テクニカルノート

組み込み用Webサーバのモジュール構造

峯村治実† 山口智久†
大野次彦† 下間芳樹†

Webアプリケーションの普及と組み込みデバイスの高機能化を背景として，近年，注目されている組み込み用Webサーバについて，機能を部品として入れ替え可能なモジュール構造の検討を行った．Webサーバの機能は，HTTPプロトコルで定義されるメソッドおよびヘッダによって処理が大きく異なるため，メソッドとヘッダを処理単位とするモジュールの組合せによってWebサーバを構成する方法をとることにした．具体的なモジュール化のメカニズムとしては，プログラム実行効率の面から，今回はDLLをベースに実現する．今後，Windowsをはじめとする種々の環境で試作・評価を行っていく予定である．

A Module Structure for Embedded Web Servers

HARUMI MINEMURA,† TOMOHISA YAMAGUCHI,† TSUGIHIKO OHNO† and YOSHIKI SHIMOTSUMA†

Increase of Web-based applications and performance improvement of embedded devices are main reasons for a rise in needs for embedded Web servers. We studied a module structure for embedded Web servers to realize easy replacement of their functions. Our method is based on DLL, and we adopted HTTP methods and headers as modularization unit. We are planning to develop experimental systems and to evaluate them on various platforms in future.

1. はじめに

インターネットと安価なPCの普及により，Webブラウザをインタフェースとするアプリケーションシステムが増えてきている．これらのシステムには，Webブラウザの統一的なユーザインタフェースによる操作の容易性と，ほとんどすべてのPCにWebブラウザが標準的にインストールされており，クライアント側に特別なソフトウェアが不要という利点がある．

一方，組み込み機器の高機能化により，モデムによるPPP接続やEthernetなど，ネットワーク接続機能を持った組み込み機器が増えてきている．これによりネットワークを経由して，組み込み機器の情報の取得や制御を遠隔から行うことが可能になりつつある．

上記のような背景から，Webサーバ機能を組み込み機器に内蔵し，遠隔からWebブラウザを用いて機器の制御や情報取得を行う，いわゆる組み込み用Webサーバという考えが現れ，製品もいくつか出てきている．我々も，組み込み用Webサーバの試作と評価により，その有用性の検証を行いつつある[1),2)]．

このような組み込み用Webサーバに必要な要件の1つとしてモジュール構造がある．組み込み機器に内蔵されるマイコンシステムは一般にメモリが少なく，組み込みシステムそれぞれに必要とされる機能にも違いが多いため，ソフトウェアの機能を部品として入れ替え可能なモジュール構造が必要となる．また，機能の追加・改版のためにも，個々のモジュールを容易に追加・入れ替えできる必要がある．

本稿では，RFC[3)]に基づいてWebサーバの機能を分析し，それに従って，組み込み用Webサーバのモジュール構造について考察した結果について述べる．

2. モジュール構造の実現方式

モジュール構造を実現するための方法として，これまで多くの方式が提案されてきている．これらのうちよく使われている方式として，以下の4つがある．

† 三菱電機株式会社情報技術総合研究所ネットワークコンピューティング部
Network Computing Department, Information Technology R&D Center, Mitsubishi Electric Corporation

(1) DLL (Dynamic Link Library)
(2) COM (Component Object Model)
(3) CORBA (Common Object Request Broker Architecture)
(4) Java クラスライブラリ

DLL は，関数のエントリポイントをあらかじめ静的にリンクしておき，そこから DLL 本体のエントリポイントを再び呼び出すことによって，擬似的に動的なリンクを実現している手法である．現状，Windows 環境に限定されているが，使用メモリ量や性能の面でプログラムの実行効率は高い．

COM[4]や CORBA[5]はオブジェクト指向の考え方を導入し，分散システム環境でオブジェクトどうしがメッセージを交換するためのメカニズムを定義している．プラットフォームには依存しないことになっているが，COM が事実上，Windows 環境でしか使えないように，実際にはかなりプラットフォーム固有の処理を必要とする．また，機構が複雑なため，DLL に比べて使用メモリ量は多くなる．

一方，Java クラスライブラリ[6]は，仮想マシン Java VM (Virtual Machine) 上で動くインタープリタ言語のモジュールであり，プラットフォームへの依存性は非常に少ないが，インタープリタであるために性能面では不利であり，Java VM のサイズが数 M バイトになるなど，メモリ使用量が大きいという欠点がある．

性能と使用可能なメモリサイズに余裕のある 32 ビットマイコンを内蔵した組み込み機器では Java を用いるのが汎用性・保守性などの点で望ましいが，現実には，まだ 16 ビット以下のマイコンを内蔵した機器も多く，それらの上でも動作する Web サーバを実現するには，使用メモリ量が少なく，高性能なモジュール化の方式が必要となる．このため，このような比較的低コストの組み込み機器向けのモジュール化方式としては，Windows 環境 (組み込み用としては Windows CE) では DLL，それ以外の環境では DLL 相当の機能を独自に作り込むのが，最も現実的と考えられる．ただし，プラットフォームによっては DLL 相当の機能を開発するのが困難なものもあり，そのプラットフォームに適した，より簡易的な手法により実現することも考慮する必要がある．

本稿では，この DLL をベースとしたモジュール化について検討した結果を述べる．比較的大容量のメモリと性能を持つ組み込み機器用の，Java をベースとしたより汎用的なモジュール化方式については別途報告する．

```
リクエストメッセージ=
  リクエスト行
  *(一般ヘッダ | リクエストヘッダ | エンティティヘッダ)
  CRLF
  [エンティティボディ]

リクエスト行=
  メソッド URL HTTPバージョン CRLF

レスポンスメッセージ=
  ステータス行
  *(一般ヘッダ | リクエストヘッダ | エンティティヘッダ)
  CRLF
  [エンティティボディ]

ステータス行=
  HTTPバージョン ステータスコード 理由フレーズ CRLF

*(): 0回以上の繰り返しを示す.
[]: 省略可能であることを示す.
CRLF: 復帰(¥r)・改行(¥n)コード.
```

図 1 リクエストとレスポンスの構造
Fig. 1 Structure of request and response.

3. 組み込み用 Web サーバの機能分析

3.1 HTTP プロトコル

Web サーバと Web ブラウザ間のデータは，HTTP プロトコルにより送受信される．Web ブラウザから送られるリクエストメッセージに対して，Web サーバがレスポンスメッセージを返すだけの単純なプロトコルである．リクエストメッセージおよびレスポンスメッセージの構造を図 1 に示す．図中，メソッドは，クライアントがサーバに要求する動作の内容を示す．また，エンティティボディは送受信されるデータ (HTML ファイルなど) であり，一般，リクエスト，レスポンス，エンティティの各ヘッダは付随的な情報の送受信に用いられるものである．

RFC では，HTTP プロトコルのメソッドやヘッダなどについて，Web アプリケーションでの実装レベルを「必須」(must),「推奨」(should),「オプション」(may) の 3 段階に分けているが，組み込み用 Web サーバには固有の特性があり，必ずしもこの「必須」機能のみを実装すればよいことにはならない．以下，メソッド，ヘッダそれぞれについて必要性を検討する．

3.2 メソッド

メソッドには，GET, HEAD, POST, PUT, DELETE, OPTIONS, TRACE の 7 種類がある．これらのうち，GET は HTML ファイルなどの静的コンテンツと CGI を用いた動的に生成されるコンテンツの両方の転送に用いられるメソッドであり，組み込み用 Web サーバでも必須である．また，HEAD はコンテンツ自体ではなく，コンテンツに関するメタ情

報（サイズや生成時刻など）のみを転送するメソッドで，必ずしも必須の機能ではないが，アプリケーションによっては使用することもあるため，選択できるようにする必要がある．

POSTは，CGIなどの機能拡張モジュールに比較的多くのデータを渡すために用いられる．GETでも代用可能であるが，Webブラウザがフォームの送信に使うことが多いため，選択可能にする必要がある．

PUTとDELETEは，コンテンツの入れ替えおよび削除に用いられるメソッドであり，遠隔からのコンテンツのメンテナンスに必須の機能である．

OPTIONSとTRACEは，サーバの能力問合せとデバッグのために用いられるメソッドであり，実際にWebブラウザから送られることはないため，優先度は低く，実装する必要はないと考えられる．

3.3 ヘッダ

ヘッダは，RFCでは一般，リクエスト，レスポンス，エンティティの4種類に分類されているが，ここでは主なヘッダをその用途によって**表1**のように分類し，それぞれの必要性について検討する．

(1) 接続制御ヘッダ：HTTP/1.1は“Persistent Connection”をサポートしており，TCP/IPの接続を切断せずに複数のコンテンツを送受信するようになっている．ただし，HTTP/1.0以下のWebブラウザではこの機能がサポートされていないため，“Persistent Connection”のオン・オフを切り替えるための接続制御ヘッダが必須である．

(2) 転送制御ヘッダ：転送制御としては，コンテンツをチャンクと呼ぶ固まりで送受信するためのチャンク形式エンコーディング機能がある．監視カメラの映像など，転送開始時に長さをあらかじめ知ることができないコンテンツの送受信を行う場合に要求される機能であり，必要に応じて選択できるようにすべきである．

(3) 条件制御ヘッダ：更新時刻などの条件を指定してコンテンツの取得を行うためのヘッダであり，必要になるケースもあると考えられるため，選択可能にしておく必要がある．

(4) 認証制御ヘッダ：ベーシック認証などの認証機能を実現するためのヘッダであり，セキュリティレベルに応じて，選択できるようにする必要がある．

(5) コンテンツ情報ヘッダ：受信可能なコンテンツ種別の指定や，送信するコンテンツ種別の通知に用いられるヘッダで，必須の機能である．

(6) キャッシュ管理ヘッダ：元々，小容量のメモリしか持たない組み込み機器では，コンテンツをキャッシュすることはできない．また，監視・制御の性格上，動的に生成されるコンテンツが大部分を占めると考えられる．このため，キャッシュ管理ヘッダは基本的に不要である．

(7) Webブラウザ必須ヘッダ：上記の各種ヘッダのほか，“Date”など，Webブラウザ側で必須のヘッダは，サーバ側でサポートしないとWebブラウザと接続できない可能性があるため，実装が必要である．

表1 ヘッダの分類
Table 1 Classification of headers.

分類	ヘッダ
(1)接続制御	Connection
(2)転送制御	Transfer-Encoding
(3)条件制御	If-Modified-Since, If-Unmodified-Since, If-Match, If-None-Match
(4)認証制御	Authorization, WWW-Authenticate
(5)コンテンツ情報	Accept, Accept-Charset, Accept-Encoding, Content-Encoding, Content-Type
(6)キャッシュ管理	Age, Cache-Control, Expires
(7)Webブラウザ必須	Date, Host

4. 組み込み用Webサーバのモジュール構造

メソッド，ヘッダとも，上記のとおり必須のものもあれば，必要に応じて選択可能にすべきものもある．必須のものについても，HTTPプロトコル自体の改版や性能改良などのために入れ替えられるようにしておく必要がある．

また，Webサーバの処理フローは**図2**に示すようになり，メソッドに応じた処理，ヘッダに応じた処理を独立して入れ替え可能にする必要がある．

以上の考察に基づいて，組み込み用Webサーバのモジュール構造の検討を行った．**図3**にその構造を示す．図において，管理モジュールはモジュール管理（登録，追加・削除など），クライアントとの接続管理およびメソッド処理モジュールの呼び出しを行い，各メソッド処理モジュールは，それぞれ必要なヘッダ処理モジュールを呼び出して処理を行う．また，共通処理モジュールは，クライアントからのデータ受信や文字列操作など，共通的な処理を行うモジュールであり，メソッド処理モジュールとヘッダ処理モジュールの両方から呼ばれる．

メソッド処理モジュールはメソッドテーブルに登録され，ヘッダ処理モジュールは各メソッドのヘッダテーブルに登録される．また，共通処理モジュールは，共通処理テーブルに登録される．各モジュールは，それぞれのテーブルを参照して呼び出される．

各メソッドおよびヘッダの処理は，リクエストメッセージ中のメソッド名やヘッダ名の検出をトリガにして起動される．メソッド名やヘッダ名の検出処理ルー

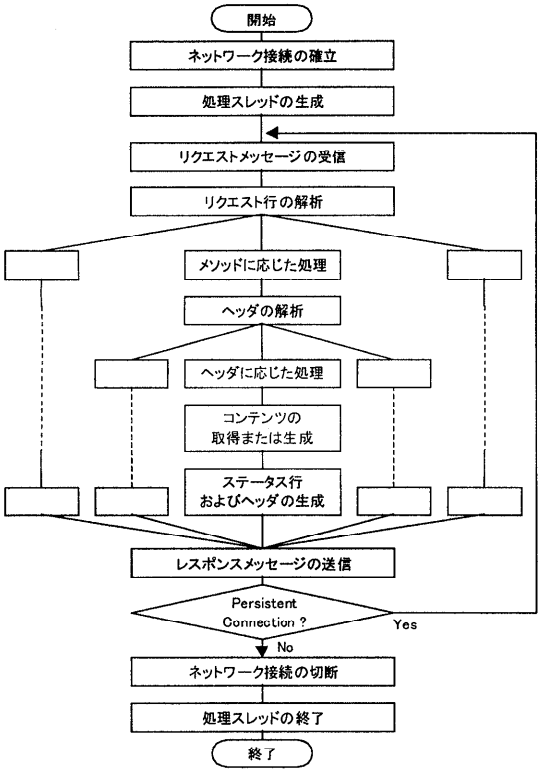

図 2 Web サーバの処理フロー
Fig. 2 Process flow of Web server.

チンは，モジュールの登録時に各モジュールの初期化ルーチンによって，共通処理モジュールの 1 つであるトークン検出ルーチンに登録される．

各モジュールは，Windows 環境では DLL として実装可能である．DLL の関数エントリを関数へのポインタの配列として，各テーブルに登録する．

5. おわりに

以上，組み込み用 Web サーバのモジュール構造について考察した．Web サーバの特性を考慮して，メソッドおよびヘッダ単位の処理モジュールをテーブルに登録することにより，モジュール構造を実現する．

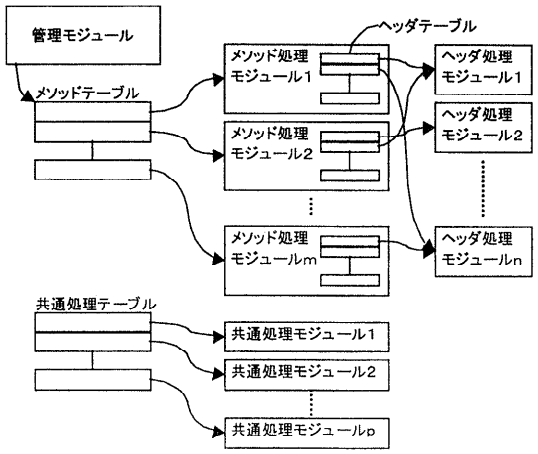

図 3 組み込み用 Web サーバのモジュール構造
Fig. 3 Module structure for embedded Web server.

今後は，この考察結果に基づいて詳細な実装方式の検討を行うとともに，種々のプラットフォーム上で試作・評価を行っていく予定である．

参考文献

1) 橋詰ほか：組み込み Web サーバ用プラットフォーム，第 58 回情報処理学会全国大会論文集，1, pp.103–104 (1999).
2) 山口ほか：組み込み用 Web サーバの試作と評価，第 58 回情報処理学会全国大会論文集，1, pp.105–106 (1999).
3) Fielding, R., et al.: Hypertext Transfer Protocol-HTTP/1.1, Request for Comments 2068 (1997).
4) Microsoft Corporation: The Component Object Model Specifications (1995).
5) Object Management Group: CORBA Services Specification (1997).
6) Sun Microsystems Inc.: http://java.sun.com.

(平成 11 年 4 月 26 日受付)
(平成 11 年 9 月 2 日採録)