Thick/Thin クライアント切換え制御システムの Android 端末への実装

本橋　史帆[1]　前野　誉[2]　小口　正人[1]

概要：近年，無線通信やモバイル通信の高速化が急速に進んだことにより，クライアント側はコマンド入力や画面表示などの限定的な処理のみを行い，サーバ側に処理を全面的に頼る，Thin クライアントモデルとして機能する端末が増加した．一方，モバイル端末では，ガラパゴス携帯からスマートフォンへと流行が移り，近年スマートフォンの高機能化が進んでいる．スマートフォンは，クライアント側にデータを保持し計算処理も主にクライアント側で行う Thick クライアントモデルとして動作し，「持ち運べるコンピュータ」としての役割が期待されている．ここで，スマートフォンやタブレットのようなモバイル端末を Thick/Thin クライアントどちらの形に進化させていくべきなのかということが論点となる．しかしスマートフォンのアーキテクチャは，リソースの制約や，プログラミングおよび実行環境が組込みシステム向けとなっていることなどから，汎用 PC とは大きく異なることにより，これまでの Thick/Thin クライアントモデルの議論は，クライアント側に汎用 PC を用いる事が多かった．
これに対し本研究では，汎用 PC と環境が大きく異なる Android 端末において，Thick/Thin クライアントモデルを議論し，両者を環境に応じて切り換えて扱うことを可能にする制御システムの実装を提案する．またこれを実現するための各要素の具体的な実装例や，本提案制御システムの実用例を示す．

A System Implementation to Control Switching of Thick/Thin Clients using Android Terminal

Shiho Motohashi[1]　Taka Maeno[2]　Masato Oguchi[1]

1. はじめに

クライアント・サーバ型のネットワークコンピューティングにおいては，クライアント側にデータを保持し，計算処理も主にクライアント側で行う Thick クライアントモデルと，クライアントではコマンド入力や画面表示などのような限定的な処理のみを行い，サーバ側の処理に全面的に頼る Thin クライアントモデルが提唱され，各時代のコンピュータやネットワーク環境においてそれぞれの実装が行われてきた．Thick クライアント端末はデータやアプリケーションを端末内に保持するため，オフラインでも操作が可能というメリットを持つ一方で，データ管理・セキュリティ面には不安が残ると言える．それに対し，Thin クライアント端末はサーバ側でデータを管理するため，セキュリティ面で注目を集めている．また，サーバ側のリッチなリソースを用いた高機能/高性能処理が可能というメリットも持つ．しかし Thin クライアント端末はサーバに接続するための一定レベルの通信帯域が必須となり，オフラインでは使用できないことがデメリットとなる．

近年，スマートフォンが爆発的に増加し，その高機能化が進んでいる．それに伴い，Thick クライアントとして動作するスマートフォンは「持ち運べるコンピュータ」としての役割が期待され，Thick クライアント化が進んでいる．その一方で，無線通信やモバイル通信の高速化が進み，通信帯域が太く安定したため，セキュリティ面で注目を集めた Thin クライアント端末が急速に普及した．モバイル端末を Thin クライアント化してしまおうという動きも見られる．しかし，前述のように Thick/Thin クライアントモ

[1] お茶の水女子大学
〒 112-8610 東京都文京区大塚 2-1-1
[2] 株式会社スペースタイムエンジニアリング
〒 101-0025 東京都千代田区神田佐久間町 3-27-3 ガーデンパークビル 7F

デルにはどちらにもメリット・デメリットが挙げられるため，どちらが良い/悪いという優劣をつけることは難しい．

増加しているスマートフォンとして主に Android 端末と iOS デバイスの 2 つが挙げられる．このうち Android 端末は世界のスマートフォンシェア率の約 80 %を占める [1] ．また，Android の開発環境は，無償で提供されており，誰でも気軽にアプリケーションを開発することが可能になっていることからも対応アプリケーションが開発しやすいというメリットがある．そのうえ Android はキャリア間の制約がなく，アプリケーション開発においても自由度及び汎用性が高いということがメリットとなる．これらの利点から，本研究では Android 端末を用いて，ネットワーク環境や起動するアプリケーションの処理の重さ，端末性能を考慮し，環境に応じて Thick クライアントモデルから Thin クライアントモデルへと切換えて処理を実行することができるような切換え制御システムを提案する．この制御機能を実現させることにより，Thick/Thin クライアント両者のメリットを生かした端末の使用ができるのではないかと考えられる．

これまで行われてきた Thick/Thin クライアントモデルの議論では，クライアント側は基本的に汎用 PC であることが多く，サーバ側に近いアーキテクチャのコンピュータをクライアント側で用いてきた．これに対し，スマートフォンのアーキテクチャは I/O インタフェースの機能の乏しさ，限られたハードウェア資源，プログラミングおよび実行環境が組込みシステム向けになっているなどの点で，従来の汎用 PC のアーキテクチャとは大きく異なり複雑である．本研究では，こういった理由もあり，今まであまり議論されてこなかったモバイル端末をクライアント側に用いて Thick/Thin クライアントモデルを議論し，環境に応じて両モデルの切換え制御を行うシステムの提案と具体的な実装を行う．また，本提案制御システムの実用例として経路探索を挙げる．

2. Android のアーキテクチャ

Android は，2013 年第 3 四半期では全世界のスマートフォン OS の中でも，81.0%とトップシェアを占める [1] ．Android のアーキテクチャを図 1 に示す [2] ．Android は Linux カーネルをベースとし，スマートフォンやタブレット端末をターゲットに，それらに適したコンポーネントが追加されている．Linux OS と大きく異なる部分は，独自に開発された，Java で記述された Android アプリケーションの実行環境である Dalvik VM(Dalvik 仮想マシン) を Android Runtime に搭載している点である．基本的にすべての Android アプリケーションは Java で記述され，この Dalvik VM を介して実行される．そのため，ネイティブコードで記述されるプログラムに比べ，Android アプリケーションは実行速度が遅くなってしまう．特に，処理するデータ量が多く，CPU に負荷をかけるようなグラフィクス処理や数値計算処理，データベース処理などに関しては，ネイティブコードで記述されたプログラムと Android アプリケーションとでは実行速度に大きな差が現れ，Android 端末での処理は不向きであると考えられる．本提案制御システムはこれらの処理についても，サーバ側とクライアント側を使い分けることにより，Android 端末からの利用でも十分な速度で実行させることを可能にすると考える．

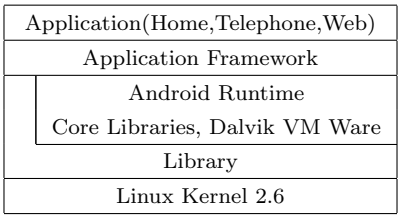

図 1 Android のアーキテクチャ

3. 関連研究

携帯性とサーバの利便性を両立する Thin クライアント端末としてモバイル端末を利用した Thin クライアントシステムの検討がなされてきた [3] ．関連研究のほとんどはモバイル端末を Thin クライアント化させることに焦点を置き，そのための要件やセキュリティ・データ管理に関するものである．実際に近年ではモバイル端末を Thin クライアント化させ，端末の紛失・盗難対策やデータやアプリケーションの管理などセキュリティ機能を考慮したものも提供されている．しかし，Thin クライアントモデルにはサーバへの接続のための通信環境が必須であり，全ての処理をサーバを介して行うことによる通信料・消費電力量などを考慮すると，必ずしも Thin クライアントの形が良いとは限らない．そこで，本研究ではモバイル端末を完全に Thin クライアント化するのではなく，Thick クライアントとしての機能を残しつつ，環境に応じて Thin クライアントモデルに切換えて動作させることを目標とする．

4. Thick/Thin クライアント性能評価

4.1 実験目的

Android 端末において，Thick/Thin クライアントモデルを切換えて扱う制御システムを提案し，両モデルを使い分けることに意義があることを示す．そのためにまず，両者の処理性能比較を行い，Thick クライアント端末で十分な速度で実行できるアプリケーションと，Thick クライアント端末では実行速度が遅く，Thin クライアントモデルの形で処理を行った方が好ましい Thin クライアント向けのアプリケーションがあることを明らかにする．

4.1.1 Android アプリケーション

Android アプリケーションの開発環境は無償で提供されており，誰でも気軽にアプリケーションを開発できるようになっている．また，Android アプリケーションはキャリア間の制約がなく，自由度も汎用性も高い．そのため，様々な機能を持つ数多くのアプリケーションが市場に出回っており，ユーザは自由に欲しいアプリケーションを入手できる．しかし，それらはもちろん Android 端末上で処理することを前提としており，十分な速さで処理ができるものに限られるため，Thick/Thin クライアントモデルの切換えは必要としないと考えられる．そこで本研究では，このアプリケーション開発環境を利用して，2 節で挙げたような実行条件によっては Android には不向きと考えられる処理のうちグラフィクス処理に焦点を当て，あえて実行速度が遅くなる可能性のある重い処理のアプリケーションを実装する．

4.2 実験概要

PNG 形式 182KB の画像を 1 枚用意し，グレースケール化するプログラムとネガポジ反転をするプログラムを作成した．Thick クライアント側は Java で Android アプリケーションとして実装し，Android 端末実機で実行させる．一方，Thin クライアント側はネイティブコードで記述したプログラム (今回は C で書いたプログラムのバイナリ) をサーバ側で処理させ，処理結果をクライアント端末に表示させる．それぞれ画像処理にかかる時間を計測することで，このような単純な画像処理における Android 端末とサーバの処理性能の比較を行う．

4.3 実験環境

本実験で使用した実験環境を表 1 に示す．

Android	Model number	Galaxy Nexus
	Android version	4.2.2
	Application	OpenCV Manager
	CPU	1.2GHz
	Memory	700MB
server	OS	Ubuntu 13.04 (64bit)
	CPU	3.60GHz
	Memory	4GB
	Application Development Tool	JDK, Android SDK, Eclipse, ADT
	Library	OpenCV-2.2.0, OpenCV for Android-2.4.7

表 1　実験環境

サーバ側に Android アプリケーション開発環境を整えた．本実験ではグラフィクス処理を扱うため，コンピュータビジョンに必要な機能を揃えた C/C++ライブラリ集である OpenCV をインストールし，さらに Android アプリケーション用に Java から OpenCV ライブラリを呼び出せるよう開発された OpenCV for Android をインストールした．また，実機で OpenCV を利用したグラフィクス処理アプリケーションを実行するために必要となる OpenCV Manager をインストールした．

4.4 実験結果と考察

図 2 に実験結果を示す．図から明らかなように，本実験環境で Thin クライアントとしてサーバ側で処理をさせた場合には，Thick クライアントとして Android 端末に処理をさせたときと比較して半分以下の処理時間で処理できていることがわかる．ただし，この結果は単に端末とサーバの処理能力を比較するものであり，サーバ接続などの諸通信にかかる時間は含まない．

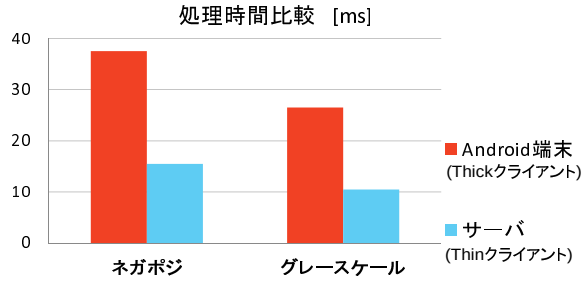

図 2　Thick/Thin クライアント処理時間比較

この結果から，本アプリケーションに関してはサーバで処理を行う Thin クライアントモデルの方が性能が大幅に良いことを示すことができた．本実験の考察として，1 枚の画像に対してこれだけの差が表れていることから，より大きな画像，もしくは 1 秒間に 20〜30 枚の静止画を連続表示する動画を処理対象にするアプリケーションでは処理時間差がより大きく表れると考えられる．このような Thin クライアント向けの処理をモバイル端末を利用してできるようにするという点で Thick/Thin クライアント切換え制御システムを提案することには十分な意義があると考えられる．

5. 切換え制御システム

5.1 システムの構成

ここで，本研究で提案する Thick/Thin クライアント切換え制御システムの概要を説明する．

本提案制御システムの動作を図 3 に示す．実験に用いたような Android 端末に不向きな処理を要するアプリケーションを起動する際に，本提案制御システムを介して，モバイル通信 (3G,LTE),Wi-Fi などといったネットワーク環境や端末性能情報，そして処理の重さなどといった情報を取得し，その情報を元にそのまま Thick クライアントの形でアプリケーションを起動するか，または Thin クライアントの形で起動するためにサーバへ接続しサーバ側に処理をさせるかを決定する．

本研究ではまず，この切換え制御システムを図 4 のよう

な切換えアプリケーションとして実装する．

このアプリケーションは，起動時にネットワーク環境・端末情報を取得する．そしてユーザに起動したいアプリケーションを選択させ，実行時の予測が可能な場合は選択されたアプリケーション情報を取得する．今後の評価実験により定める切換え基準と，得られたこれらの情報を元に Thick/Thin クライアントモデルどちら向けのアプリケーションなのか判定し，環境に応じた形でアプリケーションを実行するものとする．次節ではこのフレームワーク実現のために用いる機能を挙げる．

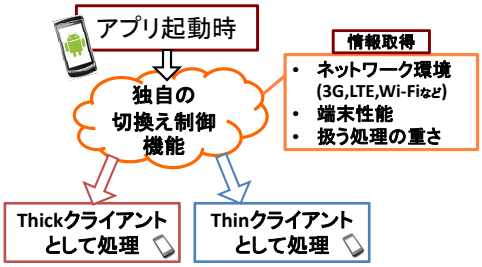

図 3　Thick/Thin クライアント切換え制御ソフトウェア

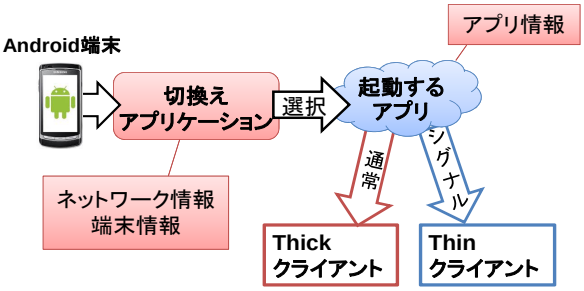

図 4　切換えアプリケーション概要

5.2　切換え制御システム実装方法

5.2.1　情報取得

本実験で実装した具体的な情報取得手法を説明する．ネットワーク情報としてネットワークの種類 (モバイル通信，Wi-Fi 等) を取得し，さらに Wi-Fi の場合は，RSSI(Received Signal Strength Indication) と呼ばれる受信信号強度を取得する．RSSI の単位は dBm で，RSSI 値は 1mW のときの信号強度を 0dBm としたときの相対数値となっており，取得した値が大きいほど電波強度が強いという指標となる．電波強度が必ずしも通信性能につながるわけではないが，強度が弱くなると通信性能も落ち込むと考え，ここでは電波強度を指標とする．これらの情報から，Thin クライアントとしてサーバに接続するために十分なネットワーク状況下であるかどうかを判断する．

次に，端末情報として，CPU の種類，性能，搭載メモリ容量，使用可能メモリ量，端末名などを取得し，選択したアプリケーションを実行するのに十分なメモリが確保できるか，端末に十分な処理能力が備わっているか等の判断基準とする．

以上に述べたネットワークや端末の情報取得機能の実装を Android 端末上で行った．選択したアプリケーション情報については，実行時に情報取得が可能な場合には，メモリ使用量等を取得し，切換え判断基準に追加する．Thick/Thin クライアントモデル切換えに必要な情報切換え判断基準は評価実験を重ねて定めていくことにする．

5.2.2　Thick クライアントとして動かす場合

Android 特有のインテントという機能を用いる．インテントとはアプリケーションの中の 1 つ 1 つの機能を橋渡しする仕組みのことであり，各アプリケーションはインテントフィルタに自身が反応できるイベントの種類を格納している [4]．インテントが発行されると，OS は端末内の全アプリケーションのインテントフィルタを調べ，そのインテントを処理できるアプリケーションを呼び出す仕組みになっている．ここで呼び出されたアプリケーションを選択し起動することで，本切換えアプリケーションを介して Thick クライアントモデルの形でアプリケーションを実行させることが可能となった．

5.2.3　Android NDK

次に，Thick クライアント側で起動する Android アプリケーションを作成するための手順を説明する．2 節で述べたように，Android アプリケーションは Java で記述されるため，サーバ側で動作するネイティブコードで書かれたプログラムを Android アプリケーション化するには大変な手間がかかる．そこで，本研究では解決策として Android NDK(Native Development Kit) を用いることにした．本節では，図 5 を用いて Android NDK の仕組みを説明する．

Android NDK とは Google 社が無償で公開している，Android 環境において端末のハードウェア上で実行できるネイティブコードのプログラムを開発するための環境のことであり，Java プログラムから JNI(Java Native Interface) を利用してネイティブコードを呼び出すことができる．ネイティブコードで記述された部分は Dalvik VM を介さずに実行されるため，アプリケーションの高速化につながり，また，ネイティブで記述されたコードの再利用が可能となる [5] ．

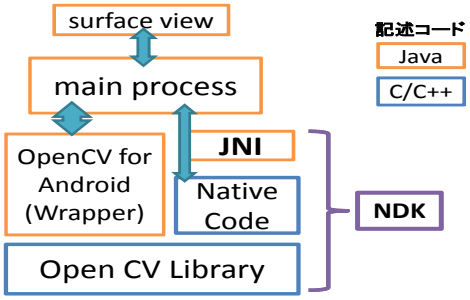

図 5　Android NDK の仕組み

Android NDK を用いたアプリケーションの構成を以下に示す．

```
《ホームディレクトリ》
 + apps
   + プロジェクトディレクトリ
   + JNI
     + Application.mk
     + Android.mk(makefile)
     + C/C++ファイル
     + ヘッダファイル
```

NDK を用いたアプリケーション作成に必要な手順は，JNI フォルダの作成，C/C++ファイルの一部書き換え，Android.mk と呼ばれる makefile の作成，Application.mk の作成と，Java 側からの呼び出し・画面表示部分の記述である．NDK を用いずにアプリケーション化させる場合と比べ，C/C++のファイルをすべて Java で書き換える必要がないということが大きなメリットといえる．

5.2.4　Thin クライアントとして動かす場合

サーバに命令を送るために socket 通信を利用する．具体的には，切換えアプリケーションで得た端末情報，ネットワーク情報等から Thin クライアントとして動作させることにした際に，起動したいアプリケーションを選択すると，そのアプリケーション名が socket 通信によりサーバに送られ，サーバ側は受け取ったアプリケーション名に対応するプログラムを実行するという仕組みである．同時に，Android 端末側ではデスクトップ共有アプリケーションを起動し，サーバ側の画面を表示することで処理結果を共有する．socket 通信を利用して，サーバ・クライアント間でデータ送受信を行い，サーバ側で指定プログラムを実行するためのプログラムをサーバ側とクライアント側にそれぞれ C 言語で実装した．このプログラムは，サーバ側・クライアント側でそれぞれ socket を生成/接続した状態でクライアントである Android 側からアプリケーション名をサーバ側へ送信し，サーバ側が受信したアプリケーション名と同名のプログラムを実行するものである．Android 側ではその処理結果を受け取り，表示する．このクライアント側のプログラムを Java で実装しなおし，切換えアプリケーションに組み込むことで，Thin クライアントとしてアプリケーションを動作させることが可能となる．

6.　今後の展開

6.1　実用例の提案

本提案制御システムの実用例として経路探索アプリケーションを挙げる．経路探索は道路ネットワークデータ (道路形状，接続情報，属性など)，地点情報データ (建物名称，住所，代表物など)，背景データ (河川，海，鉄道など) といったデータベースを要する．Thick クライアントとして動作させる場合は，端末上にあらかじめ地図データをダウンロードしておくことにより，いつ，どこでも端末上で経路探索を行うことが可能となる．一方，Thin クライアントとして動作させる場合は，サーバへクエリを投げ，サーバ側で経路探索を行い，処理結果を画面に表示するという形をとる．この場合，サーバ側で更新されている最新の交通情報を考慮した経路探索結果を受け取ることができる．

これは災害時における経路探索アプリケーションの利用にも応用できると考えられる．ネットワーク環境に影響が及ばない程度の災害時には，サーバを介して経路探索を行う Thin クライアントの形で避難経路探索を行う．一方，ネットワークが遮断されてしまうような大災害の場合，端末上に地図データさえあれば避難経路探索が可能である．しかし，地図データを持っていない場合やサーバ側で更新される最新交通情報を必要とする場合には遅延耐性ネットワーク (DTN:Delay/Disruption Tolerant Networking) を利用し，必要なデータを被災地の端末へと運び，端末上での経路探索を可能にする．これにより，本提案制御システムが有効的に実用可能であることを示す．

6.1.1　遅延耐性ネットワーク

遅延耐性ネットワーク (DTN:Delay/Disruption Tolerant Networking) について説明する．DTN とは，継続的なネットワーク接続が不可能な状況に耐えうる通信技術を幅広くとらえたものである．緊急災害時にはネットワークインフラが断たれ，必要とするデータや情報が得られなくなってしまう地域が生じる．そのような環境下で，端末とサーバが連携して動作するアプリケーションを利用可能にする DTN フレームワークが研究されている [6] ．このような DTN フレームワークを用いて，ネットワーク環境のある地域のモバイル端末でキャッシュした地図データや最新交通情報を，ネットワーク接続が困難な地域のモバイル端末へ伝達し，経路探索を行うことを実現できると考える．

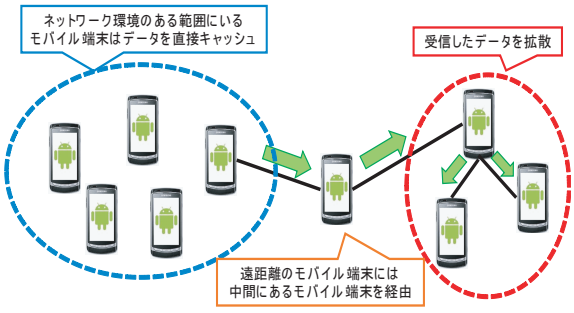

図 6　DTN を利用したデータ伝達の一例

6.2　経路探索システム

本研究ではネットワークシミュレータ Scenargie[7] に実装されたモジュールを用いて，切換え制御システムを備えた経路探索アプリケーションを作成した．本アプリケーションは，起動時に端末情報，ネットワーク情報等を取得し，環境に応じた形で経路探索モジュールを起動するもの

である．どちらの形でも起動できる場合にはユーザが希望する形で探索を行う．これを用いて，あらゆる通信環境を想定したエミュレーションを行っていく．

7. まとめと今後の課題

本研究では，Thick クライアントモデルとして動作する Android 端末を，ネットワーク環境や端末性能，起動するアプリケーションの処理の重さなどに応じて，Thin クライアントモデルとして動作させることで，両モデルのメリットを生かすことができると考え，Thick/Thin クライアント切換え制御システムを提案した．また，簡単な画像処理を行うプログラムを実装し，実験を行った結果，本実験環境においては Thin クライアント端末としてサーバ側で処理を行った方が端末上で処理を行うよりも実行速度が速く，サーバの処理性能が端末の処理性能を遥かに上回ることがわかった．この結果から，本提案制御システムを実装することに十分な意義があることが示せた．

また，本提案制御システムフレームワークを提案し，そのうちの情報取得部分，Thick クライアントとして動作させる部分，Thin クライアントとして動作させる部分をそれぞれ実装した．加えて，具体的な実用例として経路探索アプリケーションを提案し，起動時に取得するネットワーク環境や端末情報に応じた形で探索を行う，経路探索アプリケーションを Android NDK を用いて作成した．本アプリケーションは，地図データを端末にダウンロードし，端末上で経路探索を行う Thick クライアントとしての動作と，サーバにクエリを投げ，サーバ側で探索を行った結果を端末で表示する Thin クライアントとしての動作を兼ね備えるものである．

今後は，切換え制御自体がボトルネックになる可能性を踏まえて切換えアプリケーションの改良を行っていく．また，経路探索アプリケーションを用いて，様々な通信環境を想定したエミュレーションを行い，本提案切換え制御システムの性能評価を行うとともに，切換え判断基準となる情報の取捨選択・基準値の改定を行っていく．さらに，Thin クライアントとして動作させる場合のサーバ接続等にかかる通信量や端末の消費電力にも着目し，制御を行えるよう改良を行い，本提案制御システムの有効性を示すことを目標とする．

また，ネットワークが遮断されてしまうような緊急災害時を想定したエミュレーションを行い，DTN を利用して地図データや最新交通情報を運搬・拡散することを可能にする DTN フレームワークを考えていきたい．

謝辞

本研究を進めるにあたり，ご指導，ご協力賜りました UCLA の高井峰生先生に深く感謝致します．

参考文献

[1] シェア率：http://www.nikkei.com/article/DGXNASFK1301K_T11C13A1000000/
[2] アーキテクチャ：http://www.techfirm.co.jp/lab/android/outline.html
[3] 高橋竜男，高橋修，水野忠則:モバイル向けシンクライアントシステムの検討，情報処理学会論文誌，Vol.45 No.5 pp.1417-1431(May 2004)
[4] http://techbooster.org/android/application/8346/
[5] NDK：http://e-words.jp/w/Android20NDK.html
[6] 長谷川友香，高井峰生，小口正人:広域災害時に可用な Web アプリケーションのための DTN フレームワーク，DEIM Forum 2014 E7-1，2014 年 3 月
[7] Scenargie：http://www.spacetime-eng.com