3M-9　ヘッダ処理機能を柔軟にカスタマイズ可能なSOAPプラットホーム

根山 亮　　中村 祐一
日本アイ・ビーエム(株)東京基礎研究所

1. Webサービスと SOAP

Webと XML[1]の普及により、インターネット上での企業間の取引 (B2B) が今後ますます加速してゆくと考えられている。特にB2Bでは、取引相手や取引のインターフェースがあらかじめ固定されている従来型の企業間取引とは異なり、取引相手を動的に発見しその相手の持つインターフェースに柔軟に対応できる仕組みが必要とされている。このような仕組みを実現するには、(1) 取引の参加者が利用する共通のプロトコルを定め、(2) かつそのプロトコルを取引形態に合わせて柔軟に拡張できることが重要である。

これらの要件を満たすため、XMLをHTTPでやり取りするサービス(Webサービス)を定義し (要件(1))、Webサービスの定義したXMLを運ぶための拡張可能なプロトコルとして SOAP (Simple Object Access Protocol) [2]を用いる (要件(2)) ことが、B2Bにおける標準になりつつある。

SOAPでは、メッセージ本体とともに複数のヘッダを付加できる。アプリケーションのセマンティックスとは独立した汎用的なヘッダを定義した場合、SOAPを処理するプラットホームがそのヘッダを統一的に処理できることが望ましい。またSOAPでは、サービスの要求者と最終到達先サービスの間でやり取りを仲介する仲介サービス (Intermediary) が定義されており、プラットホームがこれらを考慮して設計されていることも必要である。

我々は、SOAPが持つ拡張性に注目し、メッセージのパス上に機能ごとに分離された複数のメッセージ・プロセッサを並べることによりヘッダ処理機能や仲介サービスを柔軟にカスタマイズできるSOAPプラットホーム (TRLSOAP) を設計、Java言語で実装した。TRLSOAPは現在Web上で公開されている[3]。

2. SOAPの拡張性とプラットホームに必要な機能

2.1 SOAPの拡張性

図1に示すように、SOAPではアプリケーションが使うXMLを運ぶための封筒 (**Envelope**) が定義されており、Envelopeは要求の最終到達先サービスのセマンティックスとは独立して定義できるヘッダ (**Header**) と最終到達先サービスに向けられた本体 (**Body**) から構成されている。

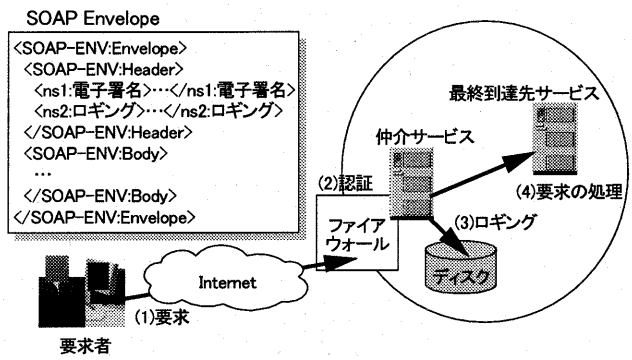

図1：ヘッダ処理を含むアプリケーションの例

Headerは Header エントリと呼ばれる子供要素を複数持つことができる。Headerエントリは XMLの名前空間[4]により修飾されるためアプリケーションが自由に定義できるようになっており、例えば図1のようにメッセージ単位の認証のための電子署名 (2) や、通過したメッセージをディスク上に保存しておき後で必要になったときに取り出して検証するロギングの指示 (3) を格納するために利用できる。

また、サービスの要求者と最終到達先サービスの間に仲介サービスを置くことで、最終到達先サービスに対するメッセージのセマンティックスとは独立して、メッセージ認証やロギングのような処理を透過的に追加できる。仲介サービスは、メッセージの型 (XMLの要素名など) に応じて最終到達先サービスを動的に選択するルーターとして使うこともできる。

2.2 SOAPプラットホームに必要な機能

SOAPメッセージを処理する汎用的なプラットホームを考えた場合、2.1節で述べたようなSOAPの拡張性を最大限に利用できる機能が提供されていることが望ましい。我々は、SOAPプラットホームに必要な機能は主に以下の3つであると考える。

(1) 仲介サービスを透過的に追加できる仕組み
(2) リクエスト/レスポンス、一方向 (One-way) のようなメッセージのやり取りのパターンをアプリケーションが容易に扱える仕組み
(3) HeaderやBodyを処理するプログラムを機能ごとの小さなコンポーネントとして実装し、それらを組み合わせることで全体のサービスを構成できる仕組み

A SOAP Engine Flexibly Customizable for Header Processing
Ryo NEYAMA, Yuichi NAKAMURA,
IBM Research, Tokyo Research Laboratory

(1)により、最終到達先サービスのプログラムを変更することなく、ロギングのような付加的な機能を追加できるようになる。また、要求者が直接最終到達先サービスのアドレスを指定せずルーターにメッセージを送ることで、要求者がWebサービスの最終到達先を知らなくてもサービスを受けることが可能になる。

SOAPで定義されているメッセージのやり取りのパターンは一方向だけである。しかし、実際にはアプリケーションのセマンティックスは一方向以外だけでなくリクエスト／レスポンス、同期／非同期のようなやり取りのパターンの上に定義することが多い。特にSOAPで主に利用されるHTTP-POSTの場合、同期のリクエスト／レスポンスが使われることが多い。そこで(2)のように基本的なやり取りのパターンを処理する機能をプラットホームが提供することにより、アプリケーション開発者の負担を減らすことができる。

(3)により、例えば電子署名やその検証のような汎用的な機能をコンポーネントとして実装し、Webサービスを配置する際に、管理者がこれらを組み合わせ、新たにプログラムを書くことなくWebサービスを柔軟にカスタマイズできるようになる。

3. アーキテクチャと実装

我々の提案するSOAPプラットホームのアーキテクチャを図3に示す。ここでは、要求者から仲介サービスを経て最終到達先サービスまでのパスをメッセージの通るパイプとして捉え、要求者、仲介サービスおよび最終到達先サービスの各コンポーネントがパイプを構成するブロックになっている。これにより、要求者と最終到達先サービスの間に複数の仲介サービスを自由に抜き差しすることができる（要件(1)を実現）。

パイプを構成するブロックは、リクエスト/レスポンスのようなやり取りのパターンごとに用意され、セッション・プロセッサと呼ばれる。アプリケーションのセマンティックスはセッション・プロセッサのAPIに自然に対応付けられる(要件(2)を実現)。例えば、リクエスト/レスポンス用のセッション・プロセッサのAPIは以下のようになる。

```
SOAPMessage onMessage(SOAPMessage request)
    throws SOAPFaultException;
```

ここでは、メソッドの引数にリクエストが渡され、リクエストを処理したセッション・プロセッサがメソッドの戻り値としてレスポンスを返す。メッセージの処理中に起きたエラーは例外として処理される。また、各送信先に対応付けられた複数のトランスポートの前にメッセージの送信先を制御する1つのセッション・プロセッサを置くことでルーターを実現できる（要件(1)を実現）。

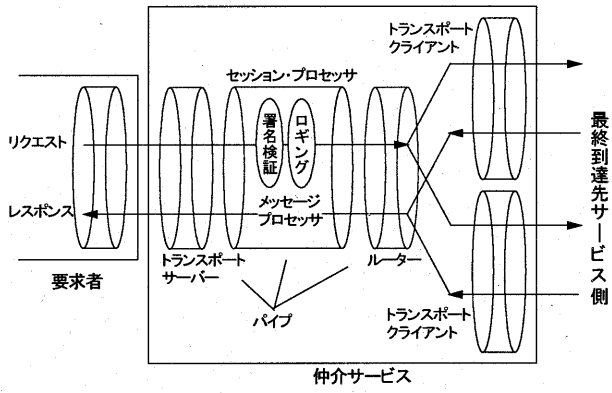

図3：SOAPプラットホームのアーキテクチャ

セッション・プロセッサは、内部にメッセージ・プロセッサと呼ばれるリクエストやレスポンスのようなメッセージを処理するコンポーネントを複数差し込めるようになっている。ここに署名検証やロギングのような機能を持つメッセージ・プロセッサを差し込むことができる（要件(3)を実現）。

このような仕組みは仲介サービスだけでなく、要求者や最終到達先サービスでも同様に利用できるため、管理者はより要求者と最終到達先サービスを柔軟に結びつけることができる。例えば、要求者と最終到達先サービスのいずれかの側にアダプタを差し込むことで、インターフェースの異なる2者がメッセージをやり取りできるようになる。

4. まとめ

本研究ではSOAPのヘッダ拡張機能と仲介サービスに注目し、これらの拡張性を最大限利用できるSOAPプラットホームを実現した。TRLSOAPで実現した機能は今後Apache SOAPエンジン[5]に組み込まれていく予定である。今後はこのSOAPプラットホームを基盤として、Webサービス同士の動的な結合を実現することが期待される。

参考文献

[1] Extensible Markup Language (XML), http://www.w3.org/XML/.

[2] Simple Object Access Protocol (SOAP) 1.1, http://www.w3.org/TR/SOAP/.

[3] TRLSOAP, http://www.trl.ibm.co.jp/projects/xml/soap/trlsoap/.

[4] XML名前空間, http://www.w3.org/TR/1999/REC-xml-names-19990114/.

[5] Apache SOAP プロジェクト, http://xml.apache.org/soap/.