組み込み用メールミドルウェアの一考察

田中 功一*　　泊　陽一郎*

*三菱電機（株）情報技術総合研究所

携帯電話向けメールシステムなど、組み込みソフトウェアの構造で重要な項目として、新しいターゲット装置、たとえば次機種開発に迅速に適用あるいは移行できることがある。特にソフトウェアの基本構成を変更することなく、機能拡張が実施できることが望まれている。

これらのソフトウェアは、ハードウェアの制約や短期間での開発の結果、十分にアーキテクチャの検討をせずに機能拡張を進める傾向があり、開発完了時にはモジュール間インタフェースが複雑で、移植性の低いソフトウェア構成となるケースが多い。

我々は、機種依存性に配慮し、メーラとしての基本機能を備えた「組み込み用メールミドルウェア」の検討を進めた結果、機種依存部の分離と機能の抽象化に基づくインタフェースを持つミドルウェア・コアを設計した。ここでは設計方針、基本機能の定義について報告する。

An Implementation of Middleware for Imbedded Mail Systems

Koichi Tanaka*　Yoichiro Tomari*

* Information Technology R & D Center, Mitsubishi Electric Corp.

We have developed a middleware for mail systems, which provides basic functions and interfaces for implementation of embedded mailer systems. We separated device dependent parts and others clearly, and designed a core layer that consists of abstraction mechanisms and mail content parse functions. We can use this middleware in the development of network appliances, PDA's, and cellular phone systems. The existence of this middleware will decrease future development costs and software bugs.

1　はじめに

インターネットに接続し、各種のサービスを実現する携帯情報機器の開発の増加に伴い、組み込み機器向けのソフトウェア開発も多くなされるようになった。

これらの開発は、製品のライフサイクルの観点から、開発期間が他のソフトウェア開発と比較し短期間であり、低位ドライバからユーザインタフェースまで、広範囲に渡り大規模な開発を特徴としている。携帯電話に代表されるメール送受信機能ソフトウェアの開発も例外ではなく、機能追加を行いながら改良開発が続いている。

組み込みソフトウェアの構造で重要な点のひとつとして、新しいターゲット装置、たとえば次機種開発に迅速に適用あるいは移行できることがある。その際、ソフトウェアの基本構成を変更することなく、機能拡張が実施できることが、良いソフトウェア部品としても望まれている。

しかしながら、実際には限られたハードウェア資源と、短期間での開発の繰り返しから、十分な検討をする時間がなく、次機種開発を進めなければならないこともある。この場合，開発完了時にはモジュール間インタフェースが複雑で、移植性の低いソフトウェア構造となり、課題を残すことになる。

そこで、機種依存性に配慮し携帯情報機器用としてのメールミドルウェアの開発を進めることとした。本稿では、設計方針、基本機能に

ついて検討したので報告する。

2　背景と課題

組み込み機器用のメーラの開発では、メモリなど利用可能な資源の観点から、ソフトウェアの極限的小型化を要望されることが多く、アーキテクチャが重要視されるとは言いがたかった。

これらはモジュール間結合度の高いソフトウェア構成となるため、機種依存性が高く再利用性に欠くソフトウェア構造となる。構造が複雑ゆえ、テストのコストも増大し品質への影響も少なくない。しかしながら、過度の構造化もまた、性能劣化につながりかねない側面も持っているのは事実である。

我々は、メーラを実現するための基本機能の整理に基づき、機種依存部分と非依存部分を明確に分離し、機能実現に必要な「コア」部品を備えたメールミドルウェアを設計することとした。メールミドルウェアは、継続的なメールソフトウェア開発における改修を最小限にし、開発コストの低減と品質の向上を目的とする。

3　基本方針

まず、メールシステムのレイヤ分割と、汎用化可能な機能範囲「メールコア部」を決定するために、メーラ機能を以下の４つとコア部に大別し、それぞれ汎用性、機種依存性について検討した。コア部以外の４つの部分は、機種依存性の高さを尺度として、分割の対象とした。

■　ユーザインタフェース処理

ユーザ操作を下位レイヤへ伝達、メーラ機能を実現する部分。画面表示や操作などユーザインタフェースは、他との差別化を図るために機種の特色が強く反映される部分である。機種依存性が非常に高いことから、ユーザ操作による一連の処理制御を行う部分はメールコア部に組み込まない方が独立性が高くなる。

ここでは、ユーザインタフェース処理部がコマンドを組み合わせて機能を実現できるよう、メールコア部は機能を抽象化した汎用的なインタフェースを提供することとする。

■　プロトコルアクセス処理

メールの送受信など、伝送プロトコルの制御。携帯情報機器の場合、一般にメールを伝送するプロトコルには統一性が無く、事業者ごとに独自のプロトコル拡張が行われるケースが多い。このような独自の伝送プロトコルは事業者のサービス展開方針により変更されていくと思われるため、メーラに対する影響を抑えるには本処理をメールコア部には組み込まない方が望ましい。

すなわち、メールコア部で機能を抽象化したインタフェース(実装要求 API)を定義し、プロトコルアクセス処理部は組替え可能な部品として設計する方が汎用性は高まる。

■　ストレージアクセス処理

メッセージデータ保存/取得などローカルストレージへのアクセス処理。事業者毎のストレージ管理ポリシーの違いや携帯情報機器特有の厳しい容量制約/性能制約など、非常に機種依存性が高い部分であるため、本機能はメールコア部には組み込まない方が望ましいと判断した。

プロトコルアクセス処理と同様、メールコア部で機能を抽象化したインタフェース(実装要求 API)を定義することとする。

■　フォーマット変換処理

指定フォーマットに従ったメッセージデータの生成/解析処理。メールフォーマット規定が RFC 準拠もしくはその拡張という形になりつつあることから、本機能は汎用メーラ内部に組み込めるものと判断した。

一般的な RFC2822[1][2]解析をデフォルト解析ルールとし、メール本文の解析及び整形(生成)を行う機能を検討する。また、拡張性を高めるため MIME[3][4]解析／独自仕様メール解析／文字コード変換等のライブラリ追加も可

能とする。

■　メールコア部

以上のことから、メールコアは「コアモジュール API」と「フォーマット変換処理」を機能範囲とし、上位レイヤ（ユーザインタフェース）と下位レイヤ（プロトコルアクセス処理/ストレージアクセス処理)をつなぐフレームワークとして位置付けることとした。

4　メールミドルウェアの設計

コア部で実現する機能は、コアモジュール API 群と、フォーマット変換処理 API 群として外部から参照される。ここではコア部の機能およびプロトコルアクセス処理、ストレージアクセス処理の設計に関して述べる。

■　コアモジュール API 群

まず、コアモジュール API 群はユーザ操作を抽象化した上位向け API と、プロトコルアクセス/ストレージアクセスを抽象化した下位実装要求 API を持ち、上位レイヤからの要求を下位レイヤの処理へマッピングする役割を果たすものと定義できる。

レイヤとしては層が薄いものとなるが、メールコア部をフレームワークとして取り入れることで以下の利点があると考えた。

- ✓ コアモジュールを上位レイヤ（ユーザインタフェース）と下位レイヤ（プロトコルアクセス処理/ストレージアクセス処理）間に置き、統一したインタフェースを定義することにより、今後の組み込み系メーラ開発におけるレイヤ分割基準を明確に提示することができる。
- ✓ レイヤ分割基準を明確にすることで、各モジュールの部品化を促進し、独立性、再利用性を高めることができる。
- ✓ 上・下レイヤの関係が過度に密接になるのを防ぐ。これにより障害や仕様変更などによる改修の影響範囲を局所化することが可能。

以下にコア部が提供するインタフェース検討の際に留意した点を記述する。

・　サーバ接続

サーバ通信時の API 抽象化について検討した。まず、「メールを１件送信する」という操作を考えた場合、以下の２とおりのインタフェースが考えられる。

① アプリケーションはサーバ接続を意識しない
　アプリケーションは、送信を指示するのみで、コア部とプロトコルアクセス部がサーバ接続、送信、サーバ切断を実施。

② アプリケーションがサーバ接続を意識
　アプリケーションがサーバ接続、送信、サーバ切断処理を実施、コア部およびプロトコルアクセス部もそれに応じで処理を実施。

①はアプリケーションからのコマンド数が少なく操作性が良いように感じる。しかし、現状の携帯情報機器ではサーバアクセスには課金が伴うことが多く、バックグラウンドで自動的に接続するのではなく、画面表示などでユーザに接続を確認するインタフェースが一般的と思われる。

このことから、サーバ接続についてはアプリケーションがサーバ接続のタイミングをコントロール可能な②のインタフェースとする。

また、プロトコルに IMAP4[5]を採用していれば、サーバ上のフォルダやメッセージを操作することが可能となる。この場合もサーバ上のフォルダ・メッセージ操作を行う前に、アプリケーション主導でサーバへ接続するインタフェースする。

・　文字列検索

メーラの標準的な機能として、メール本文や件名に指定した文字列を含むメッセージを検索し表示する機能が考えられるが、本機能は複数の機能の組み合わせで実現可能である

（例：メッセージ検索→メッセージ取得→メッセージ解析→文字コード変換→文字列比較・判断）。

また、文字列検索中はユーザに処理中であることに知らせるために何らかの画面表示を行うケースが想定されるが、それらの制御処理は機種依存性が高く共通化は難しい。

従って、コアモジュールでは文字列検索 API 自体は提供せず、組み合わせることで文字列検索が実施可能となるような API を用意することとする。

・ サーバからの Push 受信

事業者にも依存するが、携帯情報機器でのメール受信は、まず始めに短いメッセージデータで着信通知を受け、ユーザが選択してメール全体（または一部分）を受信する方式が一般的である。下位伝送プロトコルから Push 型で受けたデータをアプリケーションに渡す機能は、携帯情報機器のメーラで必須といえるため、コアモジュールの機能範囲とする。

また、サーバからの Push 型受信は、メールの他にも各種情報通知やサービス利用許可/非許可通知など、様々な用途に使用されることが予想される。

それらはメールデータだけに限定はできず、データ種別/内容は機種依存性が高いため、識別・振り分け処理はコアモジュールの機能範囲外とする。メールシステムのプロトコルスタックで振り分けるか、コアモジュールより上位のアプリケーションで振り分ける方式をとる必要がある。

■ フォーマット変換モジュール群

フォーマット変換モジュール群はメッセージの生成と解析を行うためのモジュールとして部品化する。この部品化によって以下の利点があると考えた。

✓ 現状での標準である RFC2822 形式のフォーマットや MIME 解析等は、機種に依らず再利用することができる。
✓ 事業者固有のフォーマットがある場合も、その追加や削除が容易となる。

以下、メールシステムとして必要なフォーマット変換機能について検討する。

まず、現在の携帯情報機器ではインターネットメールの送受信に対応している機種が一般的といえるため、インターネットメール標準規格である RFC2822 解析を基本とする。

しかし、インターネットメールのフォーマット仕様も事業者等によって独自拡張されている場合があり、RFC2822 標準解析処理のみでは都度カスタマイズが必要となってしまう。

また、携帯電話等のメールシステムには標準フォーマットや機種固有フォーマットなど複数の変換機能が必要であり、それらは事業者のサービス展開に対応して取捨が容易に行えるべきであるため、共通のインタフェースを持つ組替え可能な部品として設計することとした。

結論として、フォーマット変換モジュールは、上位に対して RFC2822 などメッセージ解析処理、指定されたフォーマット種別でのメッセージ生成、漢字コードなど変換機能を提供する。

■ プロトコルアクセス処理部

メールプロトコルとして一般的な SMTP／POP[6]／IMAP をモデルとしてプロトコルアクセス処理の機能を検討した。

まず受信プロトコルである POP／IMAP のコマンドを比較すると、IMAP の特徴であるサーバ上でのメール操作以外は機能的に非常に類似性が高く、抽象化することによってプロトコルの違いを隠蔽することが可能であると考えた。

そこで、「送信のためのプロトコル」「受信のためのプロトコル」と分けるのではなく、「送信と受信のためのプロトコル」を想定し、

どのように抽象化していくかを検討する必要があると考えた。

上記の検討結果を踏まえ、プロトコルアクセス処理部は上位に対してサーバの接続切断、メッセージ格納用フォルダの操作、メッセージ操作、および処理の中断処理など抽象機能を提供するモジュールとして設計する。

■　ストレージアクセス処理部

ストレージをメールアプリケーションから利用する場合は高速検索性も要求されるため、シリアルフラッシュメモリなど低速ストレージ使用時のアクセス速度を補うデータベースを組み込み、ファイルシステムとデータベースを統合管理するブロックの検討を行った。

ストレージアクセス処理部の機能は、

・ファイルシステムが提供する「保存」「読出」「消去」機能
・データベースが提供する「検索」「論理情報の保持」機能

を組み合わせたものと考えた。

ここでは、複数のストレージデバイスを一本化して処理するものではなく、デバイス毎にモジュール化することを検討した。デバイスの違いを個々のモジュール内に限定することで追加や削除を容易にするため、上位からは「どのストレージデバイスを利用するか」を意識するだけで良いというメリットがある。

上記の検討結果を踏まえ、ストレージアクセス処理部は上位に対してフォルダの作成や削除、メッセージの保存、削除、属性の変更など、抽象機能を提供するモジュールとして設計する。

5　メールミドルウェアの構成

以下、図１に従い、各部分の機能に関して説明する。

■　アプリケーション部

ユーザ操作に従い、コアモジュールAPIを呼び出し、コアモジュールからのイベントをハンドリングする。

メールコア部の上位レイヤには、実装対象によって多少変化するが、基本的には画面を表示するメールUI部､画面表示を行わないミドルウェア的なメール非UI部､メールシステム以外のアプリケーション(データ転送､Pushハンドラなど）を想定した。

■　コアモジュールAPI群

ユーザ操作を抽象化した上位向けAPIと、プロトコルアクセス/ストレージアクセスを抽象化した下位実装要求 API を持つフレームワーク。

■　フォーマット変換モジュール群

メッセージの生成と解析を行うためのモジュール群。メッセージタイプごとに部品化する。上位アプリケーションからの呼び出しはコアモジュールAPIを通し、インタフェースを共通化する。

■　プロトコルアクセス処理部

コアモジュールAPIからの要求により、通信プロトコルの制御を行う。実装対象で採用している伝送プロトコル（Push通知、SMTP、POP、IMAP など）に対応して部品化し、それらを組み合わせて実装可能な仕組みとする。

■　ストレージアクセス処理部

コアモジュールAPIからの要求により、下位ローカルストレージの制御を行う。データ記憶デバイスを意識せずにファイルやデータベースにアクセスする。また、データの整合性について保証する仕組みを持つ。

■　プロトコルスタック群

プロトコルコマンドの送受信制御を行う。Push型通信プロトコルスタックやTCP/IPプロトコルスタックなどが該当する。

■　ストレージ

データ記憶デバイスにアクセスし管理するファイルシステム・簡易データベースに相当する。実際の記憶媒体はシリアルフラッシュメモリ、不揮発 RAM、リムーバブルメディアなどである。

6　まとめ

機種依存性に配慮した組み込み用メールミドルウェアの設計を行い、方針に基づき構成、基本機能の定義について検討した。

その結果、ミドルウェア・コア部およびインタフェース、ユーザインタフェース部、プロトコルアクセス処理部、ストレージアクセス処理部、フォーマット変換処理部など大きく 4 つの機能ブロックとして全体を構成する方式を決定した。この構成では機種特化処理部分とメーラを開発するために必須の基本部分を明確に分離しており、拡張性が高いメーラ開発プラットフォームとなっている。

今後は情報家電など携帯情報機器の開発で本メールミドルウェアの適用を考えており、既存ソフトウェアとの連携を含めた開発としていく予定である。また、インスタントメッセージングなど、比較的リアルタイム性があるメッセージ交換アプリケーションへの適用も検討し、全体の性能を加味したチューニングも進める予定である。

Reference

[1] D. Crocker: Standard for the format of ARPA Internet text messages, RFC822, 1982

[2] P.Resnick: Internet Message Format, RFC2822, 2001

[3] N. Freed 他:　Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, RFC2045, 1996

[4] K. Moore: MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text, RFC2047, 1996

[5] M.Crispin: Internet Message Access Protocol – Version 4rev1, RFC2060, 1996

[6] J.Myers他: Post Office Protocol – Version 3,　RFC1939, 1996

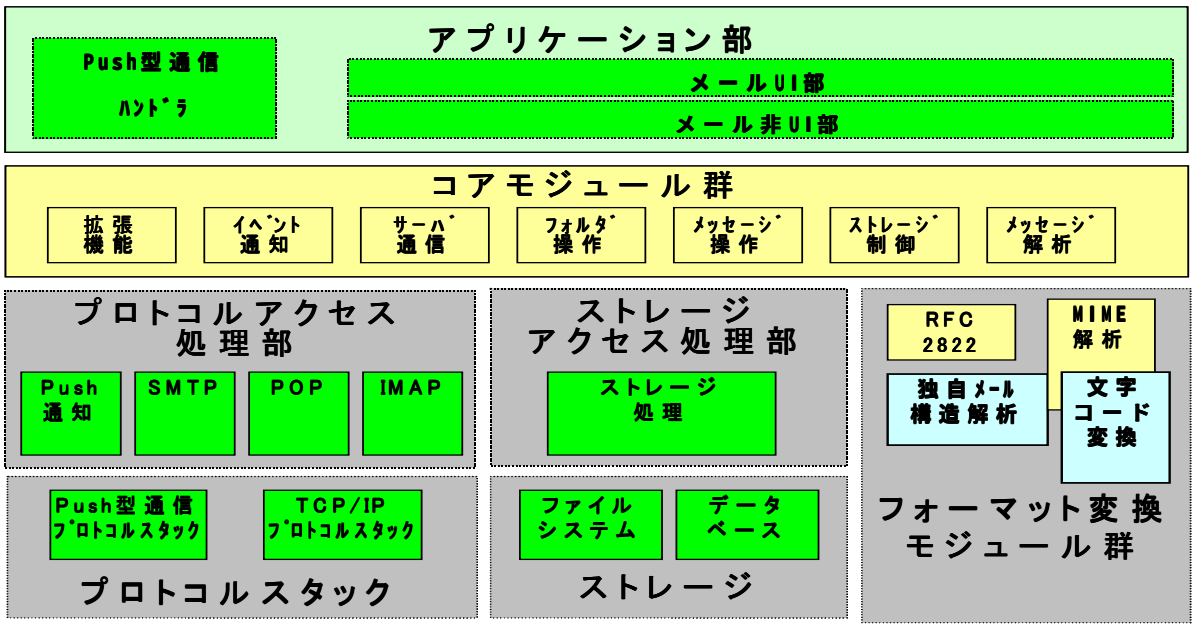

図 1　メール・ミドルウェアの構成