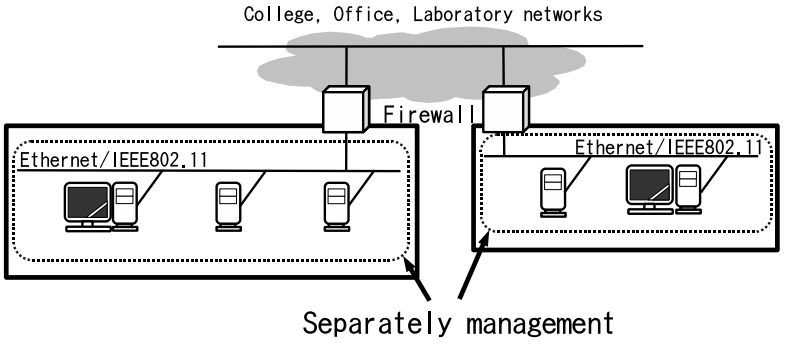


図 1.3: ネットワーク

1.2.3　利用範囲の拡大

利用者の移動範囲を拡大するために，ファイアウォールによって通信が制限されたサブネットワーク間でデバイス移動に対応する技術を実現する．また利用者の移動先に本システムによる支援が十分にない場合にもデバイスの利用継続を可能にする支援技術を実現する．例えば計算機は存在するがデバイスの接続先（USB ポートや Bluetooth 受信機）がない場合や移動先に無線 LAN アクセスポイントのみ存在する場合などでもデバイスの利用継続を可能にすることが目標である．これらを実現することにより，利用者は遠い部屋や別階などへの広範囲の移動が可能となる．

1.3　適用範囲と適用例

1.3.1　適用範囲

大学，研究所，企業内ネットワークなど，管理されたネットワークと決められた利用者を対象とする．

ネットワーク

近年セキュリティ意識の高まりにより，大学，研究所，企業レベルの対策から，研究室，部門レベルの対策も行われるようになってきた．割り当てられたネットワークに個別にファイアウォールを設置することもめずらしくない．これは管理者にとっては自分で管理するネットワーク以外は信用しないというポリシに従っ

ている．

本研究は，このように異なる管理者によって管理されファイアウォールが設置されているネットワークを想定する．利用者の移動範囲としては，これら個別に管理されるサブネットワークを跨ぐような範囲を想定する．サブネットワーク間の各計算機は許可された計算機以外は，相互通信を制限されている．例えば，サブネットワークにはプライベート IP が割り当てられたプライベートネットワークかもしれない．ネットワークに接続された計算機も，それぞれのサブネットワークで別々に管理されている．例えば，サブネットワーク内では NIS や LDAP などで個別にユーザを管理されており，サブネットワーク間では UID/GID などのユーザ識別情報が異なる場合も考えられる．

利用者

上記ネットワーク内で計算機の利用許可を与えられた利用者を対象とする．自身の計算機にシステムを導入できる．またあらかじめ共有鍵を配布可能である．パブリックな利用は対象としない．

デバイス

計算機に接続して人間が直接使うデバイスを対象とする．例えば，ヘッドセット，ヘッドフォン，スピーカ，マウス，キーボード，携帯型ディスプレイなどである．デバイスは単に計算機と接続する機能のみを具備するだけでよく，デバイスを簡潔につくることができると考えられる．また既存デバイスに特殊な機構を組み込むことなく，そのまま利用することができる．接続方式としては USB と Bluetooth を対象とする．

本研究では，ストレージデバイスは対象としない．代表的なストレージデバイスとして USB メモリが挙げられるが，これを持ち歩きながら使用することは少ないと考える．例えば，移動元の計算機に USB メモリを接続し，その状態で別の場所へ利用者が移動する状況を考える．この場合，移動元の計算機に接続したまま利用者が移動しても，USB メモリのデータ読み書きに支障はなく，利用者が持ち歩くことは考え難い．それに対して，本研究の対象デバイスは，利用者が直接使うため，持ち歩かなければ用をなさない．

また内蔵デバイス（スピーカやタッチパッド）も利用対象とする．

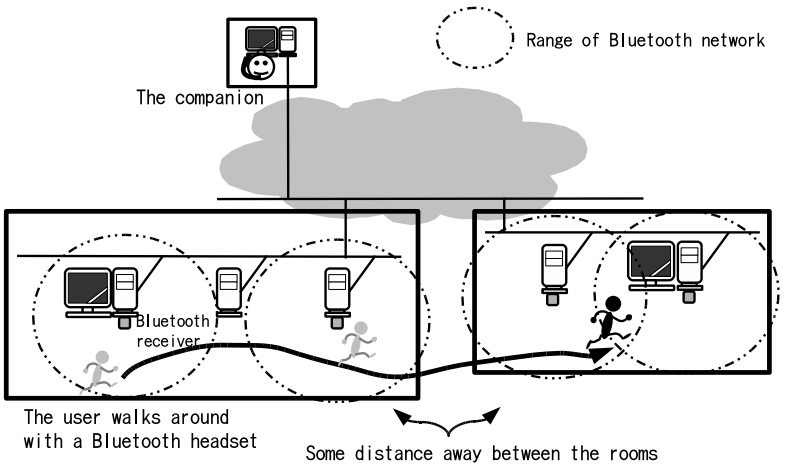


図 1.4: 移動しながら遠隔地の同僚と会話したい状況

1.3.2　適用例

提案するデバイス持ち歩き形態は例えば以下のような状況で用いる．

オフィスの自分の席で，計算機と無線接続する Bluetooth[5] ヘッドセットでどこか別の場所の同僚と会話をしているユーザがいる（図 1.4）．そのとき席から一時的に離れなければならない用事ができたとしても，同僚と会話を続ける必要がある．しかし移動先が遠く，Bluetooth 受信機の電波到達範囲から外れる場合は，会話を続けることができない．本システムのサポートがあるならば，この場合でも利用者は会話を継続できる．

1.4　本論文の構成と概要

2 章：デバイス携帯利用を支援する技術および関連研究

デバイス携帯利用，およびその支援技術の関連研究について述べる．本携帯利用法を，Bluetooth スキャッタネットによるデバイス携帯利用，無線 LAN 内蔵デバイスと比較する．また支援ソフトウェア技術として，遠隔デバイスアクセス技術，ホストモビリティ技術と比較する．

3 章：デバイス携帯利用支援システム：デバイス移動

デバイス移動が起きても利用者がデバイス利用を継続可能とするための機能を遠隔デバイスアクセス機構上に設計する．本機構は遠隔デバイスをデバイスファイルとしてアプリケーションに提供する．そのため利用者は既存アプリケーションを書き換えることなく利用できる．本機構はサブネットワーク内の計算機上にサービスグループを形成し，アプリケーションはグループ内の計算機に接続されたデバイスにアクセス可能となる．また本機構はサービスグループ内で起きたデバイス移動を検出し，アプリケーションから透過的に遠隔デバイスアクセス先を切り替える．これらは自動的に処理され，利用者に手を煩わせることはない．なお開発の容易化と拡張性を高めるため，機能の大部分はユーザレベルプログラムとして実現した．

4 章：デバイス携帯利用支援システム：移動範囲の拡大

3 章で実現したシステムを拡張し，利用者の移動範囲を拡大する支援システムを設計する．一つはファイアウォールによって通信が制限されたサブネットワーク間で，遠隔デバイスアクセスを可能とする支援システムである．これにより利用者は遠い部屋や別階などへの広範囲の移動が可能となる．もう一方は，利用者の移動先に本機構がなく無線アクセスポイントのみ存在する環境において，利用者のデバイス利用継続のための機能である．そのような環境でのアクセス継続は，デバイスの接続したモバイル計算機を利用者が携帯することで可能となる．アプリケーションからデバイスへのアクセスは，無線アクセスポイントに接続したモバイル計算機を経由して行なわれる．本機構には，利用者の移動にともないモバイル計算機の IP アドレスが切り替わっても，遠隔デバイスアクセスを継続することができる機能を実現した．

5 章：評価実験と議論

本システムの評価実験結果について述べる．デバイス特性を基にした評価実験により，本システムが遅延への要求が高いデバイスの性能要件を満たしていることを確認する．さらに通信インフラを除いた本システムの処理時間を計測する．またデバイス移動時の本システムの処理時間およびデバイス利用不可時間を計測評

価する．その結果から実用に耐える計算機の配置について議論する．

　また提案するデバイス利用法の有効適用範囲，セキュリティ，応用，将来性の観点で議論する．

第2章

デバイス携帯利用を支援する技術および関連研究

本デバイス利用法は，計算機とデバイスの新たな利用法を提案するものである．そこで，まず本提案を既存の計算機・デバイス利用方法と比較する．その後，本提案手法の支援システムを実現する諸技術である，遠隔デバイスアクセス，デバイス移動などを関連技術と比較する．

2.1　デバイス携帯利用法

提案するデバイス携帯利用法は，有線デバイスでは移動先で再接続して作業を継続，無線デバイスでは移動しながら作業を継続が可能とする．以下，移動先で作業を再開，移動中に作業を継続するという観点で，既存の計算機・デバイス利用方法と本研究とを比較する．

2.1.1　計算機の携帯

移動先で作業を継続する場合，ノート PC を持ち歩き移動先で作業を続けることが最も一般的な手段であると考えられる．ノート PC の多くは単独で利用するための機能を持っている．ただし，サイズが大きく移動場所や作業内容によっては作業が困難な場合がある [22]．また利用者が移動しながら利用するという用途には向かない．

利用者が移動しながら利用することを考慮された計算機としては PDA（Personal Digital Assistant），UMPC（Ultra Mobile PC）などの携帯端末がある．通常のノート PC に比べ小型で，端末を持ったまま作業を行なえるような入力装置，例えばペン入力を持つ．しかし，作業を行なうには画面に注意を向けなければならなず，歩きながらの利用は難しい．

Motile[22] は，電気通信サービス業者や海運業者など，別の作業を行ないながら利用することを考慮された小型端末である．片手で容易に利用できる単純なボタン式入力装置，音声フィードバックを中心とした出力装置などの特徴を持つ．ただし，この形態の小型端末は量産されておらず一般的に利用されてはいない．

別の方法として，Personal Server[49] を用いる方法がある．Personal Server はデータのみを格納するデバイスで，利用者はそれを持ち歩き，近くにある計算機の入出力装置を利用し作業する．持ち歩くデバイスは小型で携帯性が高い．このように作業内容を保存したデバイスを持ち運び，移動先に設置された共有端末上で作業を再開する．しかしディスプレイなどの入出力機器を占有して利用するため，他の利用者が使っていない時に利用は限られる．また公共端末で表示内容を盗み見られる可能性からは，自助努力で注意するか，重要な情報は表示しないようにする技術的支援が必要となる．

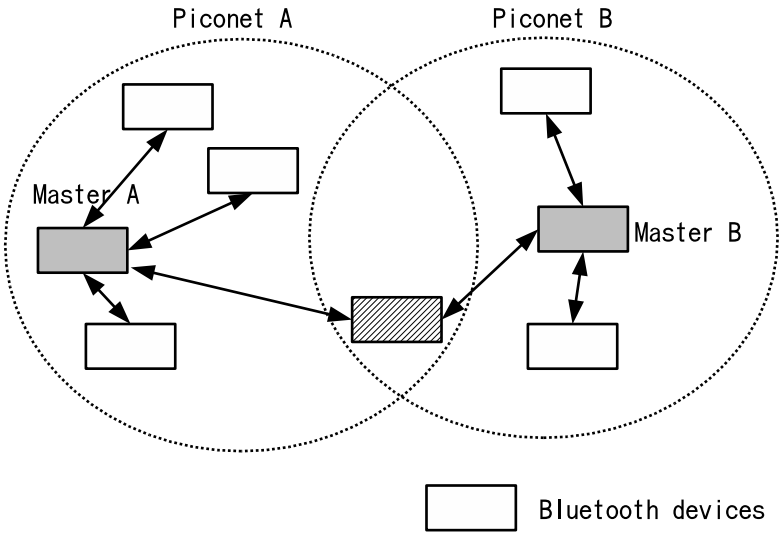


図 2.1: Bluetooth スキャッタネット

2.1.2　Bluetooth スキャッタネット

Bluetooth は，無線到達範囲内（通常 10 m）のデバイス間でピコネットを形成する．ピコネットは一台のマスタデバイスと複数のスレーブデバイスで構成され，ピコネット内の通信はマスタデバイスを経由して行われる．

一台のスレーブが複数のピコネットに所属し，中継を行なうことでスキャッタネットを形成する（図 2.1）．スキャッタネットを形成することにより，無線範囲の拡大や参加デバイスの増加を可能とする．

この仕組みを用いて，利用者のより広範囲の移動を実現することが可能である．しかし図 2.2 のように，隣り合うデバイスがお互いの無線到達範囲内に位置しなければならない．このため設置コストが大きくなる．実際には移動元と移動先の間に通路など，アクセスポイントや受信機を設置しにくい場所があることが多い [4].

またスキャッタネットは規格で策定されているものの，実際に実装されている機器はほとんどない．

2.1.3　無線 LAN 内蔵デバイス

デバイスに無線 LAN アダプタを内蔵し利用する方法が考えられる（図 2.3）．無線 LAN は Bluetooth に比べ遮蔽物による通信影響が大きいものの，通信可能範囲は 100 m と広い．デバイス内に IP プロトコルスタックと Mobile IP[32] などのモ

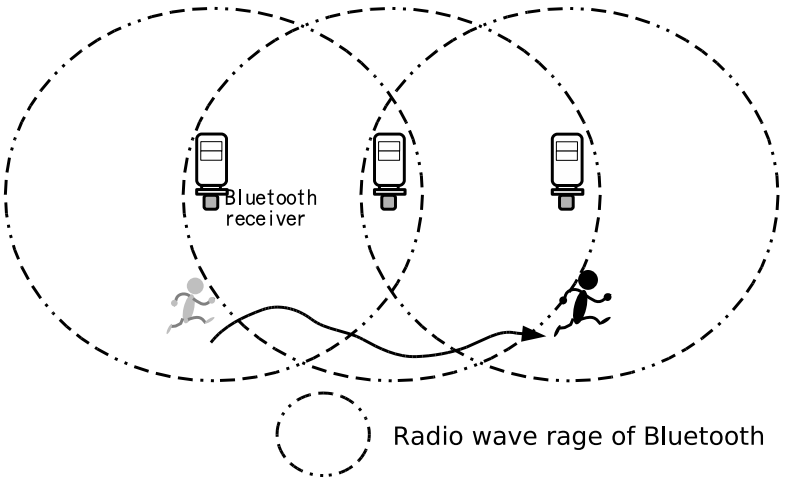


図 2.2: スキャッタネットを用いた利用者の移動範囲の拡大

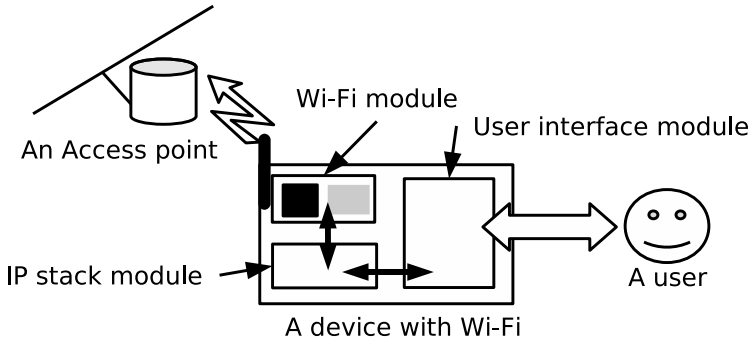


図 2.3: 無線 LAN 内蔵デバイス

ビリティを実現する機能を導入することで，利用者の広範囲の移動が可能となる．しかし，これらの機能を組み込むことは，デバイスの大型化を招き，製造コストも大きくなる．

2.1.4　遠隔ログインと遠隔プロトコル

移動先で作業を単に継続するだけならば，移動先にある計算機にログインして，ssh などで元の計算機に遠隔ログインすることも考えられる．ssh の場合は仮想端末での作業となるが，X Window System[41] や VNC[36] など遠隔画面表示プロトコルを使って移動元の計算機とほぼ同じ作業状況を再現することもできる．

これらの手法を用いるためには，移動先に利用者がログイン可能な計算機が設置されている必要がある．またディスプレイなどの入出力機器を占有して利用す

るため，他の利用者が使っていない時に利用は限られる．また画面以外のデータ，例えば音声や音楽などを聴くには別のソフトウェアの補助が必要となるなどの問題点がある．

2.1.5　まとめ

既存の計算機・デバイスの利用方法について述べた．それぞれ一長一短あるものの，利用者が持ち運ぶデバイスが小型軽量で紛失時のコストが小さく，広範囲の移動時にも継続利用できるものはない．

本提案手法は，支援システムを導入し利用者が既存デバイスを使いながら広範囲の移動を可能とするものである．デバイスが小型軽量で紛失時のコストが小さいが，複数の計算機に支援システムを導入しなければならないため，適用可能な条件は限られる．本提案手法の有効適用範囲については 5.3.1 で述べる．

2.2　デバイス携帯利用を支援するソフトウェア技術

ここでは，デバイス携帯利用を支援するソフトウェア技術として何が必要であるか明らかにする．

デバイス携帯利用では，計算機に接続して人間が直接利用するデバイスを利用者が持ち歩く．USB デバイスでは，利用者は移動先の計算機で作業を再開することができる．Bluetooth デバイスでは，移動中にも利用者は作業を継続できる．

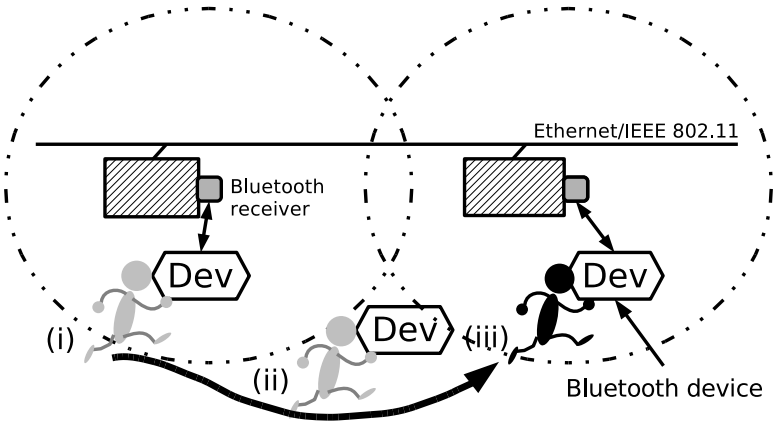
利用者はある計算機でアプリケーションを動かしデバイスを利用する．例えば，音楽再生ソフトウェアを用い USB ヘッドフォンで音楽を聴く，VoIP ソフトウェアを用い Bluetooth ヘッドセットで遠隔地の同僚と会話をする．この作業を，移動中，移動先でも継続することが支援ソフトウェアの目的である．

2.2.1　デバイス移動

デバイス携帯利用を支援するソフトウェア技術において，継続利用を実現するための最も大きな課題はデバイス移動に対応することである．デバイス移動とは，デバイスの接続先計算機が切替わることを指す（図 2.4，2.5）．利用者は，デバイスを利用する際，デバイスの他にそれに対して入出力を行なうアプリケーションを動かしている．デバイスが接続されている計算機を離れてもデバイス利用を可能とするためには，デバイス移動が起きても，移動元計算機のアプリケーションプログラムが移動先計算機のデバイスへアクセスする手段が必要となる．

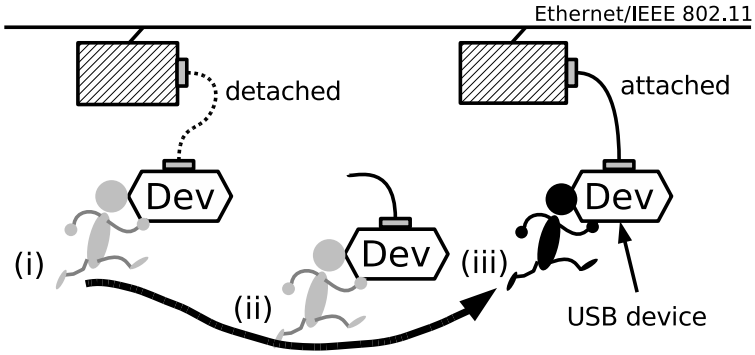
これを実現するためには，2 つの方法が考えられる．1 つは別の計算機に移動したデバイスに，アプリケーションが遠隔アクセスすることを可能とする方法（図 2.6）．もう 1 つは，デバイス移動に合わせて，アプリケーションも移動させる方法である（図 2.7）．

携帯利用法の実現のための新たなソフトウェアの導入は，利用者の移動範囲内に留めることが望まれる．例えば，利用者がアプリケーションに VoIP ソフトウェアを利用している場合，会話相手は遠隔地にいる．携帯利用法の実現のために，遠隔地にいる相手に特別なアプリケーションや支援ソフトウェアの追加インストールを強要することは望ましくない．単純なプロセス移送，もしくは follow me アプリケーションフレームワークでは，遠隔地のアプリケーションと通信を継続するために，移動後に通信を回復する機構が遠隔地にも必要となる [45, 46]．またミドルウェアなどによる実現方法では，特定の実行環境に合わせてアプリケーション



（i）計算機に接続していた Bluetooth デバイスが,（ii）利用者の移動により計算機と接続が切れ,（iii）別の計算機と接続する.

図 2.4: デバイス移動（Bluetooth の場合）



（i）計算機に接続されていた USB デバイスを利用者が取り外し,（ii）目的地へ移動,（iii）別の計算機へ接続する.

図 2.5: デバイス移動（USB の場合）

を改変または新規作成しなければならない.

Zap[31] は，Linux 上に実現した既存アプリケーションや通信を行なうアプリケーションにプロセス移送環境を提供する．利用者に合わせた計算機間のアプリケーションの移動は，通信相手のアプリケーションに透過である．移動元，移動先の計算機に LKM（Loadable Kernel Module）を導入する必要がある．また NFS（Network File System）[39] によるファイル共有を前提としているため，分割管理下にあるネットワーク環境では，NFS を中継する支援機構が必要となる.

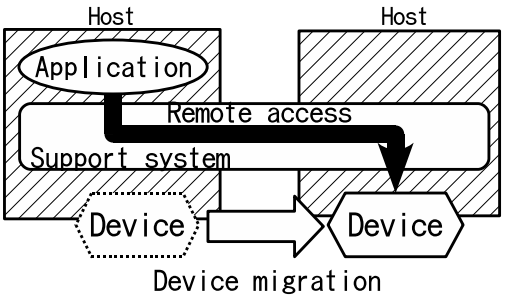


図 2.6: 遠隔アクセスによる対応

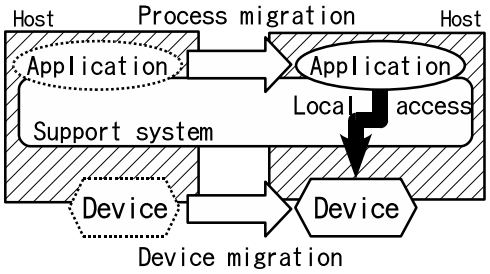


図 2.7: プロセス移送による対応

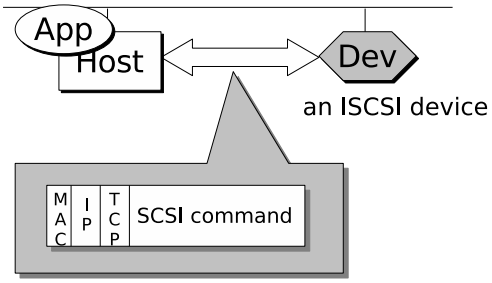


図 2.8: iSCSI

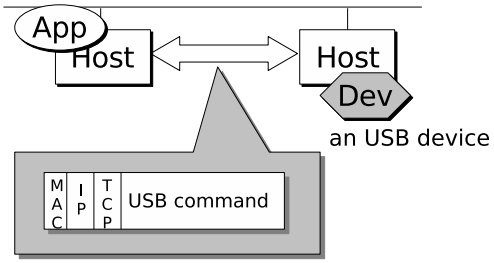


図 2.9: USB/IP

2.3　遠隔デバイスアクセス

遠隔の資源にアクセスするための技術は NFS をはじめ多く存在するが，デバイス携帯利用法において必要なのは，遠隔計算機に接続された周辺機器へアクセスする手段である．

ここでは関連技術について，実現するレイヤごとに分けて述べる．

2.3.1　ハードウェア，OS カーネル

iSCSI[40] は SCSI コマンドを TCP/IP で転送することでネットワーク接続された遠隔ストレージデバイスにアクセスする規格である（図 2.8）．SCSI はコマンド命令を発行するイニシエータと実際にコマンドを SCSI 機器に発行するターゲットに分けられる．SCSI では，イニシエータとターゲットが計算機内部のバスで接続されているが，iSCSI では，バスが Fast Ethernet などのネットワークになったと考えることができる．iSCSI イニシエータが通信を隠蔽するため，iSCSI 機器を利用する計算機では，機器別のドライバやアプリケーションは SCSI 機器がローカル計算機に接続されているように見える．

USB/IP[23] は USB コマンドを IP ネットワークで転送することで遠隔計算機の

USB デバイスに遠隔アクセスする技術である（図 2.9）．USB は 1 つのホストコントローラとデイジーチェイン接続された複数の USB デバイスで構成され，ホストと USB デバイス間は USB コマンドでデータのやり取りがなされる．USB/IP では，ローカル計算機の OS カーネルに仮想 USB デバイスドライバを導入し，USB コマンドを転送する．そのため USB ホストコントローラドライバに改変の必要はなく，USB ホストコントローラやアプリケーションは USB デバイスがローカル計算機に接続されているように見える．iSCSI と異なり，対象となるのは計算機に接続された USB デバイスであり，既存の USB デバイスも利用可能である．

2.3.2　OS

RFS（Remote File System）[37] はデバイスファイルも扱うことができる分散ファイルシステムである．また Plan 9[34] は OS やファイルシステムのプリミティブとして遠隔デバイスアクセスのための機能を実現している．いずれのシステムも遠隔デバイスのための API は OS の標準的なファイル操作インタフェースである．

Sprite[52] は RPC による計算機間通信をサポートする分散 OS であり，ユーザレベルプログラムで実現された疑似デバイスを介して遠隔デバイスへのアクセスが可能である．アプリケーションは Sprite OS が提供する API で遠隔デバイスへアクセスすることが可能であり，ユーザレベルプログラムで実現されいるため機能拡張は容易である．

2.3.3　ミドルウェア

ミドルウェアによる実現方法としては GnomeVFS[19] や EventHeap[16] などのシステムが挙げられる．ライブラリ等ミドルウェアとして実現されたこれらのシステムは，ユーザの多様な要求に応える機能拡張が容易である．OS やデバイスを隠蔽し抽象化されたインタフェース（GnomeVFS では POSIX に似た API，EventHeap では URL を用いたアクセスインタフェース）で遠隔デバイスへのアクセスを可能とする．

しかし，これらのシステムは独自のインタフェースを用いるため，既存のアプリケーションをそのまま用いることは難しい．提供される API を使うようにプログラムを書き換えるか，新たに一からプログラムを作らなければならないことが多い．

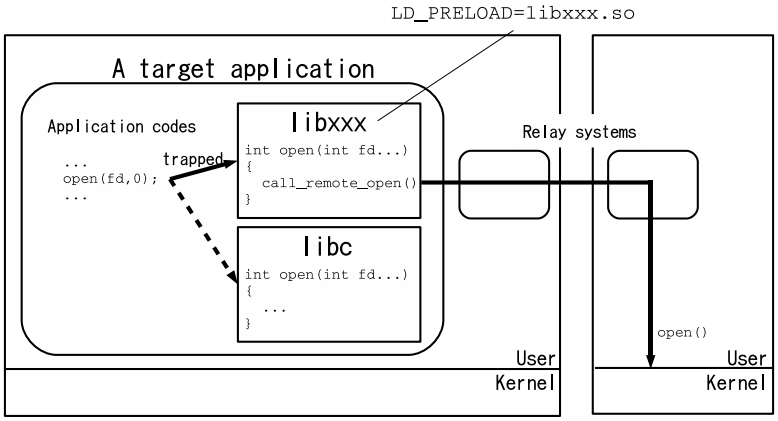


図 2.10: Preloading を用いた遠隔関数呼び出し

この制限を回避する方法として，リンカの reloading 機能を用いる手法がある．リンカは，プログラムの実行に必要な関数を持つ共有ライブラリを実行時に動的にメモリにロードする．この際，別の共有ライブラリを先にロードすることで，特定の関数呼び出しを用意した別の関数に置き換えることができる（図 2.10）．例えば，GNU Compiler Collection のリンカは，環境変数 LD_PRELOAD に指定したライブラリを先にロードする．

しかし Victor らは，リンカの preloading 機能には以下の制限があると指摘している [53].

1. 静的リンクを用いたプログラム
   Preloading は動的リンクされたライブラリに対して作用する．そのため静的にリンクされたライブラリを用いるようなプログラム，例えば setuid を用いるプログラムには適用できない．
2. システムライブラリ
   例えば Linux C ライブラリの名前解決ライブラリは socket システムコールを静的に呼び出しているため，preloading でそれを置き換えることはできない．
3. Preloading を用いる他のシステムとの共存
   Active Files[12]，Orfa[20]，Bypass[47] など preloading 機能を用いたシステムは多い．これらを同時に用いる場合，ロードされる順番やファイルディスクリプタなどの資源の取り扱いに注意しなければならない．

2.3.4　まとめ

遠隔デバイスアクセスをハードウェア・OS カーネルで実現すると，上位層からネットワークを隠蔽することができるが，対象デバイスが限られる．OS での実現では，アプリケーションへ透過的なサービスを提供できるが，導入や機能拡張が容易ではない．ミドルウェアで実現する場合は，導入や機能拡張が容易であるが，基本的に対象アプリケーションの改変が伴う．リンカの機能を用いることで既存 API を提供可能であるが，前述の通りいくつかの制限を伴う．

図 2.11: Personal Mobile Hub

2.4　補助計算機によるデバイス移動支援

利用者は補助計算機を携帯することにより，より広範囲の移動を可能とする．補助計算機は，デバイス接続機能と無線 LAN アダプタを含んだ最小限の機能だけ持てば良い．例えば，Personal Mobile Hub（PMH，図 2.11）[24] は，このような補助的な役割を果たす小型計算機である．PMH は，Bluetooth や Zigbee など短距離無線機能を持つ小型アプライアンスやセンサ・アクチュエータを集約し，外部通信の肩代り，相互イベント通知などの機能を提供する．

補助計算機は，利用者の移動後に移動元の計算機と通信を回復し，アプリケーションとデバイス間の移動透過性を提供する．

以下では，計算機の移動透過性を実現する技術・研究について述べる．

2.4.1　TCP/IP（トランスポート /インターネット）層

Mobile IP[32, 33] は固定 IP アドレスを持つホームエージェントがデータを中継することで移動ノードのモビリティを実現している．

しかし IPv4 向け Mobile IP は，IP ヘッダのソースアドレスに送信元（気付けアドレス）と異なるアドレス指定するため，IP ヘッダを厳密にチェックするルータが存在するネットワークでは利用できない．IPv6 向け Mobile IP は IPv4 の制限を克服しているが，ホームエージェントを必要とすることには変わりない．

Mobile IP は広域の計算機移動にも対応できる技術である．しかし利用環境によっては，計算機移動の移動範囲はより狭い場合が多い．例えばオフィスや大学

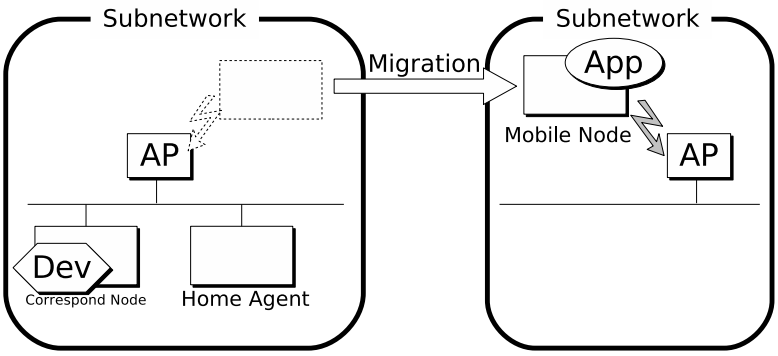


図 2.12: Mobile IP

では，利用者は建物内しか移動しない．

MSOCKS[26] は，通信相手のホストと移動計算機の間にプロキシを設置した構成で，トランスポート層で計算機移動に対応する．

Indirect TCP[1] も MSOCKS 同様，トランスポート層で計算機移動に対応する．移動計算機のアプリケーションは中継計算機上の支援機構を介して固定計算機へ間接アクセスする．支援機構は利用者の移動箇所ごとに設置され，移動の際には TCP セッション状態を転送・復元することで固定計算機から移動を隠蔽する．

Migrate[44] は，ネットワークレイヤ（IP 層）を書き換えることなく，計算機が移動してもエンドホスト間のコネクションを継続可能とする．Mobile IP と違い，ホームエージェントを必要とせず通信経路を迂回させる必要がないため，通信性能が高い．しかし，TCP プロトコルとエンドホストのアプリケーションを書き換える必要がある．

2.4.2　セッション層より上位の層

Rocks，Racks[53] は IP アドレスの変化や物理リンクの障害など，TCP では対応できない通信保障を行なう機構を実現している．Rocks は前述の preloading の機能でアプリケーションの通信データを横取りし，通信保障を行なう．一方 Racks は OS のパケットフィルタリング機構を利用している．

PlanB[3] は分散 OS Plan9 へ環境の動的変化に適応できる機能拡張を施した OS である．ファイルと資源の結合を実行時に動的解決することで，環境の変化に対応することができる．

S-proxy[46] は，ネットワークトポロジの動的な変化やユーザのネットワーク上

の移動が起きるような環境において，シームレスにサービスを連続させる技術である．S-proxy は階層構造を持ち，ノードの移動，中継経路の変更，サービスの移動といった変化する環境をそれぞれを層で対応する．

2.4.3　まとめ

計算機移動に対応する技術は実現する層によって，透過性，導入し易さ，ネットワーク制限に対する柔軟性が異なる．TCP/IP 層での実現は，透過性に優れているが，プロトコルスタックの改変が必要なため導入は容易ではない．セッション層より上位の層での実現では，導入が容易であるが，一般的に API が独自のものになる．透過性は提供するためには，preloading 等の機能を使わなければならない．

2.5　本章のまとめと本研究の特色

本章では，本提案を既存の計算機・デバイス利用方法と比較した．また，本提案手法の支援システムを実現する諸技術，遠隔デバイスアクセスを関連技術と比較した．

本研究の特色は，デバイス携帯利用法とそれを支援する技術，特にデバイス移動に対応することにある．

本提案手法は，支援システムを導入し利用者が既存デバイスを使いながら広範囲の移動を可能とするものである．デバイスが小型軽量で紛失時のコストが小さいが，複数の計算機に支援システムを導入しなければならない．

NextD は，遠隔アクセス機構の実現方法としてハイブリッド構成を採用している．デバイスファイルを遠隔デバイスアクセスのインタフェースとしてアプリケーションに提供する．また実装のほとんどはユーザレベルで実現，デバイス移動やホストモビリティの機能を実現する．Sprite[52] も同様にハイブリッド構成を採用している．Sprite はカーネル内処理はハードコーディングされているが，本システムは FUSE（Filesystem in Userspace）[9] という LKM を用いて遠隔デバイスアクセスを実現しており，他の OS への移植を可能としている．例えば，FUSE は Linux の他に FreeBSD 向けの実装が存在する．

計算機移動の機能はセッション層より上位の層で実現する．通信や計算機移動時の接続管理はユーザレベルで行なう．NextD が提供する API は，FUSE が提供するファイルシステムインタフェースである．そのためアプリケーションへ提供される API はネットワーク透過となる．しかし，適応的なアプリケーションを作成するためには，ネットワーク状態を取得する API が必要となる．そこで，NextD はネットワーク切断やデバイス接続などのイベントを取得するインタフェースも提供する．

第3章

デバイス携帯利用支援システム: デバイス移動

本章では，提案するデバイス携帯利用法を支援するソフトウェア NextD について述べる．NextD は，コア機能として遠隔デバイスアクセス機構を持ち，その上にデバイス移動が起きても利用者がデバイス利用を継続可能とするための機能を持つ．NextD は遠隔デバイスをデバイスファイルとしてアプリケーションに提供する．そのため利用者は既存アプリケーションを書き換えることなく利用できる．NextD はサブネットワーク内の計算機上にサービスグループを形成し，アプリケーションはグループ内の計算機に接続されたデバイスにアクセス可能となる．また NextD はサービスグループ内で起きたデバイス移動を検出し，アプリケーションから透過的に遠隔デバイスアクセス先を切り替える．これらは自動的に処理され，利用者に手を煩わせることはない．

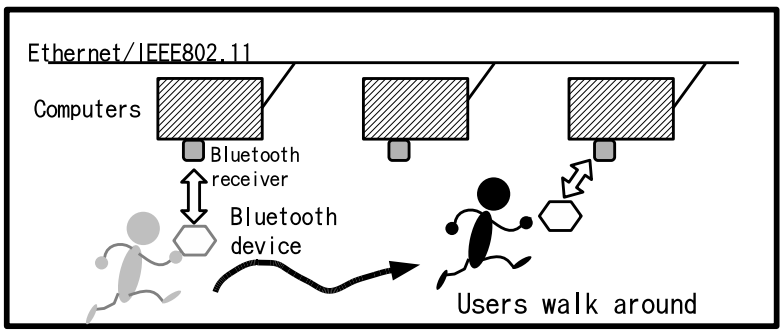


図 3.1: ネットワーク構成

3.1　設計

3.1.1　設計方針

以下の方針を基にシステムを設計する．

- 既存資産の活用

  既存アプリケーションが改変なしにそのまま利用できる．

- 利用者の負担軽減

  システムの導入や利用が容易である．計算機間のセットアップの自動化や，継続利用時の利用者の手を煩わせないほうが良い．

- 拡張性

  機能の多くをユーザレベルプログラムで実現し，容易に機能拡張できる．

3.1.2　対象とする計算機ネットワーク構成

対象とするネットワーク環境は図 3.1 のように単一のサブネットワークで構成され，利用者は計算機の付近をデバイスを持って移動する．計算機は Ethernet や IEEE802.11 で接続され，サブネットワーク内では相互に直接通信できる．USB ポートや Bluetooth 受信機を持ち，デバイスと接続する．

3.1.3　対象デバイス

計算機に接続して人間が直接使うデバイスを対象とする（表 3.1）．例えば，ヘッドセット，ヘッドフォン，スピーカ，マウス，キーボード，携帯型ディスプレイなどである．デバイスは単に計算機と接続する機能のみを具備するだけでよく，デバイスを簡潔につくることができると考えられる．また既存デバイスに特殊な機構を組み込むことなく，そのまま利用することができる．接続方式としては Universal Serial Bus（USB）と Bluetooth を対象とする．

USB は，シリアルポート，パラレルポート，PS/2 などのレガシポートの代替としてつくられた，計算機の起動中でもデバイスの抜き差しが可能な周辺機器接続規格である．現在，USB は広く普及し，ほとんどの計算機は接続ポートを持つ．USB 1.1 では 1.5 Mbps（Low Speed）と 12 Mbps（Full Speed），USB 2.0 ではそれに加え 480 Mbps（High Speed）の転送速度を持ち，キーボードやマウス，スピーカ，ストレージなど多様なデバイスをサポートする．

Bluetooth は，電波到達距離 10 m [1]，通信速度 721 kbps[2]の無線通信規格である．無線到達範囲内のデバイス間でパーソナルネットワーク（ピコネット）を形成する．ピコネットは一台のマスタデバイスと複数のスレーブデバイスで構成され，ピコネット内の通信はマスタデバイスを経由して行われる．また Bluetooth は，モバイル計算機で標準的に搭載されている無線 LAN（IEEE 802.11）に比べ消費電力が小さいという特徴をもつ [35]．

本研究では，ストレージデバイスは対象としない．代表的なストレージデバイスとして USB メモリが挙げられるが，これを持ち歩きながら使用することは少ないと考える．例えば，移動元の計算機に USB メモリを接続し，その状態で別の場所へ利用者が移動する状況を考える．この場合，移動元の計算機に接続したまま

表 3.1: 対象デバイス

接続方式	USB，Bluetooth，内蔵デバイス
対象デバイス種別	ヘッドセット，ヘッドフォン， その他オーディオ機器， マウス，キーボード， 携帯型ディスプレイ（現在は未対応）
対象外デバイス種別	ストレージデバイス（HDD，USB メモリなど）， モデム，ネットワークカード（無線 LAN など）

[1]一般的に使用されている Class 2 の場合．Class 1 は 100 m．

[2]Bluetooth v1.2 の場合．v2.0 では 2.1 Mbps であるが，流通している製品はまだ少数である．

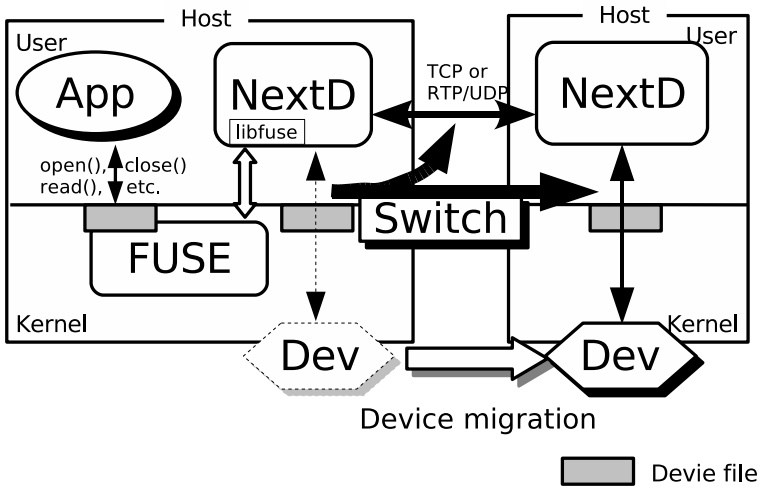


図 3.2: システム構成

利用者が移動しても，USB メモリのデータ読み書きに支障はなく，利用者が持ち歩くことは考え難い．それに対して，本研究の対象デバイスは，利用者が直接使うため，持ち歩かなければ用をなさない．

また内蔵デバイス（スピーカやタッチパッド）も利用対象とする．

3.1.4　基本構成

NextD は機能のほとんどをユーザレベルプログラムとして実現する．参加計算機のネゴシエーション処理，デバイス情報の交換・管理，遠隔デバイスアクセスのための通信コネクション管理などを行なう．OS が提供する通信インタフェース（ソケットインタフェース）を使い，TCP や UDP で他の計算機の NextD と通信を行なう．

NextD は FUSE（Filesystem in Userspace）[9] を用い実現され，Linux OS 上で動作する．FUSE はユーザレベルでファイルシステムを実装するための，Linux カーネルモジュールである．Wayback[7] バージョン管理システムや GMail FS[18] など多数アプリケーションが FUSE を用いて実装され，最新 Linux カーネルでは標準添付されている．これを利用することで NextD の機能のほとんどをユーザレベルで実現している[3]．そのため機能拡張は容易である．

[3]カーネルレベルの変更は，ioctl() 実現のため FUSE へ追加した 200 行程度である．

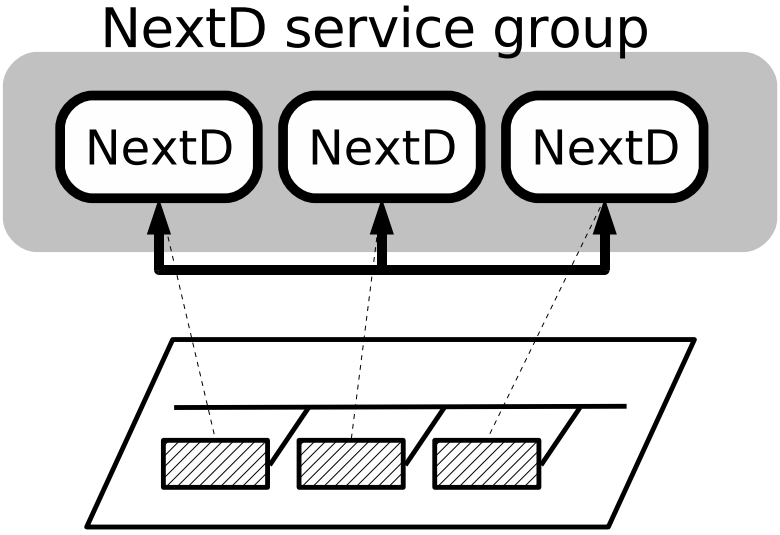


図 3.3: サービスグループ

3.1.5　サービスグループ

継続利用のため，利用者の移動範囲にある計算機群に NextD を配置し連携する．NextD は，ブロードキャスト通信で互いを自動的に発見し，**図 3.3** のようにサービスグループを形成する．サービスグループ内で，デバイス移動を検出し利用者のデバイス継続利用を可能とする．

3.1.6　通信

NextD 間のデバイスデータ通信に用いるプロトコルは，デバイスの要求に応じて TCP と RTP/UDP を使い分ける．データ欠落が起きないことが期待される HID と Audio デバイスには TCP を用い，逆にデータ欠落が起きても良いが，データ間隔が一定であることが期待される Voice デバイスには RTP/UDP を用いる．RTP の実現には oRTP ライブラリ [30] を用いた．

3.1.7　計算機とデバイスの発見

NextD は，遠隔デバイスアクセスサービスを提供する他の計算機を自動的に見つけサービスグループ（以下 **NextD サービスグループ**）を形成する．NextD サービスグループ内で，デバイスの存在が周知され，アプリケーションはデバイスに

アクセスすることができる．NextD サービスグループへの参加は，NextD 起動時と NextDが動く計算機がネットワーク接続した際に，UDPブロードキャストで参加を表明することで行なわれる．NextD サービスグループ内では，デバイス情報を共有し，デバイスの着脱イベントを随時通知し合う．上記処理は NextD で自動的に行なわれ，アプリケーションやユーザはそれを意識する必要はない．NextD 間で交換される UDP メッセージを，本研究では **NextD 制御メッセージ**と呼ぶ．

3.1.8　API

NextD は遠隔デバイスをローカル計算機のデバイスファイルとしてアプリケーションに見せる．そのためアプリケーションは，デバイスアクセスのためのインタフェースとして POSIX で定義されている API（UNIX 系 OS では open(), close(), read(), write() など）を用いることができる．これらのインタフェースは Windows OS や UNIX 系 OS などで標準的に提供されている [48]．またデバイス固有の操作のための API（UNIX 系 OS では ioctl()）も提供する．したがって遠隔デバイスアクセス機能を持たない既存プログラムを書き換えることなく利用できる．

現在の実装では，デバイスファイルを介してデバイスアクセスするアプリケーションにのみ対応している．例えば，mmap(2) を用いてデバイスに直接アクセスするアプリケーションへの対応は今後の課題とする．

また NextD は，サービスグループ内のデバイス着脱イベントやデバイス・計算機情報を取得するための API を提供する．これらの API は D-BUS[11] インタフェースで提供される．API の詳細は 3.3.3 で述べる．

3.1.9　デバイスアクセス実行方法

遠隔デバイスに対応するデバイスファイルへの操作（open(), close(), read(), write(), ioctl()）を，遠隔計算機上で NextD が代わりに行なうことで，遠隔デバイスアクセスは実施される（図 3.4）．アプリケーションがデバイスファイルを open すると，NextD は遠隔の NextD との間に TCP コネクションを張り，以降の read(), write(), ioctl() などの操作はそのコネクションで行なわれる．コネクションはアプリケーションがデバイスファイルを close した際に閉じられる．

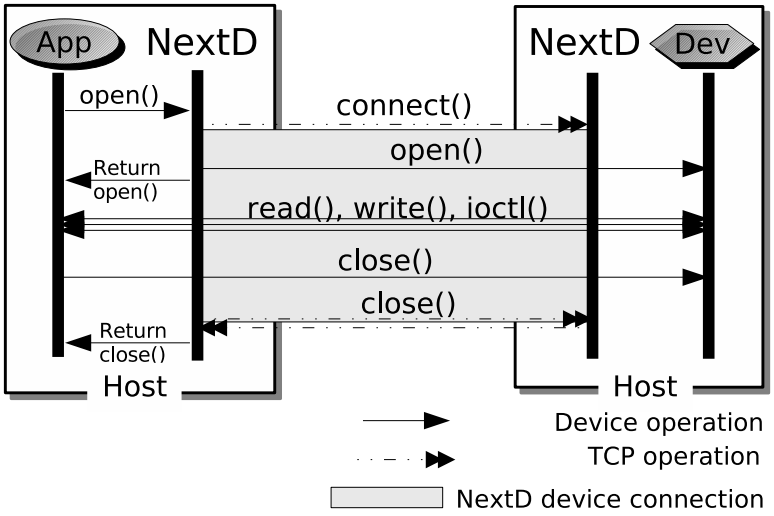


図 3.4: 遠隔デバイスアクセスと NextD デバイスコネクション

3.1.10　システム動作

セッション開始処理

セッション開始処理は，計算機がネットワークに接続されたり，計算機内で NextD が起動されるときに行なわれる．

1. 存在通知：UDP ブロードキャスト
   計算機がネットワークに接続されたり，計算機内で NextD が起動されるときに，他の計算機に自分の存在を通知する．
2. 存在確認応答：UDP ユニキャスト
   上記通知を受け取った NextD は，送信元 NextD に返答を返す．
3. デバイス情報通知：UDP ブロードキャスト
   デバイスに関する詳細な情報を通知する．
4. デバイス情報応答：UDP ユニキャスト
   上記通知を受け取った NextD は，デバイスに関する詳細な情報を返す．

アプリケーションのデバイスアクセス

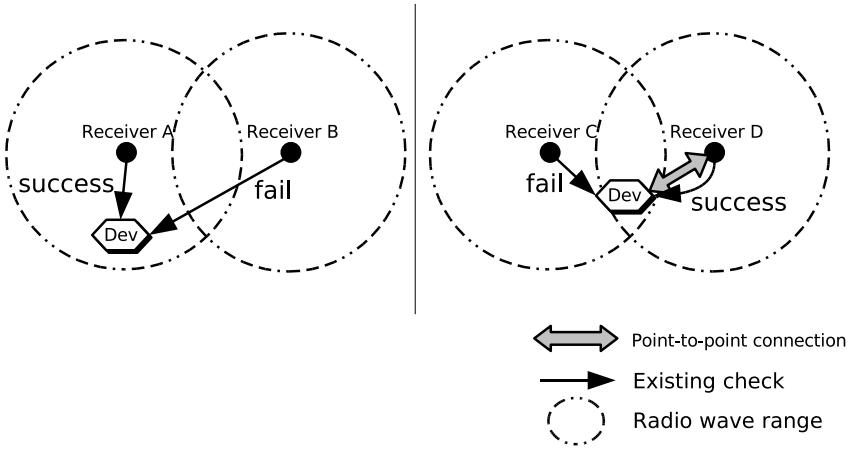
セッション開始処理を行なった NextD は，応答のあった NextD と TCP コネクション確立する．この段階で，アプリケーションはローカルに存在するかのよう

表 3.2: メッセージの種類

メッセージ	説明	捕捉
HELLO	存在通知	UDP ブロードキャスト
HOWLOW	存在通知応答	UDP ユニキャスト
DEVADD	デバイス接続	UDP ブロードキャスト
DEVDEL	デバイス切り離し	UDP ブロードキャスト
PING	存在確認	UDP ブロードキャスト
PONG	存在確認応答	UDP ユニキャスト

に遠隔デバイスにアクセスすることが可能となる．

例えば UNIX 系 OS では，参加計算機の/dev/ディレクトリを mount した状態となり，アプリケーションはインポートされたデバイスファイルを通して遠隔計算機のデバイスにアクセスする．



- （左）電波到達範囲内にデバイスが位置すれば存在確認に成功する．
- （右）受信機 C, D 双方の電波到達範囲内にデバイスが位置するが，デバイスは受信機 D と接続しているため，受信機 C は存在確認に失敗する．

図 3.5: Bluetooth におけるデバイス存在確認

3.2　デバイス移動

ここでは，デバイス移動時のアクセス継続を行なうために NextD に実現した機能について述べる．

3.2.1　デバイス追跡

デバイス切離し・接続検出

USB と Bluetooth では，デバイス切離し・接続検出の手法が異なる．

USB では，デバイスが実際に着脱されるため，OS の着脱イベント通知機構を使うことができる．NextD は D-BUS を用い Linux のデバイス管理デーモン hald からイベントを取得する．

一方 Bluetooth は，デバイス移動時にデバイス着脱が実際に起きるわけではない．そのため Bluetooth 受信機と，利用者が持ち歩く Bluetooth デバイス間の接続を監視する手法をとる．

Bluetooth は，受信機の電波到達範囲内にあるデバイスと 1 対 1 コネクションを張る．本研究では，定期的に無線デバイスの存在確認を行ない，返答がない場合に切断されたと判定する．存在確認には hcitool[6] の無線デバイス名を取得する機

能（Host Controller Interface の Remote Name Request コマンド）を用いる．無線デバイスが通信範囲内に存在し，かつ他の受信機とコネクションを張っていないときに処理に成功し，そうでない場合に失敗する（**図 3.5**）．この存在確認方法はデバイス接続判定にも用いられる．

NextD は，Bluetooth デバイスアドレスを用いてデバイスを同定する．デバイスアドレスはデバイス切断時に，サービスグループ内の NextD へマルチキャストされる．

なお本研究では，Bluetooth デバイスはサービスグループ内に一つのみ存在し，常にスレーブに固定されているものを対象とする．

デバイス移動判定

NextD は，上記方法で検出した着脱イベントを基にデバイス移動を検出する．着脱イベントは，デバイス同定に用いるシリアル番号とともに制御メッセージとしてサービスグループ内の NextD へマルチキャストされる．NextD は，切離しイベントの制御メッセージを受け取ってから，一定時間（ユーザが設定可能）以内に接続イベントの制御メッセージを受け取ったときデバイス移動が起きたと判定する．

シリアル番号を持たないデバイスの場合は，デバイスはデバイス種別[4]で同定を行なう．この方法は，NextD サービスグループ内に同種のデバイスが一つしか存在しない状況でのみ用いることができる．

3.2.2　自動切替え機構

デバイス状態の保存・復帰

デバイスによっては，利用する前にアプリケーションがデバイスに固有の設定を行うものがある．無線デバイスの場合は，移動先のデバイス（受信機）は初期化されていない．そのためデバイス移動後は，移動先のデバイス状態を移動元と同じ状態にしなければならない．本研究では，アプリケーションが ioctl() を用いて設定を行なうことに着目し，アプリケーションが発行する ioctl() を保存，デバイスが再接続した後に再発行することでデバイス状態を復帰している．

予備実験として，デバイス状態の復帰機能の効果を調べる実験を行なった．デ

[4]Vendor ID と Product ID で決まる．

```
$ strace mpg123 audio_01.mp3 2>&1 |egrep '(open|ioctl)'
...
open("/dev/dsp", O_WRONLY)              = 3
ioctl(3, SNDCTL_DSP_GETBLKSIZE, 0x806cb28) = 0
ioctl(3, SNDCTL_DSP_RESET, 0)           = 0
ioctl(3, SNDCTL_DSP_SETFMT or SOUND_PCM_READ_BITS, 0xbfe2dbf8) = 0
ioctl(3, SNDCTL_DSP_STEREO, 0xbfe2dbf4) = 0
ioctl(3, SNDCTL_DSP_SPEED or SOUND_PCM_READ_RATE, 0xbfe2dbf0) = 0
ioctl(3, SNDCTL_DSP_SETFMT or SOUND_PCM_READ_BITS, 0xbfe2dbf8) = 0
ioctl(3, SNDCTL_DSP_STEREO, 0xbfe2dbf4) = 0
ioctl(3, SNDCTL_DSP_SPEED or SOUND_PCM_READ_RATE, 0xbfe2dbf0) = 0
...
open("audio_01.mp3", O_RDONLY)          = 3
open("/dev/dsp", O_WRONLY)              = 4
ioctl(4, SNDCTL_DSP_GETBLKSIZE, 0x806cb28) = 0
ioctl(4, SNDCTL_DSP_RESET, 0)           = 0
ioctl(4, SNDCTL_DSP_SETFMT or SOUND_PCM_READ_BITS, 0xbfb69314) = 0
ioctl(4, SNDCTL_DSP_STEREO, 0xbfb69310) = 0
ioctl(4, SNDCTL_DSP_SPEED or SOUND_PCM_READ_RATE, 0xbfb6930c) = 0
open("/dev/dsp", O_WRONLY)              = 4
ioctl(4, SNDCTL_DSP_GETBLKSIZE, 0x806cb28) = 0
ioctl(4, SNDCTL_DSP_RESET, 0)           = 0
ioctl(4, SNDCTL_DSP_SETFMT or SOUND_PCM_READ_BITS, 0xbfb69434) = 0
ioctl(4, SNDCTL_DSP_STEREO, 0xbfb69430) = 0
ioctl(4, SNDCTL_DSP_SPEED or SOUND_PCM_READ_RATE, 0xbfb6942c) = 0
```

図 3.6: mpg123が発行する ioctl

バイス状態の復帰機能を有効にした実験では音声が正しく再生されたが，デバイス状態の復帰機能を止めた実験では，デバイス移動後に雑音が USB スピーカから流れてきた．これは mpg123が初期化時に行なうデバイスの設定[5]をしていないためである．これによりデバイス状態の復帰機能が有効に働くこと，またデバイス状態の復帰はデバイス移動に対応するためには必須の機能であることがわかった．

デバイスデータの転送

NextD は，切断イベントを受け取るとアプリケーションがデバイスに対して行なう処理（read() や write() など）をサスペンドし，デバイス移動が完了した後，その処理を再開させる．アプリケーションはこれらの処理を意識する必要はないが，NextD 拡張 API を用いて切断イベントを取得し，独自の処理を行なうことも可能である．

NextD はデータのバッファリングを行なうため，多くの場合はデバイスデータ

[5] ioctl() の種類でいうと SNDCTL_DSP_SETFMT, SNDCTL_DSP_STEREO, SNDCTL_DSP_SPEED である．

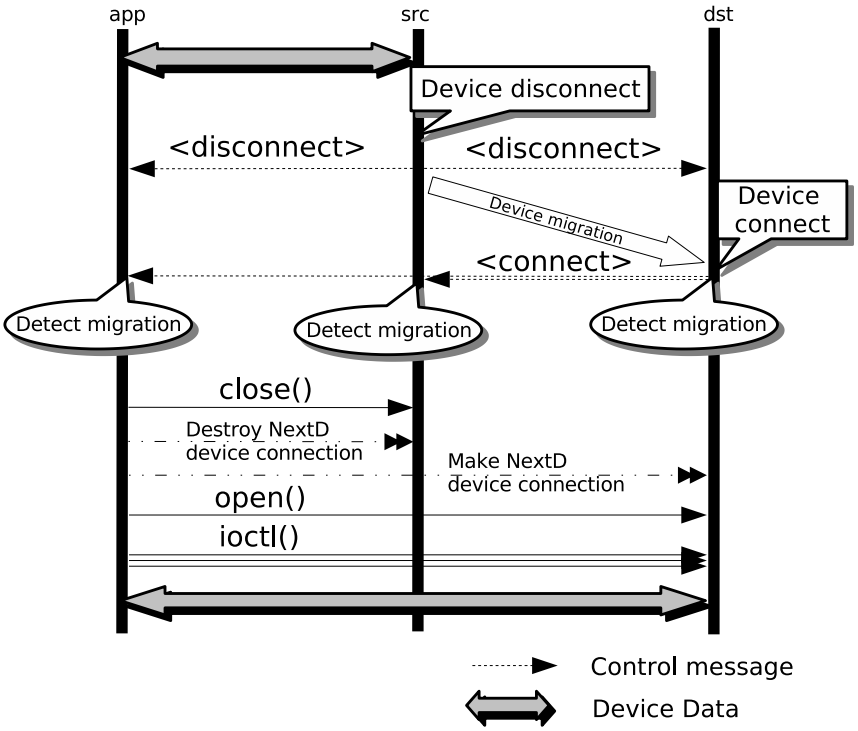


図 3.7: デバイス移動時の処理

は正しく移動先の計算機へ送られる．しかし，すべての状況において NextD はデバイスデータの完全性を保障できるわけではない．アプリケーションの処理のタイミングにより，デバイスデータが重複・欠損する．この特徴はストレージのようなデバイスには向いていない．しかし NextD が対象とするデバイスでは，影響はそれほど大きくない．例えば，Audio デバイスでは最大 200 ms 分[6]のデバイスデータが重複する可能がある．デバイス移動中に音声出力が止まった後，重複分の音声が再度再生される．

3.2.3　デバイス移動時の動作

本章では，デバイス移動時の NextD の処理を手順に沿って説明する（図 3.7）．ここでは Bluetooth デバイスの場合について述べるが，USB の場合でも NextD の動作は同じである．

[6]音楽再生ソフトウェア mpg123 が一度に write するデータサイズ

デバイス移動の検出

1. 初期状態
   ユーザはある計算機（移動元ホスト）に接続している Bluetooth デバイスを使っているとする．移動元ホストの NextD は定期的にそのデバイスが存在するかどうか 3.2.1 で述べた手法でチェックしている．その状態から，ユーザは移動元ホストから別の計算機（移動先ホスト）付近へ移動する．
2. デバイス切離し
   ユーザが移動し，移動元の受信機の通信範囲から外れると，移動元計算機では存在確認に失敗する．NextD デバイス切り離しが起きたと判断し，サービスグループ内に切離しイベントをマルチキャストし通知する．切離しイベントを受け取った NextD は，存在確認処理を開始し，計算機の受信機の通信範囲内に無線デバイスが移動してきたどうか調べる．
3. デバイス接続
   無線デバイスは受信機に対する処理に失敗すると，コネクションを破棄し新たなコネクションを受け付ける．この状態になった無線デバイスは，NextD の存在確認に応答する．NextD は検出した無線デバイスとコネクションを張り，デバイス接続イベントをマルチキャストする．これで図 3.7 のようにすべての NextD でデバイス移動が検出され，デバイス移動元と移動先の NextD でデバイスデータ転送先の切替えの準備が整う．

デバイスデータ転送先の切替え

- TCP

  アプリケーションが動いている計算機の NextD は，デバイス移動を検出すると移動先の NextD へコネクションを切替える．その際，移動元計算機のデバイスファイルを close し，移動先計算機のデバイスファイルを open する．

- RTP/UDP

  移動元の NextD との RTP セッションを終了し，移動先の NextD と新たな RTP セッションを確立する．その際 RTP のタイムスタンプを初期値に戻す．

いずれの場合もデバイスデータを転送する前に，必要ならば 3.2.2 で述べた手法でデバイス状態を復帰する．

デバイスデータのバッファリング

デバイス移動中にデバイスデータを NextD でどう扱うかは，アプリケーションに依存する問題である．例えば，ネットニュースを聴いている場合は

3.2.4　通信範囲外への移動

デバイスを持った利用者は，一時的に受信の通信範囲から外れる可能性がある．このときデバイスと受信機間の通信は途切れ，利用者はデバイスの利用が一時的に利用できなくなるが，受信機との接続が戻り次第，利用を再開できる．NextD は通信の途切れが一定時間内の場合にデバイス移動と判定し，通信を再開する．受信機とデバイス間の通信が途切れたまま復帰しない場合には，NextD はデバイスが離脱したと判断し，デバイスファイルを close，通信路を閉じてアプリケーションにエラー（EINVAL）を返す．アプリケーションは 3.1.8 で述べた API で離脱イベントを取得することもできる．

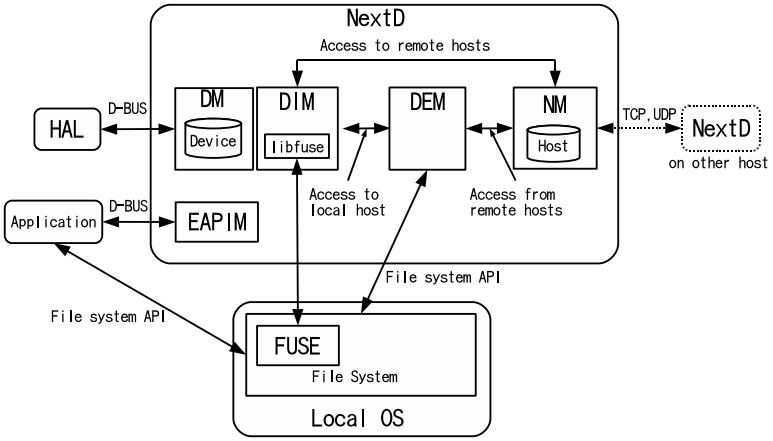


図 3.8: NextD 内部構成

3.3　実装の詳細

3.3.1　NextD 内部構成

- DM（Device module）：デバイス情報を管理するモジュール

  ローカルと遠隔計算機のデバイス情報を管理する．ローカルデバイスの情報は HAL から取得し，遠隔デバイスは，遠隔の NextD から送られる情報を NM から受け取る．デバイス移動の判定もこのモジュールで行なう．

- DIM（Device file import module）：遠隔デバイスをデバイスファイルとして見せるモジュール

  FUSE とやりとりし，アプリケーションのファイル操作を受け取り処理する．遠隔デバイスへの操作は NM を介して遠隔の NextD へ渡し，ローカルデバイスへの操作は DEM へ渡す．また DM のデバイス情報や NM のホスト情報から，ディレクトリやファイルを作り FUSE に渡す．

  デバイス移動にともなう，ファイル操作の発行先計算機の切り替えや ioctl 履歴などはこのモジュールで行なう．

- DEM（Device file export module）：デバイスファイルを遠隔計算機に渡すモジュール

  ローカルの DIM や遠隔の NextD からの要求に従い，ローカルデバイスファイルに対して処理を行なう．

- NM（network module）: 通信を行なうモジュール

  遠隔の NextD との通信，デバイスデータや制御メッセージなどのやりとりを行なう．ホスト情報を管理し，セッション開始・終了処理を行なう．

  4 章で述べる，移動したホストへの再接続処理や通信保障のためのデータバッファリング（4.3.5 参照）などはこのモジュールで行なう．

- EAPIM（Extended API module）:デバイス情報やイベントなどをアプリケーションに通知するモジュール

  アプリケーションの要求に応じ，他のモジュールが持つ情報を返したり，デバイス移動やデバイス着脱などのイベントをアプリケーションに通知する．アプリケーションとのやりとりは D-BUS で行なう．アプリケーションの問い合わせに返答する RPC（Remote Procedure Call）タイプのものと，アプリケーションが登録したイベントが起きたら通知するシグナルタイプのインタフェースを持つ．

3.3.2　FUSE の ioctl 対応

前述の通り FUSE は ioctl に対応していない．デバイスへのアクセスを想定しておらず，ioctl の代わりに getxattr(2) や setxattr(2) を使うことを要求している．

Linux オンラインマニュアルでは，ioctl の関数宣言は

```
int ioctl(int d, int request, ...);
```

となっているが，通例，引数は 3 つで，第 3 引数の型は `char *`である．ioctl の特殊なところは，この第 3 引数が構造体などへのポインタであったり単なる即値であったり，さらにドライバから結果を返すために渡されたデータ領域が書き換えられる場合がある．昔は，この挙動はアプリケーションとドライバのみが知り得たが，現在では第 3 引数の扱いを示す情報が `request` 変数に組み込まれることが推奨されている [51].

一方，Linux カーネルのファイル操作テーブルの ioctl エントリは

```
int (*ioctl)(struct inode *inode, struct file *file,
             unsigned int cmd, unsigned long arg);
```

```
size_t argsize = 0;

if ((_IOC_DIR(cmd) & _IOC_READ) && (_IOC_SIZE(cmd) > 0)) {
  char __user *userptr = (char *) arg;
  argsize = _IOC_SIZE(cmd)
  __copy_from_user(inarg.argbuf, userptr, argsize);
}
if ((_IOC_DIR(cmd) & _IOC_WRITE) && (_IOC_SIZE(cmd) > 0)) {
  argsize = _IOC_SIZE(cmd)
}

inarg.cmd = cmd;
inarg.arg = arg;
inarg.argsize = argsize;
```

inarg.argbuf にアプリケーションから渡されるデータが，argsize にアプリケーションから渡されるデータサイズが入る．なおエラー処理などは省略している．

図 3.9: FUSE 内の ioctl 処理

のように定義されている．そして Linux カーネルには cmd 変数から情報を取り出すマクロが定義されている[7]．例えば _IOC_DIR(cmd) & _IOC_READ が真ならば，アプリケーションが渡したデータ領域が書き換えられることを意味し，_IOC_SIZE(cmd) でそのサイズを知ることができる．

以上のように第 3 引数のデータ領域は，アプリケーションがドライバにデータを渡すとき，ドライバがアプリケーションにデータを返すとき，そしてその両方に用いられる．遠隔計算機で ioctl を正しく発行するために，アプリケーションからのデータを遠隔計算機に渡し，遠隔計算機のドライバからのデータをローカル計算機に戻す．遠隔計算機では，図 3.9 のようにアプリケーションから取得したデータを指すアドレスを引数に ioctl を発行する．即値の場合も考慮して，第 3 引数そのものの値も遠隔計算機へ渡される．

libfuse のファイル操作テーブルの ioctl エントリは

```
int (*ioctl)(const char *path, const int cmd, const unsigned long arg,
             char *argbuf, size_t argsize, struct fuse_file_info *fi);
```

のように定義されている．

遠隔計算機での第 3 引数が即値か否かの判定は，argsize が 0 か否かで判断される．

[7] /usr/include/asm/ioctl.h 参照

3.3.3　拡張 API

NextD は，デバイス着脱，新規ホストのサービスグループ参加・離脱などのイベントをアプリケーションに通知する拡張 API を持っている．アプリケーション開発者は，この拡張 API を用いて適合的な処理をアプリケーションに追加することができる．

NextD が通知するイベントは以下の通りである．

デバイス接続 ：

デバイスがサービスグループ内のいずれかの計算機へ接続された．

デバイス切断 ：

デバイスが計算機から取り外された．

デバイス使用開始 ：

サービスグループ内のデバイスがあるアプリケーションによって使用され始めた（デバイスファイルが open() された）．

デバイス使用終了 ：

使用中のデバイスが開放された（デバイスファイルが close() された）．

ホスト参加 ：

新たな NextD がサービスグループへ参加した．

ホスト離脱 ：

サービスグループ内の NextD がサービスグループから抜けた．

NextD から取得できるデバイス・ホスト情報は以下の通りである．

- サービスグループ内のデバイス一覧
- サービスグループ内のホスト一覧
- デバイスの接続先ホスト
- デバイスの詳細（3.3.4 参照）

```
# 登録
@bus = DBus::SessionBus.new
@bus.add_signal_receiver(handler, ‘ DEVICE ’, ’org.nextd.EAPIM’,
’org.nextd.EAPIM’, ’/org/nextd/EAPIM/EXTAPI’)

# イベントハンドラ
def handler( dbus_if, member, svc, obj_path, message )
   remote_service = @bus.get_service("org.nextd.EAPIM")
   remote_object = remote_service.get_object("/org/nextd/EAPIM/EXTAPI",
"org.nextd.EAPIM")
   devices = remote_object.get(“ ALL_DEVICES")
   …
end
```

図 3.10: Ruby プログラムにおける拡張 API の利用例

図 3.10 に利用例を示す．拡張 API は D-BUS インタフェースで提供される．アプリケーションは通知してほしいイベントを指定して，D-BUS にイベントハンドラを登録する．この利用例では，サービスグループ内のデバイスの状態変化イベント（’DEVICE’）が起きた場合，イベント通知し，handler メソッドを呼び出すように指定している．イベントハンドラ内では，イベント通知があった場合，サービスグループ内のデバイス情報をすべて取得している．

図 3.11 に応用例を示す．このプログラムは Web インタフェースを持つデバイス状態監視ツールである．NextD から，ホスト情報とデバイス情報をすべて取得し Web ブラウザに表示することができる．またデバイス状態変化を監視し，デバイス移動が起きた場合，それをリアルタイムで表示することができる．

3.3.4　デバイス情報の詳細

NextD は，デバイスの詳細やデバイスファイルのパス，ホスト ID などをサービスグループ内で共有する．デバイス固有の情報としては，hald から取得する Hardware Abstraction Layer（HAL）[14] で規定された情報が用いられる．例えば，デバイス情報としてはデバイス種別（マウス，スピーカなど），商品名，ベンダ ID，プロダクト ID などである．また利用者が持ち歩くデバイスだけでなく，内蔵スピーカや USB ポートなどの情報も含まれる．

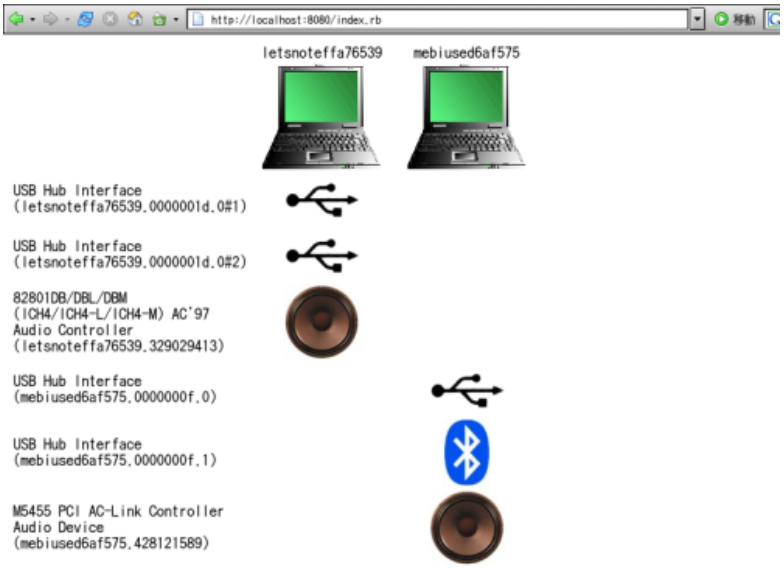


図 3.11: 拡張 API の応用例（デバイス状態監視プログラム）

3.3.5　使用中 USB デバイスの取り外し

本システムでは，利用者は使用中の USB デバイスを取り外して別の計算機に接続，利用を再開する．使用中のデバイスを取り外すことは，PnP（Plug and Play）処理の中では，予期せぬ取り外し [50] として分類され，場合によってはデータ欠損，カーネルパニックなどの障害を引き起こす可能性がある．本研究で扱うデバイス種別には，ストレージデバイスは含まれておらず，データ欠損による影響は小さい．カーネル障害は，カーネルのバージョンの違いによっては影響が異なるかもしれない．しかし，現在我々が利用している Linux カーネルバージョン 2.6.9-1.667, 2.6.12-2.3.legacy_FC3 では，カーネルパニックなどの障害が起きていない．

3.4　従来研究との比較

NextD の計算機やデバイスの発見は P2P やオーバレイネットワーク，例えば JXTA[21] のモデルと類似する点を持つ．JXTA の Advertisement と Peer discovery はそれぞれ資源の周知と検索の機能をアプリケーションに提供する．一方 NextD はアプリケーションが明示的な検索要求をしないことを前提としているため，資源（デバイス）情報をあらかじめ共有する機能のみで実現しているといえる．これはデバイス移動を即時検出する目的と合致する．また JXTA は基本的な通信機能のみをもち，デバイス移動（資源の移動）が起きてもサービスを継続させる機能に関しては JXTA モデルの範囲外である．

Rocks[53] は IP アドレスの変化や物理リンクの障害など，TCP では対応できない通信保障を行なう機構を実現している．しかし Rocks は End-to-end の通信保障を行なうため，ファイアウォール内など直接通信できない計算機との通信の継続を行なうことはできない．

3.5　本章のまとめ

本章では，デバイス移動が起きても利用者がデバイス利用を継続可能とする遠隔デバイスアクセス機構 NextD を設計した．NextD は遠隔デバイスをデバイスファイルとしてアプリケーションに提供する．そのため利用者は既存アプリケーションを書き換えることなく利用できる．サブネットワーク内の計算機上にサービスグループを形成し，アプリケーションはグループ内の計算機に接続されたデバイスにアクセス可能となる．また NextD はサービスグループ内で起きたデバイス移動を検出し，アプリケーションから透過的に遠隔デバイスアクセス先を切り替える．これらは自動的に処理され，利用者に手を煩わせることはない．

NextD を用いることによって，利用者はデバイスを携帯し手近な計算機からサービスを受けることができる．しかしその移動範囲は NextD がサービスグループ形成できるサブネットワーク内に限られる．

次章では，利用者の移動範囲を拡大する支援システムについて述べる．

第4章

デバイス携帯利用支援システム: 移動範囲の拡大

利用者の移動範囲を拡大するために，ファイアウォールによって通信が制限されたサブネットワーク間でデバイス移動に対応する技術を実現する．また利用者の移動先に本システムによる支援が十分にない場合にもデバイスの利用継続を可能にする支援技術を実現する．例えば計算機は存在するがデバイスの接続先（USB ポートや Bluetooth 受信機）がない場合や移動先に無線 LAN アクセスポイントのみ存在する場合などでもデバイスの利用継続を可能にすることが目標である．これらを実現することにより，利用者は遠い部屋や別階などへの広範囲の移動が可能となる．

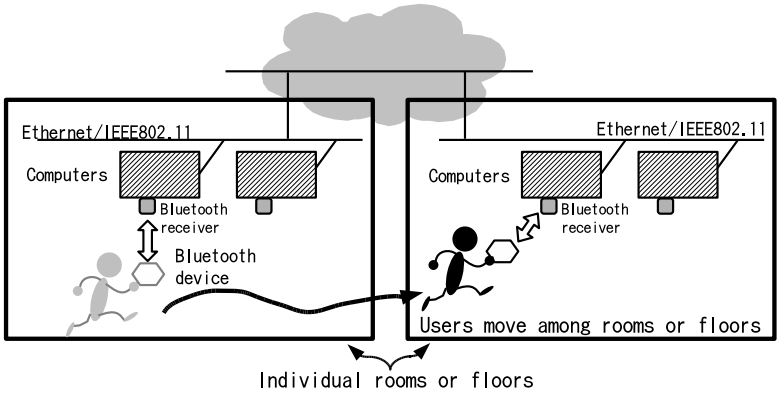


図 4.1: 計算機ネットワーク環境

4.1　背景と目的

前章までにサブネットワーク内でデバイスを携帯した利用者の移動を可能とするサポートシステム NextD について述べた．しかし，時として利用者の移動は広範囲におよび，別々に管理されたサブネットワークを跨ぐ．また比較的狭い範囲であってもネットワーク的に分断された環境も存在する．別々に管理されたサブネットワークは，ファイアウォールが設置され外部との通信が制限されたり，計算機内の利用者情報（例えば UID/GID のようなログイン情報）がサブネットワーク間で異なる場合がある．

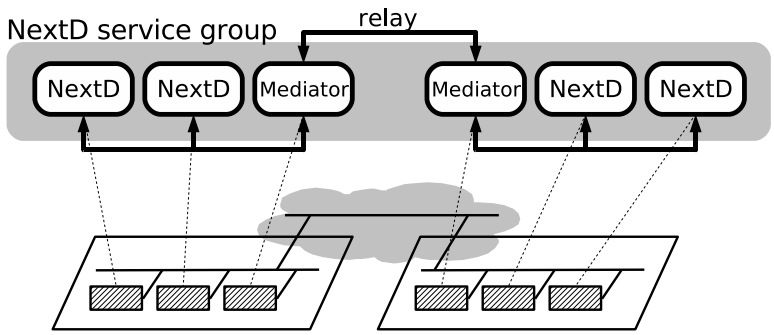
また利用者の移動範囲が広くなると，デバイスのみを持ち歩く利用形態で適合できないような場所への移動も多くなる．提案するデバイス携帯利用法では，移動先にデバイス接続先を備えた計算機が設置されている必要があるが，必ずしもそのような計算機が存在するとは限らない．

本章では，提案するデバイス携帯利用法で利用者の広範囲の移動を可能とする．支援システムについて述べる．

4.2　設計

4.2.1　対象とする計算機ネットワーク構成

対象とする計算機ネットワーク環境は図 1.3 のように，別々のサブネットワークで構成される部屋や階であり，利用者はその間をデバイスを持って移動する．計算機は Ethernet や IEEE 802.11 で接続され，サブネットワーク内では相互に直接



- サブネットワーク内の NextD は互いに直接通信する．
- Mediator はサブネットワーク間のサービスグループを繋げる．
- Mediator はサブネットワーク間の NextD の通信を中継する．

図 4.2: サービスグループ

通信できる．USB ポートや Bluetooth 受信機を持ち，デバイスと接続する．

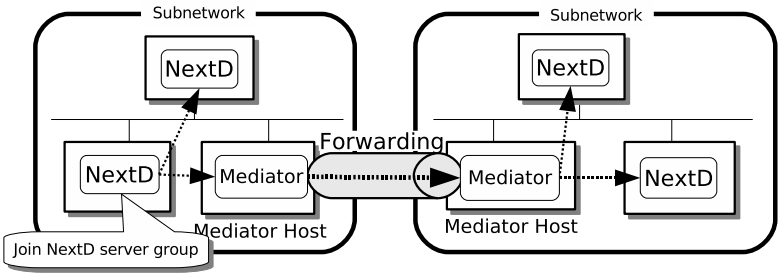
4.2.2　サービスグループの拡大

継続利用のため，利用者の移動範囲にある計算機群に NextD を配置し連携する．NextD は，ブロードキャスト通信で互いを自動的に発見し，図 4.2 のようにサービスグループを形成する．サービスグループ内で，デバイス移動を検出し利用者のデバイス継続利用を可能とする．別々の部屋や階の間は支援システムである Mediator で中継する．

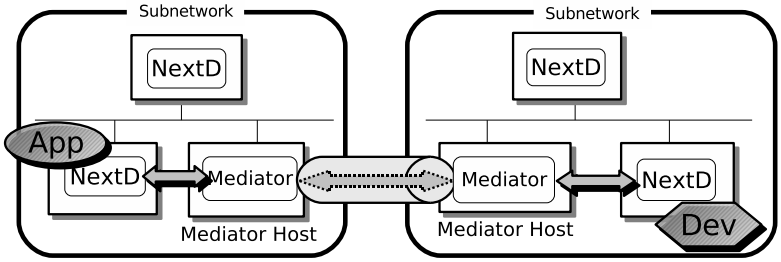
ところで，近年のセキュリティ意識の高まりにより，ファイアウォールが設置され，特定の計算機に特定のポートに対する通信のみを許す環境が一般的である．移動元と移動先の間で通信を行なう場合，そのような制限されたネットワークを考慮する必要がある．Mediator はネットワーク管理者にサブネットワーク間の通信を許可された計算機に配置する．

4.2.3　Mediator

Mediator は，ネットワークを跨いだ NextD サービスを実現する支援システムである．ネットワーク管理者が許可した特定の計算機へ配置され，外部サブネットワークとやりとりをする．あらかじめ別サブネットワークの Mediator とコネクショ



（1）NextD 制御メッセージ転送



（2）NextD デバイスコネクション中継

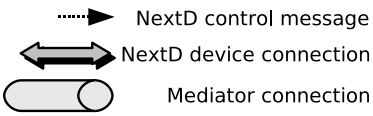


図 4.3: Mediator の機能

ンを張り，自サブネットワークの NextD 制御メッセージやデバイスデータを別サブネットワークへ転送する．こうしてサブネットワークをまたいだ NextD サービスグループが形成される．

以下では，Mediator が NextD に提供する主要機能について述べる．

- NextD 制御メッセージ転送

  Mediator は，NextD 制御メッセージを NextD サービスグループを形成する別サブネットワークの Mediator へ転送する．これによりサブネットワーク間の計算機の存在やデバイス情報を共有することができる．例えば NextD が NextD サービスグループに参加した際には図 4.3（1）のようにメッセージが転送される．

- NextD デバイスコネクション中継

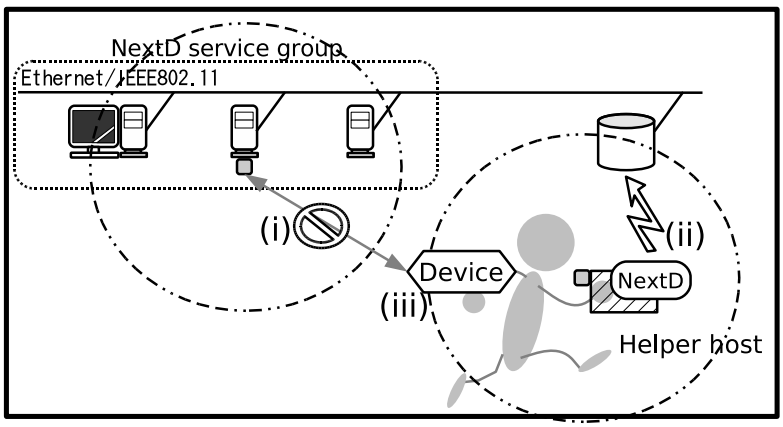
  直接 TCP コネクションを張ることができない別々のサブネットワークの NextD 間の仲立ちをする．Mediator は NextD とコネクションを張ることで別の NextD と通信を行なう．図 4.3（2）は，アプリケーションが別サブネットワークのデバイスにアクセスしているとき（open() した後）のコネクションを示している．

  NextD はファイアウォールのため直接通信できない環境を想定して設計されており，サブネットワーク間の遠隔デバイスアクセスの場合，Mediator を経由して通信を行なう．Mediator を介することで通信遅延が増えることが予想されるが，5.1.2 の実験結果より，そのオーバヘッドは小さく，遠隔デバイスアクセスの要求性能への影響はほとんどないことを確認している．

Mediator は NextD と同様，セッション開始処理（3.1.10 参照）で他の NextD に存在を通知する．その際，Mediator は外部ネットワークからアクセス可能な IP アドレスとポートも通知する．この情報は計算機が別のネットワークに移動した際に利用される．

4.2.4　アクセス制御

NextD は OS で提供される標準的なアクセス制御方法を利用する．UNIX 系 OS では，ユーザにアクセス権限（UID/GID）と，デバイスファイルのアクセスパーミッションにより，アクセス可能なデバイスを制限する．NextD は，遠隔デバイスアクセスを行なうアプリケーションの UID/GID で遠隔計算機上のデバイスファイルにアクセスする．アプリケーションの UID/GID に対するアクセス許可がデバイスファイルになければ，アプリケーションの遠隔デバイスアクセスは失敗する（open 時に EACCES エラーが返る）．この手法は，すべての計算機上でユーザのアクセス権限が共通化されていなければならないが，LDAP[25] などのディレクトリサービスを利用することでより柔軟なアクセス制御を行なうことが可能である．



- （i）デバイスが直接 NextD サービスグループ内の計算機へ接続できない．
- （ii）補助計算機の NextD が無線 LAN 経由でサービスグループへ参加する．
- （iii）デバイスは補助計算機の NextD を経由して，アクセス継続する．

図 4.4: 補助計算機を利用したデバイス移動

4.3　補助計算機を利用したデバイス移動

4.3.1　概要

また利用者の移動範囲が広くなると，デバイスのみを持ち歩く利用形態で適合できないような場所への移動も多くなる．提案するデバイス携帯利用法では，移動先にデバイス接続先を備えた計算機が設置されている必要があるが，必ずしもそのような計算機が存在するとは限らない．

そこで補助計算機を用いたアクセス継続技術を考える．補助計算機を利用者が携帯し，その計算機を経由してアクセス継続を可能とする（図 4.4）．

4.3.2　設計

本章では移動する計算機を**移動ホスト**，その計算機と遠隔デバイスアクセスを行なっている計算機を**相手ホスト**と呼ぶ．

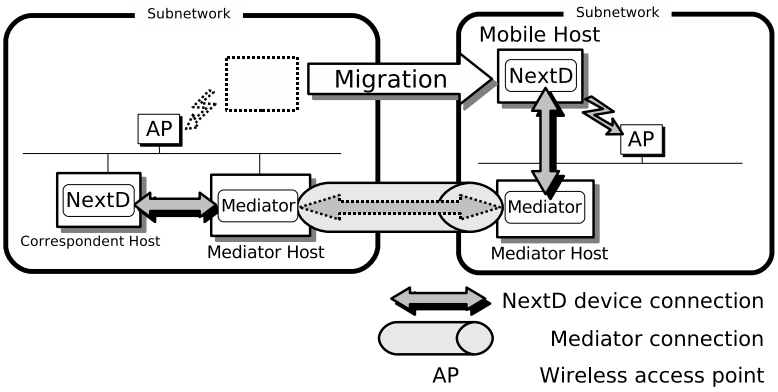


図 4.5: 計算機移動

4.3.3　移動検出

通信が起きていないときでも計算機移動を素早く検出するために NextD にビーコン機能を追加した．メッセージを定期的に交換し，返答がないホストを移動しているとみなす．一定時間過ぎてもそのホストが再接続してこなければ，ホストはいなくなったと判断する．TCP の KEEPALIVE 機能を用いることで接続先ホストの移動を検出することも可能であるが，デフォルト設定でタイムアウトが 2 時間と長い．カーネルオプションで設定を変更できるが設定の変更はホスト全体に影響する．

移動ホストがネットワークへ再接続された場合，NextD は速やかに移動を検出し，相手ホストへ再接続することが望まれる．以下では移動検出方法を移動ホストと相手ホストに分けて説明する．

- 移動ホスト

  移動ホストの NextD は，通信のリンクダウンイベント，ビーコンタイムアウト，TCP/UDP 通信エラーのいずれかにより，計算機が移動開始したと判断する．ビーコンは NextD デバイスコネクションを張った NextD 間で定期的に交換し，ホストの移動を検出する NextD 制御メッセージである．そしてリンクアップイベントにより，計算機が移動完了したと判断し再接続処理を開始する．

- 相手ホスト

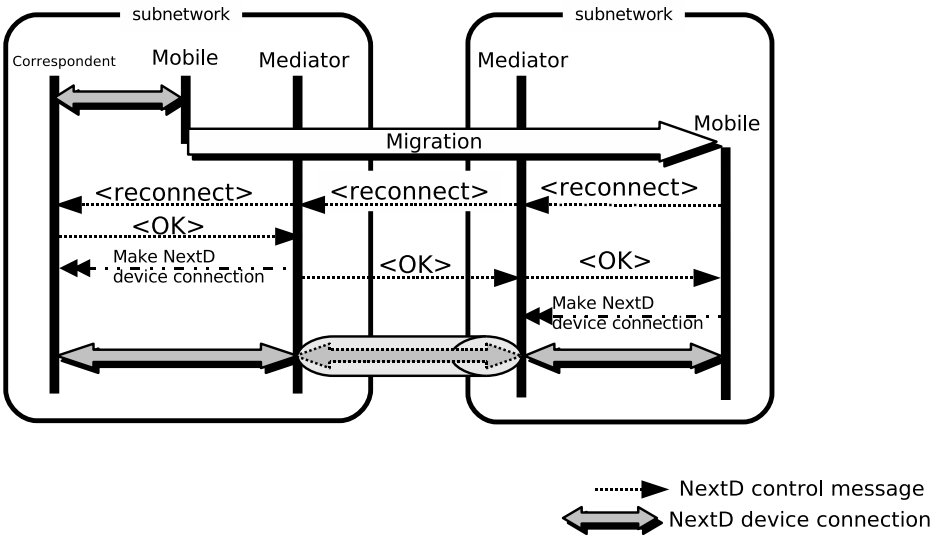


図 4.6: 再接続処理

相手ホストの NextD は，ビーコンタイムアウト，TCP/UDP 通信エラーのいずれかにより，計算機が移動中であると判断し，再接続待ち状態に入る．ここで一定時間（ユーザが設定可能）たっても移動ホストが再接続処理をしなければ，NextD サービスグループから**離脱**したと判断する．

4.3.4　再接続処理

移動ホストの NextD は，リンクアップイベントを受けたらサービス再開処理を行なう（図 4.6）．Mediator を介して相手ホストに再接続を要求し，許可が得られれば NextD デバイスコネクションを張る．コネクションは図 4.6 のように，移動ホスト・Mediator 間と Mediator・相手ホスト間に張られる．Mediator 同士の通信経路は，Mobile IP[32, 33] で経路最適化された場合と同じ経路となる．

4.3.5　移動に伴う消失データの再送

NextD は，通信を再開する前に計算機移動時に消失した可能性のあるデータを再送する．計算機が移動した場合，TCP コネクションが一度切断するため通信中のデータが一部消失する場合がある [53]．通信時にユーザプログラムから渡され

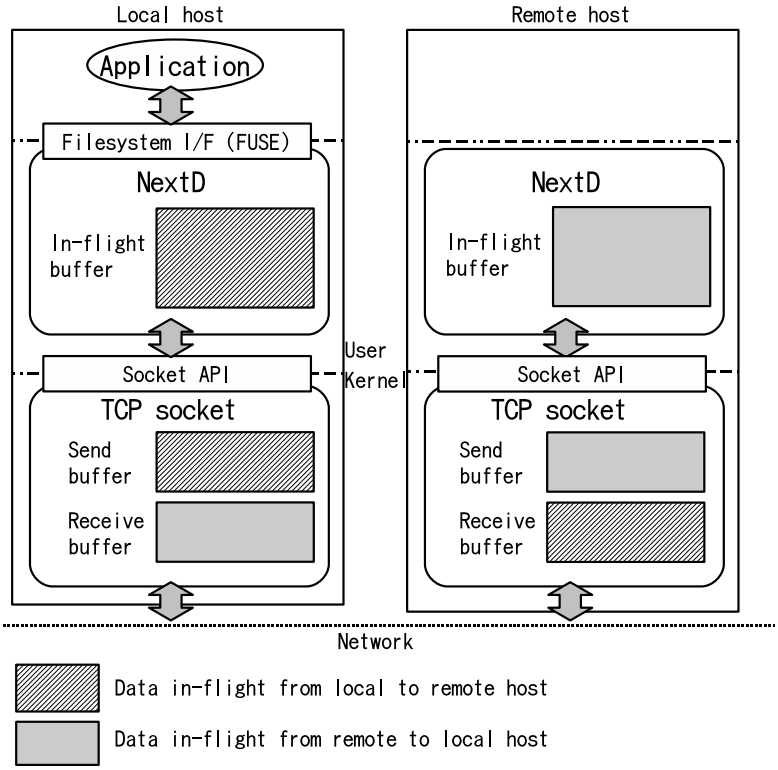


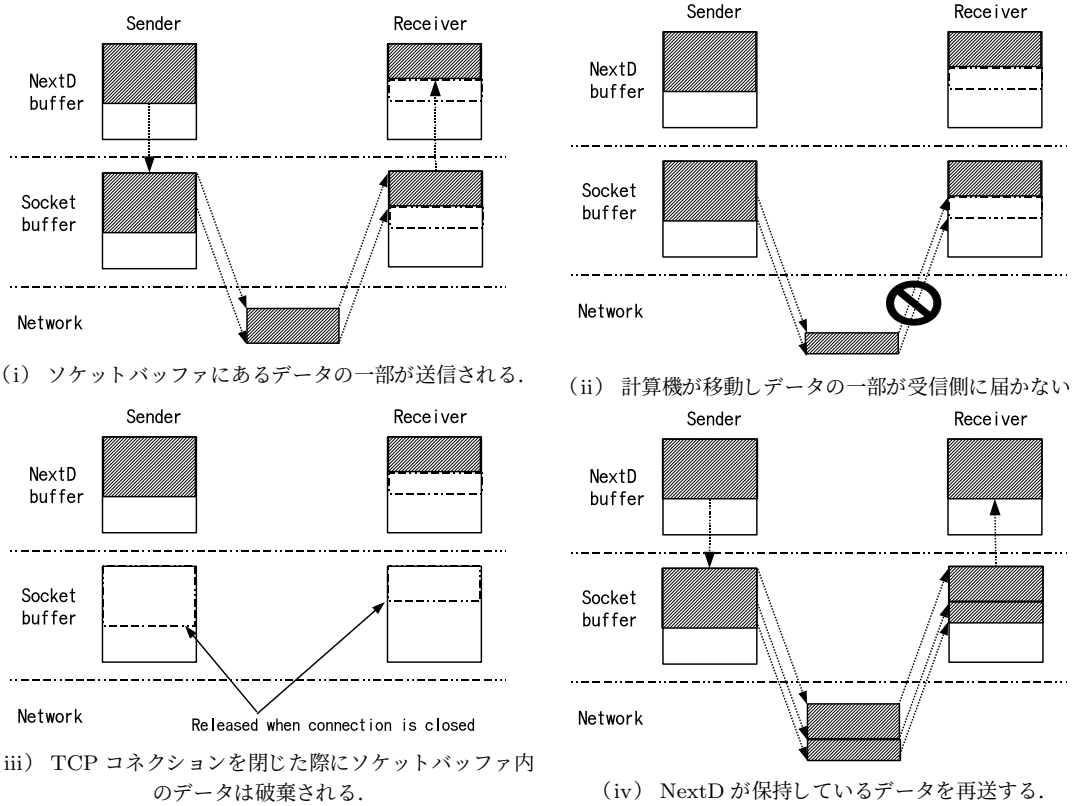
図 4.7: データ保障

るデータは，通信データ保障のため，受信ホストから受信確認が届くまでデータは OS カーネル内のソケットバッファに蓄えられる．TCP の場合，通信データ保障のため受信ホストにデータが届くまでソケットバッファのデータは削除されないが，TCP コネクションが切断したとき[1]にはこのデータは破棄される．そのため NextD でソケットバッファと同じだけのデータをプログラム側で保存し，計算機移動後そのデータを再送する（図 4.7, 4.8）．

4.3.6　Mediator を必要としない計算機移動対応

ファイアウォールによる制限がなく，相手ホストへ NextD 制御メッセージが届き，NextD デバイスコネクションを張ることができる環境では，Mediator によるサポートを必要としない．つまり NextD は，場合によっては Mediator がなくとも計算機移動に対応することができる．処理手順は，再接続処理を相手ホストへ直

[1]データ送信処理のエラー，ビーコンのタイムアウトと Mobile ホストからの再接続処理要求時の明示的なコネクション close 処理の際に起きる．



（i） ソケットバッファにあるデータの一部が送信される．

（ii） 計算機が移動しデータの一部が受信側に届かない．

（iii） TCP コネクションを閉じた際にソケットバッファ内のデータは破棄される．

（iv） NextD が保持しているデータを再送する．

図 4.8: データ保障

接行なう点を除き，前述の方法と同じである．

4.4　従来研究との比較

Mediator は JXTA[21] の Relay peer とメッセージを中継するという点では同等の位置付けになるが，サブネットワーク間の転送にのみ焦点を当てており，マルチホップ転送は行なわない．

計算機移動に対応しアクセス継続を可能とする遠隔デバイスアクセスを実現する方法として，遠隔デバイスアクセス技術に Mobile IP [32, 33] を組み合わせる方法が考えられる．Mobile IP では固定 IP アドレスを持つホームエージェントがデータを中継することで移動ノードのモビリティを実現している．NextD は計算機移動のみならずデバイス移動が起きてもアクセス継続できることを目的としている．デバイス移動をネットワーク層で対応するのは困難である．2 つの機能を統一的に実現するために，NextD は Mobile IP とは異なるアプローチをとった．また NextD は相手ホストと移動ホストが直接通信できるならば，Mediator がない環境でも，計算機の移動時の通信の継続を行なうことができる．

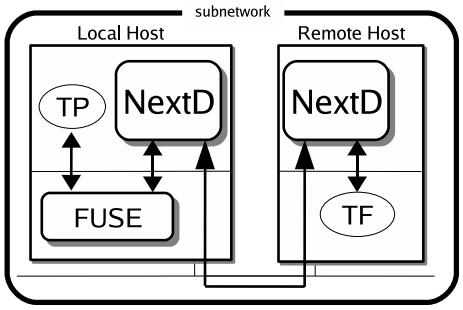
4.5　本章のまとめ

利用者の移動範囲を拡大するために，ファイアウォールによって通信が制限されたサブネットワーク間でデバイス移動に対応する技術を実現した．また利用者の移動先に本システムによる支援が十分にない場合にもデバイスの利用継続を可能にする支援技術を実現した．これにより，利用者は遠い部屋や別階などへの広範囲の移動が可能となった．

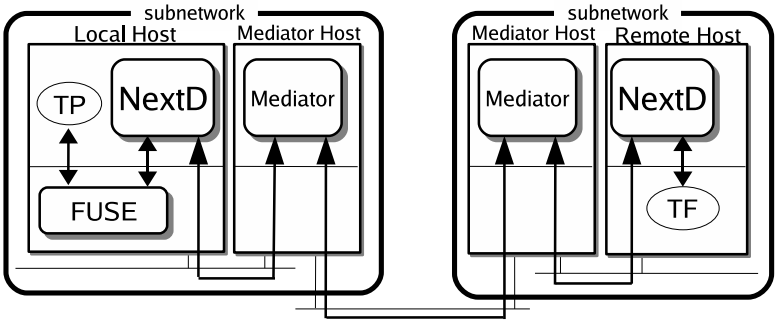
第5章

評価実験と議論

ここでは本システムの評価実験結果について述べる．デバイス特性を基にした評価実験により，本システムが遅延への要求が高いデバイスの性能要件を満たしていることを確認する．またデバイスが移動して接続先の計算機が切り替わる際のサービス中断時間を測定する．また提案するデバイス利用法の有効適用範囲，セキュリティ，応用，将来性の観点で議論する．



（A）サブネットワーク内



（B）サブネットワーク間

TP – Test Program
TF – Target File

図 5.1: 計算機・ソフトウェア構成

5.1　基礎性能評価

デバイスごとに要求される性能を明らかにし，遠隔デバイスへのアクセス遅延を計測する実験の結果について報告する．

5.1.1　デバイス性能要件

本論文では，インタラクションデバイスの代表例として，USB デバイスクラスでいう HID（Human Interface Device）と Audio を取り上げる．

- HID：　マウスであれば，ユーザがマウスを動かしたら，即座にマウスカー

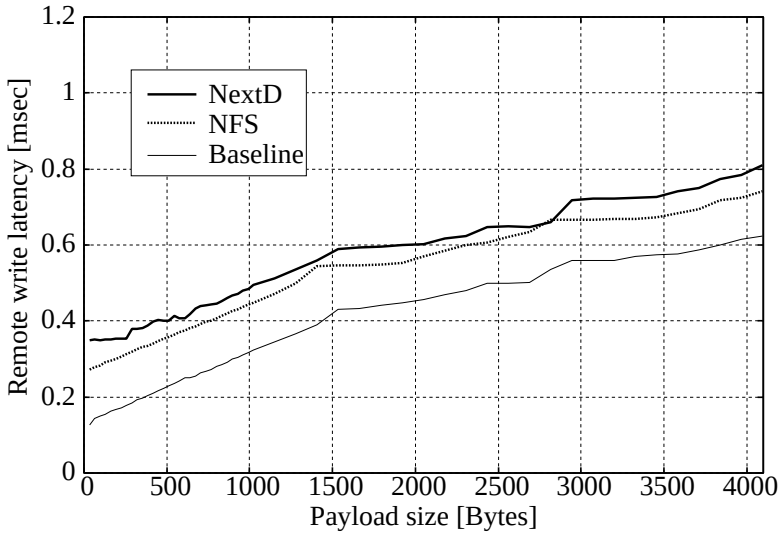


図 5.2: （A-1） データ書込み遅延 （CPU 1 GHz）

ソルが動くという即時フィードバックが求められる．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 [27]. Linux OS の場合，データサイズはマウスで 3 bytes（イベント形式ではデータサイズは 16〜48 bytes）である．つまり 64 bytes 程度のデータを 100 ms 以内に処理できる性能が HID に対する性能要件である．

- Audio： スピーカに対する音楽データ出力であれば，音が途切れないこと求められる．MP3 などの音楽データのビットレートは 56〜160 Kbps が一般的であるが，例えば音楽再生プログラム mpg123 は，160 Kbps のデータを 4 Kbytes ずつ，5 回/秒程度の頻度で出力する．つまり 4 Kbytes のデータを 200 ms（1,000 ms/5 回）以内に処理できることが，Audio に対する性能要件である．

以下の実験は，この要求を満たす性能を得られているかどうか確認することを目的とする．

5.1.2　評価実験（遅延）

実験環境

NextD と Mediator におけるデータ書込み遅延を計測するための実験を，サブネットワーク内とサブネットワーク間の 2 つのネットワーク環境で行なった（図

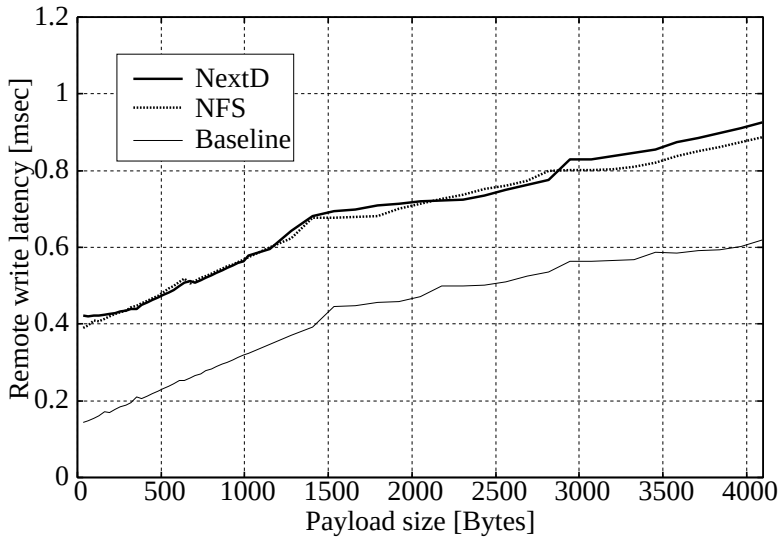


図 5.3: （A-2） データ書込み遅延 （CPU 600 MHz）

5.1）．実験（A）は Fast Ethernet で接続した 2 台の計算機上で NextD を動かし，一方の計算機から他方の計算機のデバイスファイルに書込みを行なう．この実験では計算機の CPU が 1 GHz の場合（A-1）と，600 MHz の場合（A-2）の実験を行なった．実験（B）は Gigabit Ethernet で相互接続された 2 台の計算機（ともに CPU 2.6 GHz）上で Mediator を動かし，実験（A-1）の NextD を中継する構成である．

比較として NFS（Network File System）における書込み遅延，および TCP 通信の片道通信遅延（Baseline）も計測した．

実験における計算機間の通信は，NFS を含めすべて TCP が用いられている．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 [28] を無効にした．

結果と考察

図 5.2, 5.3 の横軸はペイロードサイズ（64 bytes から 4 Kbytes），縦軸は遠隔データ書込み遅延である[1]．この実験では遅延時間のほとんどが通信遅延であるため，2 つの環境での Baseline の結果はほぼ同じとなっている．

サブネットワーク内の実験（A-1）では，ペイロードサイズ 64 bytes で書込み遅

[1]NextD は，遠隔計算機での write() 処理の返り値を得るためローカル計算機で待ち合わせしており，データ書込み処理は 2 回の通信を含む．

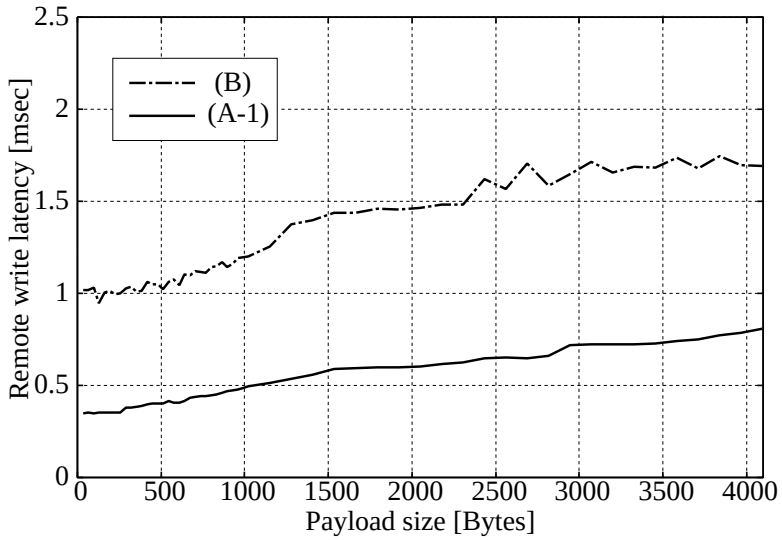


図 5.4: （B） Mediator を経由したデータ書込み遅延

延が 0.35 ms と HID の許容遅延とされる 100 ms に比べ十分小さい．また 4 Kbytes で 0.80 ms と，Audio に求めれる 200 ms に比べ十分小さい（図 5.2）．NextD は図 5.1 のように書込みデータがカーネルから一度ユーザに渡されるが，その処理時間は小さい．ペイロードサイズ 64 bytes と 4 Kbytes でも NextD の処理時間は 0.12 ms で残りは通信遅延である．また NextD は NFS と比べ遜色ない性能を得ている．

サブネットワーク間の実験（A-2）では，ペイロードサイズ 64 bytes で書込み遅延は 0.42 ms，4 Kbytes で 0.92 ms である（図 5.3）．CPU 速度が落ちた分実験（A-1）に比べ遅延時間が増えているがいずれもデバイスの要求性能に比べると十分小さい．

実験（B）の結果を図 5.4 に示す．比較として実験（A-1）の NextD の測定値も載せる．データ書込み遅延はペイロードサイズ 64 bytes で 1.0 ms，4 Kbytes で 1.7 ms であり，Mediator を介してもデバイスの性能要件を満たしている．

以上の結果から NextD と Mediator が HID や Audio など遅延への要求が高いデバイスの性能要件を満たしていることがわかった．

NextD 処理分析

NextD の遠隔デバイスアクセス処理は,（i）FUSE 処理,（ii）NextD 自身の処理,（iii）ネットワーク処理の 3 つに分けられる．

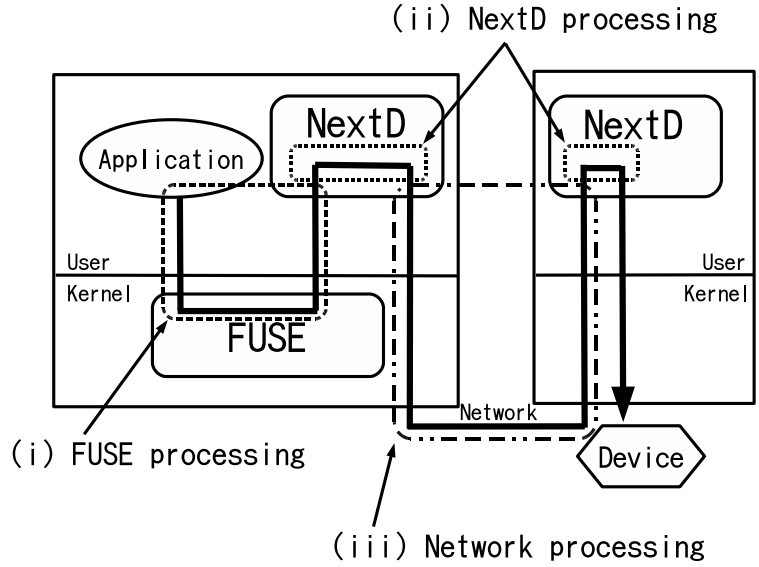


図 5.5: NextD 処理

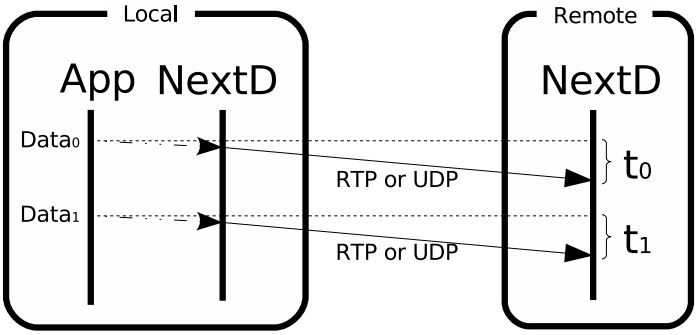


図 5.6: 書き込み時刻のずれ

5.1.3　Voiceデバイス

実験環境

NextD 間の通信を RTP/UDP と UDP にした場合の書き込み時刻のずれを計測した（図 5.1（B））．書き込み時刻のずれとは，図 5.6 のようにアプリケーションで write() した時刻と，そのデータを遠隔の NextD が write() した時刻の差とする．

実験では NIST Net エミュレータ [10] を用いて，通信遅延にゆらぎを発生させた．ローカル計算機と遠隔計算機間の遅延を 30 ms，遅延の分散幅を 10 ms に設定した．

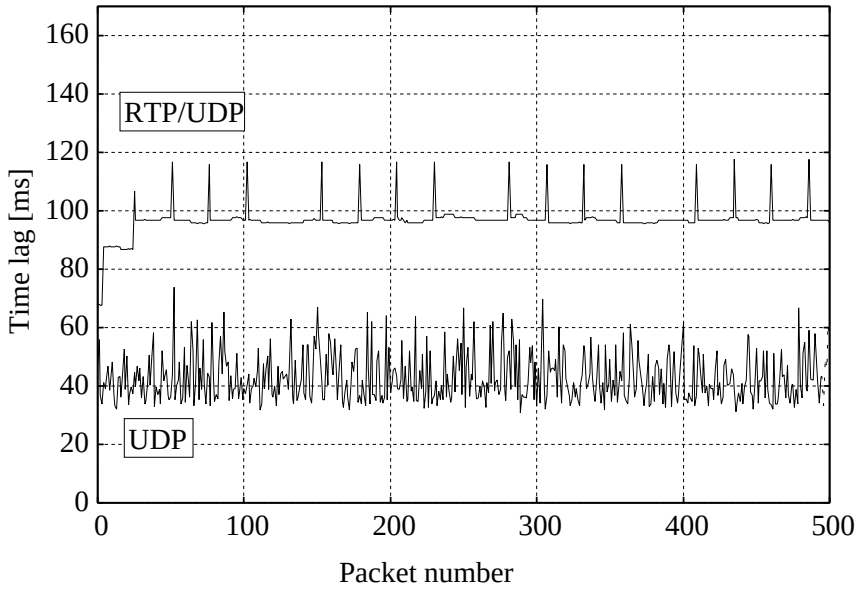


図 5.7: 書き込み時刻のずれの推移

結果と考察

実験結果を図 5.7, 5.8 に示す．UDP の書き込み時刻のずれは 20〜60 ms に分布し，遅延のゆらぎの影響がそのまま出ている．このゆらぎの分布には NIST Net がエミュレートするゆらぎ [10] がそのまま現れている．それに対し RTP の書き込み時刻のずれは 95 ms 付近に集中し安定している．定期的に発生しているスパイクは RTCP[43] によるものである．

以上の結果より，RTP/UDP を用いることによって，Voice デバイスに必要な定間隔のデータ書込みを得られていることがわかった．

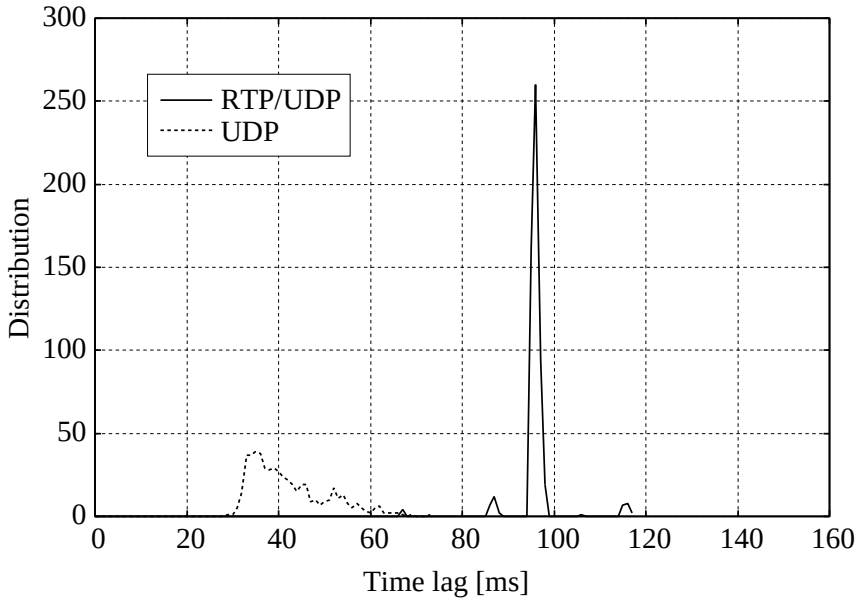


図 5.8: 書き込み時刻のずれの分布

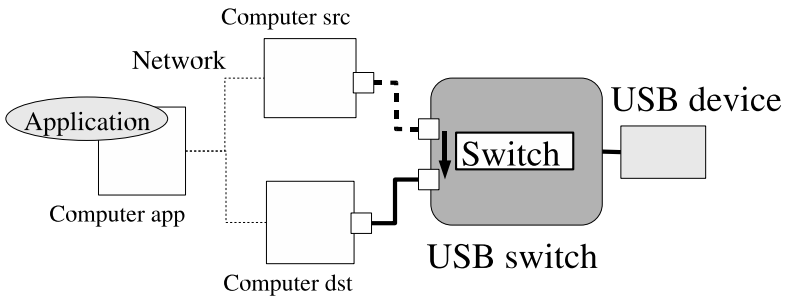


図 5.9: 実験環境

5.2　デバイス移動時間計測実験

ここでは実際にデバイス移動を起こす実験環境を構築し，実現したアクセス継続機構が正しく機能しているか確認する実験を行なった．

5.2.1　USB

実験内容

実験には 3 台の計算機と 1 台の USB スピーカを用いた（図 5.9）．app は CPU 1.0 GHz のノート PC，src は CPU 1.0 GHz のノート PC，dst は CPU 1.4 GHz の

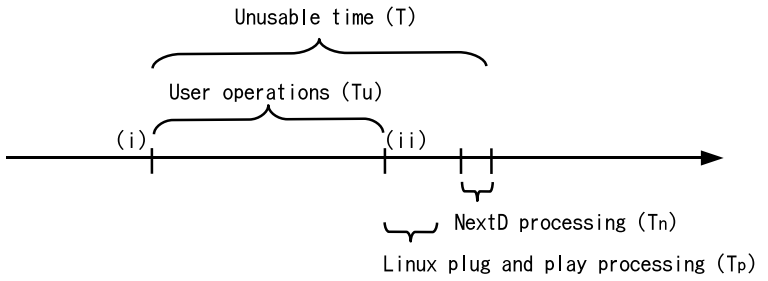


図 5.10: USB デバイス移動とデバイス利用不可期間

SMP 計算機である．app と src は 11 Mbps 無線接続，dst は Fast Ethernet で接続されている．2 台の計算機に接続した USB 切替器に USB デバイスを接続しデバイス移動を模擬した．USB 切替器を用いるデバイス移動では切り離しから接続までの時間は無視できるほど小さい．

アプリケーションには mpg123 を利用した．mpg123 はデバイスファイル（/dev/dsp）に音声データ（MP3）を固定サイズずつ出力することで音楽データを再生する．

実験処理手順は以下のようになっている．

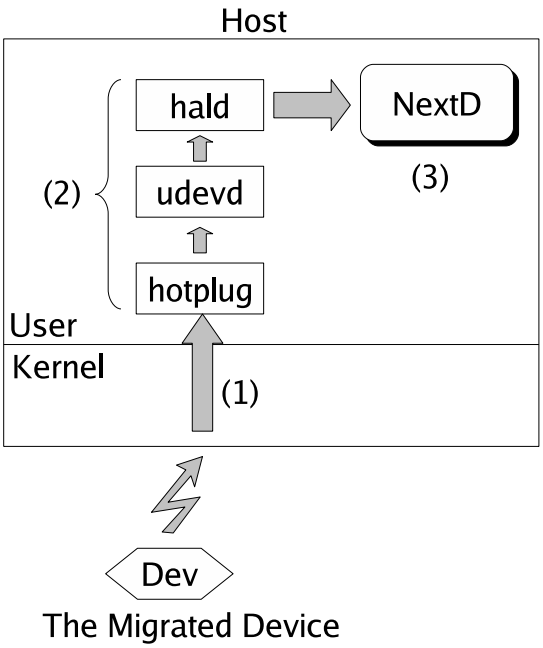
- mpg123 が音声データを NextD_{app} がインポートしたデバイスファイルを通して NextD_{src} へ出力している．
- デバイス切り離しが起きる．（USB スイッチで USB スピーカの接続先を src から別の計算機に変える．）
- デバイスが NextD_{dst} へ接続される．（USB スイッチで USB スピーカの接続先を dst にする．）

この実験により実現した機能が正しく動作しているかどうかを確認した．またデバイス移動時の NextD の切替え処理時間を評価した．

実験結果と考察

デバイス状態の復帰機能を有効にした実験では音声が正しく再生されたが，デバイス状態の復帰機能を止めた実験では，デバイス移動後に雑音が USB スピーカから流れてきた．これは mpg123 が初期化時に行なうデバイスの設定[2]をしていな

[2]ioctl() の種類でいうと SNDCTL_DSP_SETFMT, SNDCTL_DSP_STEREO, SNDCTL_DSP_SPEED である．



（1） Linux カーネルのデバイスハンドリング処理
（2） Linux ユーザプログラムのデバイスハンドリング処理
（3） NextD の処理

図 5.11: デバイス接続イベントが NextD に通知されるまでの処理順序

いためである．

デバイス移動時に音声再生が中断される時間を計測することで，NextD の切替え処理時間の評価実験を行なった．デバイス切り離し（図 3.7 における Device disconnect）から音声が再び USB スピーカから流れ出すまでに約 5 秒の時間を要した．（図 5.9 のシステムを用いているため，デバイス切り離しと再接続間の時間は含まれない．）この内訳は，Linux がデバイス装着イベントを検知して，hotplug（Linux の着脱イベント時に最初に呼び出されるユーザレベルプログラム）に通知するまでに約 0.7 秒（図 5.11 の（1）），そこから hotplug や udev などのユーザレベルプログラムの処理を行なったのち NextD に着脱イベントが通知されるまでに約 3.0 秒（図 5.11 の（2）），NextD での処理に約 1.0 秒（図 5.11 の（3））である．NextD に着脱イベントを即座に通知するように hotplug を書き換えることで，約 3.0 秒の処理時間短縮が図れる．

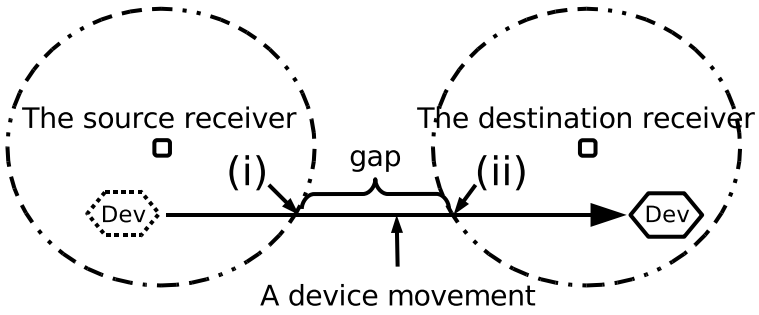


図 5.12: Bluetooth デバイス移動と電波到達範囲

5.2.2　Bluetooth

時間計算

ここでは，Bluetooth におけるデバイス移動時間を定量的に評価する．またパケットロスが起きたり，IEEE 802.11b/g（WLAN）と通信帯域を共有するような環境における影響について考察する．

デバイス移動の間は，デバイスを利用することができない（図 5.12）．その期間 T は以下のように算出される．

$$T = \begin{cases} T_g + T_{d'} + T_n & (T_g \geq T_d) \\ T_d + T_{d'} + T_n & (T_g < T_d) \end{cases} \tag{5.1}$$

- T_g：隙間時間．

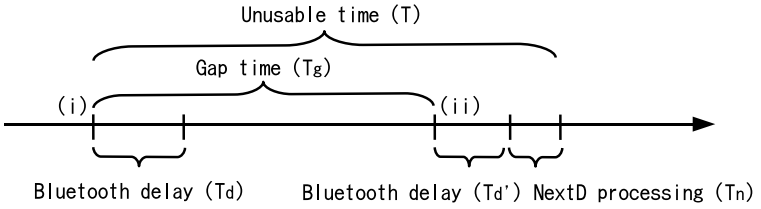
  利用者が Bluetooth デバイスを持って，Bluetooh 受信機の電波到達範囲外を移動する時間．

- T_d：Bluetooth 遅延．

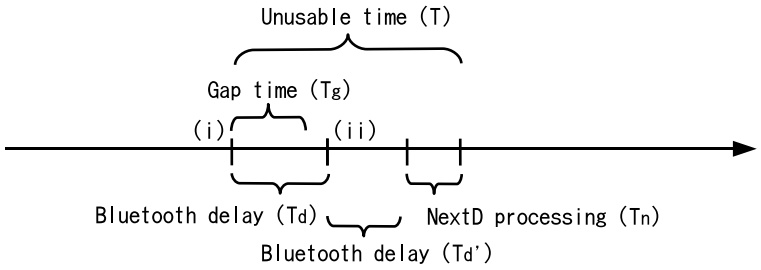
  Bluetooth が電波到達範囲から外れてから，NextD がそれを検出するまでの時間．この遅延は Bluetooth はリンクロスと判断するまでに，一定の猶予期間を設けているためである．これは一時的な通信障害をリンクロスと誤判断しないためである．

- $T_{d'}$：T_d と同様である．

  しかし $T_{d'}$ は T_d に比べて小さくなる．NextD はポーリングを行ない，移動中の Bluetooth デバイスにすぐに接続しようとするためである．



（a）電波到達範囲が離れている場合



（b）電波到達範囲が近接・重なっている場合

図 5.13: デバイス利用不可期間

- T_n : NextD 処理時間.

数式 5.1 の上の式は，移動元と移動先の計算機の Bluetooth 受信機の電波到達範囲がお互いに重なっていない場合における，デバイス利用不可期間 T である．T_g は Bluetooth 受信機を持った計算機の設置間隔によって決まる．計算機間の距離が広くなるほど，T_g は大きくなる．T_d と $T_{d'}$ は Bluetooth の設定によって変わる．Bluetooth では，デバイスと受信機がお互いにリンクロスしたかどうかを判定するまでの時間を *Link Supervision Timeout* で指定できる．この値は 0.625 ms から 40.9 sec. の間に設定可能で，初期値は 20 sec. である．本研究ではこの値を 2 sec. に設定している．T_n は 5.2.1 で述べたように，およそ 1 sec. である．結果として，T は $3 \sim 5 + T_g$ sec. となる．

数式 5.1 の下の式は，移動元と移動先の計算機の Bluetooth 受信機の電波到達範囲がお互いに非常に近いか重なっている場合における，デバイス利用不可期間 T である．つまり T_g が T_d より小さくなる場合である．これにより T_g は式から消える．一方，T_d と $T_{d'}$ は重なり合うことができない．これは，デバイスがいなくなったことを検出した移動元 NextD が送出する *disconnect* メッセージを受け取ってから，移動先 NextD は移動中のデバイスをポーリングするためである．結果として，

T は 3 ～ 5 sec. となる．

パケットロスは，デバイス移動における *connect/disconnect* メッセージ配送の遅延を引き起こす．NextD は制御メッセージの転送において 3-way ハンドシェイクによる通信保障を実現している．その再送間隔は 1 sec. である．そのため制御メッセージ転送でパケットロスが起きるとデバイス利用不可期間 T が 1 sec. だけ長くなる．

WLAN と Bluetooth は電波干渉を引き起こす．本研究で用いている Bluetooth v1.2 では，適合的周波数ホッピング機能が実装され干渉を軽減している．しかし実際には，WLAN で通信が増えると Bluetooth 通信，例えば VoIP の音声品質に影響が見られる．しかしながら，その影響によって T が増えることは観測されていない．干渉によって起きる遅延は T より小さいためだと考えられる．

運用に関する議論

導き出した結果から，利用者に使いやすいシステム運用について議論する．

デバイス移動においてデバイス利用不可期間は数秒におよぶ．長すぎるデバイス利用不可期間は利用者に受け入れられない．

USB では，利用者は明示的にデバイスを取り外し接続するため，例えば音楽再生までの数秒の遅延は利用者にとって受け入れられないものではない．しかし Bluetooth では，切断時間は利用者にとって敏感な問題となる．それゆえデバイス利用不可期間 T は小さいことが望ましい．計算機を互いに近くに設置することで T_g を 0 に近づける．また *Link Supervision Timeout* は環境に合わせて設定する必要がある．現在は 2 sec. に設定しているが，デバイス移動の素早い検出と Bluetooth リンクの誤切断とのトレードオフもとにより小さい値に設定することができる．付け加えて言うと，デバイス移動に対応するメカニズムによって誤切断された Bluetooth リンクの再接続に用いることができる．デバイスが移動元計算機にデバイス移動したとシステムで認識し，再接続を行なうためである．

5.3　議論

5.3.1　有効適応範囲

ここでは本システムを有効に利用できる，またはできない環境について議論する．

移動範囲

本システムは建物内など比較的狭い範囲での利用に適している．Bluetooth デバイスを利用する場合，で述べたとおりある一定の間隔で計算機を設置することでデバイス利用不可時間を小さくし，実用に耐えるサービスを提供することができる．USB デバイスの場合では，移動中にデバイス利用ができないものの，移動先でデバイス利用を再開することができる．利用者は自身の机での作業で利用するデバイスをそのまま持ち運ぶことができるため，移動開始までの動作がスムースである．

逆に本システムは，建物の外に出るような長距離・長時間の移動には不向きである．NextD を動作させる計算機を広範囲に設置することは現実的でない．USB の場合，例えば大学から自宅まで持ち運ぶことを考えると，移動中にデバイスが全く利用できなければ，利用者は満足できないだろう．またデバイスは簡単な機能しかもたないため，長時間の移動で生じる多様な要求には応えることができない．

ネットワーク

本システムのサービス品質はネットワーク負荷に大きく左右される．そのため，ネットワーク状態が大きく変化するような環境での利用は，利用者を満足させることができないかもしれない．例えば，ネットワーク状態が予測できないインターネットがデバイス接続先計算機とアプリケーションが動作する計算機の間に存在すると，デバイスの安定した利用は難しい．

デバイス

本システムでは，永続データなどの状態を持たないデバイスの利用を想定している．音楽データ，ストリーミング，音声通話やマウスやキーボードなどがそれにあたる．逆にストレージデバイスなどはあまり向いていない．本システムはデバ

イス移動後に再接続することは必ずしも保障されていないため，ストレージデバイスにおけるデータの整合性を維持することができない．再接続するかどうかは利用者の行動にかかっているためである．しかし，本システムの想定移動範囲，つまり狭い範囲の移動で，利用者が元の位置に戻ってくるような状況ではストレージデバイスを持ち歩く必要はないと考える．ストレージデバイスは利用者とのインタラクションを必要とせず，移動元の計算機に接続したままでも用が足りるためである．

5.3.2　セキュリティ

ここではセキュリティに関して議論する．分散システム―原理とパラダイム（タネンバウム他著）[2] に従い，アクセス制御，セキュアチャネル，セキュリティ（鍵）管理の点について，NextDが提供する機能について述べる．

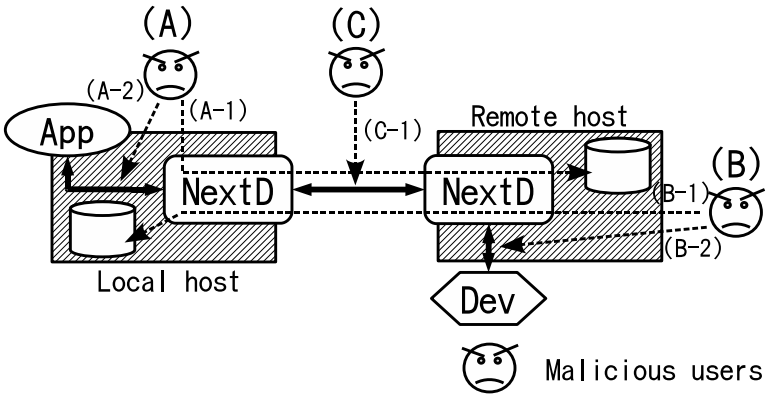
またいくつかの妨害者を想定し，その攻撃からいかにしてNextDが管理する資源を守るか述べる．

アクセス制御

NextDは ACL（Access Control List）を用いたアクセス制御機構を持つ．アプリケーション（ログインユーザ）とデバイスファイルの UID/GID のマトリクスで，利用者はアクセス可能なデバイスを指定できる．例えば，自分の管理する計算機の特定のデバイスへは遠隔計算機からはアクセスできないようにする，特定の UID を持つアプリケーションからのアクセスだけは許す，といった設定が可能である．アプリケーションは，アクセスを許可されていないデバイスのデバイスファイルを見ることはできない．

セキュリティ（鍵）管理

デバイス携帯利用法では，計算機や利用者は既知で管理されていることを前提としている．そのため鍵をあらかじめ NextD 間に配布することができる．



（A-1）ローカルの妨害者が遠隔ストレージを攻撃

（A-2）ローカルの妨害者がローカルのアプリケーション入出力処理を攻撃

（B-1）遠隔の妨害者がローカルデータを攻撃

（B-2）遠隔の妨害者が遠隔計算機のデバイス入出力処理を攻撃

（C-1）妨害者がデバイスデータを攻撃

図 5.14: 攻撃者

セキュアチャネル

現在の NextD はセキュアチャネルをサポートしていない．しかし NextD 間でセキュアチャネルを確立することは難しくない．NextD はユーザレベルプログラムで実現され，通常のソケットインタフェースで通信を行なう．そのため，例えば Secure Sockets Layer[17] や Transport Layer Security[13] など暗号化ソケットを導入することは，前述の通り，認証や暗号化に必要な鍵を持つことが可能なため容易である．

守るべき資源と妨害者

脅威にさらされる恐れのある資源は以下の 3 つである．

- ローカルストレージ
- 遠隔ストレージ
- デバイスデータ

それぞれの資源に対し，以下の妨害者が考えられる．

- （A）ローカル計算機のユーザ
- （B）遠隔計算機のユーザ
- （D）それ以外の計算機のユーザ

考えうる攻撃の種類

そのうち NextD が考慮すべき攻撃を図 5.14 に示す．それぞれについて攻撃内容と NextD における対処法を述べる．

（A-1）は，ローカル計算機のユーザが NextD を通して，遠隔計算機のストレージを不正に読み書きする攻撃である．（B-1）は，遠隔計算機のユーザが NextD を通して，ローカル計算機のストレージを不正に読み書きする攻撃である．これらは本質的に同じであるので，対処法はまとめて書く．NextD の遠隔デバイスアクセス機構は，ローカルファイルシステムのディレクトリの一部を他の計算機にエクスポートし遠隔のアプリケーションに見せる．エクスポートするディレクトリはユーザレベルプログラムである NextD が読み出すため，NextD がアクセス可能なファイルはすべてエクスポート対象となる．そのため，不正なファイルパス（“../” を含むパスなど）を指定することにより許可されていないファイルへアクセスを許すかもしれない．しかし，NextD は必要なデバイスファイルのみしかエクスポートしないように，ユーザが指定することができる．さらに chroot や SELinux[42] などで NextD のファイルアクセス範囲を制限することでより強固な安全性を確保できる．

（A-2）は，妨害者がローカルのアプリケーションとローカルの NextD とのやりとりを不正に横取・改変する攻撃である．（A-2）では，妨害者は（A-2-1）アプリケーション・FUSE 間，（A-2-2）FUSE・NextD 間において攻撃機会を持つ．（A-2-1）の攻撃は，NextD は通常のファイルシステム API（open(), close(), read(), write() など）でデバイスファイルにアクセスため，この攻撃はローカル OS の安全性と同じである．（A-2-2）は FUSE の安全性によって保障される．

（B-2）は，（A-2-2）と同様 NextD のデバイスファイル入出力を横取・改変する攻撃であるが，（A-2-2）とは別の攻撃方法が考えられる．NextD は HAL から渡されるデバイスファイルのパスを信用するため，妨害者が HAL を改変し，妨害者がデータを横取りするデバイスファイルを間に挟むことができる．これは，計算機の

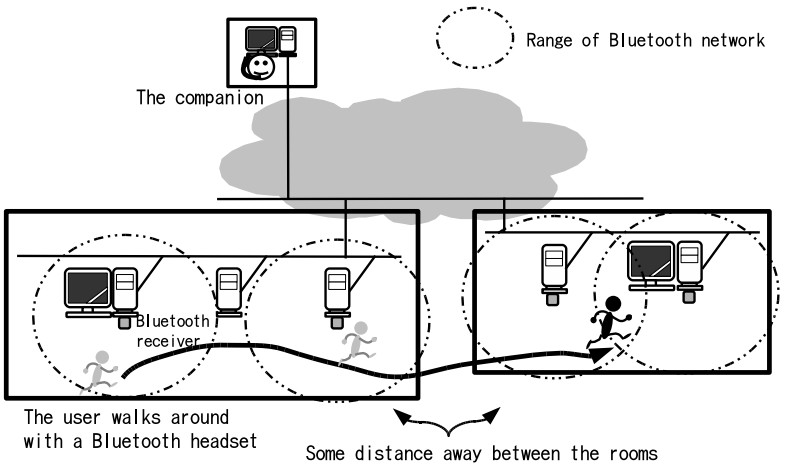


図 5.15: 無線デバイスを携帯した利用者の移動

管理者ならば可能である．この攻撃を回避することは難しい．例えばアプリケーションとデバイス間でデータを暗号化することが考えられるが，デバイスに暗号・復号機能が必要となる．この問題には，信用できる管理者の計算機のみを利用するといった，回避策をとることを推奨することで対応することになる．

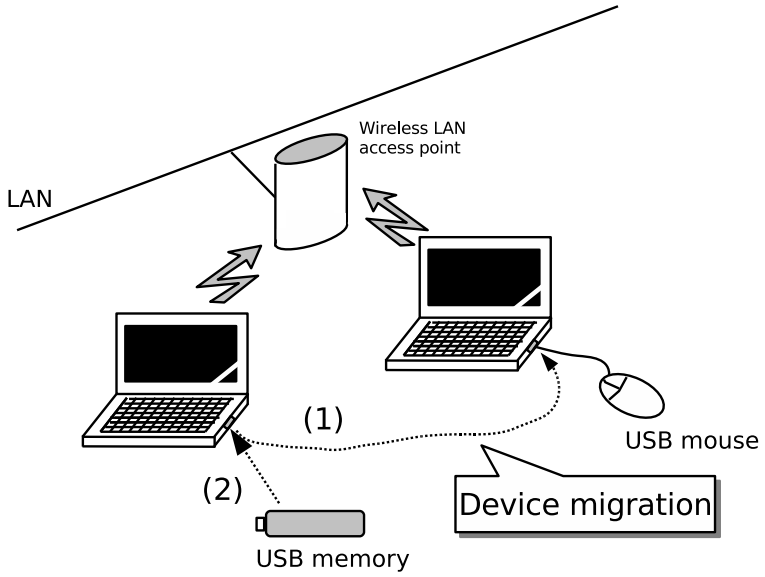
（C-1）は，第三者が NextD 間の通信を傍受し，デバイスデータを横取・改変する．これは NextD 間でセキュアチャネルを形成することで回避可能である．

5.3.3　NextD の応用

応用例 1：無線デバイスを携帯した利用者の移動

提案するデバイス持ち歩き形態は例えば以下のような状況で用いる．

オフィスの自分の席で，計算機と無線接続する Bluetooth[5] ヘッドセットでどこか別の場所の同僚と会話をしているユーザがいる（図 5.15）．そのとき席から一時的に離れなければならない用事ができたとしても，会話を続けたいとする．しかし移動先が遠く，Bluetooth 受信機の電波到達範囲から外れる場合は，会話を続けることができない．本システムのサポートがあるならば，この場合でも利用者は会話を継続できる．



(1) USB マウスを取り外し別の計算機に付け直す.

(2) 空いた接続ポートに USB メモリを接続する.

図 5.16: 接続ポートの一時的利用

応用例 2：接続ポートの一時的利用

有線接続ではデバイス用ポートが足りなくなることがあるため，別の計算機のポートを借りてデバイスを利用することが考えられる（図 5.16）．特にモバイル計算機ではポート数が少ないため，この問題は起きやすい．ポートが既に埋まっているモバイル計算機でストレージデバイス等の扱うデータが大きいデバイスを利用したいときに，モバイル計算機に接続されているデバイスを一時的に別の計算機に再接続して，ストレージデバイスをモバイル計算機に直接接続するとデータ転送を早く終えることができる．

5.3.4　デバイス携帯利用法の将来性

現在はマウス・ヘッドフォン・ヘッドセットなどの無線デバイスしか存在しない．しかし，サインは VGA[3]（図 5.17）や USB NIVO[4]（図 5.18）などの USB 接続の外部ディスプレイ出力ユニットが製品として売り出されておりこれが無線 USB と

[3] http://plusd.itmedia.co.jp/pcupdate/articles/0406/22/news100.html
[4] http://www.newnhamresearch.com/

図 5.17: サインは VGA

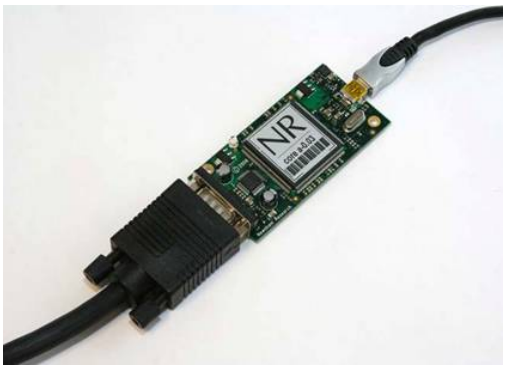


図 5.18: USB NIVO

組み合わされ，携帯可能なディスプレイとなることは十分考えられる．そうなると提案する利用モデルは，今後より有用なデバイス利用方法となると考える．

また NextD は write() でデバイスアクセス可能なフレームバッファデバイス[5] を用いるようなアプリケーションにのみ対応している．より一般的なアプリケーションに対応するためには，mmap(2) などメモリに直接アクセスするアプリケーションの出力データを取得する機構を NextD に実現しなければならない．またデータ転送量を抑えるために，USB NIVO の仮想ドライバが持つようなデータ圧縮技術を実現する必要がある．

[5] http://sourceforge.net/projects/linux-fbdev/

5.4　本章のまとめ

本章では，NextD の評価実験について報告した．デバイス特性を基にした評価実験により，本システムが遅延への要求が高いデバイスの性能要件を満たしていることを確認した．遠隔デバイスに対するデータ書込み処理時間は，データサイズが 4KB の場合，同一サブネットワーク内では約 0.8ms、サブネットワークを超えた場合（Meidator による中継が 2 回の入る）で 1.8ms であった．これは人間が直接使うデバイスとしては，十分実用的な性能である．またデバイスが移動して接続先の計算機が切り替わる際のサービス中断時間は 3~5 秒であり、実用に耐えるといえる．またデバイス携帯利用法の有効適用範囲，セキュリティ，応用，将来性に関して議論した．

第6章

結論

本論文では，計算機に接続し利用者が直接扱うデバイスの携帯利用法およびその支援技術について議論し，以下の結果を得た．

提案したデバイス携帯利用法は，提案する利用形態は，デバイスを利用者が持ち歩き，手近にある計算機を経由して，必要なデータへのアクセスや遠隔地との相手と会話というものである．この利用法は高い安全性を持ち，広範囲の移動でも継続利用でき，持ち歩くデバイスは小型軽量でシンプルである，という特徴を持つ．

また本研究は，提案する利用形態で利用者が広範囲の移動をしてもデバイスの継続利用を可能とするシステムを実現した．利用者の移動に伴いデバイスの接続先計算機が切替わる，デバイス移動が起きても利用者がデバイス利用を継続可能とする．

第 3 章では，デバイス移動が起きても利用者がデバイス利用を継続可能とするための機能を遠隔デバイスアクセス機構上に設計した．本機構は遠隔デバイスをデバイスファイルとしてアプリケーションに提供する．そのため利用者は既存アプリケーションを書き換えることなく利用できる．本機構はサブネットワーク内の計算機上にサービスグループを形成し，アプリケーションはグループ内の計算機に接続されたデバイスにアクセス可能となる．また本機構はサービスグループ内で起きたデバイス移動を検出し，アプリケーションから透過的に遠隔デバイスアクセス先を切り替える．これらは自動的に処理され，利用者に手を煩わせることはない．なお開発の容易化と拡張性を高めるため，機能の大部分はユーザレベルプログラムとして実現した．

第 4 章では，前章で実現したシステムを拡張し，利用者の移動範囲を拡大する支援システムを設計した．一つはファイアウォールによって通信が制限されたサブネットワーク間で，遠隔デバイスアクセスを可能とする支援システムである．これにより利用者は遠い部屋や別階などへの広範囲の移動が可能となる．もう一方は，利用者の移動先に本機構がなく無線アクセスポイントのみ存在する環境において，利用者のデバイス利用継続のための機能である．そのような環境でのアクセス継続は，デバイスの接続したモバイル計算機を利用者が携帯することで可能となる．アプリケーションからデバイスへのアクセスは，無線アクセスポイントに接続したモバイル計算機を経由して行なわれる．本機構には，利用者の移動にともないモバイル計算機の IP アドレスが切り替わっても，遠隔デバイスアクセスを継続することができる機能を実現した．

第 5 章で，本システムの評価実験結果について述べた．デバイス特性を基にした評価実験により，本システムが遅延への要求が高いデバイスの性能要件を満たしていることを確認した．遠隔デバイスに対するデータ書込み処理時間は，データサイズが 4KB の場合，同一サブネットワーク内では約 0.8ms、サブネットワークを超えた場合（Meidator による中継が 2 回の入る）で 1.8ms であった．これは人間が直接使うデバイスとしては，十分実用的な性能である．またデバイスが移動して接続先の計算機が切り替わる際のサービス中断時間は 3~5 秒であり、実用に耐えるといえる．またデバイス携帯利用法の有効適用範囲，セキュリティ，応用，将来性に関して議論した．

本研究の成果は，新たなデバイス利用形態を提案し，重要な情報を扱い複雑な作業に従事する環境において，利用者に計算機利用法の代替手段をもたらしたこと，支援システムを実現し評価実験によりその有効性を実証したことにある．

謝辞

本論文は，様々な方々のお力添えのもとに完成いたしました．

本論文の主査であり，主任指導教官でもありました丸山勝巳教授におきましては，多大なるご支援とご協力を頂きました．研究ミーティングを始め，さまざまな面で，いつも真摯にご指導くださいました．幾度となくくじけそうになったときも，叱咤激励とともに我慢強くご支援くださりました．筆者のわがままを聞き入れて頂き，研究テーマを自由に決めることを許してくださいました．感謝しても，し足りないご支援を頂きました．本当にありがとうございました．

また論文審査員の先生方にも多くのご助力を頂きました．

橋爪宏達教授におきましては，ハードウェアに関する深い知識を元にしたご指摘とともに，研究者としての心得をご教授いただきました．本研究を，違った角度からの視点で考察・評価して頂き，研究を深めることができました．

多田好克教授におきましては，システムソフトウェアに関する深い知識を元にした幅広いご指摘を頂きました．微に入り細を穿つご指摘により，本論文をより質の高いのものにすることができました．

佐藤一郎教授におきましては，ハードなスケジュールの中，時間を割いて様々なご指摘を頂きました．また第一線を走る研究者としての心得をご教授いただきました．

児玉和也助教授におきましては，研究に関していつも鋭いご指摘を頂きました．児玉先生がいたからこそ，この論文が完成したといっても過言ではありません．

また研究ミーティングに参加いただいた日高宗一郎助手におきましても，ご支援を頂きました．いつも的確なご指摘をしてくださり，筆者の至らない所を正してくださいました．

この他，学部・修士学生時代の指導教官であり，国立情報学研究所という研究の場を紹介して頂きました中山泰一助教授にもお礼申しあげます．

また電気通信大学時代の諸先輩方や国立情報学研究所の同僚の皆様にも，公私の両面でご支援をいただきました．

最後に，博士後期課程への進学を快く許し，いつも温かく見守ってくださいました，両親にも深く感謝しております．博士号取得できたのは，ひとえに両親が温かく見守ってくださったが故です．本当に感謝しております．

ここに論文を完成させることができ，博士号を取得できたのは，皆様のおかげです．本当にありがとうございました．

参考文献

[1] Ajay, B. and Badrinath, B.R.: I-TCP: Indirect TCP for Mobile Hosts, Proc. *International Conference on Distributed Computing Systems (ICDCS '95)*, pp.136–143, (1995).

[2] アンドリュー・S・タネンバウム, マールティン・ファン・スティーン: 分散システム—原理とパラダイム, ピアソンエデュケーション (2003).

[3] Ballesteros, F. J., Castro, E. M., Muzquiz, G. G. and Algara, K. L.: A New Network Abstraction for Mobile and Ubiquitous Computing Environments in the Plan B Operating System, Proc. *Sixth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'04)*, pp.112–121 (2004).

[4] Bellotti, V. and Bly, S.: Walking Away from the Desktop Computer: Distributed Collaboration and Mobility in a Product Design Team, Proc. *the 1996 ACM Conference on Computer Supported Cooperative Work (CSCW '96)*, pp.209–218 (1996).

[5] Bluetooth SIG. `http://www.bluetooth.com/`

[6] BlueZ. `http://www.bluez.org/`

[7] Brian Cornell, P. A. D. and Bustamante, F. E.: Wayback: A User-level Versioning File System for Linux, Proc. *the USENIX 2004 Annual Technical Conference, FREENIX Track*, pp.19–28 (2004).

[8] ブレント・ミラー, チャトシック・ビスディキアン: Bluetooth テクノロジーへの招待, ピアソン・エデュケーション (2002).

[9] FUSE: Filesystem in Userspace. `http://fuse.sourceforge.net/`

[10] Carson, M. and Santay, D.: NIST Net: a Linux-based network emulation tool, SIGCOMM Comput. Commun. Rev., Vol.33, No.3, 111–126 (2003).

[11] D-BUS. `http://www.freedesktop.org/wiki/Software/dbus`

[12] Dasgupta, P., Itzkovitz, A. and Karamcheti, V.: Active Files: A Mechanism for Integrating Legacy Applications into Distributed Systems, Proc. *the 20th International Conference on Distributed Computing Systems (ICDCS 2000)*, pp.680–690 (2000).

[13] Dierks, T. and Allen, C.: The TLS Protocol Version 1.0, *RFC 2246* (1999).

[14] HAL - Hardware Abstraction Layer. `http://www.freedesktop.org/wiki/Software/hal`

[15] Hirofuchi, T., Kawai, E., Fujikawa, K. and Sunahara, H.: USB/IP - A Peripheral Bus Extension for Device Sharing over IP Network, Proc. *the USENIX 2005 Annual Technical Conference, FREENIX Track*, pp.47–60 (2005).

[16] Johanson, B. and Fox, A.: The Event Heap: A Coordination Infrastructure for Interactive Workspaces, Proc. *the Fourth IEEE Workshop on Mobile Computing Systems and Applications*, pp.83–93 (2002).

[17] Freier, A.O., Karlton, P. and Kocher, P.C.: The SSL Protocol Version 3.0, *Transport Layer Security Working Group, Internet Draft* (1996).

[18] GMail FS. `http://richard.jones.name/google-hacks/gmail-filesystem/gmail-filesystem.html`

[19] GNOME Resources. `http://www.gnome.org/`

[20] Goglin, B. and Prylli, L.: Transparent Remote File Access through a Shared Library Client, Proc. *the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA '04)*, pp.1131–1137 (2004).

[21] Gong, L.: JXTA: A Network Programming Environment, *IEEE Internet Computing*, Vol.5, No.3, pp.88–95 (2001).

[22] Kristoffersen, S. and Ljungberg, F.: "Making Place" to Make IT Work: Empirical Explorations of HCI for Mobile CSCW, Proc. the International ACM SIGGROUP Conference on Supporting Group Work (GROUP '99), pp.276–285 (1999).

[23] Hirofuchi, T., Kawai, E., Fujikawa, K. and Sunahara, H.: USB/IP - A Peripheral Bus Extension for Device Sharing over IP Network, Proc. *the USENIX 2005 Annual Technical Conference, FREENIX Track*, pp.47–60 (2005).

[24] Husemann, D., Narayanaswa, C. and Nidd, M.: Personal Mobile Hub, Proc. *8th IEEE International Symposium on Wearable Computers (ISWC'04)*, pp.85–91 (2004).

[25] Lightweight Directory Access Protocol (v3), RFC 2251 (1997).

[26] Maltz, D.A. and Bhagwat, P.: MSOCKS: An Architecture for Transport Layer Mobility, Proc. *17th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '98)*, pp.1037–1045 (1998).

[27] Miller, R.B.: Response time in man-computer conversational transactions, Proc. AFIPS Fall Joint Computer Conference, Vol.33, pp.267–277 (1968).

[28] Nagle, J.:Congestion Control in IP/TCP Internetworks, RFC 896 (1984).

[29] Noble, B.D., Satyanarayanan, M., Narayanan, D., Tilton, J.E., Flinn, J. and Walker, K.R.: Agile application-aware adaptation for mobility, *SIGOPS Operating Systems Review*, Vol.31, No.5, pp.276–287 (1997).

[30] oRTP. `http://freshmeat.net/projects/ortp/`

[31] Osman, S., Subhraveti, D., Su, G. and Nieh, J.: The design and implementation of Zap: a system for migrating computing environments, Proc. *SIGOPS Oper. Syst. Rev.*, Vol.36, No.SI, pp.361–376 (2002).

[32] Perkins, C.: IP Mobility Support, RFC 2002 (1996).

[33] Perkins, C. E. and Johnson, D. B.: Mobility Support in IPv6, Proc. *ACM International Conference On Mobile Computing And Networking (MobiCom'96)*, pp.27–37 (1996).

[34] Pike, R., Presotto, D., Dorward, S., Flandrena, B., Thompson, K., Trickey, H. and Winterbottom, P.: Plan 9 from Bell Labs, *Computing Systems*, Vol.8 (1995).

[35] Raghunathan, V., Pering, T., Want, R., Nguyen, A. and Jensen, P.: Experience with a low power wireless mobile computing platform, Proc. *the 2004 international symposium on Low power electronics and design (ISLPED '04)*, pp.363–368 (2004).

[36] Richardson, T., Stafford-Fraser, Q., Wood, K. R. and Hopper, A.: Virtual Network Computing, *IEEE Internet Computing*, Vol.2 (1998).

[37] Rifkin, A. P., Forbes, M. P., Hamilton, R. L., Sabrio, M., Shah, S. and Yueh, K.: RFS Architectural Overview, Proc. *USENIX Conference*, pp.248–259 (1989).

[38] Salz, J., Snoeren, A. and Balakrishnan, H.: TESLA: A Transparent, Extensible Session-Layer Architecture for End-to-End Network Services, Proc. *4th Usenix Symposium on Internet Technologies and Systems*, pp.211–224 (2003).

[39] Sandberg, R., Goldberg, D. Kleiman, S. Walsh, D. and Lyon, B.: "Design and Implementation of the Sun Network Filesystem," Proc. *Summer 1985 USENIX Conference*, pp.119–130 (1985).

[40] Satran, J., Meth, K., Sapuntzakis, C., Chadalapaka, M. and Zeidner, E.: Internet Small Computer Systems Interface (iSCSI), RFC3720 (2004).

[41] Scheifler, R. W. and Gettys, J.: The X Window System, *ACM Transactions on Graphics*, Vol.5 (1986).

[42] Security-Enhanced Linux. `http://www.nsa.gov/selinux/`

[43] Schulzrinne, H., Casner, S., Frederick, R. and Jacobson, V.: RTP: A Transport Protocol for Real-Time Applications, RFC 1889 (1996).

[44] Snoeren, A. C. and Balakrishnan, H.: An end-to-end approach to host mobility, Proc. *the 6th annual international conference on Mobile computing and networking (MobiCom '00)*, pp.155–166 (2000).

[45] Takashio, K., Soeda, G. and Tokuda, H.: A Mobile Agent Framework for Follow-Me Applications in Ubiquitous Computing Environment, Proc. *the 21st International Conference on Distributed Computing Systems Workshops (ICDCSW '01)*, pp.202–207 (2001).

[46] 高杉耕一, 中村元紀, 田中聡, 久保田稔, 小柳恵一: 動的な環境に適応するシームレスなサービス連続技術, 情報処理学会論文誌, Vol.46, No.2, pp.608–622 (2005).

[47] Thain, D. and Livny, M.: Bypass: A Tool for Building Split Execution Systems, Proc. *the Ninth IEEE Symposium on High Performance Distributed Computing (HPDC9)*, pp.79–85 (2001).

[48] Tsegaye, M. and Foss, R.: A comparison of the Linux and Windows device driver architectures, *ACM SIGOPS Operating Systems Review*, Vol.38 (2004).

[49] 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. *the 4th international conference on Ubiquitous Computing*, pp.194–209 (2002).

[50] 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).

[51] Rubini, A. and Corbet, J.: Linuxデバイスドライバ 第2版, オライリー・ジャパン (2002).

[52] Welch, B. B. and Ousterhout, J. K.: Pseudo Devices: User-Level Extensions to the Sprite File System, Proc. *the 1988 Summer USENIX Conference*, pp.37–49 (1988).

[53] Zandy, V. C. and Miller, B. P.: Reliable Network Connections, Proc. *the 8th annual international conference on Mobile computing and networking*, pp.95–106 (2002).

関連論文の印刷公表の方法及び時期

学術雑誌論文

1. 尾崎 亮太, 日高 宗一郎, 児玉 和也, 丸山 勝巳: 計算機移動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 (和文論文誌 B) シームレス通信サービスのためのネットワーキング技術特集号, Vol.J89-B, No.8, pp.1357-1366 (2006).
2. Ryota Ozaki, Soichiro Hidaka, Kazuya Kodama and Katsumi Maruyama, "Design and Implementation of Remote Device Access Facility to Support Device Migration," IEICE Transaction on Information & Systems, (採録決定).

国際会議（査読付）

1. Ozaki, R., Hidaka, S., Kodama, K. and Maruyama, K.: Accessing Remote Devices Using Conventional Interfaces in Mobile Computing Environment, Proc. *8th International Workshop on Mobility in Databases and Distributed Systems*, pp.1063–1067 (2005).

その他

学会口頭発表

1. 尾崎 亮太, 日高 宗一郎, 児玉 和也, 丸山 勝巳: 単機能ユーザインタフェースデバイスの携帯利用法の提案とその支援システムの設計と実現, 電子情報通信学会 第 4 回 次世代ネットワークソフトウェア研究会, p.11 (2006).
2. 尾崎 亮太, 日高 宗一郎, 児玉 和也, 丸山 勝巳: デバイス移動に対応するデバイスアクセス継続技術, 第 9 回 プログラミングおよび応用のシステムに関するワークショップ (SPA 2006) (2006).
3. 尾崎亮太, 日高宗一郎, 児玉和也, 丸山勝巳: 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, 情報処理学会 第 36 回 モバイルコンピューティングとユビキタス通信研究発表会, MBL-36, pp.251–220 (2006).
4. 尾崎 亮太, 日高 宗一郎, 児玉 和也, 丸山 勝巳: 遠隔デバイス管理機構の設計, 電子情報通信学会 2005 年総合大会, D-3-4, p.24 (2005).
5. 尾崎 亮太, 日高 宗一郎, 児玉 和也, 丸山 勝巳: ネットワークの動的変化に対応する遠隔デバイス管理機構, 情報処理学会 第 32 回 モバイルコンピューティングとユビキタス通信研究発表会, MBL-32-8, pp.53–59 (2005).
6. 尾崎 亮太, 丸山 勝巳, 日高 宗一郎, 児玉 和也: 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, 並列／分散／協調処理に関するサマー・ワークショップ (SWoPP 2004), 2004-OS-97-11, pp.81–88 (2004).

ポスタ発表

1. Ozaki, R., Hidaka, S., Kodama, K. and Maruyama, K.: Continuous Access to Remote Devices in the Presence of Device Migration, Proc. *The International Conference on Dependable Systems and Networks 2005 (DSN-2005)*, pp.86–87 (2005).