Vol.2012-SLDM-155 No.20
Vol.2012-EMB-24 No.20
2012/3/3

Androidにおける遠隔サービス呼出し機能について

中尾和弘[†1]　中本幸一[†1]

Androidは，スマートフォンだけではなく組み込みシステムとして家電やカーナビ等への利用にも注目されており，今後はスマートフォンやAndroidを搭載した端末相互間における連携が必要になると考えられる．しかし，Android端末相互間における情報のやりとりについてはこれまであまり注目されてこなかった．そこで，本稿ではAndroid端末を用いたアプリケーション層での遠隔サービス呼び出し機能の試作を行い，有用性と課題について検討した．また，アプリケーション層以下での遠隔サービス呼び出しについて，Android OSレベルでの実装のためにBinderとAIDLについて考察し，どのような設計が考えられるか検討した．

On Remote Service Calls Based on Android

Kazuhiro Nakao[†1] and Yukikazu Nakamoto[†1]

Android has been attracting attention as the OS for not only smartphones but also embedded systems. It is necessary that smartphones and embedded systems which Android OS is installed have to coordinate with each other. However, sending and receiving information between Android devices has been gotten less attention. Therefore, in this paper we demonstrate a prototype of the function which calls remote services based on Android and make a feasibility study on utilities and problems. Moreover, we discuss Binder and AIDL to implement calling remote services under the application layer in Android OS and we examine possible architectures.

†1 兵庫県立大学大学院 応用情報科学研究科
Graduate Schoole of Applied Informatics University of Hyogo
〒650-0047 兵庫県神戸市中央区港島南町 7 丁目 1 番 28 計算科学センタービル 5-7 階
E-mail:aa10l208@ai.u-hyogo.ac.jp

1. はじめに

Google社の発表したAndroidは，スマートフォンやタブレット端末のプラットフォームとして広がってきている．ところが，その“ソフトウェア”としての実力と可能性が明らかになるにつれて，携帯電話以外の組込み業界全般からも注目される存在になっている[1)]．今後，組み込みシステムとして家電やカーナビなどにAndroidが採用された場合に，スマートフォンとの連携をはじめとする端末相互間の連携は，必要不可欠である．iPhoneを利用した端末相互間の情報のやりとりについては，先行研究として文献2) などがあるが，Android端末相互間での情報のやりとりについては，これまであまり注目されてこなかった．そこで本稿では，Android端末を用いたアプリケーション層での遠隔サービス呼び出し機能として以下の試作を行い，有用性と課題について検討した．なお，本稿で使用したAndroidのバージョンは，Android1.6である．

2. システム試作の目的と流れ

本稿の目的は，Androidを搭載したスマートフォンや，家電などの組み込み機器の相互間における協調動作のための遠隔サービス呼び出し機能の試作と検討である．具体的には，インテントを利用したネットワーク経由のアクティビティ起動，ネットワーク経由によるリソースの共有，画面データの共有，VNCサーバによる遠隔操作の有効性確認を試みた．

本稿では，以下の流れで試作を行った．

(1) ネットワーク経由のインテントによるアクティビティ起動
インテントはもともと，同一デバイス内での利用を前提としている．本稿では，ネットワーク経由でのインテント処理として，暗黙的インテントによるアクティビティ起動の容易性を検証した．

(2) ネットワーク経由によるリソースの共有
通常，Androidアプリケーションは，apk内のリソースを利用する．本稿では，apk内にリソースを保持せずに，サーバからダウンロードしたデータをリソースとして扱うためのプログラム設計と有効性を考査した．

(3) 画面共有システム
複数台の端末間で協調作業を行ったり，端末を操作したりする場合は，画面共有が必要となる．モバイル端末における画面共有には，複数のPDAによる画面共有システム[3)]や，iPhoneによる分散共有ワークスペース[2)]などがある．本稿では，Android

© 2012 Information Processing Society of Japan

Vol.2012-SLDM-155 No.20
Vol.2012-EMB-24 No.20
2012/3/3

端末における画面共有を目的とし，Drawable クラスと SurfaceView クラスによる描画処理と，VNC サーバによる画面データ転送を行った．

(4) VNC サーバ機能を用いた操作
VNC は，画面データを転送する他に，端末の遠隔操作を行う場合にも有効である．本稿では，Android1.0 に実装されていた VNC サーバ機能を 1.6 にマージし，VNC による遠隔操作の有効性確認を目的とした．

また，アプリケーション層レベルでの遠隔サービス呼び出しについて，フィージビリティを確認するとともに，今後アプリケーションに修正をしなくてもすむように AndroidOS レベルでネットワークを介したサービス呼出しの実装を検討する．

2.1 Android におけるネットワークプログラミング

Android 上でのプログラミングは Java 言語で開発を行うが，Java 言語での RPC 機能を提供する JavaRMI パッケージは含まれていないため，JavaRMI を用いた RPC を利用することができない．したがって，ソケットを利用したネットワークプログラミングが基本となる．プロセス間通信 (IPC) については，Android では後述のインテントとよばれる仕組みがある．本稿では，インテントとソケットを用いて試作を行った．

3. ネットワーク経由のインテントによるアクティビティ起動

3.1 インテント

インテントとは，Android におけるプロセス間通信の特徴である．Android Reference[4] の説明箇所を要約すると，“インテントは，Android アプリケーションのコンポーネントであるアクティビティ，サービス，ブロードキャストレシーバをアクティブ化する非同期のメッセージ” ということである．

インテントは，開発当初は同一デバイス内の利用に限定されていた[5]．しかし，現在では一人で複数台の Android 端末を所有することが多くなった．例えば，タブレット端末を家で使って，外ではスマートフォンを使うという場合などである．そのときに，タブレット端末からインテントをスマートフォンに送信し，タブレット端末で閲覧していた情報をそのままスマートフォンで見ることができるというように，複数の端末間でやりとりができれば便利である．そのような背景から，インテントを同一端末内だけではなく複数端末間でやりとりする仕組みが，試験的ではあるが Google から発表され，Android2.2 から実装された．その技術が，Cloud To Device Messaging(C2DM) である[6]．しかし，C2DM では Google の C2DM サーバを経由しなければならず，またブラウザ等の一部のアプリケーションしか対応していないため，本稿では C2DM を用いないで試作を行った．

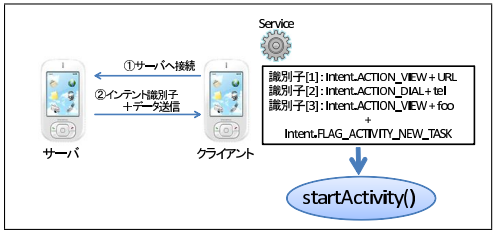

図 1 ネットワーク経由でのインテントによるアクティビティ起動の仕組み

3.2 明示的インテントと暗黙的インテント

通常，特定のアクティビティを起動する場合は，明示的インテントを用いる．しかし，明示的インテントは，同一アプリケーション内のアクティビティ遷移に使用されるのが一般的であり，呼び出し元と異なるアクティビティを起動する際は，暗黙的インテントを使用する．暗黙的インテントで，ブラウザや電話機能以外の特定のアクティビティを起動させるには，`AndroidManifest.xml` にインテントフィルタとして`<action android:name>`に該当するアクション，また`<data android:scheme>`にユニークな値（本稿では “foo”）を登録し，`<android:mimeType>`を削除する．加えて，当該アクティビティやサービス以外から別のアクティビティを起動する場合は，`addFlags(Intent.FLAG_ACTIVITY_NEW_TASK)` の記述が必要となる．これで，暗黙的インテントの引数として，scheme にインテントフィルタに登録した値を渡すことで，特定のアクティビティを起動させることが可能になる．

3.3 システム概要

システムの概要は，図 1 のとおりである．クライアントは，サービスからサーバへ接続する．サーバからは，インテント識別子とデータが送信される．クライアントは，受信したインテント識別子から，該当するアクションを決定する．また，データをパースして URI を得る．これらのアクションと URI を元に，アクティビティを起動する．

3.4 実行結果および考察

実行結果は，図 2，図 3，図 4 のとおりである．

インテントは，アクティビティ起動の他にもサービスを開始したり，ブロードキャストを開始することができる．本稿では，アクティビティ起動のみを実装したが，サービスやブロードキャストにも応用できると考えられる．

© 2012 Information Processing Society of Japan

Vol.2012-SLDM-155 No.20
Vol.2012-EMB-24 No.20
2012/3/3

図 2 実行結果 1
（起動画面）

図 3 実行結果 2
（ACTION_DIAL 実行）

図 4 実行結果 3
（特定アクティビティ起動）

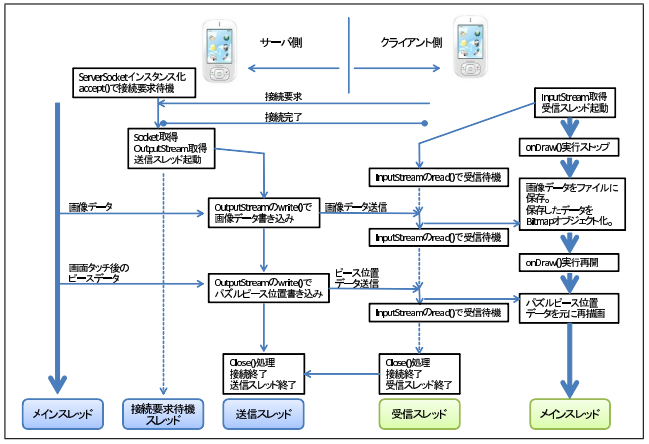

図 5 パズルゲームアプリケーション実行の流れ

4. ネットワーク経由によるリソースの共有

4.1 リソース

リソースとは，ローカライズされた文字列，ビットマップ，その他プログラムが必要とする小さなコード以外の情報のことである[7]．リソースは，res ディレクトリ下に作成し，ファイルフォーマット別にサブディレクトリに格納する．例えば，画像データは，res/drawable ディレクトリに格納し，画面レイアウトを記述する xml ファイルは，res/layout ディレクトリに格納する．リソースは，ビルド時にアプリケーションに取り込まれ，Android アプリケーションの実行形式である apk ファイル内に組み込まれる．通常アプリケーション実行時に必要なリソースは，この apk ファイルに組み込まれたものが使用される．

Android では，プログラム実行時にリソースが必要となるため，遠隔のサービス起動時にもリソース共有を容易に行うことができるかが問題となる．

4.2 システム概要

本稿では，クライアントにおいて res ディレクトリ下に画像リソースを保持せずに，サーバからダウンロードした画像データを元にアクティビティを起動する．アクティビティは，パズルゲームである．このパズルゲーム部分のプログラムは，文献 8) を参考にしている．サーバからダウンロードする画像データは，パズルの元となる png 形式のデータである．アプリケーション実行の流れは，図 5 の前半部分である．

4.3 実行結果および考察

実行結果は，図 6 のとおりである．

この方法を用いることで，res ディレクトリ下に画像リソースを保持せずに，ダウンロード形式で利用することができた．Android におけるリソースは，画像以外にも音声などがあり，それに応用することも可能である．

またリソースは，apk 内の res ディレクトリに存在し，その他に layout ディレクトリがあり，その下にアプリケーションの UI を定義する xml ファイル（以下，レイアウト xml）がある．レイアウト xml によって，アクティビティ上のボタンの大きさや場所などが定義されている．したがって，レイアウト xml をダウンロードし，そのレイアウト xml を元に UI を定義できるようにすれば，ユーザ独自の UI をカスタマイズすることが可能になると考えられる．

5. 画面共有システム

5.1 システム概要

本稿では，Drawable クラスを用いたパズルゲームと，SurfaceView クラスを用いた画像ビューア，VNC サーバ機能による画面データ転送の 3 つのシステムを試作した．

© 2012 Information Processing Society of Japan

Vol.2012-SLDM-155 No.20
Vol.2012-EMB-24 No.20
2012/3/3

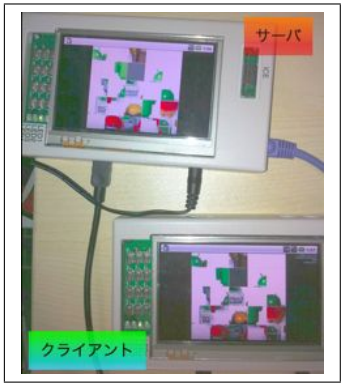

図 6 パズルゲーム実行結果

パズルゲームは，4 章での実装と同じものである．サーバで画面をタッチすると，タッチした部分が移動する．パズルのピース位置は int 型の配列で管理しており，タッチしてピースが移動した場合はこの配列が更新される．配列が更新されると，サーバはクライアントに対して更新した配列データを送信する．クライアントは，受信した配列データを元に画面の再描画を行う．具体的なアプリケーション実行の流れは，図 5 のとおりである．

画像ビューアは，サーバでタッチした座標データをクライアントに送信し，クライアントではその座標データを元に 2 種類の画面表示を行う．同一画面表示は，クライアント 2 台でそれぞれサーバと同じ画面を表示する．共有画面表示は，クライアント 2 台で 1 画面としてサーバの画面を表示する．

VNC の詳細については，6 章で述べる．本稿では，Android1.0 に実装されていた VNC サーバ機能を Android1.6 にマージした．VNC サーバを Android 上で実行し，クライアントでは VNC クライアントソフトを用いて画面表示を行った．

5.2 実行結果および考察

実行結果は，パズルゲームが図 6，画像ビューアが図 7，図 8，VNC による画面共有（パズルゲーム）が図 9 のとおりである．

VNC による画面共有と，パズルゲーム，画像ビューアの違いは，VNC が画面情報の差分を送信しているのに対して，後者はパズルのピース位置を示す配列や，画像表示位置を表す座標を元にクライアントで描画処理を行っている点である．つまり，VNC による画面共有では，多くの帯域を使用している．実際に，HVGA 解像度のエミュレータにて，VNC 経由でパズルゲームを行った場合の帯域使用量は，図 10 である．エミュレータ起動時 (40s) からパズルゲーム終了時 (100s) までの 1 分間で，平均して 1 秒間に $12868228bytes \div 60s = 214Kbytes$ のデータを送信している．

一方で，パズルのピース位置を示す配列では，TCP/IP のオーバヘッドを $20bytes$ と仮定した場合，一回のピース移動で送信されるサイズは，$20bytes + int\ data[24] = 20bytes + 4bytes \times 25 = 120bytes = 0.12Kbytes$ となる．

図 7 画像ビューア実行結果 1
（同一画面表示）

図 8 画像ビューア実行結果 2
（共有画面表示）

6. VNC による遠隔操作

VNC とは，Virtual Network Computing の略で，ネットワーク上の離れたコンピュータに対してリモートアクセスを行うソフトウェアである．VNC では，リモートアクセスされる側をサーバ，リモートアクセスする側をクライアントと呼ぶ．Android でのサーバアプリケーションとしては，“droid VNC Server”[9] などがある．

OS レベルでは，Android1.0 において，`/frameworks/base/libs/surfaceflinger` に VNC サーバ機能が実装されていたが，Android1.5 以上からは，その機能が削除されてい

© 2012 Information Processing Society of Japan

Vol.2012-SLDM-155 No.20
Vol.2012-EMB-24 No.20
2012/3/3

る[10)]

VNC サーバ機能が削除された理由として考えられることは，セキュリティの問題である．VNC サーバ機能を使用することができれば，スマートフォンを操作して他人へメールを送信したりすることも可能である．VNC サーバを利用することによるメリットとデメリットは以下のようなものがある．

- メリット
  メリットは，リモートアクセスによる端末操作が行えることである．例えば，企業のネットワーク管理者が，複数の Android 端末のメンテナンスや完全性検証[11)] などを行う際，VNC を利用してリモートアクセスで行うことによって時間やコストを削減できる．また，Android 端末を紛失した場合でも，リモートアクセスによって端末のロックや位置情報の取得などが可能になる．
- デメリット
  一番のデメリットは，前述のとおり，端末への不正アクセスなどの問題である．二つ目は，VNC の通信が暗号化されていないことである．そのため，通信内容を盗聴される可能性がある．しかし，これについては SSH や VPN を利用してセキュアな通信を行うことが可能である．三つ目は，画面データ転送による帯域使用量が多いことである．

6.1 Android1.6 への VNC サーバ機能のマージ

Android1.5 への VNC サーバ機能のマージは，すでにウェブページ 10) で行われている．しかし，Android1.6 以上へのマージについては，まだ行われていないようである．したがって，本稿では Android1.6 への VNC サーバ機能のマージを行った．Android1.6 に VNC サーバ機能をマージする際に，変更するソースコードは以下の 5 つである．

(1) RFBServer.cpp
(2) RFBServer.h
(3) SurfaceFlinger.cpp `framework/base/libs/surfaceflinger`
(4) SurfaceFlinger.h `framework/base/libs/surfaceflinger`
(5) Android.mk `framework/base/libs/surfaceflinger`

`RFBServer.cpp` と `RFBServer.h` は，ソースコードから削除されているので，Android1.0 のソースコードからコピーする．2012 年 01 月 07 日現在では，ウェブページ 12) からソースコードをダウンロードすることが可能である．

6.2 実行結果および考察

実行結果は，図 6 のとおりである．VNC サーバによる遠隔操作は，無線 LAN を使って

図 9 Android VNC サーバ機能実行結果
(パズルゲーム実行)

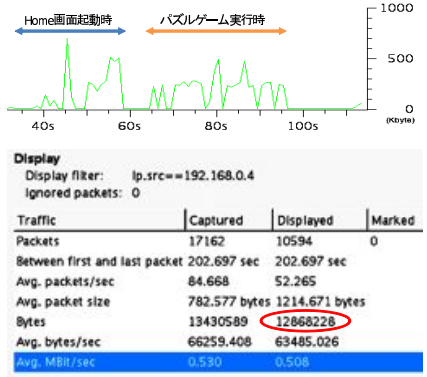

図 10 Android VNC サーバ実行時の帯域使用量

行ったが，レスポンスも十分実用的なものであった．しかし，描画更新が少ないパズルゲーム実行時でも，毎秒 214$Kbytes$ の帯域を使用した．ムービー再生など，画面更新が頻繁に行われるアプリケーションを実行した場合は，これよりさらに多くの帯域が必要である．したがって，今後は帯域使用量を削減する新たな RFB プロトコルが必要になると考えられる．

7. Android OS での実装の検討

これまでの試作は，ソケット通信を用いたアプリケーション層での実装であった．次のステップとして，Android OS レベルでネットワークを介したサービス呼出しの実装を検討する．利用するのは Android OS で提供している IPC 機能である Binder である．

まず，Binder について略述する．Binder は Palm 社による OpenBinder[13)] をベースにしていると言われ，OpenBinder の遠隔手続き呼出しの機能を Android のプロセス間通信 (IPC) に適用している．AIDL(Android Interface Denifition Language) で，リモートプロセスの手続きのインタフェースを記述し，当該プロセスを呼び出しに必要なクライアントプロキシとサーバスタブを生成する．例えば，図 11 のようにそれぞれ別プロセスに存

© 2012 Information Processing Society of Japan

Vol.2012-SLDM-155 No.20
Vol.2012-EMB-24 No.20
2012/3/3

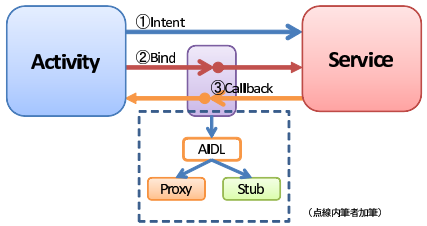

図 11 AIDL とアクティビティ／サービス間の関連

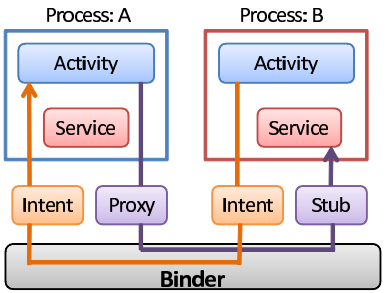

図 12 Binder によるプロセス間通信

在するアクティビティとサービス間で IPC を行う場合に，IPC のインタフェース記述を AIDL で行い，これをもとに`/frameworks/base/tools/aidl/generate_java.cpp` をテンプレートにしクライアントプロキシとサーバスタブが生成される．このクライアントプロキシとサーバスタブを利用して IPC を行う (図 12)．Binder を利用した IPC は実行する処理を引数で指定して，Binder クラスの transact() メソッドで実行する．Android のサービスでは提供する機能に応じて transact() メソッドを含むメソッドを拡張している．それは，`generate_java.cpp` 内に定義する場合，Binder クラスを継承してクラスを設ける場合がある．

一方，ソケット通信は socket クラスライブラリで管理している．ネットワークを介したサービス呼出しを Andoriod OS で実が元する場合，`generate_java.cpp` 内に定義するよりも，Binder クラスを拡張して，ソケットによる遠隔サービス呼び出しの Binder クラスを提供するのが自然と考えている．

8. まとめと今後の課題

本稿では，Android 端末を用いたアプリケーション層での遠隔サービス呼び出し機能として，ネットワーク経由のインテントによるアクティビティ起動，ネットワーク経由によるリソースの共有，画面共有システム，VNC サーバ機能のマージの 4 つを試作した．また，OS レベルで RPC の手法を取り入れるために Binder や AIDL について考察し，どのような設計が考えられるか検討を行った．

今後の課題は 2 つある．第一にアドホック通信での実装，第二に，RPC を組み入れることである．本稿では，検証に用いた端末上に無線 LAN や Bluetooth デバイスがなかったことから，有線 LAN によるルータを介した接続であった．しかし，スマートフォンのモバイラビリティを活かすには無線 LAN や Bluetooth によるアドホック通信での実装が必要である．また今後，Binder や AIDL に対して RPC の手法を取り入れた設計についてさらに検討し，実装していくつもりである．

謝　　辞

本研究の一部は，日本電気株式会社の支援を受けている．

参 考 文 献

1) 安藤恐竜：組み込みフレームワークとしての Android ハードウェア移植のツボ，SoftwareDesign，技術評論社，no.3，p.58，(2009).
2) 工藤聖広，辻野友孝，佐野博之，白松俊，大囿忠親，新谷虎松：複数スマートフォンを用いた分散共有ワークスペースの試作，第 24 回人工知能学会全国大会論文集 (CD-ROM)，1D3-3 (2010).
3) 野田敬寛，吉野孝，宗森純：GDA:複数の PDA による画面結合および共有システム，情報処理学会論文誌，vol.44, no.10 pp.2478–2489 (2003).
4) Android developers 〈Intent-filter〉 (online), `http://developer.android.com/intl/ja/guide/topics/intents/intents-filters.html` (accessed 2012-01-05).
5) Some information on APIs removed in the Android 0.9 SDK beta (online), `http://android-developers.blogspot.com/2008_08_01_archive.html` (accessed 2012-01-07).
6) Android Cloud to Device Messaging Framework (online), `http://code.google.com/intl/ja/android/c2dm/` (accessed 2012-01-07).
7) Eb Burnette，長尾高弘：初めての Android，p.22，オライリージャパン (2011).
8) 布留川栄一：Android プログラミングバイブル，pp.324–331，ソシム株式会社 (2011).
9) Android Market/droid VNC server BETA (online), `https://market.android.com/details?id=org.onaips.vnc&hl=ja` (accessed 2012-01-27).
10) Android VNC Server (online), `http://jfyi.blogspot.com/2009/05/android-vnc-server.html` (accessed 2012-01-03).
11) Index of /android/mydroid (online), `http://www.netmite.com/android/mydroid/` (accessed 2012-01-07).
12) 竹森敬祐：セキュリティ技術でスマートフォンの未来を支える〜環境へスマートに適応する携帯端末〜，情報処理学会論文誌，Vol.52，No.1，pp.38–39 (2011).
13) PalmSource:OpenBinder version 1.0 (2005). (online), `http://www.angryredplanet.com/~hackbod/openbinder/docs/html/index.html` (accessed 2012-01-30).

© 2012 Information Processing Society of Japan