情報処理学会論文誌　コンシューマ・デバイス＆システム　Vol. 1　No. 1　1–11 (Dec. 2011)

コンシューマ・システム論文

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[†1]　伊藤栄信[†1]　中村洋介[†1]
藤野信次[†1]　三浦隆文[†2]

PC がスリープ状態であってもネットワークサービスを継続するサブシステムを導入することで，いつでもサービスを享受できる．その実現のために，従来では通信モジュール内に変更を加えるアプローチが一般的であった．しかし，その場合，新しい通信モジュールへの乗り換えが困難になるという課題があった．そこで本稿では，汎用 WWAN モジュールを使用し，かつ PC などのホストシステムへの変更が不要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に動作可能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 など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,[†1] Hidenobu Ito,[†1]
Yousuke Nakamura,[†1] Nobutsugu Fujino[†1]
and Takafumi Miura[†2]

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ても，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．

たとえば，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 をつ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，たとえば，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たび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．これらは労力をともなうものであり，開発コストへの影響が懸念される．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．

†1 株式会社富士通研究所
Fujitsu Laboratories Ltd.
†2 富士通株式会社
Fujitsu Limited

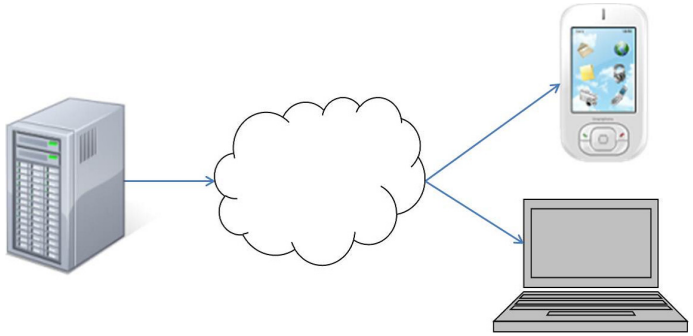

図 1　ネットワークサービス
Fig. 1　Network service.

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変更が不要な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ともに，フィールドにおける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・実行するための端末側の仕組みを提案する．これは従来の Windows よりも平易なアプリケーション実行を可能にし，コンシューマ PC など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 章では関連研究に触れる．3 章では，提案アーキテクチャを示し，4 章ではインプリメンテーション詳細を示す．5 章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 章では，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 章でまとめる．

2. 関 連 研 究

Proxying[7)–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よ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 1 つのアーキテクチャを提示するとともに，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などに渡すべきパラメータなどを定義している．製品としては，Apple の製品に mDNS[15)] を用いて Wake on WLAN を扱う機能が組み込まれている[16)]．

しかし，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．LAN および WLAN が基本アクセスにおいてユーザによる接続手続きを必要としないのに対して，WWAN はダイヤルアップ接続を必要とする．また特に WWAN デバイスに依存せずサービスを受ける仕組みが確立されていなかった．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への変更を不要にすること」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どお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

サブシステムをともなうシステムの大枠構成として 2 つが考えられる．1 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 1 つは，ネットワークデバイスの制御をつね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．上記要件を満たすため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，たと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というメリットがある．

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．

要件に対応する機能概要は以下のとおりである．

(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．

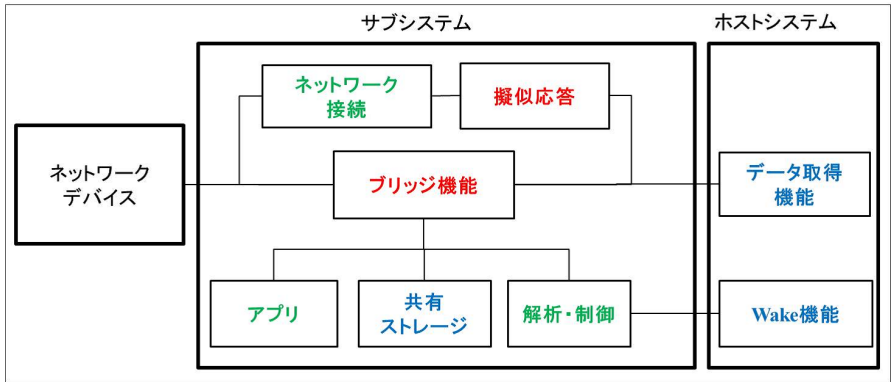

図 2　サブシステムとホストシステムのアーキテクチャ
Fig. 2　Architecture of the sub system and host system.

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の 3 つのステートに関する定義を示す．このステートのいず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．

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 にゆくに従い大きくなる．

また，次章以降に関して，5 章で述べるホスト透過性の評価は，SS2 と SS3 の間の状態遷

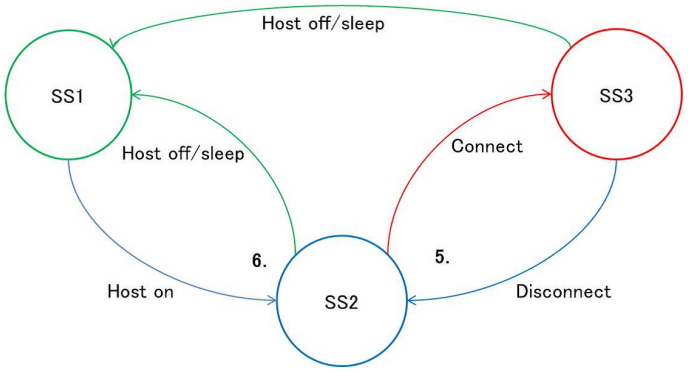

図 3　状態遷移
Fig. 3　State transition.

移を扱っており，6 章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ために最低限必要となるハードウェア構成は，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，具体的なハードウェアとして，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 ファミリの CPU（LPC2388 ARM7TDMI-S）を搭載し，最大周波数 72 [MHz] で動作，512 KB のオンチップ・フラッシュ・メモリと，64 KB の CPU 用 SRAM，USB（Universal Serial Bus）が利用する 16 KB 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．

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

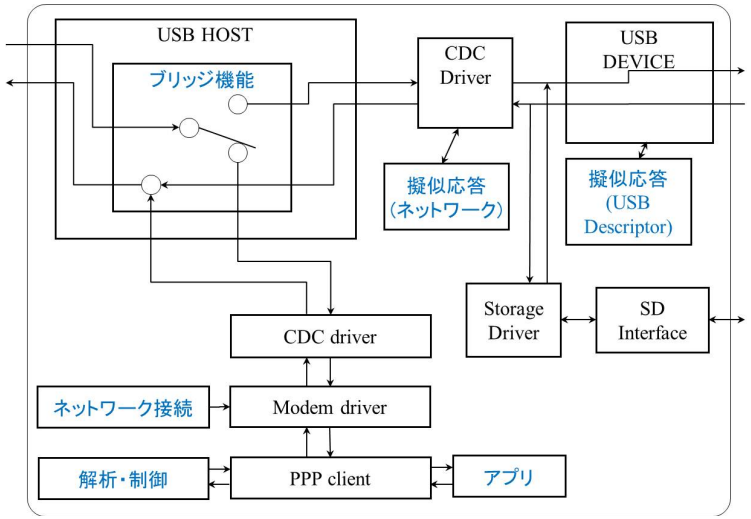

図 4　サブシステムの構成
Fig. 4　Structure of sub system.

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，このとき，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，つねにネットワーク接続が維持されるように実装している．ここでサブシステムのアプリケーションとは，ネットワークからの指示を受けて動作するサービスのことである．開発者はアプリケーション部分のみ記述することでサービスを追加できる．本実装ではマルチタスク Real-Time Operating System を用いたため，複数のサービスを同時に動作させることが可能である．しかし低スペックな CPU を用いたため動作検証は行っていない．より高スペックの CPU を使用すれば，複数のサービスの提供も問題ないものと考えられる．

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

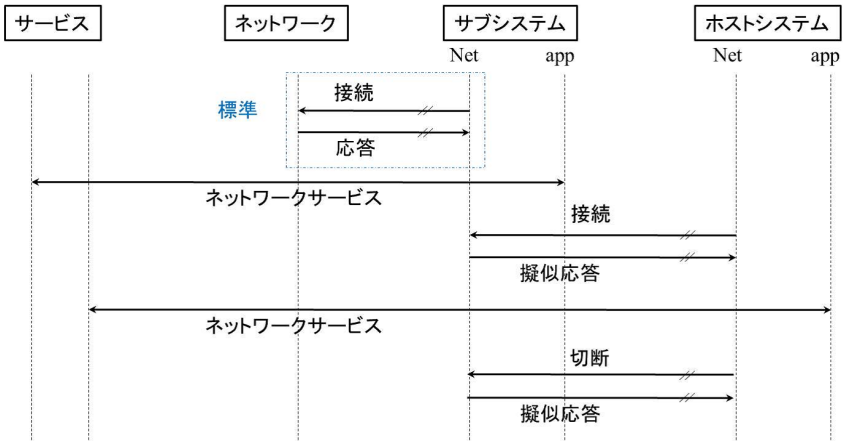

図 5　提案システムの通信接続/切断
Fig. 5　Connection and disconnection protocol on proposed system.

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，あらかじ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あらかじ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．

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ときやデバイス挿入時には，デバイスの Enumeration が行われる．このとき，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あらかじめ保持しておき，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である．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，あらかじ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，整合性を保ちつつまと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ときに，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ないようにする．
- 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である．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，あらかじ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，たと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　システムハードウェアの仕様

次章以降の実評価に利用したハードウェアの仕様は以下のとおりである（**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 [120 mA] 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．

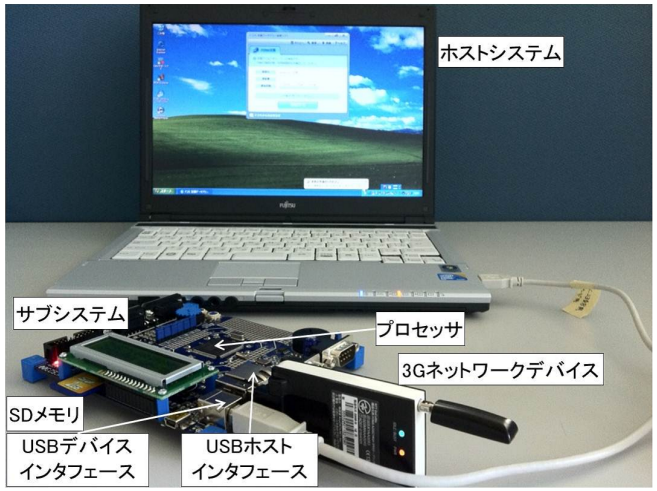

図 6　システム構成
Fig. 6　System configuration.

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．

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影響が懸念される．特に，定義したサブシステムのうち，ブリッジが機能するステート 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．

(A)　ホストシステムからのデバイス認識の透過性
(B)　接続・切断動作におけるユーザビリティへの影響
(C)　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 (A)，(B) について，以下のように動作に問題がないことを確認した．

- 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．

また，前述の (A) について，レイテンシの観点から以下のような測定を行った．

- TCP 1 フレームのレイテンシを実測と補足計算から求める．具体的な測定ポイントは，USB Host でのデータ受信開始時（P1）と，USB Device でのデバイスへデータ送信後（P2）である．これらのタイミングで，UART にキャラクタを出力し所要時間を測定した．その結果は CPU 周波数が 72 [MHz] の場合 244 [μs] であった（ping データ（98 [Byte]）を送信し，サブシステム内でデータ受信してから，ホストシステム側へ送信するまでの時間を測定）．1 フレームは 1,460 [Byte] で，USB バッファサイズが 512 [Byte] であることから 3 回処理が発生し，$244 \times 3 = \underline{732}$ [μs] となる．

この測定では，測定区間（P1-P2）前後の処理が含まれていないため，それらについてはアセンブラコードから処理時間を算出した．調査の結果，アセンブラの処理ステップ数は 144 ステップであった．アセンブラは 1 クロックで処理される場合がほとんどであると考えられるため CPU の 1 クロックである 1/72 [MHz] = 14 [ns] とかけ合わせると所要時間は 2.02 [μs] となる．

さらに USB Host と USB Device の DMA 処理時間を算出すると，DMA の 1 [Byte] 転送時間である 4 クロックと CPU の 1 クロック時間をかけ合わせた 4×14 [ns] = 56 [ns] と，1 フレームのバイト数である 1,460 [Byte] とをかけ合わせ，さらに 1 転送で 2 回の DMA が発生することをふまえて，$0.056\,[\mu s] \times 1,460 \times 2 = 163.5\,[\mu s]$ が所要時間となる．

これら 3 つの和である $732\,[\mu s] + 163.5\,[\mu s] + 2.02\,[\mu s] = 898\,[\mu s]$ が TCP1 フレームのレイテンシになる．

また，上り方向における ping に対する Ack の測定結果は 460 [μs] であった．上記と同様に測定ポイント外の処理に関してアセンブラのステップ数を調査したところ 100 ステップであった．よって所要時間は 1.4 [μs] となる．さらに，これに関しては 60 [byte] の DMA 処理が 2 回発生するため 6.72 [μs] が加わる．これらの 3 つの和である 468 [μs] が Ack のレイテンシになる．

そして，Ping と Ack を加えた 1366 [μs] が 1 ラウンドのレイテンシとなる．

5.2　フィールドでの実地性能評価

前述の (C) 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**）．

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．このときのスループット結果は，2.87 [Mbps] であった．

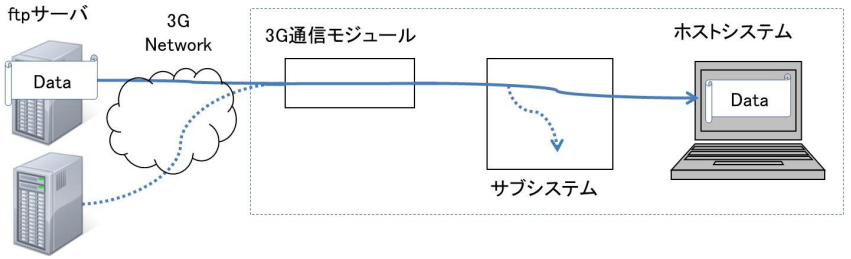

図 7　評価構成
Fig. 7　Evaluated system structure.

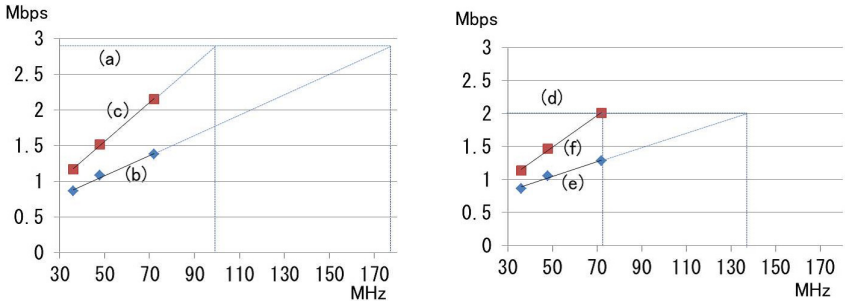

図 8　スループット評価
Fig. 8　Results of throughput tests.

(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，サブシステムをともなわない (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) と遜色のないパフォーマンスに

なることが予測できる．

5.2.2　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ないことが分かる．(e) 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) と遜色のな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プッシュす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．

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．

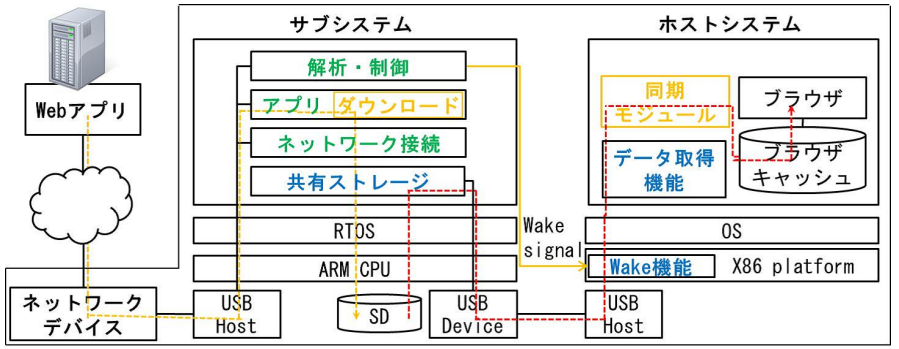

図 9　アプリケーション実行システムの構成
Fig. 9　Structure of application execution system.

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な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などからの復帰時，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 9 の実装および動作フローは以下のとおりである．

- 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ともに，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 1 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 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．左がダウンロード前で，中央がダウンロード後を表しており，1 つアプリが追加されている．右はアプリが自動実行された状態を表している．前述のとおり HTML5 では，ネットワーク接続を実行しなくて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，たと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．

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

7. ダウンロードと電力評価

ここでは実アプリケーションを利用した際の消費電力およびレイテンシに関する測定結果を示す．

まず消費電力について，サブシステムが存在しない場合には，PC にアプリケーションをダウンロードし，サブシステムが存在する場合には，ブリッジ 2 を用いてサブシステム内の SD メモリにダウンロードする．439.7 [KB] のアプリ（図 10 右にも示した HTML5 Pacman[18] を zip したもの）をダウンロードしたときの電流測定結果を**表 1** に示す．このときの所要時間はサブシステムが存在しない場合は 4.7 [s] であり，電力量は 35 [mWh]．サブシステムが存在する場合は 25 [s] であり，電力量は 25 [mWh] であった．

これによりサブシステム導入による優位性があることが分かったが，ダウンロード時間をさらに削減できれば電力量の低減が期待できる．そこで，サブシステムの処理内のどこがボトルネックとなっているのか調査を行った．ダウンロードでは，図 4 のアプリ部分に http アクセスによるダウンロード処理機能が置かれている．以下の 4 つの経路について，キャラクタ出力によって所要時間を測定した．図 4 の USB Host から CDC ドライバの経路の所要時間は 0.113 [ms]，CDC ドライバから http 処理（開始点）の経路の所要時間は 16.5 [ms]，http 処理（開始点）から SD アクセスを経て CDC ドライバに戻る経路の所要時間は 21.8 [ms]，CDC ドライバから USB ホストへの経路の所要時間は 4.4 [ms] であった．この結果で顕著なのは CDC ドライバを含む経路の 2 つの処理が遅いことである．この原因の 1 つとして，利用した通信ライブラリが，シリアル送受信の処理を 1 バイトずつ処理するインタフェースを採用している点があげられる．今回ソースコードがなかったため測定はできなかったが，これを複数バイトで扱えるようにすることで処理回数を減らすことが可能になり効率を改善できると考えられる．またネットワーク側への送信制御処理において，1 つの USB Host ディスクリプタで処理を行っていた．このため 1 つの処理が終わらないと次の処理が動作しないため非効率である．これを複数のディスクリプタを扱えるようにする

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

	ステータス	電力 [W]
サブシステムなし	ダウンロード	26.5
サブシステムあり	PC スリープ時のダウンロード	3.54

ことで待ち時間を減らすことが可能になり効率を改善できると考えられる．これらの改良などにより，ダウンロード時間を改善することが可能であると考えられる．

次に，レイテンシを求めるために，ダウンロード先を同じ PC にして，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において，同じアプリケーションのダウンロードを http で実施した．その結果は，サブシステムなしの場合には 4.7 [s] であったが，サブシステムありの場合には 5.9 [s] であった．これらの差は 1.2 [s] である．レイテンシは増えるが，スリープ時にサービスができるメリットの方が大きいといえる．

8. おわりに

通信モジュールおよび PC への変更が不要で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，サブシステムを導入することで，電力消費量においても優位性があることが分かった．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く，より低電力のものがすで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によっても，電力消費の低減が期待できる．

参 考 文 献

1) 鈴木裕紀，平石絢子，竹田千沙，中矢恭介，高津利樹：2008 年秋冬モデル搭載アプリケーション機能（1）i コンシェル機能およびユーザメモリ一括バックアップ機能の開発，NTT ドコモテクニカル・ジャーナル，Vol.16, No.4, pp.40–45 (2009).
2) Karayi, S.: PC ENERGY REPORT 2009 UNITED STATES, UNITED KINGDOM, GERMANY, Technical report, 1e (2010).
3) Intel: Introducing Intel Anti-Theft Technology version 3.0 (2011), available from ⟨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，Vol.61, No.2, pp.94–99 (2010).
5) Surhone, L.M., Tennoe, M.T. and Henssonow, S.F.: *Chromium* (*Web Browser*), Betascript Publishing (2010).
6) W3C: HTML5 A vocabulary and associated APIs for HTML and XHTML, available from ⟨http://dev.w3.org/html5/spec/Overview.html⟩.
7) Nordman, B. and Christensen, K.: Proxying: *The Next Step in Reducing IT Energy Use*, pp.91–93, IEEE Computer Society (2010).
8) Nordman, B.: Proxying: Reducing PC energy use with network technology, *EEDN Seminar slide* (2010).
9) Nedevschi, S., Chandrashekar, J., Nordman, B., Ratnasamy, S. and Taft., N.: Skilled in the art of being idle: reducing energy waste in networked systems, *Proc. 6th USENIX Symposium on Networked Systems Design and Implementation* (*NSDI*) (2009).
10) Jimeno, M., Christensen, K. and Nordman, B.: A Network Connection Proxy to Enable Hosts to Sleep and Save Energy, *IEEE International Performance Computing and Communications Conference* (2008).
11) Christensen, K.J., Gunaratne, C., Nordman, B. and George, A.D.: The next frontier for communications networks: power management, *Computer Communications*, Vol.27, pp.1758–1770 (2004).
12) Agarwal, Y., Hodges, S., Chandra, R., Scott, J., Bahl, P. and Gupta, R.: Somniloquy: Augmenting Network Interfaces to Reduce PC Energy Usage, *NSDI'09: 6th USENIX Symposium on Networked Systems Design and Implementation* (2009).
13) Agency, U.S.E.P.: ENERGY STAR Program Requirements for Computers Version 5.0 (2008).
14) ECMA: ECMA-393 proxZzzy for sleeping hosts, 1st edition (2010).
15) Internet Engineering Task Force: Multicast DNS, draft-cheshire-dnsext-multicastdns-14 (2011).
16) Apple: Mac OS X v10.6 About Wake on Demand (2011), available from ⟨http://support.apple.com/kb/HT3774⟩.
17) 3rd Generation Partnershp Project: Technical Specification Group Core Network; Packet Domain; Mobile Station (MS) supporting Packet Switched Services, 3gpp ts 27.060 v3.8 (2003-06) edition (2003).
18) Harvey, D.: HTML5 Pacman (2010), available from ⟨http://arandomurl.com/2010/07/25/html5-pacman.html⟩.

(平成 23 年 4 月 15 日受付)
(平成 23 年 9 月 6 日採録)

二村　和明

昭和 44 年生．平成 6 年東京電機大学大学院情報通信工学専攻修士課程修了，同年富士通株式会社入社．平成 9 年より（株）富士通研究所．平成 13 年より 5 年間（株）米国富士通研究所勤務．モバイルコンピューティング，ヒューマンセントリックコンピューティングの研究開発に従事．

伊藤　栄信

昭和 43 年生．平成 5 年大阪府立大学大学院工学研究科数理工学専攻博士前期課程修了．同年（株）富士通研究所入社．モバイルコンピューティング，ヒューマンセントリックコンピューティングの研究開発に従事．

中村　洋介

昭和 52 年生．平成 12 年横浜国立大学工学部電子情報工学科卒業．平成 14 年同大学大学院工学研究科電子情報工学専攻修士課程修了．同年（株）富士通研究所入社．パーソナルコンピュータの先行技術開発，ヒューマンセントリックコンピューティングの研究開発に従事．

藤野　信次（正会員）

昭和 35 年生．昭和 61 年大阪府立大学大学院工学研究科電子工学専攻博士前期課程修了．同年（株）富士通研究所入社．現在，同ヒューマンセントリックコンピューティング研究所主管研究員．デジタル無線システム，マルチエージェント応用，無線用 TCP，異種網ローミング・統合，機器連携，車車間通信，電力平準化，ヒューマンセントリックコンピューティングの研究開発に従事．博士（情報学）．平成 15 年情報処理学会業績賞受賞．平成 16 年通信放送機構先端・基盤技術賞受賞．

三浦　隆文

昭和 40 年生．平成 2 年早稲田大学大学院理工学研究科修士課程修了．同年富士通株式会社入社．現在，同パーソナルビジネス本部シニアマネージャー．パーソナルコンピュータの先行技術研究開発に従事．