B-037

携帯機器向け適応型サービスプラットフォームの提案と評価

A Proposal and Evaluation of Adaptive Services Platform

奥山 玄† 中川 好久† 才田 好則† 臼井 和敏†
Gen Okuyama Yoshihisa Nakagawa Yoshinori Saida Kazutoshi Usui

1 はじめに

近年, 携帯電話やスマートフォンなどの携帯機器において, ユーザが機能を追加・カスタマイズして使用する機会が増加している. しかし, 現状のアプリケーションダウンロードサービスやスマートフォンにおけるカスタマイズ機能は制約がある. すなわち, 機器ごとの独自 API による開発が必要であることがアプリケーション開発者に対しての制約である [1]. また, カスタマイズソフトウェアのインストール時に PC 連携が必要であり [2], 携帯機器ユーザに対しての敷居になっている. そしてこれらが開発効率やカスタマイズ性の低下の要因となっている.

我々は, アプリケーション開発者および携帯機器ユーザの負担を軽減することで, 開発効率およびカスタマイズ性の向上を図る, 適応型サービスプラットフォーム (Adaptive Services Platform: AS-PF) の研究を行っている [3]. AS-PF は, ユーザインタフェース (UI) とサービスモジュールの呼出し手順から構成されるアプリケーションと, アプリケーションから利用されるサービスモジュール (ライブラリ, ミドルウェア) を分けて管理する. また, それらアプリケーションとサービスモジュールを入替え可能とする. これにより, 開発効率およびカスタマイズ性の向上を実現する.

2 適応型サービスプラットフォーム (AS-PF) 概要

本章では, AS-PF の概念, 機能, および効果について述べる.

2.1 概念

AS-PF は, UI とサービスモジュールの呼出し手順から構成されるアプリケーションと, アプリケーションから利用される機能単位であるサービスモジュールを分けて管理する. 前者を AS アプリ, 後者を AS モジュールと呼ぶ.

AS アプリは, AS-PF を通して AS モジュールを利用できる (図 1). AS-PF は, AS モジュールの要求があった際, 必要に応じて検索・認証サーバからドメイン情報を取得する. AS-PF は, AS アプリおよび AS モジュールのドメインを比較することにより, AS モジュールの利用可否を判定する. また, AS モジュールが機器内にない場合は, サービス提供サーバから動的ダウンロードを行う.

2.2 機能

AS-PF は, 主要機能として, 以下に示す 4 つの機能を提供する.

- モジュール管理機能
  AS モジュールを動的ダウンロードし, AS アプリとリンクする機能を提供する. また, どこからも参照されなくなったときなど, 不要時の動的削除を行う. これにより, 必要なモジュールのみが機器内に残るため, 機器のユーザに応じた (適応した) モジュール管理が可能である.
- PF 間連携機能
  Java や Flash などの既存 PF から, AS-PF が管理する AS モジュールへアクセスするためのインタフェースを提供する. これにより, 異なる PF 環境に適応した連携が可能である.
- サーバ連携機能
  スタブとしての AS モジュールを介してサーバ上のサービスを利用する. また, サービスのキャッシュにより, オフライン制御を行う. すなわち, ネットワークが使用できない場合でもサービスを利用できる環境を提供する. これにより, ネットワーク環境に適応したサービス利用が可能である.
- コンテキスト制御機能
  ネットワーク状況, およびメモリ使用量などの端末内状況 (コンテキスト) から, AS モジュールの適正配置 (サーバに置くか携帯機器に置くか) を行う. また, 携帯機器のユーザや AS アプリの特性に応じて予測ダウンロード, プリフェッチを行う. これにより, 各々のコンテキストに適応した AS モジュール制御が可能である.

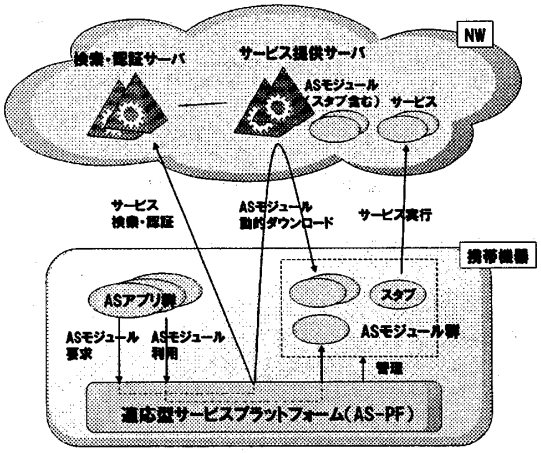

図 1: 適応型サービスプラットフォーム概念図

2.3 効果

AS-PF は, AS アプリと AS モジュールを分けて管理し, それらを入替え可能とする. これにより, 動的ダウンロード, 削除を行うことで, バージョンアップや不具合時のサービスモジュール入替えなどに対応できる. また, これらの処理を自動で行うため, 携帯機器のユーザは, サービスモジュールを意識する必要はない.

また, アプリケーション開発者は, AS モジュールを組み合わせることでアプリケーションを構築できる. 専門性の高い技術者が AS モジュールを作成して機能単位として提供し, アプリケーション開発者はそれを利用するという役割分担ができる. これにより, アプリケーション開発者の負担を軽減できる.

3 試作

我々は, AS-PF のアーキテクチャ検証を行うため, PC 上で試作を行った. 以降, 目的およびアーキテクチャについて述べる.

3.1 目的

本試作の目的は, 2 章で示したモジュール管理機能の動作検証を行うことである. すなわち, AS モジュールの呼出し方法や利用方法, および AS モジュールの動的ダウンロードを検証する.

3.2 アーキテクチャ

試作アーキテクチャを図 2 に示す. 図 2 の網掛け部分は, 今回試作したモジュールおよびインタフェースを示している. なお, 本試作では, AS アプリとしてネイティブアプリケーション, AS モジュールとしてライブラリを使用した.

† 日本電気 (株) システムプラットフォーム研究所
System Platforms Research Laboratories,
NEC Corporation

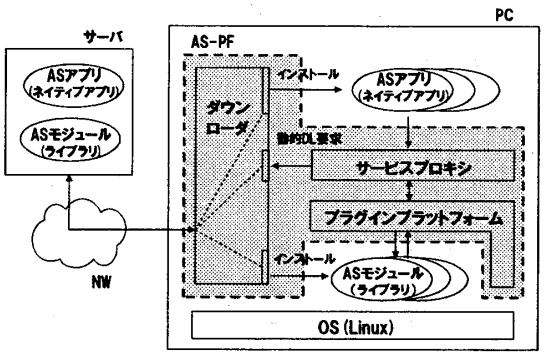

図 2: 試作アーキテクチャ

```
// ASモジュール利用要求
mod_num = aspf_request("http://foo.bar.com/serv/sample");

// ASモジュール利用
aspf_use(mod_num, "sum", &ret1, "%d %d", x, y);
aspf_use(mod_num, "square", &ret2, "%d", x);

// ASモジュール利用終了
aspf_close(mod_num);
```

図 3: AS モジュール利用例

AS アプリは, サービスプロキシを通して AS モジュールを要求する. サービスプロキシは, AS モジュールを検索し, 機器内にない場合はダウンローダに動的ダウンロード要求を行う. ダウンロードした AS モジュールはプラグイン PF が管理する. 同時に, AS モジュールは AS アプリにリンクされ, 利用可能な状態となる.

3.3 AS アプリ記述例

AS モジュール (ライブラリ) の利用例を図 3 に示す. AS アプリは, AS モジュール利用要求時, AS モジュールの URL を指定する. ここで返されたモジュール番号 (図 3 中の *mod_num*) を指定し, AS モジュール利用や終了処理を行う. このように, AS モジュールの呼出しを組み合わせることで AS アプリを作成できる.

4 評価

我々は, AS-PF のカスタマイズ性能評価として, AS モジュール要求時の動的ダウンロードにかかる時間計測評価を行った. 本章では, 評価内容, および評価結果・考察について述べる.

4.1 評価環境

評価は, PC2 台をネットワーク (10BASE-T/100BASE-TX) で接続した環境で行った. PC1 上で AS-PF が動作し, PC2 上で HTTP サーバが動作する. 各 PC のスペックを表 1 に示す.

4.2 評価項目, 評価方法

本試作における評価は, AS モジュールが PC1 にインストールされていない状態で AS アプリから AS モジュールを呼び出す. これにより, AS アプリからの AS モジュール呼出し, および PC2 からの動的ダウンロードを評価する.

評価方法は, 図 3 で示した AS アプリ内の AS モジュール利用要求前後に gettimeofday() 関数を挿入することで, AS モジュール呼出しにかかる時間を測定した. AS モジュールのサイズは, 100KB, 500KB, 1MB の 3 種類を用意した.

表 1: マシンスペック

-	PC1	PC2
CPU	Intel Pentium3 866MHz	Intel Pentium4 3.2GHz
RAM	256MB	512MB
OS	Redhat Linux 9	Redhat Linux 9
HTTP サーバ	-	Apache/2.2.6

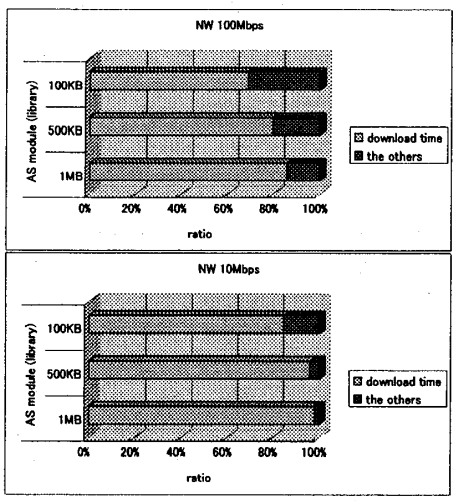

図 4: 動的ダウンロード時間の比率

4.3 評価結果と考察

AS モジュールを呼び出し使用できる状態になるまでの時間に対し, 動的ダウンロードにかかる時間の比率を図 4 に示す. これを参照すると, 動的ダウンロード時間がその他の時間 (モジュール間通信など) に比べかなりの割合を占めていることがわかる.

この動的ダウンロードの時間比率は, AS-PF を携帯機器に適用した場合, より高くなることが予想される. それは, 携帯機器は無線でネットワークに接続することが多く, その特性から不安定になることがあり, 常に高速通信ができるとは限らないからである.

このため, AS モジュール要求時は, すでにインストール済みで即座に呼び出せる状態であることが望ましい. AS モジュールは, モジュール管理機能の 1 つである動的削除の対象にならない限り, 一度インストールされれば以降の動的ダウンロードは発生しない. よって, 図 4 に示したように, 携帯機器内のオーバヘッドは動的ダウンロードに比べて少ないため, その分高速に呼び出せる.

上記を実現するためには, 携帯機器のユーザや AS アプリの特性から必要な AS モジュールを判断し予測ダウンロードを行うような仕組みが必要である. これにより, 携帯機器ユーザからみた TAT(Turn Around Time) を減少することが可能である.

5 おわりに

本稿では, AS-PF の提案, およびカスタマイズ性能評価として AS モジュールの動的ダウンロード評価について述べた. 本 PF は, AS アプリと AS モジュールを分けて管理し, それぞれを入替え可能とすることで, 開発効率およびカスタマイズ性の向上を図る.

今後は, AS モジュールの予測ダウンロード機構を検討していく. 予測ダウンロードにより, 携帯機器ユーザに対して動的ダウンロードを意識させることなく, 携帯機器上の機能として提供することを目的とする.

参考文献

[1] 中道理. “特集 1　携帯電話新時代へ”. 日経コミュニケーションズ. 2008/02/01 号, p.44-54.

[2] Microsoft Corporation. “Microsoft Windows Mobile - 機能概要 - ”. http://www.microsoft.com/japan/windowsmobile/wm6/prodinfo/customization.mspx, (参照 2008-07-01).

[3] 奥山玄, 中川好久, 才田好則, 渡邊光洋, 臼井和敏. “携帯機器向け分散環境適応型サービス提供プラットフォームの提案”. FIT2007(第 6 回情報科学技術フォーラム), 2007.