Vol.2011-CDS-1 No.11
2011/6/16

ネットワークサービス継続のためのサブシステム開発とアプリ実行システムへの応用

二村和明[†]　伊藤栄信[†]　中村洋介[†]　藤野信次[†]
三浦隆文[††]

PC がスリープ状態であっても動作するサブシステムを導入することで、継続的なネットワークサービス等のメリットを享受できる。しかし、その実現のために、通信モジュール内に変更を加えるアプローチをとった場合、新しい通信モジュールが出てくる度にその変更を実装し直す作業が必要になる。そこで本稿では、汎用 WWAN モジュールを使用し、かつ PC 等のホストシステムへのインパクトが無い、独立した常時ネットワークサービスが可能なサブシステムのアーキテクチャを提案する。そして実際にプロトタイプを作り、フィールド評価した結果から実用的な手法であることを示す。また、このサブシステムを利用したサービス例としてコンシューマ PC 等を利用しやすくするアプリ実行システムを提案し、HTML5 アプリケーションと Web ブラウザを使って実装・動作させた結果を示す。

Development of Transparent Network Sub System for Continuous Network Service and Extension to Application Execution System

KAZUAKI NIMURA[†]　HIDENOBU ITO[†]
YOUSUKE NAKAMURA[†]　NOBUTSUGU FUJINO[†]
TAKAFUMI MIURA[††]

By adding a sub system that provides continuous network service to a host system such as PC, even while the host system is sleeping it can achieve smarter use of the system and reducing energy use. When considering the design of the sub system, it should be independent of the network device in terms of development cost because once modifications were made that were dependent on a network device this would lead to much arduous extra work being needed if we wished to use new network devices and keep the modification. In this paper, we propose a unique architecture for the sub system that does not require any modification to wireless network devices and host system. We hence prototype and evaluate it using actual 3G network. In addition we propose a system that can attribute to improve the usability on consumer PC etc. Then we prototyped the system by using HTML5 applications, Web browser, and the sub system.

1. はじめに *

ユーザが PC・タブレット・スマートフォンなどのコンピューティングデバイスの前で直接指示しなくとも、常時ネットワークと接続することでリモートから提供するサービスが増えており(図 1)、このための技術開発も進んでいる。

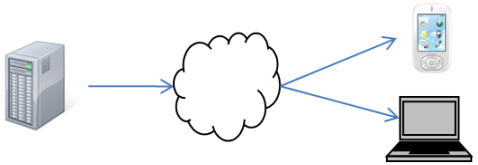

図 1　ネットワークサービス
Figure 1　Network Service.

例えば、携帯電話では i コンシェル[1]のような常時ネットワーク接続を前提にしたようなサービスが広く使われている。これは端末が待ち受け状態にあっても利用できる。一方で PC においては、一般的にオン状態においてのみ遠隔サービスを受けることが可能である。このため夜間も PC を点けっぱなしにしておく場合があるとの指摘がある[2]。これを避けるために、スリープ状態にある PC を、Wake on LAN や Wake on Wireless LAN のような遠隔から起こす機能を利用することが考えられる。さらに、場所を選ばずどこでもサービスを受けるためには、WWAN (Wireless Wide Area Network) のような通信網を利用することが有効である。具体的には例えば、PC の盗難対策である Intel Anti-Theft 3.0 [3]や Fujitsu CLEARSURE [4]がある。これらは、通信モジュールに機能変更を加えるアプローチをとっている。しかし、一般的に通信技術の進展が早いために、このようなアプローチでは新しい通信モジュールが出てくる度に専用機能を組み込む作業が必要になることが予想される。また WWAN においては、電波法により、修正内容によって再認証を取得することが必要になる場合も考えられる。これらは労力を伴うものであり、開発コストへのインパクトがある。このため、汎用の通信モジュールを使って、外部に置いたサブシステムにより機能を実現するアプローチをとることで、コストメリットを追求することが理想的である。

そこで、本稿では、汎用の WWAN モジュールを使用しても通信制御が可能で、かつ PC などのホストシステムへのインパクトが無いサブシステムのアーキテクチャを検討・提案する。そして、そのプロトタイプを開発すると共に、フィールドにおける

* [†] (株)富士通研究所
Fujitsu Laboratories Ltd.
[††] 富士通株式会社
Fujitsu Limited

パフォーマンス検証を行うことで実現可能性を確認する。その結果 70%以上の電力低減が期待でき、かつ通信スループット実測からも実用的であることが分かった。

また、このサブシステム応用したサービス例として、共有ストレージを用いてホストがオフ・スリープ時にアプリケーションを配信・実行する為の端末側の仕組みを提案する。これは従来の Windows よりもカジュアルなアプリケーション実行を可能にし、コンシューマ PC 等を利用しやすくできる。実装ではブラウザ[5]で HTML5[6]アプリを実行させるようにした。これにより新しい使い方ができることを示す。

本稿の以下の構成を示す。2.では関連研究に触れる。3.では、提案アーキテクチャを示し、4.ではインプリメンテーション詳細を示す。5.ではホスト対して透過性を保ったサブシステムであることを実環境における評価結果を示す。6.では、サブシステムを使ったサービス提案と実装および動作確認を示す。7.でまとめる。

2. 関連研究

Proxying [7][8][9]は、PC がスリープ状態にある場合にも、ネットワークサービスを継続するための手法について検討している。これに先立ち TCP コネクションのような Full network presence を保つために、サブシステムにあたるものが何を保持すれば良いのか調査が行われている[10][11]。Somniloquy[12]は、ネットワークサービスを継続する Proxying の一つのアーキテクチャを提示すると共に、LAN や WLAN デバイスを使って、IM(Instant Messaging)に対する Network presence を維持するなどのアプリによる評価を行っている。

標準化では、ENERGY STAR Version 5.0[13]と ECMA (European Computer Manufacturers Association) 393[14]が Proxying を扱っている。ENERGY STAR では、これまでの off や sleep 状態に加えてネットワーク接続を提供する proxying という状態を追加定義している。ECMA 393 では LAN と WLAN に関して具体的にサブシステム等に渡すべきパラメータ等を定義している。製品としては、Apple の製品[15]に mDNS[16]を用いて Wake on WLAN を扱う機能が組み込まれている。

しかし、より広範囲なサービスを受けるために必要な PC のモバイル運用に適用可能な WWAN を使った手法に関して研究がなされていない。そこで、このエリアでの課題をまとめ、解決のための仕組みの検討および実証を行うことにした。

3. アーキテクチャ

以下では課題解決のための提案アーキテクチャについて示す。

3.1 課題と要件

解決すべき課題は、「ネットワークサービスを提供するサブシステムを導入しても、通信モジュールとホストシステムにインパクトを与えないこと」である。これをサブシステムの要件に落とし込むと以下のようになる。サブシステムは、

i) ネットワークデバイスに対して独立して動作しネットワークを制御できること。
ii) ホストシステムが従来通り通信が利用できるように透過性を提供すること。
iii) ネットワークサービスを提供するアプリを組み込めること。
iv) ホストシステムへの通知やデータの授受ができること。

また、サブシステムと連携して動作するホストシステムの要件は以下のようになる。ホストシステムは、

v) サブシステムからの通知でホストシステムのパワーステートを変更できること。
vi) サブシステムによって置かれるデータを判断し、必要な情報を取り込めること。

3.2 アーキテクチャ

サブシステムを伴うシステムの大枠構成として２つが考えられる。一つは、ネットワークデバイスの制御をホストシステムのパワーステートに応じて、ホストシステムとサブシステムの間でスイッチさせる構成。もう一つは、ネットワークデバイスの制御を常にサブシステムに行わせるために、サブシステムをネットワークデバイスとホストシステムの間に配置する構成(図 2)である。上記要件を満たす為には後者を用いる必要がある。このようにすることで、ホストシステムに依存せず、サブシステムを動作させることができる。すなわち例えホストシステムが何らかの理由で動作しなくなったとしても、サブシステムはサービスを継続することができるメリットがある。

また、上述のそれぞれの要件を実現するため、サブシステムおよびホストシステムは図 2 に示した機能を組み込む。

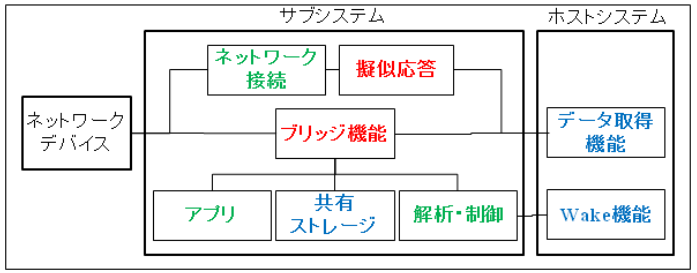

図 2 サブシステムとホストシステムのアーキテクチャ
Figure 2 Architecture of the sub system and host system.

要件に対応する機能概要は以下の通りである。

I) ネットワーク接続機能は、ネットワーク接続を司る。
II) 擬似応答機能は、ホストシステムの要求に擬似応答し、ネットワーク接続状態を通知する。ブリッジ機能は、ネットワーク通信をホストシステムに提供する。
III) 解析・制御機能は、メッセージの解析を行い、必要に応じてホストシステムに Wake を通知する。アプリケーションは、ネットワークを利用したサービスを提供する。
IV) 共有ストレージは、ホストシステムに対してデータを提供する。
V) Wake 機能は、サブシステムの要求に応じてホストシステムを電源オンの状態にする。
VI) データ取得機能は、共有ストレージに置かれるデータを監視・管理する。

ここで、前半 3 つは独立したサブシステムを構築するために必要な基本機能であり、後半 3 つはホストシステムとの連携を行うための拡張機能である。

3.3 3 つのステート

以下では、ホストシステムの動作に応じてサブシステムがとる以下 3 つのステートに関する定義を示す。このステートの何れかにおいて前述の機能が動作する。

SS1: ネットワーク接続、解析・制御、アプリが機能している状態。これらは、サブシステムに電源供給がされている限り、どのステートにあっても機能する共通機能である。この状態になるのはホストシステムがオフ・スリープ時の場合、すなわちサブシステムのみが機能している場合である。

SS2: SS1 に加えて、共有ストレージ、Wake 機能、データ取得機能が機能している状態。この状態になるのは、ホストシステムが起動してストレージアクセスを提供する場合である。共有ストレージ機能は SS3 の状態においても機能する。

SS3: SS2 に加えて、ブリッジ機能、疑似応答が動作している状態。これらは SS3 の状態においてのみ機能する。この状態になるのは、ホストシステムがネットワーク接続を駆動した場合である。

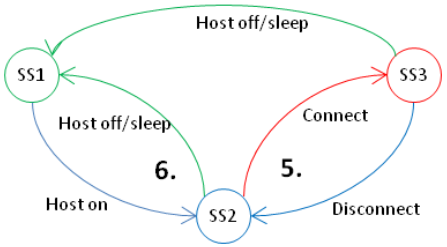

図 3 状態遷移
Figure 3 State transition.

これらの状態遷移を示したものが図 3 である。サブシステムへの負荷は、SS1 より SS2、さらに SS3 に行くに従い大きくなる。

また、次章以降に関して、5.で述べるホスト透過性の評価は、SS2 と SS3 の間の状態遷移を扱っており、6.で述べるサブシステムを使ったサービスでは、SS1 と SS2 の間の状態遷移を扱っている。

4. 実装

ここではサブシステムおよびホストシステムの実装方法について述べる。提案アーキテクチャを実現する為に最低限必要となるハードウエア構成は、プログラムを実装可能な CPU、ホストシステムとアクセス可能なインタフェース、そして、ネットワークデバイスとアクセス可能なインタフェースである。

そこで、具体的なハードウエアとして、Keil MCB2388 評価ボードを用いることにした。この評価ボードは、NXP Semiconductor 製の ARM7 ファミリーの CPU(LPC2388 ARM7TDMI-S)を搭載し、最大周波数 72[MHz]で動作、512KB のオンチップ・フラッシュ・メモリと、64KB の CPU 用 SRAM、USB (Universal Serial Bus)が利用する 16KB の DMA (Direct Memory Access)用 SRAM、1 つの USB DEVICE と 1 つの USB OTG/Host、SD (Secure Digital)メモリカード用インタフェースなどを備える。また、オペレーティングシステムとして、同時に複数の機能を実行可能な RTX Real-Time Operating System を備える。

ソフトウエア開発環境としては、MDK-ARM Microcontroller Development Kit を用いた。これは、TCP Networking Suite や、 USB HOST とデバイスのスタック、その他のライブラリが利用できる。これらを用いてアーキテクチャを実装した。

MCB2388 の USB HOST はネットワークデバイスと接続し、USB DEVICE はホストシステムと接続する。後者はバスパワーによるサブシステムとネットワークデバイスへの電力供給も兼ねている。

また、サブシステムの GPIO とホストシステムを接続し、GPIO が駆動されると、ホストシステムに電源が入るように配線した。アーキテクチャ検証のため開発したソフトウエアは、オンチップのフラッシュメモリに置かれ、ブート時に SRAM に読み込まれて実行される。

4.1 サブシステムの各構成要素の実装

以下ではサブシステムの各構成要素の実装について記述する。図 4 はサブシステムのソフトウエア構成を示している。このうち擬似応答には、ネットワーク接続に対するものと、USB Descriptor に対するものの 2 つある。

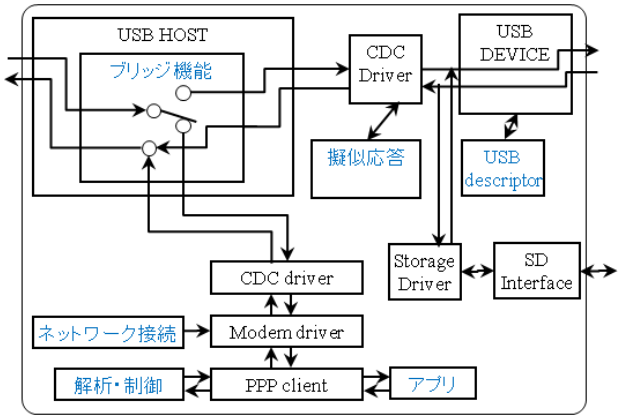

図 4　サブシステムの構成
Figure 4　Structure of sub system.

4.1.1 ネットワーク接続と擬似応答

ここではサブシステムのネットワーク接続と、ホストシステムがネットワーク接続する際のサブシステムの擬似応答について記述する。サブシステムによるネットワーク接続は、サブシステムの起動時に接続プロトコルが駆動されるように実装し、この時、3GPP TS27.060 [17]に”Example command sequences for dial-compatibility mode”として例示された標準の AT および PPP (Point-to-Point-Protocol)コマンドシーケンスの送受を通して接続を行う。これを図 5 上部に簡略表示している。これにより接続が確立されると、サブシステムのアプリケーションとネットワーク間で通信が利用できるようになる。仮にネットワークが何らかの理由で切断された場合、それを検知して再接続する機能を持たせることで、常にネットワーク接続が維持されるように実装している。

擬似応答は、ホストシステムからネットワーク接続あるいは切断が発行された場合に機能する(図 5)。ホストシステムが発行する標準の AT および PPP コマンドに対して、サブシステムが擬似応答を行う。具体的には、ホストシステムのネットワーク接続ソフトウエアが、標準コマンドを発行した場合に、予め準備してあるネットワークデバイスが返すものと同等の擬似応答 をサブシステムが送出する。これにより、ホストシステムにサブシステムの存在を意識させないようにする。また、接続時に、ネットワークのパラメータとしてサブシステムが確立した際の情報(IP アドレスなど)をそのまま渡すようにする。さらにホストシステムがネットワークを確立する際には、後述するブリッジ機能が、ホストシステムとサブシステムのパスを結合してネットワークを利用できるようにする。

ホストシステムが通信を切断する場合には、ホストシステムが発行する切断コマンドに対して予め準備してある擬似応答を返すことによって、ホストシステムがネットワーク切断されたかのように動作させる。この状態においても、サブシステムとネットワーク間の通信は維持されている。

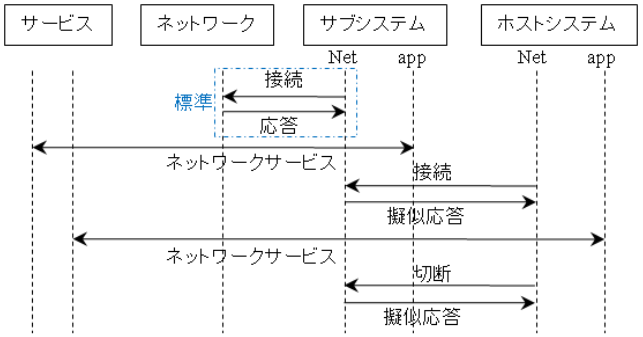

図 5　提案システムの通信接続/切断
Figure 5　Connection and disconnection protocol on proposed system.

4.1.2 USB 擬似応答と共有ストレージ

ホストシステムがブートする時やデバイス挿入時には、デバイスの Enumeration が行われる。この時、サブシステムは、CDC(Communication Device Class)と MSD(Mass Storage Device)の情報をマルチファンクションとしてホストシステムに通知する。USB の仕様では、USB DEVICE が備えるべきコマンドが定義されているが、ホストシステムからの GET 系のコマンド(GET_CONFIGURATION, GET_DESCRIPTOR, GET_INTERFACE, GET_STATUS)に擬似的に応答させる。CDC に関しては、ネットワークデバイスの情報をサブシステムに予め保持しておき、ホストシステムの要求に応じて渡す。これによりネットワークデバイスの存在を OS に伝えることができる。MSD に関しては、USB ストレージとしての定義をホストシステムに渡す。ホストシステムからは、リムーバブル記憶域にあるデバイスとして利用できる。これにより、サブシステムで用意したデータなどをホストシステムが共有利用可能にできる。

4.1.3 ブリッジ機能

本稿でのブリッジ機能の定義は図 4 左上に示したような、ネットワークから来たデータをホストシステムとサブシステムに分配する機能と、ホストシステムとサブシステムから来たデータを統合してネットワークに送出する機能のこと。この実現にあたり以下２つの方式を実装した。

ブリッジ 1: これは基本的な手法で、USB Host 制御部分に分配と統合の 2 つのタスクを追加することで実現する。分配タスクは、USB Host の受信 Buffer と連動して TCP 組み立てと、その Port 番号の解析により、入ってきたデータがホストシステムに向かうものか、サブシステムに向かうものか判断する。このとき予め定められたサブシステム向けのポートへのアクセスであれば、サブシステム側にスイッチを倒し、それ以外であればホストシステム側にスイッチを倒すような動作をさせる。統合タスクは、USB Device の受信 Buffer からの入力、サブシステムからの入力を、整合性を保ちつつ纏めてネットワーク側に送出する。このために、ホストシステムとサブシステムから来るデータを保持する 2 つのバッファを用意する。統合タスクはデータを取り込む時に、どちらのバッファから取得するか制御する。仮に片方にしかデータが入っていない場合には、そのデータを送信するが、両方にデータがある場合には、現在送信している USB データに続きがあるのかチェックを行う。そして、終了していなかった場合には、もう片方の情報送信を待たせることにより、ネット側に向かうデータに、不整合データが紛れ込むようなことが無いようにする。

ブリッジ 2: これは、ブリッジ 1 をベースとして、パフォーマンス改善のため分配タスクに修正を加えたもの。この方式では、Ping をチェックするタスクを追加する。具体的には、特定の Ping をサブシステムへの情報送信のトリガと見なし、それが送られてきた場合に、スイッチをサブシステム側に切り替える。通常状態では、スイッチはホストシステム側に倒されており、このイベントがあった場合に切り替わる。サブシステムの処理が終わると、デフォルト位置であるホストシステム側に戻す。

4.1.4 解析・制御

ブリッジによってサブシステム側に入ってきたデータを解析し、予め定めたデータ形式であった場合に、サブシステム内で処理を実行するようにする。制御は、例えば特定のコマンドが入ってきた場合に、GPIO (General Purpose Input Output)を駆動してホストシステムに通知を伝える。ホストシステムはその通知により動作を変えるトリガにできる。

4.2 システムハードウエアの仕様

次章以降の実評価に利用したハードウエアの仕様は以下の通りである(図 6)。

サブシステムとして利用した MCB2388 のハード仕様は、最大 CPU 周波数 72[MHz]、動作電圧 5.0[V]、定格電流 65[mA]、最大電流[120mA]である。ネットワークデバイスは、AnydATA Inc.製の FOMA A2502 HIGH-SPEED を用いた。これは、3G 通信を提供し、ダウンリンクの最大データレートが 7.2[Mbps]、最大電流 650[mA]、平均電流 440.6[mA]、最大スタンバイ電流 60[mA]、平均スタンバイ電流 54.7[mA]である。ホストシステムとして利用したのは、富士通製のノート PC: FMV-BIBLO NF/C80N である。これは、Intel Core 2 Duo Processor P8600 2.4 [GHz]、4 [GB]のメインメモリ、80 [GByte]のハードディスクを備える。OS (Operating System)は Microsoft Windows XP を利用した。また、通信接続ソフトウエアは、ネットワークデバイスに標準で添付されてきたものを用いた。共有ストレージとして、SD メモリを用いた。

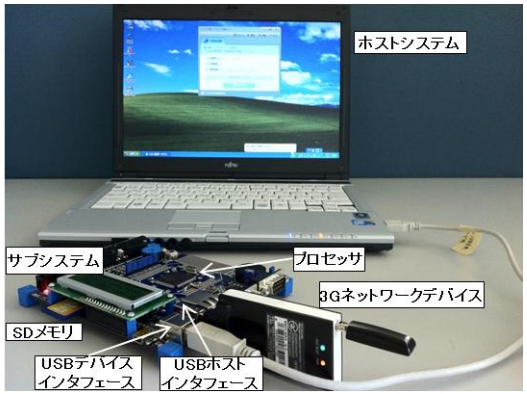

図 6 システム構成
Figure 6 System configuration.

5. ホスト透過性の評価

ここでは、サブシステムのホストに対する透過性の観点から、課題解決するシステムが構築できているかどうか実証を行う。本システムでは、通信モジュールとの独立性を持たせたことにより、サブシステムにおけるパフォーマンスへのインパクトが懸念される。特に、定義したサブシステムのうち、ブリッジが機能する SS3 が性能的に最もシビアになるはずであるが、ここでのオーバヘッドが許容できるレベルであれば実現可能性が高いと判断できる。以下では、この状態を中心に３つの観点から評価を行った。

① ホストシステムからのデバイス認識の透過性
② 接続・切断動作におけるユーザビリティへの影響
③ ホストシステムのパフォーマンスへの影響

5.1 基本動作

前述の①②について、以下のように動作に問題が無いことを確認した。

- Windows のデバイスマネージャから、通信デバイスとしての定義が見えること。
- ネットワーク接続ソフトを起動して、ネットワーク接続を開始することで、ブリッジが機能し、Web ブラウズなどの通信ができること。
- ネットワーク接続ソフトで切断することによって Windows に対するネットワー

クが途切れたことが認識されること。

5.2 フィールドでの実地性能評価

前述の③について、2 つの実際のオフィスで実性能評価を行った結果を示す。

5.2.1 電波状態の良好なオフィスでのパフォーマンスの実評価

電波状態が良好なオフィスにおいて行った評価と結果を示す。この環境で、サブシステムの CPU 周波数を 36, 48, 72[MHz]の 3 つに設定し、ブリッジ(ブリッジ 1, ブリッジ 2)が駆動している状態で測定を行った。パフォーマンステストには FTP (File Transfer Protocol)を用いた(図 7)。

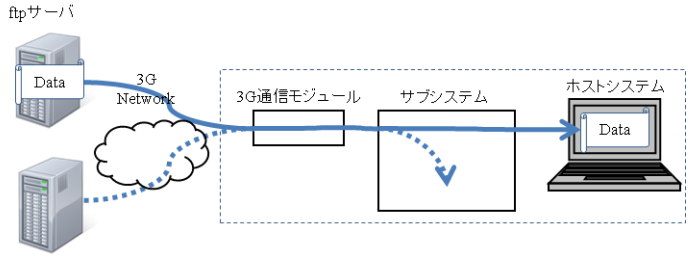

図 7　評価構成
Figure 7　Evaluated system structure.

FTP サーバとして、Yahoo GeoCities の FTP Web ホスティングサービスを利用し、FTP クライアントとして、Windows XP のコマンドプロンプトから標準搭載の FTP クライアントを利用した。FTP サーバには、2[MByte]のバイナリデータをおき、それを Get することでスループットを計測した。

(a) 最初にサブシステムを置かずに、ホストシステムとネットワークデバイスを直結状態において測定を行った。この時のスループット結果は、2.87[Mbps]であった。
(b) 次にサブシステムを配置し、ブリッジ 1 を使って測定を行った。その結果は、36[MHz]で 0.86[Mbps], 48[MHz]で 1.08[Mbps], 72[MHz]で 1.38[Mbps]であった。
(c) さらにブリッジ 2 を使って測定を行った。結果は、36[MHz]で 1.17[Mbps], 48[Mbps]で 1.52[Mbps], 72[MHz]で 2.15[Mbps]であった。

この結果をプロットしたものが 図 8(左)である。72[MHz]の(b)と(c)のスループットが、サブシステムを伴わない(a)に到達していないことから、サブシステムの CPU 性能が十分でないことが分かる。そこで、CPU 周波数をどこまで上げる必要があるか、線形近似によりスループット性能予測を(b)と(c)について算出し、同図に点線で示した。ブリッジ 1 では約 177 [MHz]、ブリッジ 2 では約 98.6 [MHz]で CPU を駆動できれば(a)と遜色の無いパフォーマンスになることが予測できる。

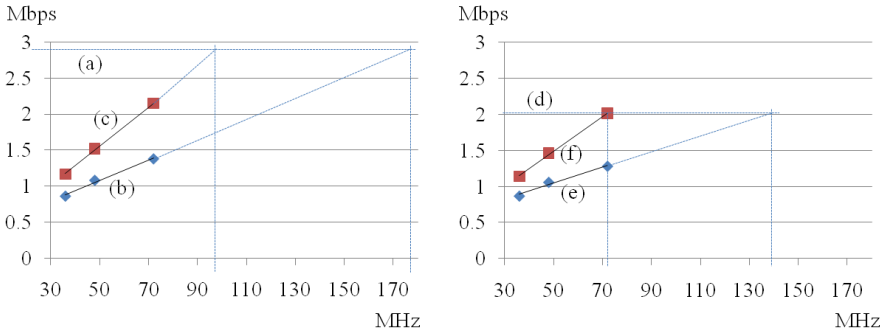

図 8　スループット評価
Figure 8　Results of throughput tests.

5.2.1 平均的なオフィスでのパフォーマンス実評価

電波状態が平均的なオフィスにおいて行った評価と結果を示す。評価方法は前項と同じである。

(d) サブシステムを置かずに、ホストシステムとネットワークデバイスを直結状態において測定を行ったスループット結果は、2.02[Mbps]であった。
(e) サブシステムを配置し、ブリッジ 1 を使って測定を行った結果は、36[MHz]で 0.87[Mbps], 48[MHz]で 1.06[Mbps], 72[MHz]で 1.28[Mbps]であった。
(f) ブリッジ 2 を使って測定を行った結果は、36[MHz]で 1.14[Mbps], 48[Mbps]で 1.46[Mbps], 72[MHz]で 2.01[Mbps]であった。

この結果をプロットしたものが 図 8(右)である。(f)は 72[MHz]動作であれば(d)と同性能であり、すなわちブリッジ 2 では、このままの動作周波数で問題無いことが分かる。(b')はサブシステムの駆動周波数が十分でない。同様に線形近似により、(e)においてとるべき駆動周波数を算出すると、ブリッジ 1 では、約 137 [MHz]であれば、(a)と遜色の無いパフォーマンスになることが予測できる。

6. サブシステムを使ったサービスの提案と開発評価

サブシステムは、ホストシステムがオフ・スリープであっても、あらゆるネットワークサービスが実行可能である。ここでは、最初に、サブシステムを使った応用例として、アプリケーションの配信と実行機能の提案および実装・評価について述べる。その後に、ダウンロードにおける電力評価の結果を示す。

6.1 アプリ配信と実行機能の開発と動作確認

サブシステムの応用例として、ネットワーク越しにサブシステムに対してアプリケーションを降らせる仕組みを提案する。これは、サブシステムに対してダウンロードを指示し、サブシステムがダウンロードしたアプリケーションを、ホストシステムが実行することを可能にするシステムである。これにより、PC において、ユーザが簡単に新しいアプリケーションを利用可能なシステムを提供できることになる。図 9 には、図 2 の構成のうちステート SS2 に関するものに加え、ホストシステム内に同期モジュールと呼ぶ機能を追加する。

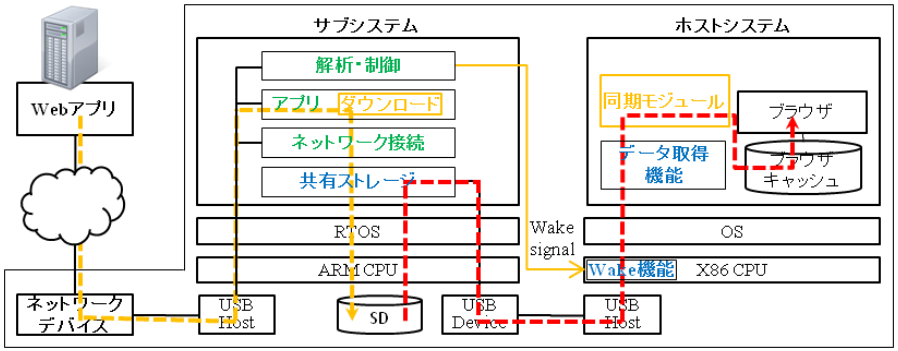

図 9　アプリケーション実行システムの構成
Figure 9　Structure of application execution system.

以下では、ホストシステム側の機能であるデータ取得機能への補足説明と、同期モジュールの機能説明を記載する。ここでは、Web アプリを扱っているが、通常のネイティブアプリとサイレントインストールを組み合わせることも可能である。

6.1.1 データ取得機能と同期モジュール

データ取得機能は、サブシステムの共有ストレージに関連する情報が無いかどうかチェックする。確認のタイミングは、ホストシステムの起動時およびスリープ等からの復帰時、ホストシステム動作中におけるデータ取得機能からのポーリングである。

同期モジュールは、Web アプリを扱うための機能である。この機能は、共有ストレージに置かれた Web アプリをブラウザキャッシュに挿入する。これにより、ブラウザの起動時やリロードを実行した際に、取得した Web アプリを利用できるようになる。

6.1.2 実装および動作フロー

図 10 の実装および動作フローは以下の通りである。

ダウンロード：サーバから、アプリ取得先を記述したメッセージを HTTP PUT によりサブシステムに通知する。このメッセージを受け取ったサブシステムのメッセージ解析・実行機能がメッセージを解析し、サブシステムのアプリとして実装したダウンロード機能がクラウド側に置かれたアプリケーションのダウンロードを実行する。ダウンロードしたアプリはサブシステムの共有ストレージ(SD カード)上に置く。

Wake：PC がスリープ状態にある場合にメッセージ解析・実行機能が、PC に対して Wake signal を発行する。この Signal を受け取った PC はスリープ状態から復帰する。

データ取得：PC のデータ取得機能は、起動時監視および常時監視により、SD に有効なデータがあるか確認する。

ブラウザ同期：PC の同期モジュールは、有効なデータがあった場合に、Web アプリをブラウザキャッシュに入れると共に、ブラウザを起動またはブラウザが起動されている場合にはリロードを実行する。ローカルキャッシュに 2 つ以上のアプリがある場合には、どのアプリがインストールされているかを表示するようにした。

実行：ブラウザは、同期モジュールによって組み込まれたアプリを実行する。

6.1.3 構成・動作確認

Web アプリとして HTML5[6]を用い、Chrome ブラウザ[5]を使って開発・評価を行った。HTML5 は、Web Hypertext Application Technology Working Group (WHATWG)が標準化を行っている新しい HTML 標準である。HTML5 で導入された機能の一つに、Application Cache を利用するオフライン Web アプリケーションがある。これは manifest と呼ばれるファイルに、必要なリソースファイルを記載することで、オフラインのときに Web アプリケーションを動作させることができるようになる機能である。ブラウザは最初の.html ファイルをアクセスするときに、オフラインで必要なファイルの一覧を取得し、ローカルのキャッシュに入れる。

動作確認として、サーバ側からアプリのダウンロードを指示し、ダウンロードを完了すると PC を起動してブラウザキャッシュに取り込み HTML5 のローカルアプリが実行できることを確認した。

図 10　Web ブラウザへのアプリインストールの様子
Figure 10　Application installation on Web Browser.

図 10 はアプリが追加される様子とアプリが実行された状態を表している。図 10 左がダウンロード前で、図 10 中央がダウンロード後を表しており、一つアプリが追加されている。図 10 右はアプリが自動実行された状態を表している。前述の通り HTML5 では、ネットワーク接続を実行しなくともアプリが動作する。このように本システムにより、PC においても遠隔からアプリケーションをインストールするだけでなく、自動実行することが可能になり、例えば人手によるアプリケーション・セットアップを不要にできるなど、従来とは違った新しい使い方ができるようになる。

6.2 ダウンロードと電力評価

サブシステムが存在しない場合と、サブシステムが存在する場合で、ダウンロード時の電力測定を行った結果を Table1 に示す。これに対して、上述 5.2.1 の FTP によるデータダウンロード時間により補正して、電力量の算出を行った。サブシステム無しの場合、ダウンロード時間 5.9[Sec]で、電力量は 43.8[mWh]。サブシステム有りの場合は、ブリッジ 1(72[MHz])において、ダウンロード時間は 12.2[Sec]で、電力量は 12.2[mWh]。よってサブシステム無に比して約 73[%]の電力低減が期待できることが分かる。またブリッジ 2(72[MHz])において、ダウンロード時間は 7.88[Sec]で、電力量は 7.8[mWh]。よってサブシステム無に比して約 82[%]の電力低減が期待できる。例えば Microsoft Windows 7 の Service Pack 1 は、x64 ベースのスタンドアロンインストールのパッケージダウンロードサイズが 903[MByte]である。これをブリッジ 1 駆動のサブシステムでダウンロードすると 14.3[Wh]、ブリッジ 2 では 16.3[Wh]の電力をセーブできることになる。

表 1　電力測定結果
Table 1　Evaluation on power consumption

システム構成	ステータス	電力[W]
サブシステム無	データダウンロード中	26.7
サブシステム有	PC スリープ時のデータダウンロード	3.55

7. おわりに

通信モジュール及び PC へのインパクトが無くネットワークサービスを提供できるサブシステムのアーキテクチャと、その応用例としてアプリ配信と実行サービスを提案した。そして ARM7 評価ボードを用いてプロトタイプの実装を行い、3G 通信モジュールおよび PC と合わせて実環境での評価を実施した。ホストから見た透過性の実証では、接続・切断に関して従来の操作感と変わらないこと、また CPU 性能としては 100[MHz]程度以上あれば 3G 性能に影響を与えないこと、平均的なオフィスにおいては、本システムで用いた 72[MHz]でも十分に機能することが分かった。さらに応用例では、PC がスリープの間に HTML5 アプリをダウンロードし、PC を起動してアプリを実行する仕組みを提案した。そして、実現できることを実証し、PC の新たな使い方の可能性を示した。また、ダウンロード中に関してはサブシステムを用いることにより 70[%]以上の電力消費低減が可能であることを示した。今回我々が選択した ARM ベースのチップおよび通信デバイスは新しいものではない。より低電力のものが既に世の中に出ているため、それらに単純に置き換えることで、さらなる電力消費の低減が期待できる。

参考文献

1. NTT docomo i コンシェル, http://www.nttdocomo.co.jp/service/customize/iconcier/
2. PC ENERGY REPORT 2009 UNITED STATES, UNITED KINGDOM, GERMANY, http://www.1e.com/energycampaign/downloads/PC_EnergyReport2009-US.pdf
3. Introducing Intel Anti-Theft Technology version 3.0, http://antitheft.intel.com/Anti-Theft-30.aspx
4. 坂巻健士、永利秀之、向地賢記、二村和明, ノートパソコンの遠隔操作による情報漏えい対策ソリューション:CLEARSURE, 雑誌 FUJITSU 2010-3 月号(VOL.61, NO.2), http://img.jp.fujitsu.com/downloads/jp/jmag/vol61-2/paper01.pdf
5. Chromium an open-source browser project, https://sites.google.com/a/chromium.org/dev/Home
6. HTML5, http://dev.w3.org/html5/spec/Overview.html
7. Bruce Nordman, Ken Christensen, Proxying: The Next Step in Reducing IT Energy Use, IEEE Computer Society, pages 91-93, JANUARY 2010
8. Bruce Nordman, Proxying: Reducing PC energy use with network technology — EEDN Seminar slide, September 10, 2010
9. S. Nedevschi, J. Chandrashekar, B. Nordman, S. Ratnasamy,and N. Taft., Skilled in the art of being idle: reducing energy waste in networked systems, In Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2009.
10. M. Jimeno, K. Christensen, and B. Nordman. A Network Connection Proxy to Enable Hosts to Sleep and Save Energy. In IEEE International Performance Computing and Communications Conference, 2008.
11. Kenneth J. Christensen, Chamara Gunaratne, Bruce Nordman, and Alan D. George, "The next frontier for communications networks: power management", Computer Communications 27 (2004) 1758-1770.
12. Yuvraj Agarwal, Steve Hodges, Ranveer Chandra, James Scott, Paramvir Bahl, and Rajesh Gupta, Somniloquy: Augmenting Network Interfaces to Reduce PC Energy Usage, NSDI '09: 6th USENIX Symposium on Networked Systems Design and Implementation, 2009
13. ENERGY STAR Program Requirements for Computers Version 5.0, October 2008.
14. ECMA-393, 1st Edition, proxZzzy for sleeping hosts, February 2010
15. Apple Wake on demand, http://support.apple.com/kb/HT3774
16. Multicast DNS, draft-cheshire-dnsext-multicastdns-14, Internet Engineering Task Force, http://files.multicastdns.org/draft-cheshire-dnsext-multicastdns.txt
17. 3GPP TS 27.060 V3.8 (2003-06), 3rd Generation Partnershp Project; Technical Specification Group Core Network; Packet Domain; Mobile Station (MS) supporting Packet Switched Services (Release 1999).