平成23年度 修士論文

モバイルデバイス間における
省電力を考慮したファイル同期方式

電気通信大学 大学院情報システム学研究科

情報システム基盤学専攻

1053016　高見澤 拓郎

指導教員　鶴岡 行雄　客員准教授
多田 好克　教授
近藤 正章　准教授

提出日　　平成24年1月26日

目 次

図 目 次

表 目 次

第 1 章

はじめに

現在スマートフォンやノート PC 等のモバイルデバイスが普及しており，1 人のユーザが複数台所有することも増えている．ここでは複数のモバイルデバイスを持ち運ぶことを考える．複数台のモバイルデバイスを利用する場合，デバイス間でのデータの共有が必要となる．

図 1.1 はデータの共有の例を示している．ユーザが電車の中でスマートフォンを利用中に，ネットワークから動画をダウンロード (更新) したとする．すると，移動中にスマートフォンからノート PC へダウンロードした動画が転送される．これによって，ユーザが喫茶店でノート PC を起動したとき，ダウンロードした動画をすぐに視聴することができる．また，Web はスマートフォンのアプリケーションの中でも利用頻度が高い [3]．スマートフォンで閲覧したウェブページをノート PC 側で閲覧したい場合，キャッシュデータを共有することで，ネットワークに接続していない状態でもウェブページを閲覧することができ，データがローカルにあるため，すぐにウェブページを閲覧することが可能となる．さらに，メールの本文データやアドレス帳を共有することで，スマートフォンで受信したメールデータをノート PC で閲覧できる．

モバイルデバイス間でデータを共有する方法としては，データを持つデバイスを他のデバイスが参照する方法 (データの一元化) と，それぞれのデバイスがデータを持ち，更新を通知しあう方法 (データの同期) がある．一元化は同期と比べて管理の手間が少なくなるが，データを持たないデバイスはデータのアクセスに時

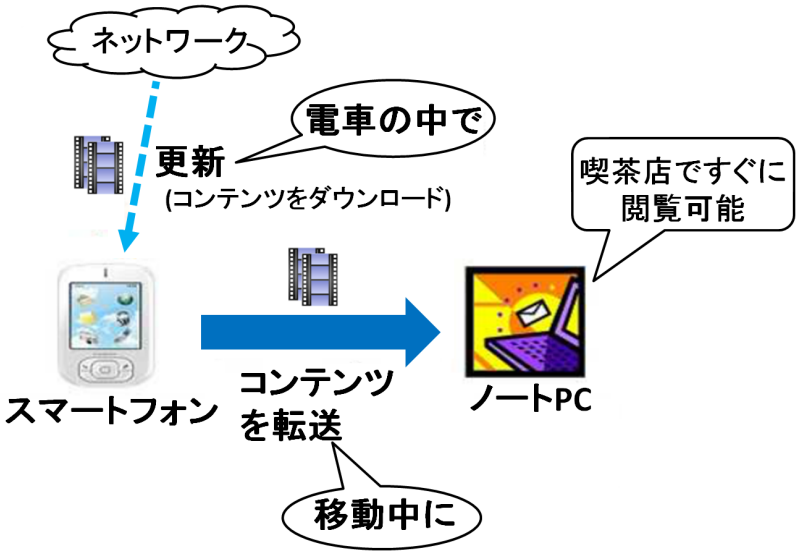

図 1.1: ファイル同期の例

間がかかる．また，もしデータを持つデバイスを自宅に置き忘れてしまった場合，データを参照することができない．ここでは各デバイスを独立に使うことを考え，データの同期を前提とする．

デバイス間の同期方法としてインターネット上のサーバを経由して同期を行う方法と，持ち運ぶデバイス間で直接同期を行う方法がある．サーバを経由する場合，まず，同期元のデバイスとサーバ間で同期を行う．その後にサーバと同期先デバイス間で同期を行う．サーバは常に起動中であることから，同期元のデバイスと同期先のデバイスが同時に起動中である必要がなく，デバイス間の距離が離れていたとしても同期を行うことができる．しかし，インターネットに接続しなければ最新の情報を得ることができない．また，インターネットを経由する分，転送に時間がかかる．サーバの代わりに NAS のようなデバイスを新たに持ち運ぶことで，インターネットに接続する必要がなくなるが，持ち運びに手間がかかるといった問題がある．本研究では持ち運ぶ 2 つのデバイス間において PAN や LAN 経由で直接同期を行う．これによってインターネット上のサーバに接続しなくても同期を行うことができるため，常に最新の情報が取得できる．また，インターネット上のサー

バを経由する場合と比べて転送にかかる時間も少なくなる．

ここでは，モバイルデバイスとしてスマートフォンとノート PC からなる 2 つのデバイス間で直接同期を行うことを考える．ノート PC はバッテリー持続時間が短いため，未使用時は停止しているものとする．一方，スマートフォンは通話待ちのため常時起動しているものとする．ユーザがノート PC を利用する時に電源を投入した場合，ノート PC が停止中にスマートフォン上に発生した更新は，起動時にスマートフォンからノート PC へまとめて転送される．このため，同期が完了しユーザがデータにアクセスできるまでには待ち時間が生じる．このような待ち時間はサービスの利便性を損ねるため問題となる．待ち時間の長さはノート PC が停止中にスマートフォン上で発生した更新量に依存する．

同期に伴う待ち時間を軽減するためには，ノート PC を事前に起動し同期を行う方法が考えられる．この場合，ノート PC の起動と停止に伴う消費電力がオーバヘッドとなる．もしスマートフォン側で更新が発生する度に頻繁に転送を行った場合，ユーザの待ち時間は軽減される半面，起動と停止に伴う消費電力量の蓄積が問題となる．このように，待ち時間と消費電力はトレードオフの関係にある．

同期ソフトウェアとして Dropbox[4]，Windows Live Mesh[5]，iCloud[6] などがある．しかし，これらについては停止中の間に発生した更新は起動後にすべてまとめて転送されるため，更新量が多い場合には待ち時間が問題となる．

通信の消費電力を削減するための関連研究として以下の 2 つが挙げられる．Niranjan らは通信を伴うスマートフォン上のアプリケーションの省電力化を提案している [1]．通常，スマートフォンの持つ無線 LAN デバイスは未使用時には低消費電力状態であり，更新があると高消費電力状態へ切り替わる．転送中は高消費電力状態である．転送終了後，再び高消費電力状態から低消費電力状態へと切り変わる．このように，転送を行うためには電力状態の切り替えが必要であり，更新がある度に切り替えを行うことで，電力状態の切り替えのオーバヘッドが問題となる．Niranjan らの研究では，更新をまとめ，無線 LAN デバイスの電力状態を切り替え

る回数を減らすことによって消費電力を削減している．Dogar らは転送中における消費電力を削減するシステム Catnap を設計，実装している [2]．これは，転送中にデバイスの無線 LAN やデバイス本体を低消費電力状態 (802.11 PSM[14] や S3 状態 [7]) にすることで，消費電力を削減している．削減方法としては，基地局と AP(アクセスポイント）間の転送速度と AP とデバイス間の転送速度の差を利用している．これらの 2 つの研究では起動中であるデバイスの通信の省電力化に着目しているが，停止デバイスに対する同期について十分に考慮がされていなかった．

本研究ではモバイルデバイス間のファイル同期において停止中のノート PC にユーザが電源を投入してからの待ち時間を一定の範囲内に抑えつつ，消費電力を削減する同期方法を提案する．提案法では，まずユーザが許容できる待ち時間すなわち，ユーザが電源を投入してから同期が完了するまでにかかる時間を定める．この待ち時間を超えないことを条件として，転送に伴う起動と停止の回数を減らすことで消費電力を削減する．また，ノート PC の停止中の電源状態 (休止状態とスリープ状態) の違いに着目すると，単位時間あたりの更新量の違いによって，各電源状態において同期に必要な消費電力は異なる．これらの各電源状態に基づいた 2 つのモードのことを休止モードとスリープモードとし，2 つのモードを適切に切り替えることにより，更なる省電力化を図る．また，提案法を実現するシステムを設計し，プロトタイプシステムを実装した．そして，実装したシステムを動作させ，正しく動くことを確認した．また，シミュレーションによる評価を行い，休止モードとスリープモードと提案法の 3 つの場合において同期にかかる消費電力量を比較した．結果として，消費電力量が削減できることを確認した．

本論文の構成を以下に示す．第 2 章では一般的なモバイルデバイス間での同期として，2 つのデバイスが共に起動中である場合について述べる．第 3 章ではモバイルデバイスとして，常時起動中のスマートフォンと未使用時は停止しているノート PC 間の同期に限定した場合について述べる．第 4 章では本研究で扱う電源状態について説明し，第 5 章では本研究で扱う通信技術について説明する．第 6 章では

提案法について述べる．第 7 章では提案法を実現するためのシステムの設計について述べる．第 8 章では提案法の実装について述べる．第 9 章では提案法のアルゴリズムの評価した結果を示す．第 10 章では本研究で扱わない技術や，さらなる省電力化の方法について考察する．第 11 章では本研究の結論について述べる．

第 2 章

モバイルデバイス間のファイル同期

本章では，一般的なモバイルデバイス間のファイル同期についてその概要と処理の流れについて述べる．2 つのモバイルデバイス (以下，デバイスと呼ぶ) をそれぞれデバイス A，デバイス B とする．同期とは，それぞれのデバイスがデータを持ち，更新を通知しあうことである．

2.1 モバイルデバイスとファイル同期処理

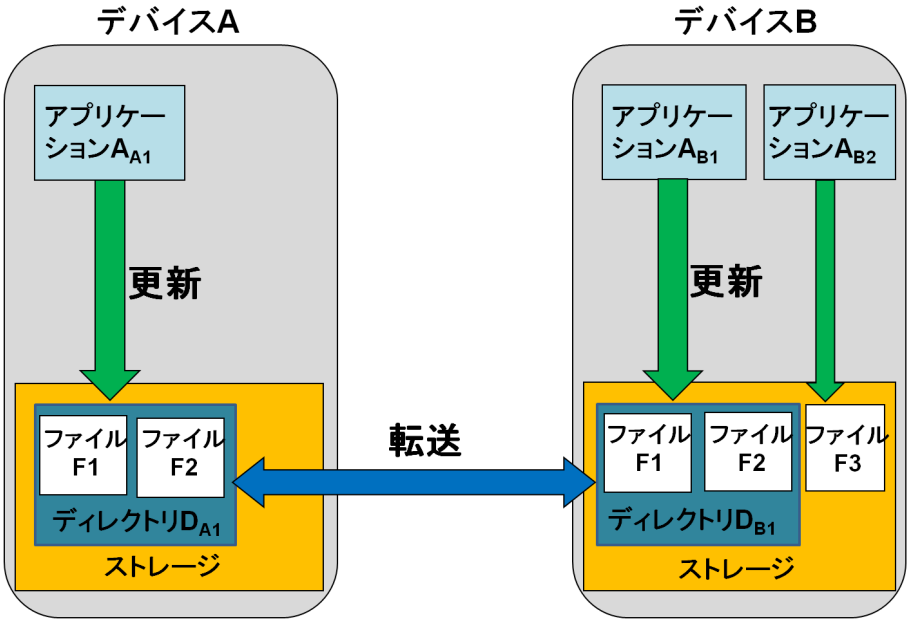

図 2.1: モバイルデバイスからなるシステム

前提とする 2 つのデバイスからなるシステムを図 2.1 に示す．各デバイスはアプリケーションおよびアプリケーションがアクセスするデータを持つ．データとはたとえば音楽や動画コンテンツなどを格納したファイルである．ここで，アプリケーション A_{A1} と A_{B1} は同じフォーマットのデータにアクセスするアプリケーションとする．たとえば Real Player と Windows Media Player である．また，それぞれのアプリケーションは予め決められたディレクトリにアクセスする．

それぞれのアプリケーションからそれぞれのストレージへのファイルの変更を更新と呼ぶ．更新の種類としてはファイルの生成，修正，削除の 3 つとする．また，あるデバイスで更新が発生した時，同じ更新を他のデバイスで行うために必要なデータを更新情報と呼び，更新の種類が生成，修正の場合には更新情報はファイルとなり，削除の場合にはファイルを削除する命令となる．更新情報の大きさは更新量と呼び，更新の種類が生成，修正の場合には更新量はファイルの大きさとなり，削除の場合には 0 となる．また，更新が発生したデバイスをソースデバイスとし，ソースデバイスより更新情報を受け取るデバイスをデスティネーションデバイスとする．そして，更新情報を伝えるための，ソースデバイスからディスティネーションデバイスへのデータの送信を転送と呼ぶ．同期の方向はデバイス A からデバイス B へ，またはその逆があり得る．

たとえば，デバイス A 上のアプリケーション A_{A1} がインターネット上から動画ファイルをダウンロードし，ディレクトリ D_{A1} にファイル F1 が追加されたとする．すると，デバイス A はデバイス B へとファイル F1 を転送し，デバイス B のディレクトリ D_{B1} にファイル F1 が生成される．なお，デバイス B 上のアプリケーション A_{B1} からデバイス A 上のディレクトリ D_{A1} へは直接アクセスできないものとする．

同期の範囲，更新の単位

同期の範囲とは同期の対象となる範囲を表し，ファイル，ディレクトリ，ストレージ全体の 3 つが考えられる．更新の単位とは，更新時に更新情報をどの程度の細かさで記録するかを表し，ディレクトリ単位，ファイル単位，ファイルの一部とする

かの 3 通りが考えられる．更新の単位がファイルの一部であるなら，差分の抽出を行うことで転送量を減らすことができる．

本研究では簡単化のため，同期の範囲はディレクトリ，更新の単位はファイルとする．ディレクトリの更新に対応する場合にはディレクトリを tar や zip などのアーカイブ形式にすることでファイルとして転送可能である．

同時更新について

デバイス A とデバイス B において更新が同時に起こる場合，各デバイスの同期対象となるディレクトリ間で不整合が生じる問題がある．この問題は各デバイスのファイルへの操作と転送のタイミングに依存する．例えば，デバイス A 上でファイル F1 に更新 (修正) があったとする．更新後のファイルをファイル F1' とする．後に，ファイル F1' が転送される前にデバイス B 上で同じファイルへ更新 (修正) が生じファイル F1" に変更されたとする．この場合，デバイス B ではデバイス A からの転送により，ファイル F1' に変更されるが，デバイス A ではデバイス B からの転送によりファイル F1" へ変更されてしまう．

この問題に対処するためには 2 つの方法がある．1 つ目は同時更新を許す方法であり，競合が生じるような場合にはファイルが 2 つ作られ，利用者がそれぞれのファイルのマージを取るなどをして対処する．2 つ目は排他制御を行い，同時更新が発生しないようにすることである．本研究では後者を選択する．このため，各デバイスでクリティカルセクションを設けることで，同時更新が起こらないようにする．詳しくは第 8 章で述べる．

ファイルフォーマットの違い

アプリケーションの種類ごとに対応するファイルフォーマットが異なる場合がある．たとえば，Windows Media Player では mp4 などの非対応のフォーマットがあり，変換ソフトを使わなければ再生することができない．対処方法としては理想的

にはHTMLのような共通フォーマットを定めることで解決できる．現状でこの問題を解消するためには，たとえば拡張子変換ソフト[13]などによりフォーマットを統一する手法が考えられるが，本研究では扱わないものとする．

表 2.1: 同期を実現する基本機能

名称	機能
更新の通知	更新を通知するイベントの発生
transfer(X，Y)	デバイスXからデバイスYへ更新情報を転送する
write()	ストレージへの書き込み

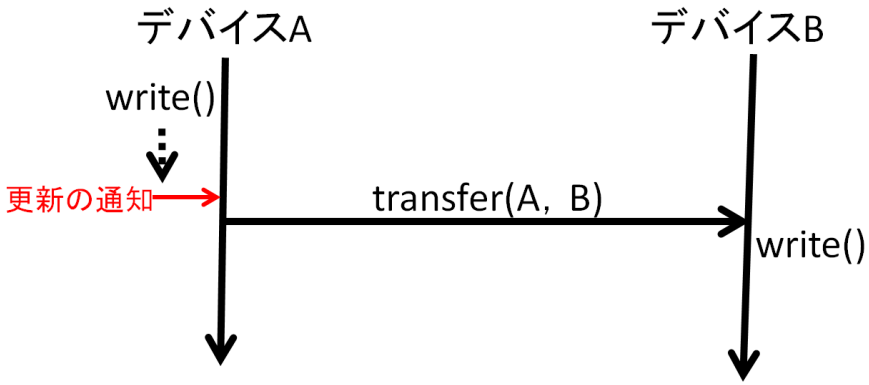

図 2.2: 一般的な同期のシーケンス

表2.1は同期を実現する機能を表したものである．デバイスXとデバイスYにはそれぞれデバイスAとデバイスBのいずれかが当てはまる．前提として，デバイスAとデバイスBは常に起動中であるため，更新があるとすぐに相手側のデバイスへ転送を行う．この時の同期の処理手順を図2.2に示す．

たとえば，デバイスA上でアプリケーションから同期対象のディレクトリに更新(write())があったとする．すると，更新を通知するイベントが発生し，その後

に transfer(A, B) によりデバイス A からデバイス B へと更新情報が転送される．デバイス B は受け取った更新情報に基づき，更新 (write()) を行う．また，逆方向も同じ動作となる．

第 3 章

停止デバイスへのファイル同期

本章では，同期対象となるデバイスをスマートフォンとノート PC とした場合に着目する．デバイス A をスマートフォン，デバイス B をノート PC とする．前提としてノート PC はバッテリー持続時間が短いため，未使用時は停止しているものとする．一方，スマートフォンは通話待ちのため常時起動しているものとする．本研究では，同期を行うために停止中のノート PC を起動し，転送を行った後に再び停止させることを前提とする．

3.1 同期の処理手順

転送を要求するデバイスをイニシエータデバイス，要求を受けるデバイスをレスポンダーデバイスとする．ソースデバイス，ディスティネーションデバイス，イニシエータデバイス，レスポンダーデバイス，それぞれの関係を表 3.1 に示す．イニシエータデバイスとソースデバイスが等しい場合，ソースデバイスの判断によりソースデバイスからデスティネーションデバイスへと転送が行われる．これを PUSH 同期と呼ぶ．たとえば，スマートフォンがイニシエータデバイスかつソースデバイスである場合，PUSH(S) と表す．対して，イニシエータデバイスとディスティネーションデバイスが等しい場合，ディスティネーションデバイスの判断によりディスティネーションデバイスからソースデバイスへ転送を要求し，その応答としてソースデバイスからデスティネーションデバイスへと転送が行われる．これを

PULL 同期と呼ぶ．たとえば，ノート PC がイニシエータデバイスかつディスティネーションデバイスである場合，PULL(N) と表す．また，添え字の S と N はそれぞれスマートフォンとノート PC を表す．

表 3.1: 同期の処理手順

同期の種類	イニシエータ	レスポンダー	ソース	ディスティネーション
PUSH(S)	スマートフォン	ノート PC	スマートフォン	ノート PC
PULL(S)	スマートフォン	ノート PC	ノート PC	スマートフォン
PUSH(N)	ノート PC	スマートフォン	ノート PC	スマートフォン
PULL(N)	ノート PC	スマートフォン	スマートフォン	ノート PC

表 3.2: 起動と停止を前提とした同期の処理手順

起動要求デバイス	同期の種類	図 3.1 との対応
スマートフォン	PUSH(S)	(1)
スマートフォン	PULL(N)	(2)
ノート PC	PUSH(S)	(3)
ノート PC	PULL(N)	(4)
スマートフォン	PUSH(N)	(5)
スマートフォン	PULL(S)	(6)
ノート PC	PUSH(N)	(7)
ノート PC	PULL(S)	(8)

起動と停止を前提とした場合の同期の処理手順は，表 3.1 で示される 4 パターンとノート PC への起動要求を出すデバイスを考慮すると，8 つのパターンに分類で

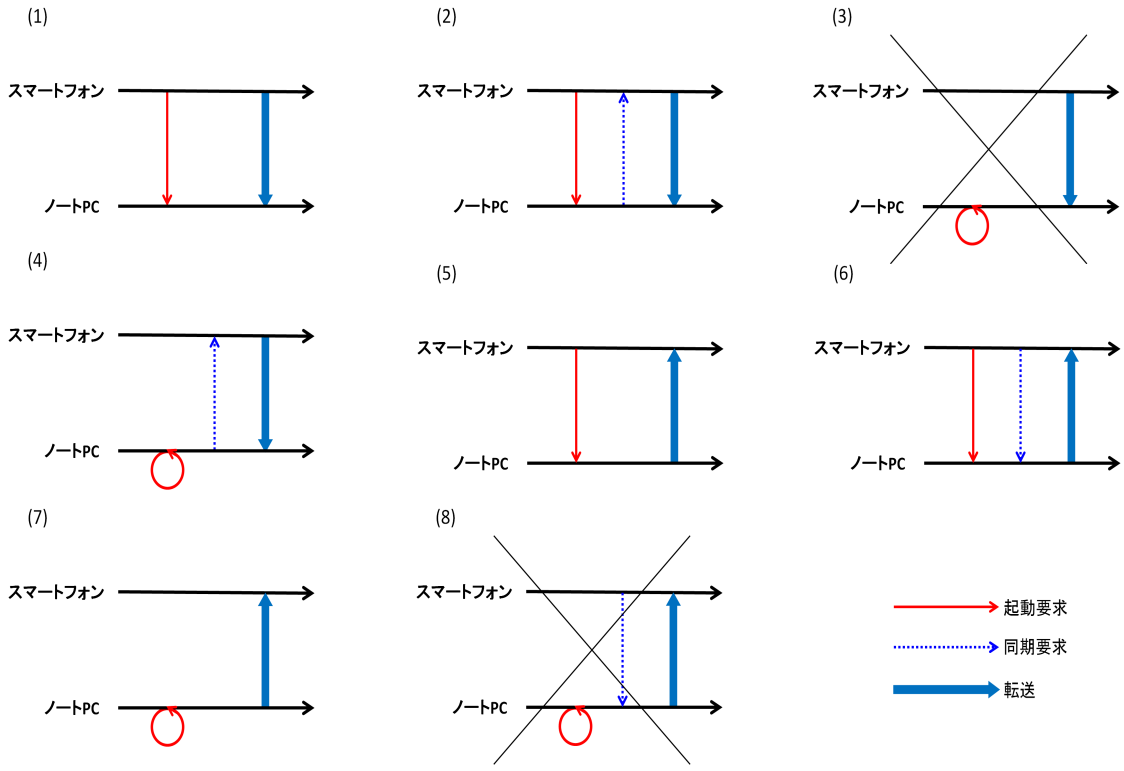

図 3.1: 表 3.2 で示した各処理手順

きる．表 3.2 に 8 つのパターンを示す．また図 3.1 は各処理手順を表したものである．(3) と (8) はシーケンスとして成り立たない．また，停止中のノート PC に更新が届くことはないため，(5)〜(8) は考えないものとする．(1) と (2) はイニシエータデバイスが異なるだけでほぼ同じ動作だが，(1) については，起動要求を行ってから転送を開始するまでの時間が一意に定まらないため考えないものとし，本研究では (2) と (4) の処理手順を用いる．起動と停止を前提とした場合の同期の機能を表したものを表 3.3 とし，これらを用いて (2) と (4) の処理手順の詳細を示したものをそれぞれ図 3.2 と図 3.3 に示す．

図 3.2 の例では，スマートフォン上でアプリケーションから同期対象のディレクトリに更新 (write()) があったとする．すると，更新を通知するイベントが発生し，start_up(S, N) によりスマートフォンからノート PC へ起動要求が出される．

表 3.3: 起動と停止を前提とした場合の同期を実現する機能

名称	機能
更新の通知	更新を通知するイベントの発生
transfer(X，Y)	デバイス X からデバイス Y への更新情報の転送
transfer_request(Y，X)	デバイス Y からデバイス X への transfer(X，Y) の要求
write()	ストレージへの書き込み
start_up(X，Y)	デバイス X からデバイス Y への起動要求

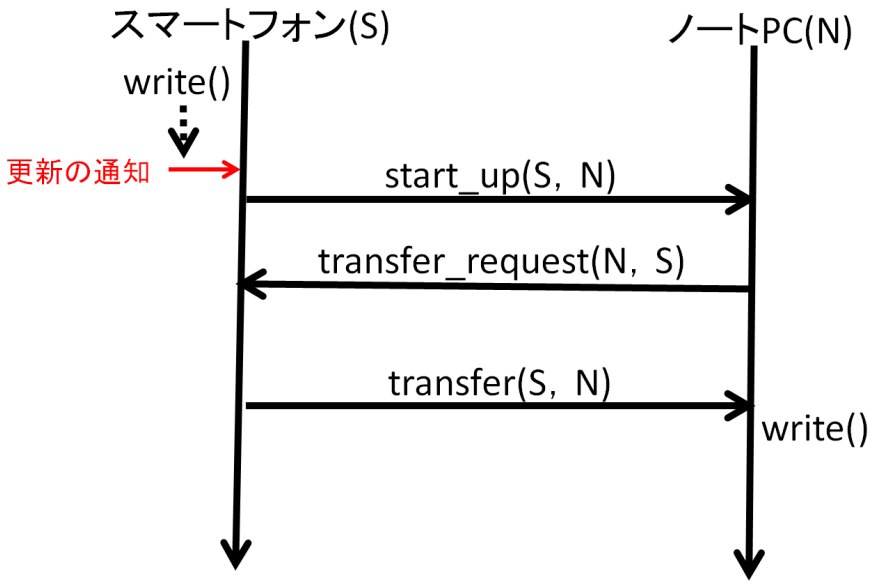

図 3.2: (2) の処理手順の詳細

ノート PC は start_up(S，N) を受け取ると起動処理を行い，起動完了後に transfer_request(N，S) によりノート PC からスマートフォンへ転送を要求する．スマートフォンは transfer_request(N，S) を受け取ると transfer(S，N) により更新情報を

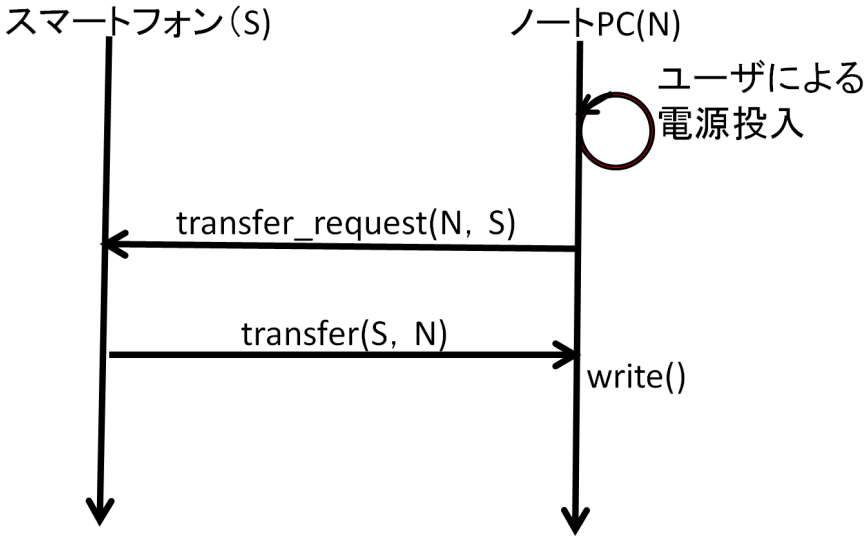

図 3.3: (4) の処理手順の詳細

転送する．転送終了後，ノート PC は受け取った更新情報に基づき，更新 (write()) を行う．

図 3.3 の例は，ユーザがノート PC に電源を投入した際の処理である．ユーザがノート PC に電源を投入すると，起動処理が行われ，起動完了後に transfer_request(N，S) により転送を要求する．スマートフォンは transfer_request(N，S) を受け取ると transfer(S，N) により，更新情報を転送する．転送終了後，ノート PC は受け取った更新情報に基づき，更新 (write()) を行う．

3.2 同期に伴う待ち時間

ユーザがノート PC の電源を投入した時，ノート PC の停止中にスマートフォン上に蓄積したすべての更新情報はまとめて転送される．蓄積した更新量が大きい場合，ユーザの待ち時間が問題となる．ここでの待ち時間とは，ユーザが電源を投入してから同期が完了しファイルにアクセス可能になるまでの時間と定義する．待ち時間は起動にかかる時間と転送にかかる時間に分けられる．図 3.4 は同期に伴

う待ち時間を表したものである．横軸が時刻 [sec]，縦軸が消費電力 [J/sec] となる．

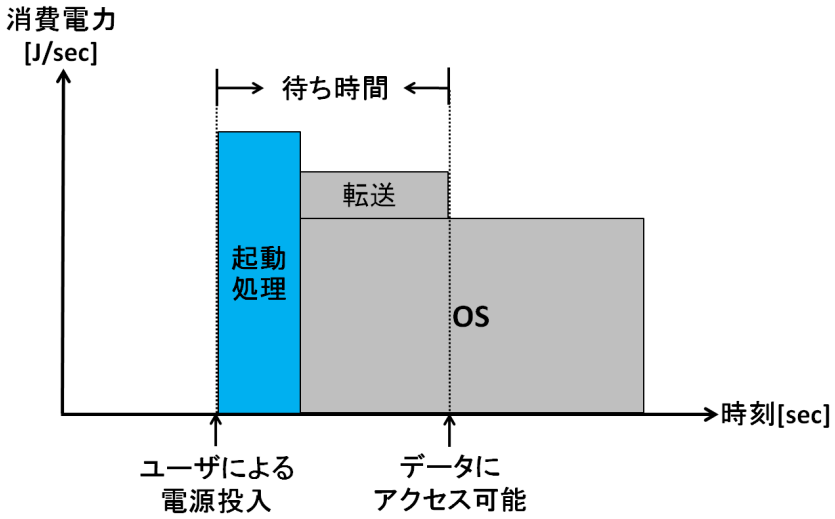

図 3.4: ノート PC の同期に伴う待ち時間と消費電力

3.3 同期に必要な消費電力

ノート PC とスマートフォンの同期に必要な消費電力について図 3.5 に示す．それぞれ横軸が時刻 [sec]，縦軸が消費電力 [J/sec] を表す．

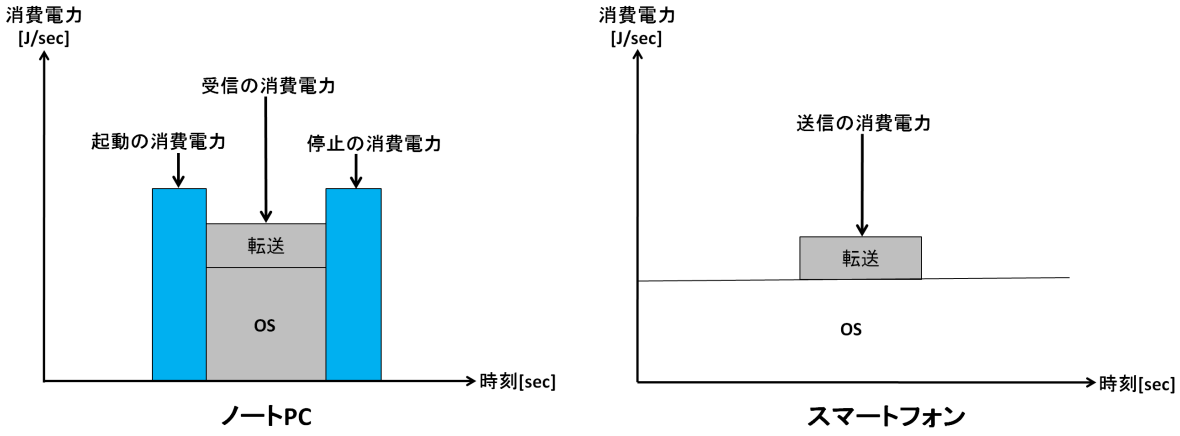

図 3.5: 同期に必要な消費電力

ノート PC での同期に必要な消費電力は，デバイスの起動に必要な消費電力と図中の OS と転送の部分が示すファイルの受信に必要な消費電力，停止に必要な消費電力の 3 つの和である．スマートフォンでの同期に必要な消費電力は図中の転送の部分が示すファイルの送信に必要な消費電力である．本研究では，スマートフォンと比べてノート PC の消費電力が支配的であるため，以下ではノート PC の消費電力に着目して考える．スマートフォンとノート PC の同期に必要な消費電力の測定結果については 8.1 節で詳細を示す．

3.4 待ち時間と消費電力の関係

本研究では，停止中のノート PC と同期を行うことを想定するため，ユーザがノート PC を利用していない停止中の間に何度か同期が行われるものとする．待ち時間の長さは同期が行われるタイミングに依存する．同期のタイミングにおける待ち時間の違いを表したものを図 3.6 に示す．

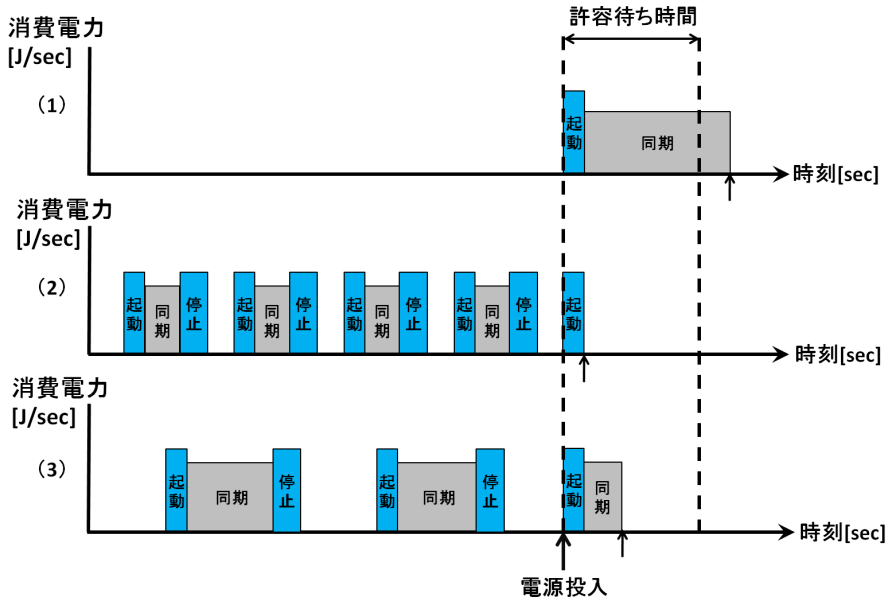

図 3.6: 待ち時間と消費電力の関係

(1) はユーザによってノート PC が起動される前に同期を行わない例であり，Dropbox などの既存のソフトウェアはこのような動作となる．この場合，ユーザによる起動前にスマートフォン上で更新が発生してもただちに同期は行わず，ユーザによる起動後に一括して転送が行われるため，蓄積した更新量が最大となり，待ち時間が最も長くなる．対して (2) は更新が発生する度にノート PC を起動した後に転送を行い，再び停止させる例である．この場合ユーザによる起動後の待ち時間は最小となるが，転送に伴うノート PC の起動，停止に必要な消費電力量は最大となる．(1)，(2) の例から待ち時間と消費電力はトレードオフの関係にあることがわかる．また，(3) は更新情報が一定量蓄積したら転送に伴う起動と停止を行う例である．

3.5 同期の要求条件

同期の要求条件は，以下の 1 と 2 の条件を同時に満たすこととする．

1. 待ち時間を一定の許容範囲内に収めること
   ユーザが許容できる待ち時間を予め決めておく．ユーザが電源を投入してからの待ち時間を許容待ち時間内に収めることが求められる．
2. 消費電力を抑えること
   各デバイスはモバイル環境においてバッテリー駆動で動作するため，各デバイスに供給できる電力には制限がある．そのため，同期に必要な消費電力を少なくすることが求められる．

第 4 章

デバイスの電源状態

前提としてノート PC は未使用時に停止していることから，停止中の電源状態を考慮する必要がある．本章ではデバイスの電源状態について説明する．

4.1 ACPI

Advanced Configuration and Power Interface（ACPI）[7] は，電源制御に関する統一規格である。

4.1.1 6 つの電源状態

ACPI で定義されている電源状態は S0 から S5 までの 6 つの状態に分けられる．数字が大きければ大きいほど起動にかかる時間が長くなるが，各電源状態の定常消費電力は少なくなる．各電源状態においての違いはメモリ，CPU のキャッシュやレジスタに電源が供給されているかどうかによって分けられる．

- S0 :
  メモリ，CPU のキャッシュ，レジスタすべてにおいて電源が供給されている状態
- S1 :
  メモリ，CPU のキャッシュ，レジスタすべてに電源が供給されているが，CPU はほとんどの割り込み処理を停止させている状態

- S2 :

  メモリ，CPU のキャッシュに電源が供給されており，CPU のレジスタに電源が供給されていない状態

- S3 :

  メモリのみに電源が供給されており，CPU のキャッシュ，レジスタ共に電源が供給されていない状態

- S4:

  メモリの内容をディスクに書き出しておき，メモリ，CPU のキャッシュやレジスタすべてに電源が供給されていない状態

- S5:

  メモリ，CPU のキャッシュやレジスタすべてに電源が供給されていない状態

4.1.2 状態間の遷移

それぞれの電源状態と状態間の遷移関係を図 4.1 に示す．なお，S1 と S2 については，対応しているデバイスが限られるため，本研究では扱わないものとする．S0 をオン状態，S3 をスリープ状態，S4 を休止状態，S5 をオフ状態と呼び，S3〜S5 をまとめて停止状態と呼ぶ．

状態遷移として以下の 6 つがある．オン状態からオフ状態への遷移 (1) では実行中の処理を終了させ，給電を絶つ．オフ状態からオン状態への遷移 (2) では OS を新たに起動する．オン状態から休止状態への遷移 (3) では実行中のメモリの内容をディスクへ退避させ，給電を絶つ．休止状態からオン状態への遷移 (4) ではディスクからメモリの内容が読み出され，実行が再開される．オン状態からスリープ状態への遷移 (5) ではメモリ以外の給電を絶ち，メモリの内容は保持される．スリープ状態からオン状態への遷移 (6) では給電によって保持されていたメモリの内容から，実行が再開される．停止とは (1)，(3)，(5) のいずれかの状態遷移を表し，起

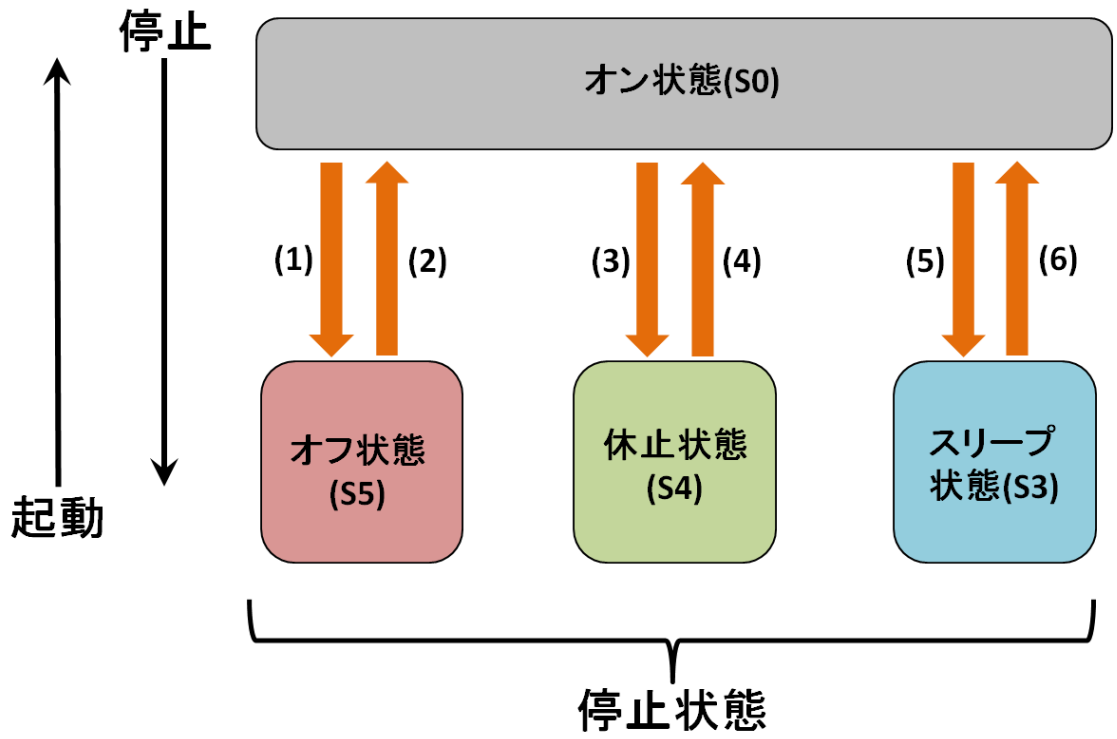

図 4.1: 各電源状態と遷移関係

動とは (2)，(4)，(6) のいずれかの状態遷移を表す．また，停止状態のうち起動にかかる時間と消費電力量が最も大きくなるのはオフ状態，次に休止状態，最も小さくなるのはスリープ状態である．一方，停止状態の定常消費電力が最も大きくなるのはスリープ状態であり，休止状態とオフ状態はほぼ 0 となる．

4.1.3 スリープ状態から休止状態への遷移

Windows Vista 以降の OS ではスリープと休止を交えたハイブリッドスリープ [11] と呼ばれる電源状態がある．初めにスリープ状態と同様にメモリにのみ電源が供給される．18 時間経過するか，バッテリーの残量が少なくなると，メモリの内容を自動的にディスクへ書き込み，休止状態に切り替わる．ハイブリッドスリープは OS への依存性が強く，切り替えを任意に行うことができないといった問題点が

あるため，本研究では利用していない．もし，今後切り替えを行うための API などが公開され，任意に切り替えが可能になれば，スリープ状態から休止状態へ切り替えるために一度電源を投入する必要がないため，本研究にも適用できると考えられる．

4.2 Wake–on–Wireless–LAN

ネットワークにつながっているデバイスに対して，自動的に電源を投入する方法として，Wake–on–LAN(WOL)[9] や Wake–on–wirelessLAN(WoWLAN)[10] がある．WOL は通信メディアとして IEEE 802.31(Ethernet) を用い，WoWLAN では IEEE 802.11 を用いる．

具体的な起動方法としては，外部の起動中のデバイスから起動待ちの停止中のデバイスへ特定のパケットを送信することによって停止中のデバイスを起動させる．特定のパケットとはヘッダ情報として宛先アドレスを FF:FF:FF:FF:FF:FF(ブロードキャストアドレス) とし，ヘッダの後ろに起動待ちのデバイスの MAC アドレスを 16 回以上連続して表記したものである．このようなパケットのことをマジックパケットと呼ぶ．利用条件としては，起動待ちのデバイスのネットワークアダプタやマザーボード，BIOS，OS などが Wake–on–LAN や Wake–on–WirelessLAN に対応している必要がある．

本研究ではモバイル環境であるため，WoWLAN の利用が想定されるが，現状として対応する機器が少ないといった問題がある．

第 5 章

通信メディア

本章では本研究においてデバイス間の通信に用いる通信メディアについて説明する．モバイルデバイス間の通信であるため，通信方法は無線経由を前提とする．

- IEEE 802.11(Wi-Fi)[14]

  IEEE によって定められた無線 LAN の標準規格である．策定後，様々な改定がなされており，周波数，減衰距離，転送速度等が異なる．Wi-Fi について規格別に周波数や公称速度などを比較したものを表 5.1 に示す．

表 5.1: Wi-Fi の規格別の比較表

規格名	周波数	公称速度	メリット	デメリット
802.11a	5GHz	54Mbps	混信がない	互換性が低い
802.11b	2.4GHz	11Mbps	互換性が高い	混信がある，速度が遅い
802.11g	2.4GHz	54Mbps	b の上位互換	混信がある
802.11n	2.4or5GHz	150-600Mbps	速度が速い	未対応のスマートフォンがある

- IEEE 802.15.1(Bluetooth)[16]

  Bluetooth は近距離無線通信技術の 1 つであり，数 m 程度の機器間接続に用い

られる．転送速度は Wi-Fi や 3G[18] に劣るが，送信に必要な電力は約 1mW であることなどから省電力性を重視している．

- WiGig[19]

  WiGig (Wireless Gigabit) は，WiGig Alliance により定められた 60GHz 帯の無線通信規格であり，帯域幅は 7〜9GHz である．転送速度は最大で 7Gbps と高速であることから，HDTV 動画などの大容量のデータを非圧縮で伝送することができる．しかし，通信可能な距離は 10m 程度と短く，直進性が強い為，遮蔽物を越えられないなどの問題がある．

- IrDA[17]

  IrDA は，赤外線による光無線データ通信規格であり，ノート PC や携帯電話，デジタルカメラなどで利用されている．転送速度は約 16Mbps であり，他の無線通信の規格と比較すると容量の大きいデータを通信するのに不向きである．

第 6 章

提案法

本章では，3.5 節で示した要求条件を満たす同期方法を提案する．6.1 節では，転送を行うタイミングを制御することで起動と停止の回数を減らし，起動と停止に伴う消費電力量を削減する手法について述べる．6.2 節では，6.1 節で示した手法を用いた同期を前提とした上で，2 つのモード (休止モードとスリープモード) を切り替えることで消費電力量を削減する手法について述べる．

6.1 転送のタイミング制御

3.4 節では，同期に伴う待ち時間と同期に必要な消費電力がトレードオフの関係にあることを示した．

ここでは，更新情報がある一定量蓄積するまで起動と停止を伴う同期を行わず，転送をまとめることで起動と停止の回数を減らすことを考える．どれぐらいの更新情報をまとめることができるかは，ユーザが許容できる待ち時間に依存する．

まず，ユーザが許容できる待ち時間を T_W[sec] とする．待ち時間は起動にかかる時間と転送にかかる時間に分けられる．T_W のうち，転送にかけることのできる時間を T_T[sec] とすると，

$$T_T = T_W - \text{起動にかかる時間}$$

となる．ここで，データの転送速度を V_C[byte/sec] とおく．V_C はデバイス間の通信に用いる通信メディアの速度によって決まる．T_T 中に転送することができる更

新量を最大転送データ量 D[byte] とすると，

$$D = T_T V_C$$

となる．更新情報の蓄積が D バイトを超える度に，起動をし，転送を行い，再び停止させる．これによって，ユーザが電源を投入してからの待ち時間を T_W 以内に収めつつ，起動と停止の回数を減らし，これに伴う消費電力量を削減することができる．

図 6.1 は転送のタイミング制御を行った場合の時刻 [sec] と消費電力 [J/sec] の関係を示している．

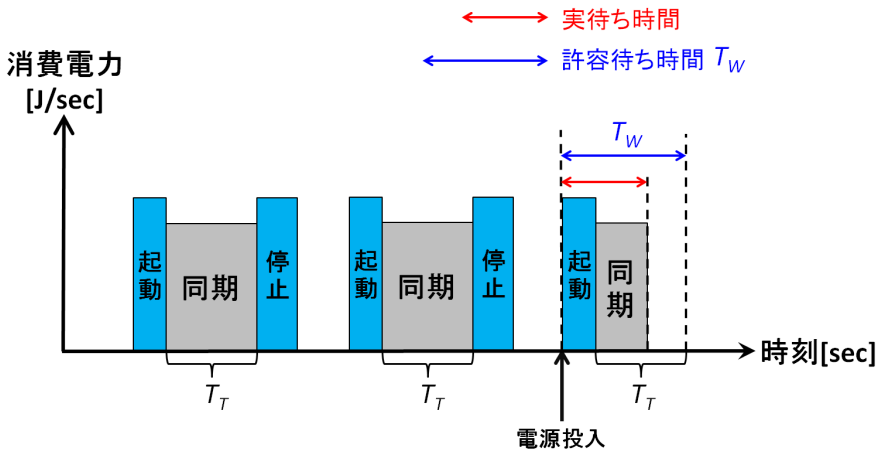

図 6.1: 転送のタイミング制御

例として，$T_W = 60$[sec]，$V_C = 1.2 \times 10^6$[byte/sec]，起動にかかる時間 $= 55$[sec] とすると $T_T = 60 - 55 = 5$[sec] となり，$D = 6.0 \times 10^6$[byte] となる．もし T_W が大きい場合には，最大転送データ量 D は大きくなり，1 度に蓄積可能な更新量が増えるため，起動と停止の回数が少なくなる．

- T_W 内に収まらない場合

  例外として，D バイトを超えて起動した時とほぼ同時にユーザがノート PC

の電源を投入した場合，更新量は D バイトを超えていることから，ユーザの待ち時間が T_W を超える可能性がある．本研究では実際に運用する上では問題ないとし，考慮しないものとする．

6.2 休止モードとスリープモードの切り替え

4.1.2 節で示したように，停止状態の違いによって消費電力が異なる．オフ状態と比べて休止状態は起動と停止にかかる消費電力量は少ない．一方で，オフ状態と休止状態の定常消費電力は等しく，0 である．このため，以降ではオフ状態は考えないものとし，オン状態，休止状態，スリープ状態の 3 つの電源状態を扱うものとする．

以降では，停止する時に停止状態を休止状態とする利用形態を休止モードとし，停止する時に停止状態をスリープ状態とする利用形態をスリープモードとする．図 6.2 は 6.1 節で示した同期処理を前提とした時の，休止モードとスリープモードにおける消費電力の違いを表したものである．スリープモードはメモリへ給電がされている分，停止中に電力を消費するが，起動にかかる時間，消費電力共に休止モードと比較して小さくなる．

ここで，休止モードにおける最大転送データ量を D_H，スリープモードにおける最大転送データ量を D_S とする．各モードにおいてノート PC の起動にかかる時間を比較すると，休止状態からオン状態への遷移にかかる時間よりもスリープ状態からオン状態への遷移にかかる時間の方が短い．このため，$D_H < D_S$ であり，最大転送データ量はスリープモードの方が大きくなる．たとえば，$T_W = 60[\mathrm{sec}]$，$V_C = 1.2 \times 10^6[\mathrm{byte/sec}]$，休止状態からオン状態への遷移にかかる時間 $= 55[\mathrm{sec}]$，スリープ状態からオン状態への遷移にかかる時間 $= 10[\mathrm{sec}]$ とすると，$D_H = 6.0 \times 10^6[\mathrm{byte}]$，$D_S = 6.0 \times 10^7[\mathrm{byte}]$ となる．したがって，同じ更新量の場合，スリープモードは休止モードに比べ起動と停止の回数が少なくなる．

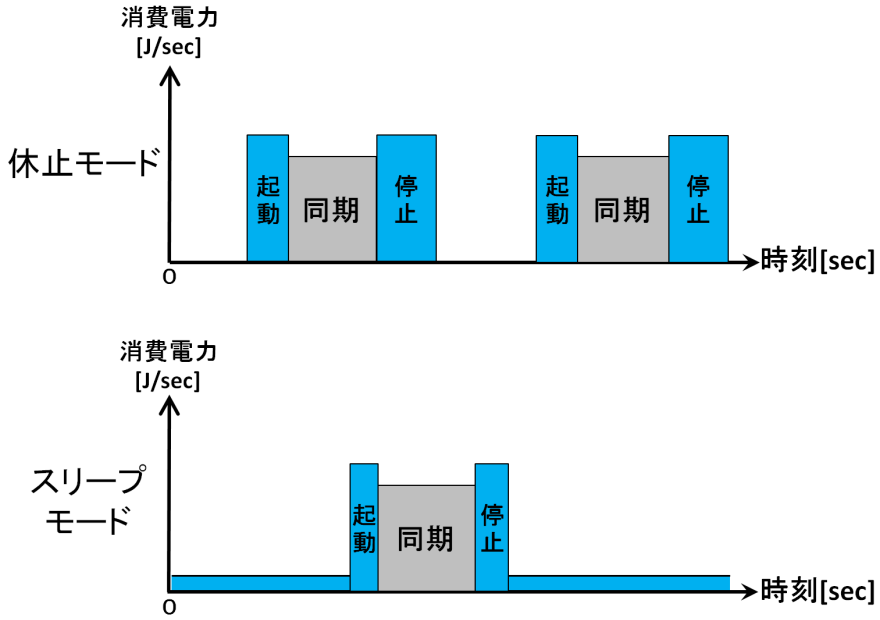

図 6.2: 6.1 節での同期処理を前提とした場合の各モードにおけるノート PC の消費電力

以降では単位時間あたりの更新量のことを更新速度と呼ぶ．スマートフォン上での更新速度が大きい場合，転送に伴う起動と停止が頻繁に必要となる．このため，起動と停止にかかる消費電力量がより少なく，同じ更新量の場合に起動と停止の回数がより少ないスリープモードが休止モードより省電力となる．逆に，更新速度が小さい場合，起動と停止がほとんど起こらないため，停止状態の定常消費電力が少ない休止モードがスリープモードより省電力となる．そこで本研究では，更新速度の違いによって休止モードとスリープモードを切り替えることで，さらなる省電力化を計る．

図 6.3 は横軸を更新速度 [byte/sec]，縦軸を平均消費電力 [J/sec] とし，2 つのモードの違いを表したものである．図で示すように，提案法では休止モードとスリープモードをしきい値 V_{th} で切り替える．

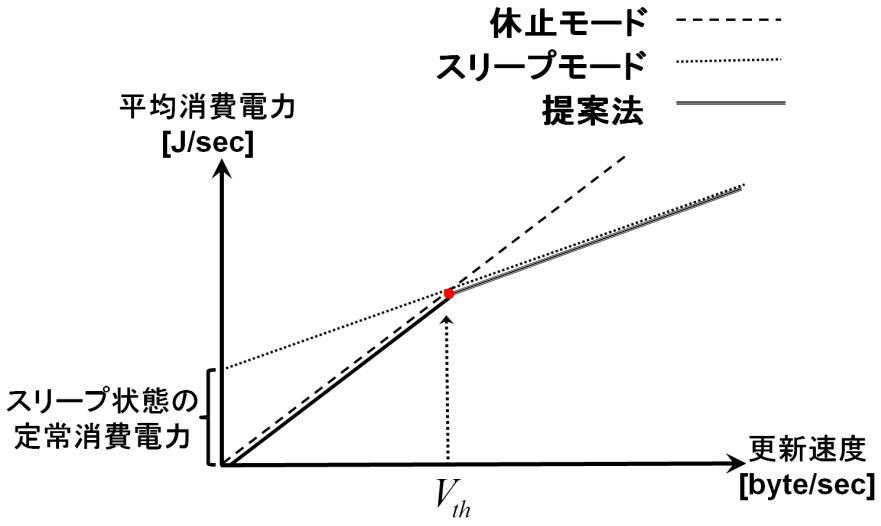

図 6.3: スマートフォン上の更新速度とノート PC の平均消費電力の関係

6.3 しきい値 V_{th} の導出

しきい値 V_{th} を解析するために，更新が細かい頻度かつ一定速度で届くような更新のパターンを考える．図 6.4 の各モードにおける時刻と消費電力の関係を元に，各モードの解析を行う．

休止状態からオン状態に遷移する間の消費電力量を Q_1[J]，オン状態から休止状態に遷移する間の消費電力量を Q_2[J]，スリープ状態の場合，それぞれ Q_3[J]，Q_4[J]，さらに休止状態からオン状態に遷移するのに必要な時間を T_1[sec]，オン状態から休止状態に遷移するのに必要な時間を T_2[sec]，スリープ状態の場合，T_3[sec]，T_4[sec] とする．この時，休止モードでの 1 回の転送にかけることのできる時間 T_H[sec]，スリープモードでの 1 回の転送にかけることのできる時間 T_S[sec] はそれぞれ

$$\begin{aligned} T_H &= T_W - T_1, \\ T_S &= T_W - T_3 \end{aligned}$$

で表すことができる．また，休止モードでの 1 回の転送にまとめることのできる最大転送データ量 D_H[byte]，スリープモードでの 1 回の転送にまとめることのでき

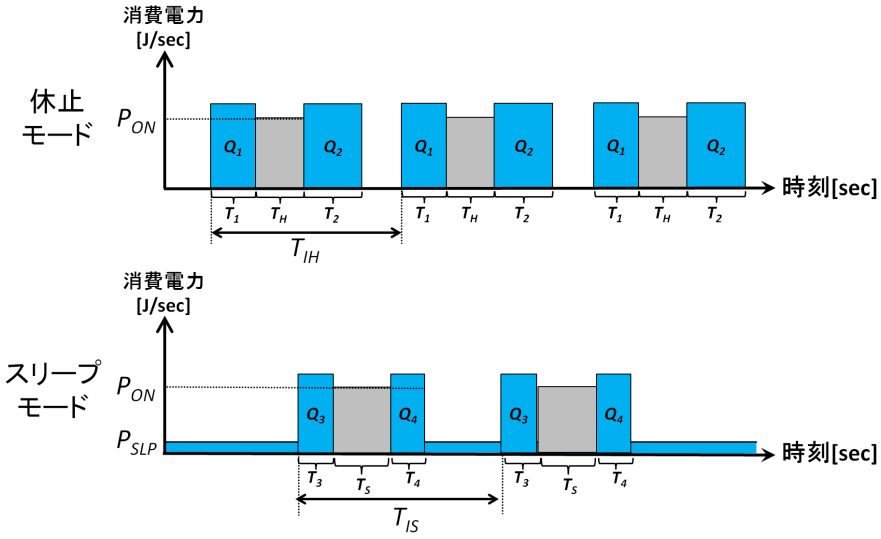

図 6.4: 休止モードとスリープモードにおける時間と消費電力の関係

る最大転送データ量 D_S[byte] はそれぞれ

$$\begin{aligned} D_H &= V_C\, T_H, \\ D_S &= V_C\, T_S \end{aligned}$$

と表せる．さらにオン状態での定常消費電力を P_{ON}[J/sec]，休止状態での定常消費電力を P_{HIB}[J/sec]，スリープ状態では P_{SLP}[J/sec] とする．また，更新速度を V_U，ある転送から次の転送が起こるまでの時間を転送間隔とする．休止モードにおける転送間隔 T_{IH}[sec]，スリープモードにおける転送間隔 T_{IS}[sec] はそれぞれ

$$\begin{aligned} T_{IH} &= D_H \,/\, V_U, \\ T_{IS} &= D_S \,/\, V_U \end{aligned}$$

と表せる．また，T_{IH} 間の消費電力量 Q_{IH}[J] は

$$\begin{aligned} Q_{IH} &= Q_1 + P_{ON}\, T_H + Q_2 + P_{HIB}(T_{IH} - (T_1 + T_H + T_2)) \\ &= Q_1 + P_{ON}\, T_H + Q_2 \qquad (P_{HIB} = 0 \text{ より}) \end{aligned}$$

となる．T_{IS} 間の消費電力量 Q_{IS}[J] は

$$Q_{IS} = Q_3 + P_{ON}\,T_S + Q_4 + P_{SLP}(T_{IS} - (T_3 + T_S + T_4))$$

となる．これより休止モードにおける平均消費電力 P_{IH}[J/sec] は

$$P_{IH} = \frac{Q_{IH}}{T_{IH}} = \left(\frac{Q_{IH}}{D_H}\right) V_U$$

となり，スリープモードにおける平均消費電力 P_{IS}[J/sec] は

$$\begin{aligned} P_{IS} &= \frac{Q_{IS}}{T_{IS}} \\ &= \left(\frac{Q_3 + P_{ON}\,T_S + Q_4 - P_{SLP}(T_3 + T_S + T_4)}{D_S}\right) V_U + P_{SLP} \end{aligned}$$

となる．

しきい値 V_{th}[byte/sec] を求めるために，$P_{IH} = P_{IS}$ とおいて V_U について解くと，

$$\begin{aligned} V_U &= \left(\frac{D_H\,D_S\,P_{SLP}}{D_S\,Q_{IH} - D_H(Q_3 + P_{ON}\,T_S + Q_4 - P_{SLP}(T_3 + T_S + T_4))}\right) \\ &= \left(\frac{V_C\,T_H\,T_S\,P_{SLP}}{T_S(Q_1 + Q_2) - T_H(Q_3 + Q_4) + T_H\,P_{SLP}(T_3 + T_S + T_4)}\right) \\ &= V_C \left(\frac{Q_1 + Q_2}{T_H\,P_{SLP}} - \frac{Q_3 + Q_4}{T_S\,P_{SLP}} + \frac{T_3 + T_4}{T_S} + 1\right)^{-1} \end{aligned}$$

となる．したがって V_{th} は

$$V_{th} = V_C \left(\frac{Q_1 + Q_2}{P_{SLP}\,T_H} - \frac{Q_3 + Q_4}{P_{SLP}\,T_S} + \frac{T_3 + T_4}{T_S} + 1\right)^{-1} \tag{6.1}$$

と表すことができる．

なお，(6.1) 式より以下が成り立つ．

$$\begin{aligned} \lim_{T_W \to \infty} V_{th} &= V_C, \\ \lim_{P_{SLP} \to 0} V_{th} &= 0 \end{aligned}$$

許容待ち時間 T_W が極端に長い場合，V_{th} は V_C に近づく．P_{SLP} を 0 に近付けた場合にはスリープモードのオーバヘッドが 0 に近づくため，V_{th} は 0 に近づく．

第 7 章

設計

本章では，提案法で示したタイミング制御，また，休止モードとスリープモードの切り替えを実現するシステムの設計を行う．7.2 節では同期の処理手順および各処理の詳細について述べる．7.3 節では更新速度の計算方法について述べる．

7.1 システムの構成

システムの構成図を図 7.1 に示す．デバイスはそれぞれスマートフォンとノート PC があり，各デバイス上でハードウェア，OS が動作し，その上で同期アプリケーションが動くものとする．

7.2 同期の処理手順

同期の処理はそれぞれスマートフォンとノート PC 側に分けられる．2.1 節では 2 つのデバイスが起動中の同期処理について説明した．本設計において，ノート PC が起動中の場合には 2.1 節の処理を行う．また，3.1 節ではノート PC が停止中の起動と停止を前提とした処理手順について説明した．ノート PC が停止中の処理手順については 8 つのパターンを示したが，本設計ではこのうち (2) と (4) の 2 パターンを用いる．

以降では更新の内容（更新の種類，ファイル名，ファイルサイズ，更新時刻，転送フラグ）を保存するリストを *updatelist*，現在蓄積している更新量を

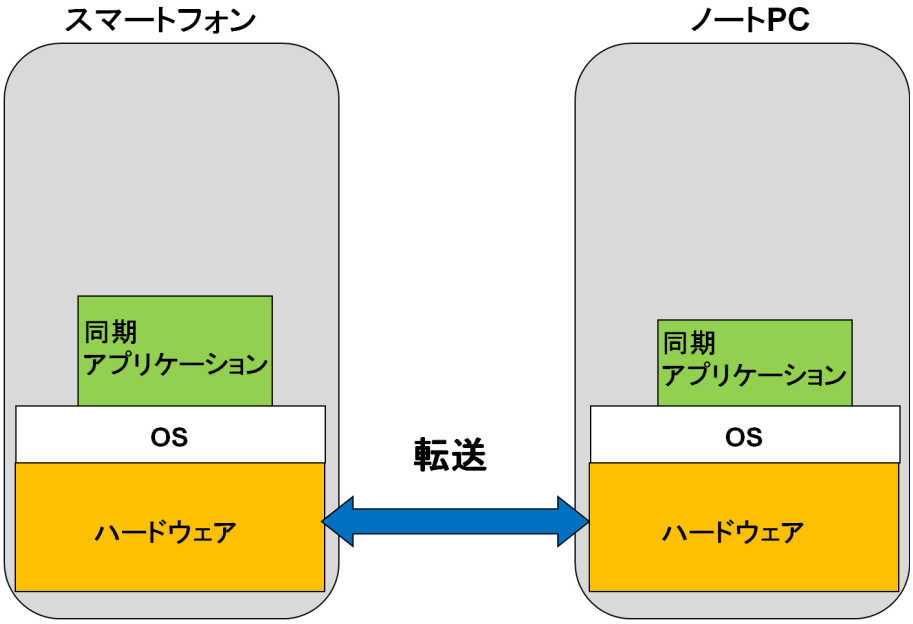

図 7.1: システムの構成図

$CurrentBuffer$(単位は [byte]，初期値は 0) とする．休止モードにおける最大転送データ量を D_H[byte]，スリープモードにおける最大転送データ量を D_S[byte] とし，現在の最大転送データ量を $CurrentD$[byte]（D_H または D_S）とする．また，ノート PC の現在のモードを $Currentmode$（休止モードまたはスリープモード），更新速度を V_U[byte/sec]，切り替えのしきい値を V_{th}[byte/sec] とする．

7.2.1 スマートフォン側の処理手順

スマートフォンの処理手順を図 7.2 に示す．なお，図の条件分岐 1 については，スマートフォンの処理中における step6 の処理手順の中で詳しく説明する．また，更新情報の受信処理については一般的であるため，省略する．

step1[更新の確認]

更新があるかどうかを確認する．更新がなくノート PC が停止中である場合 step4 へ移る．更新がなくノート PC が起動中である場合 step10 へ移る．

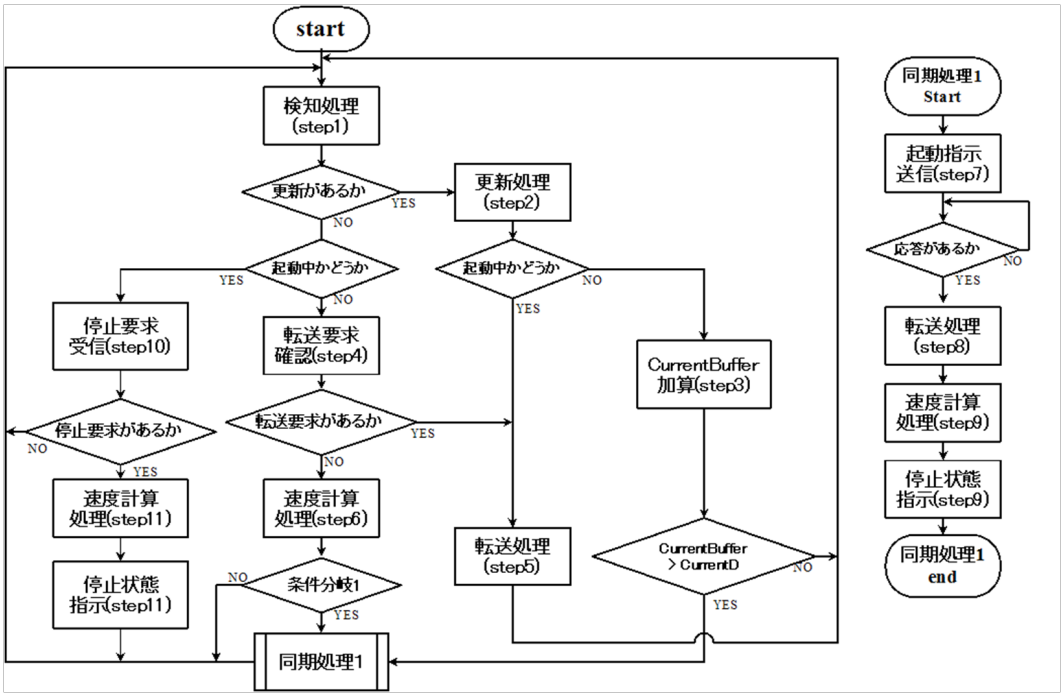

図 7.2: スマートフォンの処理手順

step2[更新処理]

updatelist に更新の内容を保存する．転送フラグは 0 とする．更新の種類が修正であり，*updatelist* 内の未転送のファイルの中に同一名のファイルがある場合には，二重転送を行わないように，$Currentbuffer$ から古いファイルのファイルサイズを減算し，転送フラグを 1 とする．ノート PC が起動中であれば step5 へ移る．

step3[$CurrentBuffer$ 加算]

$CurrentBuffer$ に発生した更新のファイルサイズを加算する．$CurrentBuffer < CurrentD$ であれば，step1 へ戻る．$CurrentBuffer > CurrentD$ であれば，$CurrentBuffer$ を 0 とし，step7 へ移る．

step4[転送要求確認]

ノート PC からの転送要求があるかどうかを確認する．転送要求がない場合

step6 へ移る．

step5[転送処理]

発生した更新の種類が生成または修正の場合は，スマートフォンからノート PC へ対象ファイルを更新情報として転送し，削除の場合はノート PC 内の対象ファイルを削除する命令を更新情報として転送する．step1 へ戻る．

step6[速度計算処理]

現在の更新速度 V_U を計算する．もし，しきい値 $V_{th} < V_U$ かつ現在がスリープモード (図中の条件分岐 1) であれば，step7 へ移る．そうでない場合 step1 へ戻る．

step7[起動指示送信]

ノート PC へ起動指示を送信する．起動応答があるまで待機する．

step8[転送処理]

updatelist を参照し，未転送のファイルを転送し，転送フラグを 1 とする．更新の種類が生成または修正の場合はノート PC へ対象ファイルを更新情報として転送し，削除の場合はノート PC 内の対象ファイルを削除する命令を更新情報として転送する．

step9[速度計算処理+停止指示送信]

現在の更新速度 V_U を計算し，$V_U < V_{th}$ であれば $Currentmode$ を休止モード，$CurrentD$ を D_H とする．$V_U > V_{th}$ であれば $Currentmode$ をスリープモード，$CurrentD$ を D_S とし．ノート PC へ停止指示を出す．step1 へ戻る．

step10[停止要求受信]

スマートフォンからの停止要求の確認を行う．要求がなければ step1 へ戻る．

step11[速度計算処理+停止指示送信]

現在の更新速度 V_U を計算し，$V_U < V_{th}$ であれば $Currentmode$ を休止モード，

$CurrentD$ を D_H とする．$V_U > V_{th}$ であれば $Currentmode$ をスリープモード，$CurrentD$ を D_S とする．ノート PC へ停止指示を出す．step1 へ戻る．

7.2.2 ノート PC 側の処理手順

次に，ノート PC 側の処理手順を図 7.3 に示す．スマートフォンと同様に更新情報の受信処理については一般的であるため，省略する．

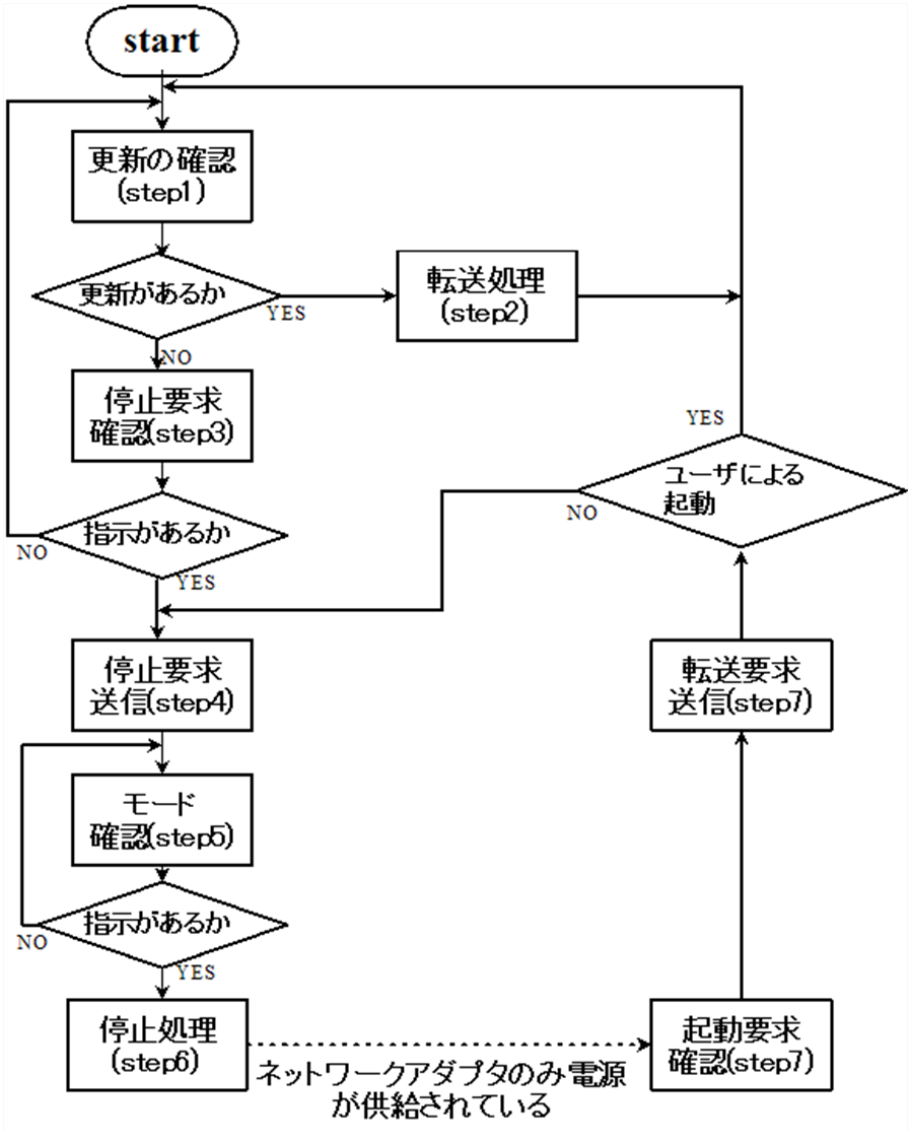

図 7.3: ノート PC の処理手順

step1[更新の確認]

更新があるかどうかの確認を行う．更新がない場合，step3 へ移る．

step2[転送処理]

発生した更新の種類が生成または修正の場合はノート PC からスマートフォンへ対象ファイルを更新情報として転送し，削除の場合はスマートフォン内の対象ファイルを削除する命令を更新情報として転送する．step1 へ戻る．

step3[停止要求確認]

ユーザからノート PC の停止指示があるかどうか確認を行う．指示がなければ step1 へ戻る．

step4[停止要求送信]

ノート PC からスマートフォンへ停止要求を出す．

step5[モード確認]

スマートフォンからの停止指示があるか確認を行う．停止指示がない場合 step5 の最初に戻る．

step6[停止処理]

スマートフォンから指定されたモードが休止モードである場合，休止状態で停止する．指定されたモードがスリープモードである場合，スリープ状態で停止する．

step7[起動要求確認+転送要求送信]

起動要求の確認を行う．起動要求を受け取った場合，スマートフォンへ転送要求を出す．もしスマートフォンから起動要求を受け取った場合，step4 へ，それ以外の場合はユーザから電源が投入されたものとみなし，step1 へ戻る．

7.3 更新速度の計算

ノート PC の休止モードとスリープモードの切り替えを行うために，スマートフォン上では適切なタイミングで更新速度 V_U を計算する必要がある．

7.3.1 更新速度の計算方法

図 7.4 は更新のパターンの例を示したものである．横軸に時刻 [sec]，縦軸は更新量 [byte] を表す．1 つ 1 つの更新は不規則なタイミングで発生し，様々な更新量を持つものとする．

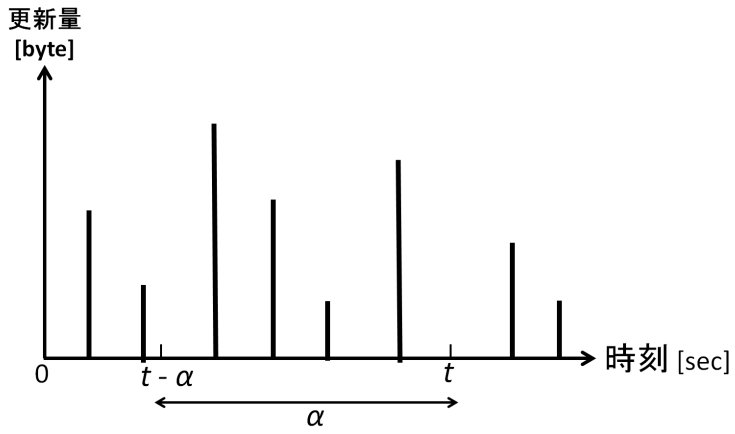

図 7.4: 更新のパターンの例

ここである時刻 t にて更新速度を求めることを考える．計測時間を α [sec] とおくと，時刻 t での更新速度 $V_U(t, \alpha)$ は

$$V_U(t, \alpha) = \frac{\text{時刻}\,(t - \alpha)\,\text{から}\,t\,\text{までの間の更新量}}{\alpha}$$

として算出できる．

例として，$t = 0$ を始めとして，10 秒ごとに 100KB のファイルが生成されるよ

うな更新があったとする．α = 30[sec] とした場合，$t = 45$ の時点での更新速度は

$$V_U(45, 30) = \frac{100 + 100 + 100}{30} = 10[\mathrm{KB/sec}]$$

となる．

7.3.2 更新速度の測定範囲

更新速度は計測時間αの値によって変動する．αが短い場合，更新量の増減に敏感に反応し，V_U がしきい値 V_{th} を前後する回数が増える．そのため，休止モードとスリープモードが切り替わり易くなる．対してαが長い場合，V_U が V_{th} を前後することが少ないため，休止モードとスリープモードが切り替わりにくくなる．

- αを固定長とする場合

  最も基本的なのは，αを固定値として計算を行う場合である．固定長の場合，計測時点 - α より以前に発生した更新の情報は失われてしまうといった問題がある．

- αを可変長とする場合

  αの値が長いか短いかによって同じ更新のパターンの場合でも消費電力が異なる．このため，更新のパターンによってαの値を動的に切り替える方法が考えられる．これによって，更新が多い場合や少ない場合に対して適切な計測時間を用いて計算することができ，固定長と比べてより様々な更新パターンに対応することができる．

- 加重平均法を利用する場合

  加重平均法とは値を平均するのではなく，値の重みを加味して平均する手法である．更新速度の計算の場合，計測時刻からみて直近の更新は重要だとし，重みを高くする．計測時刻と更新時刻が離れれば離れるほど重みは小さくなる．具体的な制御方法としては更新速度を 1 回だけ計算して判断するのでは

なく，複数回更新速度を計算し，それぞれの結果を重み付けして判断する．これによって，固定長と比べて，過去に起こった更新も更新速度に反映させることができるため，より正確な更新速度を計測することができる．

本研究ではプロトタイプシステムの設計であるため，α の値を固定値として与えるものとする．

7.3.3 更新速度の測定タイミング

図 7.2 を見ると更新速度の測定タイミングとして 2 つのタイミングがある．

- ノート PC を停止する時に計算

  ノート PC の停止時には電源状態を決める必要がある．このため，ノート PC が停止する直前で更新速度を計算する．たとえば，同期のためにノート PC を起動し，ノート PC へ転送が行われた後に停止する時や，ユーザからノート PC の停止指示を受け取った時などである．

- 定期的に計算

  上記のタイミングでのみ計算を行う場合，スリープモード時に問題が生じる．スリープモード時に更新がほとんどない場合，ノート PC は停止中であるため，モードを切り替えるタイミングがない．このため，継続的に電力を消費し，消費電力に無駄が生じる．この問題に対処するために，更新速度の計測間隔を T_p[sec] とし，ノート PC が停止している時に，T_p 秒毎にスマートフォン上で更新速度を計算する．図 7.2 のうちの条件分岐 1 の部分が定期的に更新速度を計算している部分である．また，V_U が V_{th} を下回った時，ただちに起動するべきかどうかは，現在蓄積している更新量に依存する．もし蓄積している更新量が最大転送データ量に近い場合，直ちに起動せずに次の更新を待ってから起動したほうが省電力になる場合が考えられる．本研究では簡単化のために V_{th} を下回ったらただちに起動するものとする．

第 8 章

実装

本章では，前章で設計したシステムの実装について述べる．休止モードとスリープモードの切り替えを行うためには，V_{th} を決める必要がある．V_{th} は各電源状態の定常消費電力や，ノート PC における各電源状態の遷移にかかる時間および消費電力量などのパラメータに依存するため，これらの測定を行った．本章の構成は以下の通りである．8.1 節では，V_{th} を決めるためのパラメータを測定する実験について，その測定概要と測定結果について述べる．8.2 節では，提案法を実現する同期プログラムの構成および各処理の実現方法について述べる．最後に，8.3 節では実装の動作確認を行い，正しく動作することを確認した．

8.1 同期に必要な消費電力の測定

8.1.1 測定項目

以下の 4 つの項目について測定を行った．

1. ノート PC における各電源状態の定常的な消費電力
2. ノート PC における各電源状態の遷移にかかる時間と消費電力量
3. ノート PC におけるファイルの受信に要する時間と消費電力量
4. スマートフォンにおけるファイルを送信に要する時間と消費電力量

8.1.2 同期対象

測定に用いたデバイスを表 8.1 に示す.

表 8.1: 測定に用いたデバイス

デバイスの種別	ノート PC	スマートフォン
型番	VAIO VGN-Z91JS	HTC Desire a8181
OS	Windows Vista Home premium	Android OS，v2.2
CPU	Intel Core 2 Duo T9800	Qualcomm Snapdragon ZSD8260
クロック	2.93GHz	1GHz
内部メモリ	RAM 4GB	RAM 572MB
2 次記憶	500GB	(内蔵)1GB+(SD カード)4GB
無線 LAN	IEEE 802.11 a/b/g/n	IEEE 802.11 b/g
Bluetooth	Bluetooth 2.1	Bluetooth 2.1
質量	1.35〜1.57kg	135g

8.1.3 測定方法

- ノート PC

  消費電力の計測機器として 1 秒おきに電力を測定できるワットチェッカーを利用した．測定方法としては，動作の開始から終了までの 1 秒ごとの消費電力の積算値を計算した．また，バッテリへの充電分の電力消費をさけるためにノート PC のバッテリを外し，常時 AC 電源に接続された状態とした．この

状態において手動で電力モードを VAIO 標準モードになるように設定を行った．VAIO 標準モードとは，必要に応じてディスプレイの輝度などを変更する機能である．

- スマートフォン

  消費電力の測定には電流による磁界を測定することによって電流の大きさを測定できるクランプメータを用いた．スマートフォンはバッテリを外すことができないため，バッテリーが満タンな状態でほとんど充電のために供給されない状況で測定を行った．

8.1.4 測定結果

オン状態，休止状態，スリープ状態，オフ状態における定常消費電力の測定結果を表 8.2 に示す．これらの結果から，停止状態の中でスリープ状態のみ電力を消費していることがわかる．

表 8.2: ノート PC における各電源状態の定常消費電力

電源状態	定常消費電力 [J/sec]
オン状態	25.5 ($= P_{ON}$)
スリープ状態	1.0 ($= P_{SLP}$)
休止状態	0 ($= P_{HIB}$)
オフ状態	0

次に，ノート PC における各電源状態の遷移にかかる時間と消費電力量の測定結果を表 8.3 に示す．(1) から (6) は図 4.1 の遷移と対応している．このうち V_{th} の

導出に必要なのは，休止状態からオン状態への遷移にかかる時間 (T_1) と消費電力量 (Q_1)，オン状態から休止状態への遷移にかかる時間 (T_2) と消費電力量 (Q_2)，スリープ状態からオン状態への遷移にかかる時間 (T_3) と消費電力量 (Q_3)，オン状態からスリープ状態への遷移にかかる時間 (T_4) と消費電力量 (Q_4) の合計 8 個のデータである．

表 8.3: ノート PC における各遷移にかかる時間と消費電力量

遷移の種類	消費電力量 [J]	消費時間 [sec]
(1) オン状態からオフ状態	848	29
(2) オフ状態からオン状態	3280	87
(3) オン状態から休止状態	1924 ($= Q_2$)	51 ($= T_2$)
(4) 休止状態からオン状態	2056 ($= Q_1$)	56 ($= T_1$)
(5) オン状態からスリープ状態	307 ($= Q_4$)	10 ($= T_4$)
(6) スリープ状態からオン状態	522 ($= Q_3$)	17 ($= T_3$)
(1) + (2)	4128	116
(3) + (4)	3980	107
(5) + (6)	829	27

(1)+(2)，(3)+(4)，(5)+(6) は各電源状態の遷移にかかる時間と消費電力量の和である．(1)+(2) と (3)+(4) を比較すると，消費時間，消費電力量共に (1)+(2) の方が大きい．また，(3)+(4) と (5)+(6) を比較すると，消費時間，消費電力量共に (5)+(6) の方が (3)+(4) より小さい．このことから，スリープ状態は休止状態と比べて，起動と停止にかかる消費電力量が少ないことがわかる．

次にノート PC のファイルの受信に要する時間と消費電力量を測定した．ファイ

ルの送信用デバイスにはPCを用いた．なお，本研究における転送中の消費電力とは無線LANの消費電力とOSが動作するための消費電力の和であり (図3.5より)，無線LANの消費電力と比べてOSが動作するための消費電力の方がより支配的である．通信メディアとしてはWi-Fi(IEEE 802.11g) とBluetoothを用いて測定し，転送時の単位サイズあたりの消費電力量 (以後，電力効率とする)[J/MB] と転送速度 [byte/sec] を求めた．この結果を表8.4に示す．結果からWi-Fiの平均電力効率と平均転送速度はそれぞれ，約21.8[J/MB] と約1.2×10^6[byte/sec] であり，Blutoothの電力効率と転送速度はそれぞれ，約267.3[J/MB] と約0.15×10^6[byte/sec] であった．Wi-FiとBluetoothの平均電力効率を比較すると，Wi-FiがBluetoothの約10

表 8.4: ノートPCのファイル受信時の電力効率と転送速度

更新量	Wi-Fi		Bluetooth	
	電力効率 [J/MB]	転送速度 [byte/sec]	電力効率 [J/MB]	転送速度 [byte/sec]
101KB	24.7	1.01×10^6	495	0.05×10^6
1.0MB	18.7	1.33×10^6	302	0.16×10^6
5.1MB	20.1	1.30×10^6	138	0.19×10^6
9.7MB	23.6	1.07×10^6	134	0.19×10^6

分の1である．これは，Wi-Fiの方が転送速度が速く，転送時間が短いため，転送中においてより支配的であるOSの消費電力が少なくなるためである．よって，同じ更新量の転送を行う場合にはWi-Fiの方がBluetoothよりも消費電力量が少なくなり，本研究では，通信メディアとしてWi-Fiの方がより適しているといえる．

これらの測定結果と許容待ち時間T_Wが決まれば，V_{th}を定めることができる．許容待ち時間$T_W = 60$[sec]，転送速度$V_C = 1.0 \times 10^6$[byte/sec]，$T_1 = 56$[sec],

$T_3 = 17$[sec], $T_4 = 10$[sec], $Q_1 = 2056$[J], $Q_2 = 1924$[J], $Q_3 = 522$[J], $Q_4 = 307$[J] とすると

$$V_{th} = 1.02 \times 10^3 [\text{byte/sec}]$$

となる．

次に，スマートフォンにおけるファイルの送信に要する時間と消費電力量を測定した．ファイルの受信用デバイスにはPCを用いた．通信メディアにはWi-Fiを用いて測定し，電力効率 [J/MB] と転送速度 [byte/sec] を求めた．

表 8.5 にその結果を示す．表 8.5 からスマートフォンの平均電力効率が約

表 8.5: スマートフォンのファイル送信時の電力効率と転送速度

更新量 [MB]	電力効率 [J/MB]	転送速度 [byte/sec]
1.0	2.0	0.9×10^6
5.1	2.2	0.8×10^6
9.7	2.5	0.7×10^6

2.2[J/MB], 平均転送速度が約 0.8×10^6[byte/sec] であることがわかる．通信メディアをWi-Fiとした場合のスマートフォンとノートPCの電力効率を比較すると，スマートフォンの電力効率はノートPCの電力効率の約10分の1である．このことから，ノートPCの消費電力の占める割合はノートPC側のほうが大きく，スマートフォンよりノートPCの消費電力がより支配的であるといえる．

8.2 提案法の実装

本研究ではスマートフォンとノートPCの2つのデバイスを対象としたが，本実装では，スマートフォンの代わりにLinuxをOSとしたPCを用いた．また，同期

の機能はアプリケーションとして実装した．以後，スマートフォンの代わりに用いたPCのことをデバイスAと呼ぶ．実装に用いた各デバイスの詳細を表8.6に示す．また，デバイスA上で動く同期アプリケーションのソースコードはC言語で約600行であった．対して，ノートPC上で動く同期アプリケーションのソースコードはC言語で約200行であった．

表 8.6: 実装に用いたデバイス

デバイスの種別	デバイスA	ノートPC
型番	NEC VY12AMZR-6	VAIO VGN-Z91JS
OS	Fedora core 15	Fedora core 16
CPUクロック	Intel Core 2 Duo U9300 1.20GHz	Intel Core 2 Duo
内部メモリ	3GB	4GB
ディスク	80GB	500GB
無線LAN	IEEE 802.11 a/b/g/n	IEEE 802.11 b/g/n

次に各処理の実装方法について説明する．

- 更新の検知

  アプリケーションレイヤーで利用することのできるライブラリ関数として，Linux kernel 2.6 で利用できる inotify[24] 関数を用い，指定されたディレクトリ内で，ファイルの生成，修正，削除の際に発行されるシステムコールを捕捉することで更新の検知を行った．

- 起動要求処理

  ノート PC の起動要求には 4.2 節で示した方法の中で，WoWLAN を利用することにより無線経由で実現することができるが，対応機種が限られるため本実装では代わりに有線経由の Wake–On–LAN を用いた．

- 転送処理

  更新情報の転送には rcp コマンドを用いた．また，更新の種類が削除の場合，rsh コマンドを利用して rm コマンドをリモート実行させた．

- デバイス間の通信メディア

  8.1 節での測定実験の結果から通信メディアとしては IEEE 802.11g を用いた．

- 同時更新への対応

  2 つのデバイスからの同時更新を禁止するための方法としては，C 言語で利用できる flock 関数 [23] を用いた．flock 関数はファイルに対してロックの適用・解除を行うものであり，共有ロックと排他ロックの 2 種類がある．このうち本研究ではファイルの参照と編集を禁止する排他ロックを用いた．実現方法としては，まずスマートフォン上に共有ファイルを作っておく．スマートフォンはこの共有ファイルに対して flock() によって排他ロックをかけることで実現している．ノート PC から共有ファイルにアクセスするためには rsh コマンドを用いて flock() をリモート実行している．

8.2.1 プログラムの構成

プログラムの構成について説明する．実装の拡張性を高くするために，処理ごとにスレッド単位で分割して実装した．デバイス A 上ではスレッド A1〜A4 が動作し，ノート PC 上ではスレッド B1，B2 が動作する．

- デバイス A 側の各スレッドの処理

  スレッド A1：検知処理と更新処理

  スレッド A2：定期的な更新速度の計算処理

  スレッド A3：転送処理

  スレッド A4：更新速度計算処理及び停止要求処理

- ノート PC 側の各スレッドの処理

  スレッド B1：起動処理，停止処理

  スレッド B2：検知処理，転送処理

8.3 実装の動作確認

4 つのスレッド A1〜A4 をデバイス A 上で，2 つのスレッド B1〜B2 をノート PC 上で動かし，実装したシステムの動作確認を行った．同期対象のディレクトリをそれぞれ D_A，D_N とする．また，しきい値 $V_{th} = 1.02 \times 10^3$[byte/sec]，更新速度の計測時間 $\alpha = 600$[sec]，許容待ち時間 $T_W = 60$[sec] とする．

- ノート PC が起動中の場合：

  デバイス A，ノート PC 共にオン状態とする．この時，同期は双方向に行われる．まず，デバイス A のディレクトリ D_A 上でファイル A〜E，ノート PC のディレクトリ D_N 上でファイル F〜J を生成した．ファイル A〜J のサイズ

は 1MB とする．生成終了後，2 つのデバイス間で正しく転送が行われ，D_A および D_N 上ではファイル A〜J があることを確認した．次に，デバイス A 上では，ファイル A〜E を削除し，ノート PC 上ではファイル F〜J を削除した．削除終了後，D_A と D_N 共にディレクトリが空であることを確認した．

- ノート PC が停止中の場合：

  初期状態としてノート PC が休止モードである場合の動作確認を行った．本実装において休止モードにおけるノート PC の最大転送データ量 D_H は 3.9MB である．ファイル A，B，C，D の順にデバイス A のディレクトリ D_A 上にファイルを生成したところ，ファイル D の生成後に最大転送データ量 D_H を超えたため，Wake–on–LAN によりノート PC が起動することを確認した．その後，スマートフォンからノート PC へとファイル A〜D が転送され，転送終了後に更新速度が計算された．ファイル A〜D の生成はいずれも 1 分以内の間隔で行ったため，更新速度は V_{th} を上回り，スリープ状態で停止するのを確認した．その後，定期的な更新速度の計算による動作を確認するために，約 10 分間，更新を行わなかったところ，デバイス A からの Wake–on–LAN により，ノート PC がスリープ状態から起動するのを確認し，その後休止状態にて停止するのを確認した．

第 9 章

評価

本章では，提案法のアルゴリズムをシミュレーションにより評価し，その結果について考察する．

評価方法として，様々な量の更新を入力として与え，休止モードと，スリープモードと，提案法の 3 つについて同期にかかる消費電力量の比較を行った．また，検知処理を行う代わりに，予め作成した更新の時間とファイルサイズを指定する疑似データを用いた．また，各電源状態の遷移にかかる消費電力量および転送にかかる消費電力量については，8.1 節で測定したデータを用いて計算した．具体的には，休止状態からオン状態への遷移にかかる消費電力量 $Q_1 = 2056$[J]，オン状態から休止状態への遷移にかかる消費電力量 $Q_2 = 1924$[J]，スリープ状態からオン状態への遷移にかかる消費電力量 $Q_3 = 522$[J]，オン状態からスリープ状態への遷移にかかる消費電力量 $Q_4 = 307$[J]，オン状態の定常消費電力 $P_{ON} = 25.5$[J/sec]，転送速度 $V_C = 1.0 \times 10^6$[byte/sec] を用いた．さらに，切り替えのしきい値 $V_{th} = 1.02 \times 10^3$[byte/sec]，許容待ち時間 $T_W = 60$[sec]，更新速度の計測時間 $\alpha = 300$[sec]，計測タイミング $T_p = 60$[sec] とした．なお，初期状態として，ノート PC は停止しているものとする．

更新のパターンとして，4 つのケースについて考える．

- ユースケース 1(更新量が多い場合)

  2 時間の間にファイルサイズが 1KB〜10MB のファイルが生成されるような更新を，ランダムなタイミングで 20 回発生するようなパターンを図 9.1 に

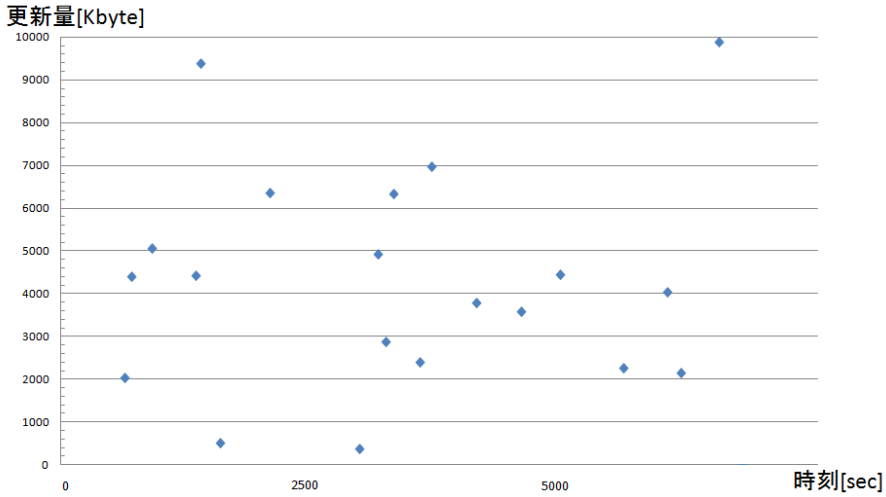

図 9.1: ユースケース 1：更新のパターン

示す.

この時の，休止モード，スリープモード，提案法の 3 つの場合について，消費電力量の比較結果を表 9.1 に示す.

表 9.1: ユースケース 1 の比較結果

	起動回数	消費電力量 [J]
休止モード	12	49957.0
スリープモード	1	10152.3
提案法	1	10152.3

ユースケース 1 では，1 つ 1 つの更新量が大きく，転送に伴う起動と停止の回数が多いことから，休止モードの消費電力量が大きくなる．提案法の消費電力量はスリープモードの消費電力量と等しくなり，休止モードの消費電力量と比べて小さくなる.

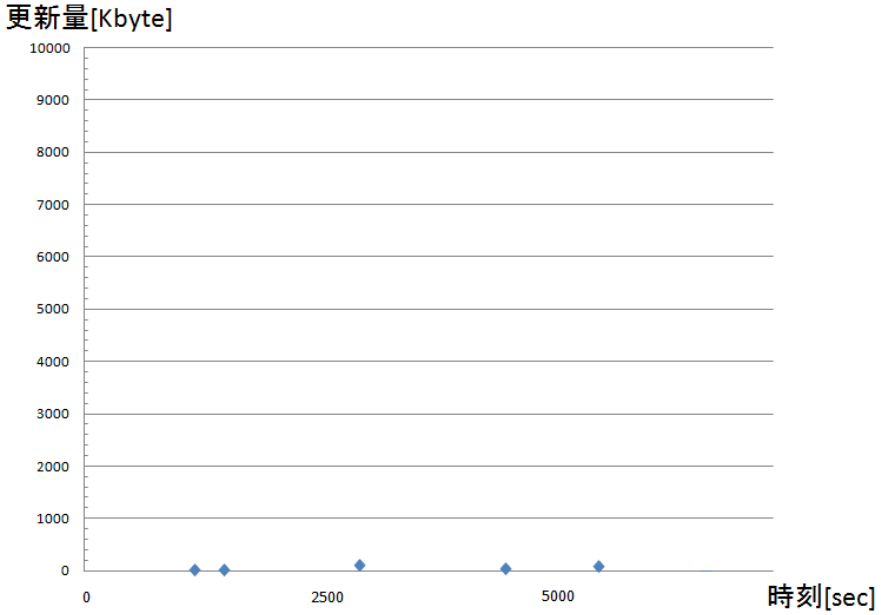

図 9.2: ユースケース 2：更新のパターン

- ユースケース 2(更新がほとんどない場合)

  2 時間の間に，ファイルサイズが 1KB〜100KB のファイルが生成されるような更新をランダムなタイミングで 5 回発生するようなパターンを図 9.2 に示す.

  この時の，休止モード，スリープモード，提案法の 3 つの場合について，消費電力量の比較結果を表 9.2 に示す．ユースケース 2 では，ユースケース 1

表 9.2: ユースケース 2 の比較結果

	起動回数	消費電力量 [J]
休止モード	0	0
スリープモード	0	7206.2
提案法	0	0

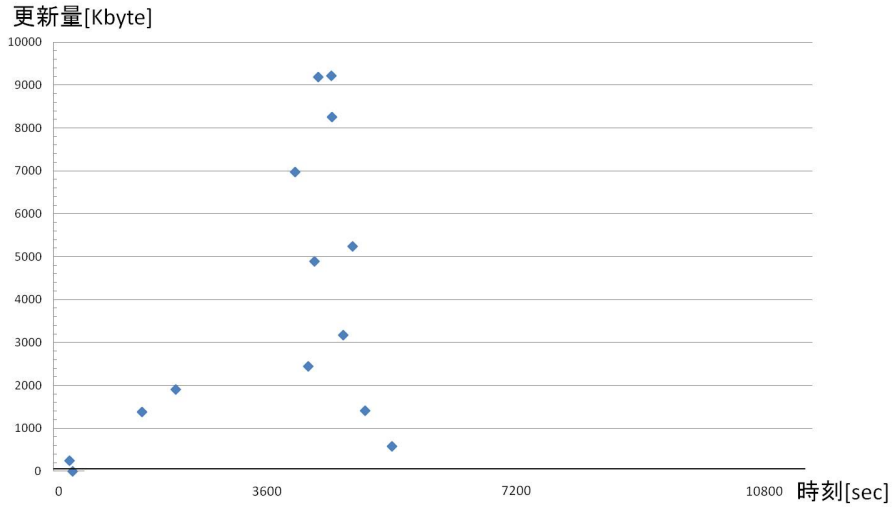

図 9.3: ユースケース 3：更新のパターン

とは対照的に転送に伴う起動と停止が発生しない．スリープモードは停止中にメモリへの給電分の電力を消費することから，最も消費電力量が大きくなる．提案法の消費電力量と休止モードの消費電力量は等しくなり，スリープモードの消費電力量と比べて小さくなる．

- ユースケース 3(ユースケース 1 と 2 の混在)

  3 つ目はユースケース 1 とユースケース 2 が混在する場合つまり，ある一定の時間だけ頻繁に更新が発生する場合である．3 時間うち，開始から 1 時間経過するまでの間はファイルサイズが 1KB〜100KB のファイルが生成されるような更新がランダムなタイミングで 5 回発生するようなパターンであるとする．開始から 1 時間経過した時点から 30 分間はファイルサイズが 1KB〜10MB のファイルが生成されるような更新がランダムなタイミング 10 回発生するようなパターンであるとする．開始から 1 時間半経過した時点から終了までの 1 時間半の間は更新がないものとする．この場合の更新のパターンを図 9.3 に示す．

また，休止モード，スリープモード，提案法の 3 つの場合についての消費電力量の比較結果を表 9.3 に示す．このような場合，更新が多い時間帯ではスリープモードがより省電力であり，更新がほとんどない時間帯では休止モードがより省電力となる．このため，提案法によるモードの切り替えの効果が大きくなり，提案法は休止モード，スリープモードと比較して消費電力量が小さくなる．

表 9.3: ユースケース 3 の比較結果

	起動回数	消費電力量 [J]
休止モード	6	25279.2
スリープモード	1	12956.7
提案法	3	9826.9

- ユースケース 4(更新のパターンが頻繁に入れ替わる場合)

  4 つ目はユースケース 3 の更新量が多い場合とほとんどない場合が頻繁に入れ替わるようなパターンである．具体的には，サイズが 2.2MB のファイルが生成されるような更新が 600 秒毎に発生するようなパターンを考える．この時の休止モード，スリープモード，提案法の 3 つの場合について，消費電力量の比較結果を表 9.4 に示す．この場合，休止モード，スリープモードと比較して，提案法は消費電力量が最も大きくなる．本研究では，スリープモード時に更新速度が V_{th} を下回った場合，休止モードへ遷移するために起動するとしたが，更新のパターンが頻繁に入れ替わる場合，スリープモードから休止モードへの切り替えのための起動と停止が頻繁に起こる．スリープモードから休止モードへの切り替えを行うためには 1 度起動してから停止させる必要がある．このため，スリープ状態からオン状態への遷移にかかる消費電

表 9.4: ユースケース 4 の比較結果

	起動回数	消費電力量 [J]
休止モード	5	20517.1
スリープモード	0	7817.1
提案法	10	27344.1

力量＋オン状態から休止状態への遷移にかかる消費電力量分がオーバヘッドとなる．よって，このような場合にはスリープモードから休止モードへの切り替えを制限する必要がある．たとえば，更新速度がしきい値 V_{th} よりも低い状態から高い状態へ遷移してから，次に V_{th} よりも高い状態から低い状態に遷移するまでの間隔が短い場合には，α を引き延ばし，切り替えが起こらないようにする．このように起動と停止の回数を減らすことで，提案法での消費電力量を減らすことが期待できる．

以上の 4 つのユースケースにおける消費電力量の比較結果から，ユースケース 1 からユースケース 3 において，提案法の消費電力量が休止モード，スリープモードと比べてより少なくできることが確認できた．問題となるのは更新のパターンが頻繁に入れ替わるような場合であり，これらに対処するためには更新のパターンによって α を変動させる必要がある．

第 10 章

考察

本章では，本研究で扱っていない同期に関連する技術や，本研究を発展させるためのさらなる省電力化の方法について考察する．

ディスク容量の制限

通常，ノート PC と比べてスマートフォンの方がディスク容量が小さいため，完全に同期することはできない．これらに対処するための 1 つの方法としてはファイルに優先度（ファイルの新しさ，アクセス頻度，通信手段）を付けて選択的に同期することで解決することができる．

また，ディスク容量は年々飛躍的に増加しており，microSDHC カードなどでは 30GB 以上のデータを保存することができる．このため，本研究ではスマートフォンのディスク容量がノート PC より小さいことは考慮しないものとし，ノート PC で発生した更新はすべてスマートフォン側へ転送されるものとする．

バックアップについて

同期対象のディレクトリ内で誤ってファイルを消してしまった場合，データが損失してしまう恐れがある．誤操作時にファイルを復元するためにはあらかじめバックアップを取っておく方法が挙げられる．Dropbox[4] ではリビジョン管理によるバックアップ機能をサポートしている．

本研究において，ローカル間でバックアップを取る場合，どちらかのデバイスに復元可能となるデータを保存しておくことで，誤操作時にデータを復元することが可能になる．ディスク容量の観点からノート PC に保存することが望ましい．しかし，ノート PC 上のディスクにバックアップを取る場合，ノート PC のディスク自体が読めない等の不具合によりバックアップが復元できない問題がある．この問題に対処するためにはローカルデバイス以外の第 3 のデバイスにデータを保存しておく必要がある．本研究では簡単化のため，バックアップ機能は扱わないものとした．

同期対象となるデータ

データの種類によって待ち時間の長さは異なる．

- **シーケンシャルアクセスが可能なデータ**
  シーケンシャルにアクセスされるようなデータの場合，転送の終了を待つことなく，データの一部が転送された状態で，データを利用することができるため，同期にかかる待ち時間はあまり生じなくなる．例えば動画や画像などのコンテンツである．

- **シーケンシャルアクセスができないデータ**
  最初から最後までのデータを 1 度に必要とするようなデータの場合，転送が終了するまでデータを利用することができない．このため，同期にかかる待ち時間は大きくなる．例えば，インターネット経由での画像のダウンロード，カメラ機能による画像の撮影データ，Web のキャッシュデータなどがある．

更新時のデータの配信方法

シーケンシャルアクセスが可能なデータの更新時には，データを受信しながら同時にアクセスすることが可能となる．この方式をストリーミングと呼ぶ．ストリー

ミングを採用している動画配信サービスとしては youtube[25] や google 動画 [26] などがある．

今日では，モバイル端末からのデータ通信によるネットワーク帯域の利用が急速に拡大しており，この問題に対処するために WAN のかわりに LAN の利用を促したり，帯域制限する機種も増えている．このため，動画や音楽の共有サービスとしても，今後は帯域の利用を減らすために，ストリーミングではなく，ダウンロード (蓄積型) 主体のサービスが増えていくことが予想される．日本では，2012 年 4 月からスマートフォン向けの動画配信サービスとして NOTTV[20] が開始される予定である．これはアナログテレビの VHF 帯を再利用したものであり，リアルタイムでの視聴が可能なほか，見たいコンテンツをダウンロードすることも可能である．このようなダウンロード型に対応したサービスの普及により，データを同期する際の待ち時間が問題となる．

クライアントサーバ間の同期

本研究では 2 つのデバイス間で直接同期を行うことを前提とした．仮に，クライアントサーバ間においても同様に提案法を適用することができる．サーバは常時起動しているため，待ち受け用としてスマートフォンの代わりに利用し，起動要求を出す役割を任せることができる．またスマートフォンを持ち歩き，ノート PC が自宅に置いてある場合，出先からスマートフォンでダウンロードしたデータはサーバを利用してノート PC に同期されるため，遠隔地からでも同期を行うことができ，帰宅後に許容待ち時間内にデータにアクセスすることが可能となる．

NILFS の利用

NILFS[21] とは，スナップショットを自動かつ連続的に取得するようなログ構造化ファイルシステムである．具体的には過去の複数の地点でのファイルシステムの状態を保存しておくことができる．転送のためにノート PC を起動した際に NILFS

ファイルシステムによって直接バッファにログを取っておく．これによって，転送処理やバックアップ処理などを容易に実現することができる．

起動に伴う発熱の問題

持ち運び中にノート PC を自動で起動，停止する場合，発熱の問題や，振動等により HDD が回転することにより故障等が起こる心配があるが，これには例えば保冷機能のあるバッグを使う方法や，CPU の性能向上による低熱化を期待する．また，SSD など書き込みの際に回転を伴わないため，故障する可能性は低くなる．

アプリケーションごとに異なる点

第 1 章で様々なアプリケーションの同期の例を示したが，これらのアプリケーションの違いによって，同期の測定範囲 α を変化させることが考えられる．例えばメールなどのデータ容量が小さいアプリケーションの場合には，更新量が少なく，起動と停止のオーバヘッドがないため，α の値は長めとする．逆に音楽動画などのデータ容量が大きい場合、α は短めにするなどの制御が考えられる．

実装のレイヤー

本研究ではアプリケーションにより提案法を実装した．このため，同期アプリケーションを利用するためにはノート PC に電源を投入した後に OS を動かす必要がある．OS を動かすためには様々な処理が必要であり，起動と停止にかかる時間，消費電力量は大きくなる．このため，アプリケーションではなく，ファイルシステムやシステムソフトウェア等で実装することで，アプリケーションのように同期を行うために OS を起動する必要がなくなり，さらなる省電力化ができると考えられる．

- **ファイルシステムと軽量** OS

  同期機能のみを動かすようなファイルシステムとその下でほとんど起動に時間がかからない OS を動かすことで，起動と停止のオーバヘッドを減らすこと

ができる．また，アプリケーションによる実装と比べて，発生する前に更新を検出することが容易であるため，排他制御などの実装が容易になる．ファイルシステムの構築には一般的にカーネルコードに手を加える必要があるが，FUSE[22] のような技術を用いることでユーザ空間で独自のファイルシステムを構築でき，実装の手間を省くことができる．

- BIOS+**ローダの修正**

  本研究では WoWLAN 経由での休止状態からの復帰の場合，hybernation.sys ファイルを用いて実行状態をメモリに展開した後に同期を行っているが，ユーザはノート PC を利用しないため，sys ファイルを読み込む必要はない．代わりに BIOS やローダを修正し，sys ファイルを読み込む前にネットワーク上からファイルをハードディスクに書き込むような処理を行うことで，同期にかかる時間が軽減し，省電力化が期待できる．

- **同期機能の仮想アプライアンス化**

  同期機能を仮想アプライアンスとして実現する方法が考えられる．この場合，ファイルシステムによる実装と同じように，同期の際に OS を動かす必要がないため，起動と停止のオーバヘッドを少なくすることが期待できる．また，仮想アプライアンス上で動くホスト OS が同期を意識する必要がなくなるといった利点も挙げられる．

転送速度について

第 5 章において，転送速度がより早い技術として WiGig について説明した．このように，転送速度がより早いメディアを使う場合，V_{th} が大きくなるため，休止モードがより省電力となる場合が多くなる．一方で，転送速度の増加と共に，ユーザのデータ通信の量も増えることが見込まれている [8]．データ通信の量が増加すれば，起動と停止の回数が増えるため，起動と停止に伴う消費電力量の小さいスリープ

モードがより省電力となる．このため，転送速度が増えたとしても，休止モードとスリープモードの切り替えの必要性はあるといえる．

第 11 章

おわりに

本研究では，モバイルデバイス間のファイル同期においてユーザが電源を投入してからの待ち時間を一定の範囲内に抑えつつ，消費電力を削減する同期方法を提案した．提案法では，まずユーザが電源投入した後に許容できる待ち時間を定め，この待ち時間をオーバーしないことを条件として，転送に伴う停止デバイスの起動と停止の回数を減らすことで消費電力を削減させた．また，ノート PC の停止中の電源状態 (休止状態とスリープ状態) の違いに着目すると，単位時間あたりの更新量の違いによって，各電源状態において同期に必要な消費電力は異なる．これらの各電源状態に基づいた 2 つのモードとして休止モードとスリープモードを考え，これらの 2 つのモードを適切に切り替えることにより，更なる省電力化を図った．また，提案法を実現するシステムを設計し，プロトタイプシステムを実装した．そして，実装したシステムを動作させ，正しく動くことを確認した．また，評価としてシミュレーションにより休止モードのみとスリープモードのみと提案法の 3 つの場合について同期にかかる消費電力量を比較した．結果として，消費電力量が削減できることを確認した．

今後の課題としては，システムの実用化が挙げられる．本研究ではプロトタイプ実装として，スマートフォンの代わりに PC を用いて実装したが，実用化のためにはスマートフォン上に実装し，動作を確認する必要がある．また，スマートフォン上で実装したシステムを用いて，実際に通勤，通学の際の同期にかかる消費電力量を測定する評価実験を行い，提案法の効果を確認することが必要である．また，

改善点として，実装ではV_{th}を固定値として用いたが，実際のモバイル環境では場所や電波状況などにより，転送速度V_Cは変化する．V_{th}はV_Cの関数であるため，スマートフォン側で定期的に転送速度V_Cを計算し，常に最適なしきい値V_{th}を用いてシステムを動かすことで，より正確なV_{th}の値を用いて，モードの切り替えの判断を行うことが可能となる．

謝辞

本研究を遂行するにあたり，いろいろな方々にお世話になりました．まず，指導教員の鶴岡先生には，お忙しい中，日頃より研究の方針等について多大な御助言と御指導を頂きました．厚く御礼申し上げます．また，ノート PC や電力測定機器などの実験器材を貸していただいたことを重ねて御礼申し上げます．

また，多田先生，佐藤先生には，講座内のゼミでの議論だけでなく，研究会発表の共著者にもなっていただき，研究を進める上で貴重な意見や助言等を頂きました．ここに感謝の意を表します．

そして本研究が行えた事は，鶴岡研究室をはじめとした，同じ講座内の学生諸氏の方々の助言があったおかげであります．最後にこれらの皆さんに感謝致します．

参考文献

[1] Niranjan Balasubramanian, Aruna Balasubramanian and Arun Venkataramani: “Energy consumption in mobile phones: a measurement study and implications for network applications,” Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference, pp . 280–293, 2009.

[2] Fahad R. Dogar, Peter Steenkiste and Konstantina Papagiannaki: “Catnap: exploiting high bandwidth wireless interfaces to save energy for mobile devices,” Proceedings of the 8th international conference on Mobile systems, applications, and services, pp . 107–122, 2010.

[3] Hossein Falaki, Dimitrios Lymberopoulos, Ratul Mahajan, Srikanth Kandula, Deborah Estrin: “A first look at traffic on smartphones,” Proceedings of the 10th annual conference on Internet measurement, pp . 281–287, 2010.

[4] “ Dropbox Simply your life, ”http://dropbox.com.

[5] “ Windows Live Mesh 2011, ”http://explore.live.com/windows-live-mesh.

[6] “ iCloud, ”http://www.apple.com/icloud/.

[7] “ ACPI : Advanced Configuration & Power Interface, ”http://www.acpi.info.

[8] “ Cisco Visual Networking Index：全世界のモバイル データ トラフィックの予測 2010 〜 2015 アップデート ”, http://www.cisco.com/web/JP/solution/isp/ipngn/literature/pdf/white_paper_c11-520862.pdf.

[9] “ Wake-On-Lan, ”http://en.wikipedia.org/wiki/Wake-on-LAN.

[10] “ Intel(R) Centrino(R) Mobile Technology Wake on Wireless LAN (WoWLAN) Feature Technical Brief, ”http://application-notes.digchip.com/027/27-45534.pdf.

[11] “ hybrid sleep, ”http://en.wikipedia.org/wiki/Hybrid_sleep#Hybrid_sleep.

[12] “ APM:Linux Ecology-HOWTO, ”http://tldp.org/HOWTO/Ecology-HOWTO/ecology-howto-power-management.html

[13] “ CCCP Codec software, ”http://www.cccp-project.net/.

[14] “ IEEE 802.11TM WIRELESS LOCAL AREA NETWORKS, ”http://www.ieee802.org/11/.

[15] “ Wi-Fi Direct Alliance, ”http://www.wi-fi.org/discover-and-learn/wi-fi-direct%E2%83%A2.

[16] “ IEEE 802.15 Working Group for WPAN, ”http://www.ieee802.org/15/.

[17] “ Infread Data Association : IrDA, ”http://www.irda.org/.

[18] “ International Telecommunication Union About 3G Technology, ”http://www.itu.int/osg/spu

[19] “ Wireless Gigabit Alliance, ” http://wirelessgigabitalliance.org/.

[20] mmbi 株式会社, “ nottv ”, http://mmbi.co.jp/.

[21] “ NILFS Continuous Snapshotting Filesystem for Linux, ”http://www.nilfs.org/en/.

[22] ”Filesystem in user space”, http://fuse.sourceforge.net/

[23] flock(2), Linux Programmer’s Manual.

[24] inotify(2), Linux Programmer’s Manual.

[25] “ youtube, ”http://www.youtube.com/.

[26] “ Google Videos, ” http://www.google.com/videohp.

[27] 高見澤拓郎，鶴岡行雄, 佐藤喬, 多田好克,“ モバイルデバイス間における省電力を考慮した同期について ”, 信学技報, vol. 110, no. 376, pp . 43–48, 2011.