南山大学 数理情報学部 情報通信学科 卒業研究 2004 年度卒業論文要旨集

非同期型 Web サービスに関する研究

2001MT021 服部 隆尚　2001MT066 森 晃

指導教員 青山 幹雄

1. はじめに

現在 Web サービスは, 組織間システム連携技術として注目されている. しかしビジネスプロセスの連携は多岐にわたり, 多くの問題がある. なかでも Web サービスは, 同期型であるため, 長時間の処理や, 一方向のサービス利用には適さない. 本研究では, 非同期型の Web サービスを提案, 実装して, その有効性を評価する.

2. 同期型 Web サービスの問題点と解決策

2.1. 同期型 Web サービスの問題点

同期型 Web サービスの多くは, トランスポートプロトコルに HTTP (HyperText Transfar Protocol)を使用した SOAP (Simple Object Access Protocol) over HTTP が使用される[1]. クライアントは, 図 1 に示すように, レスポンスメッセージが返信されるまで, 実行を待ち合わせる. その間, 待ち状態にあるクライアントは長時間, 他の処理ができなくなる. 同期型 Web サービスでは, サービスの形態によって長時間クライアントのリソースを消費する問題がおこる. さらに, 同期型 Web サービスには, 一方向のメッセージ送信ができない問題点もある.

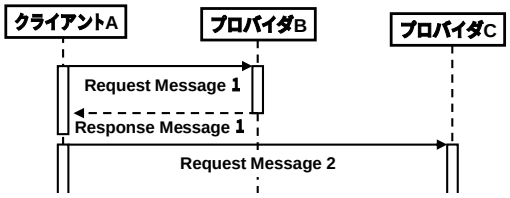


図 1: 同期型 Web サービス

2.2. 非同期型 Web サービスを用いた解決策

同期型 Web サービスの問題を, 非同期型 Web サービスを用いて解決する方法を提案する. 非同期型 Web サービスでは, 図 2 に示すように, クライアントからの一方向型のリクエストメッセージ送信が可能である. そのため, レスポンスメッセージを待たず, 次々にリクエストメッセージを送信できる. この結果, 長時間の処理を必要とするサービスや, Web 上に通信網等の障害が発生してもクライアントが待ち状態にならない Web サービスを提供できると考える.

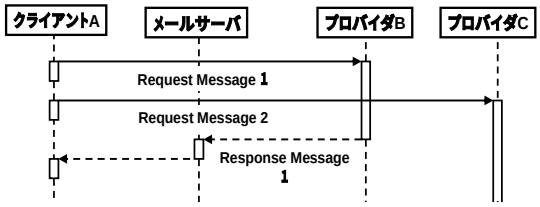


図 2: 非同期型 Web サービス

3. 非同期型 Web サービスの実現

3.1. SOAP over SMTP

SOAP メッセージを非同期で送信する場合, トランスポートプロトコルの選択が問題となる. 本研究では以下の理由により SMTP(Simple Mail Transfar Protocol)を用いた SOAP over SMTP を実現する.

(1) 一方向のメッセージングが可能
(2) 企業間などのファイアウォールを透過
(3) 既存のメールサーバを使用可能

3.2. SOAP over SMTP による非同期メッセージング

SOAP over SMTP の非同期メッセージングを図 3 に示すアーキテクチャで実現する[2]. サービスプロバイダにメールサーバを使用し, 電子メールと同様にメールアドレスを指定して SOAP メッセージを送信する.

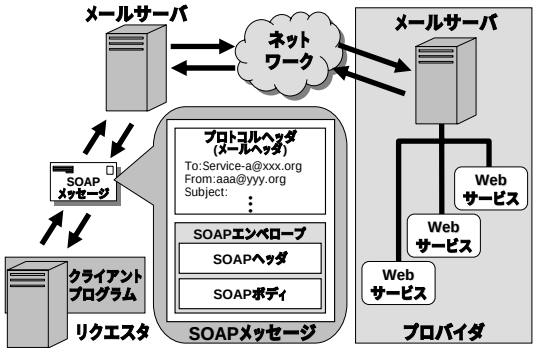


図 3:SOAP over SMTP による非同期メッセージング

3.3. SOAP メッセージの識別

SOAP メッセージの非同期での送受信は, 一方向のみのメッセージングを行うためリクエストとレスポンスの対応関係を明確にし, メッセージを識別する必要がある.

メッセージの識別には, メッセージにユニークな ID を付加する方法が利用可能である. SOAP メッセージへ ID を付加する方法として以下の 2 つが考えられる.

(1) SMTP ヘッダへの Message-Id･In-reply-to の付加

SMTP ヘッダ部に Message-Id を付加して送信し, それに対応するレスポンスメッセージには, リクエストメッセージの Message-Id の値を In-reply-to に付加して返信する.

(2) SOAP ヘッダへの ID 要素の付加

SOAP ヘッダに ID 要素を定義し, レスポンスメッセージにはリクエストメッセージに付けられた ID を付加して返すことにより, リクエストとの対応関係を明確にする.

(2)の方法は, 単体のリクエストに対して複数のレスポンスに同一の ID を付加するなどの, 一対多の対応関係や, リクエスト/レスポンス以外の関連性のあるメッセージに対して対応関係を示すことが可能である. そこで, 本研究では SOAP ヘッダに ID 要素を付加する方法を用いる.

4. SOAP over SMTP の実装

4.1. Apache Axis を用いた SOAP over SMTP の実現方法

SOAP over SMTP を用いた非同期型 Web サービスを実現するために, Apache Axis を拡張して SOAP over SMTP を実装した.

Apache Axis は図 4 に示すように, メッセージの処理を行う Handler と, 一連の Handler を纏めた 3 つの Chain で構成されている[3]. 各 Chain は責任を分離し, 独立した処理を行う. Apache Axis に SOAP over SMTP を実装するには, メッセージを送信するための処理を行う TransportSender と, 受信したメッセージを Axis Engine に渡す TransportListener を SMTP 用に変更することで実現できると考える.

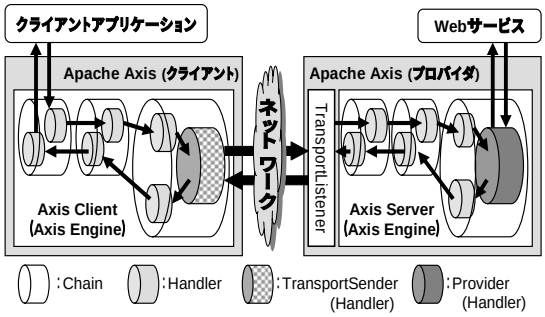


図 4:Apache Axis のアーキテクチャ

SOAP over HTTP では, TransportListener に HTTP サーブレットを使用しているが, SOAP over SMTP ではメールサーバ上で Axis Engine を起動しなければならないため, サーブレットは使用できない. そこで, 本研究では図 5 に示すように TransportListener に Apache James の Mailet を用いて, メールサーバに届いたメッセージを受信し, AxisEngine に渡す方法を提案する[4].

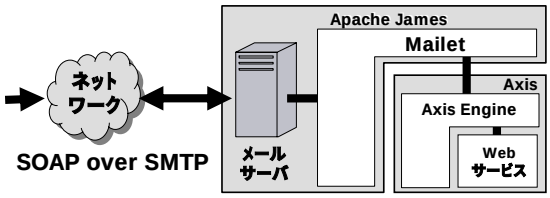


図 5:Apache Axis と Apache James によるメッセージング

4.2. Apache Axis を用いた非同期メッセージの送信

SOAP over SMTP で SOAP メッセージを送信するために図 6 に示すように, クライアント側の Apache Axis に SMTP 用の TransportSender である SMTPSender を実装･配置した. SMTPSender では, 各 Handler で処理された SOAP メッセージのバインディングヘッダに SMTP ヘッダを付加して, メールサーバにリクエストメッセージを送信する処理を行う.

クライアントアプリケーションはリクエストメッセージを送信するときに, トランスポートに"smtp"を, 宛先にプロバイダのメールアドレスを指定することで, SMTPSender を用いた SOAP メッセージの送信を可能にする.

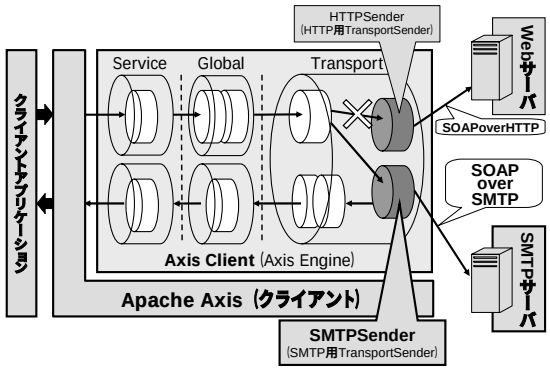


図 6:クライアント側のメッセージパス

4.3. Mailet を用いた Web サービスプロバイダ

前節で説明した方法を用いて SOAP over SMTP で送信された SOAP メッセージを受信し, AxisEngine を起動して Web サービスを呼び出すために, 図 7 に示すように Apache James の Mailet を用いて, TransportListener に Apache Axis 用の Mailet である AxisMailet を実装･配置した.

AxisMailet は, Apache James メールサーバに, Web サービス用のメールアドレス宛にメッセージが届いたときに起動するように Apache James の設定ファイルを変更することで, サーバが受信する他のメールと区別をする.

前節の SMTP 用 TransportSender の実装を含む, Apache Axis への SOAP over SMTP の実装で作成した Java プログラムのソースコードは, 計 411 行となった.

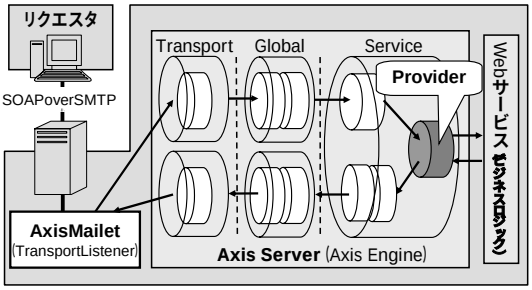


図 7:プロバイダ側のメッセージパス

4.4. WS-Routing を用いた SOAP ヘッダへの ID 付加

3.3 節で議論した SOAP メッセージへの ID の付加を WS-Routing を用いて実現する[5].

WS-Routing にはメッセージを識別し, あるメッセージを他のメッセージに対応付ける id 要素と releaseTo 要素が定義されている. この 2 つの要素を SOAP ヘッダに記述することでメッセージを識別する.

識別する ID の値には, 可能な限りの一意性を持たせるために UUID(Universally Unique IDentifier)を用いる.

5. PDF 変換サービスの試作と評価

SOAP over SMTP で送信される非同期型の Web サービスを用いることで, 効率的なメッセージ送信を行うことができるかどうかを評価するために, プロバイダで長時間の処理を行う PDF 変換サービスを同期型と非同期型の 2 つの方法で試作・実行し, Web サービスの動作の違い, 効率等を比較した.

5.1. PDF 変換サービスの試作

PDF (Portable Document Format)変換サービスは, PS (PostScript)ファイルを PDF ファイルに変換するサービスである. クライアントは, PS ファイルをリクエストメッセージに添付してプロバイダに送信する. プロバイダは, 受信した PS ファイルを Adobe Acrobat Distiller に渡し, PDF ファイルに変換する. 変換された PDF ファイルは SOAP over HTTP ではレスポンスメッセージに添付し, SOAP over SMTP ではリクエストで指定したアドレスへメールに添付して返信する. PDF 変換サービスを試作するために実装した Java プログラムのソースコードは, 550 行となった.

5.2. 同期型・非同期型 Web サービスの動作比較

図 8 に同期型 PDF 変換サービスのシーケンス図を示す.

同期型 PDF 変換サービスでは, クライアントアプリケーションから PS ファイルを送信すると TransportSender は SOAP メッセージを送信する. しかし同期型では, SOAP メッセージの送信が完了しても, プロバイダで PDF 変換処理が終了し, PDF ファイルを受信するまでは, クライアントアプリケーションに処理が戻らない状態となる. クライアントがサービス利用に必要とする時間 T_s は式(1)で定義できる.

$$T_s = T_1 + T_p + T_2 \qquad (1)$$

T_p : PS ファイルから PDF ファイルへの変換処理時間
T_1 : SOAP over HTTP リクエスト送信時の遅延時間
T_2 : レスポンス受信時の遅延時間

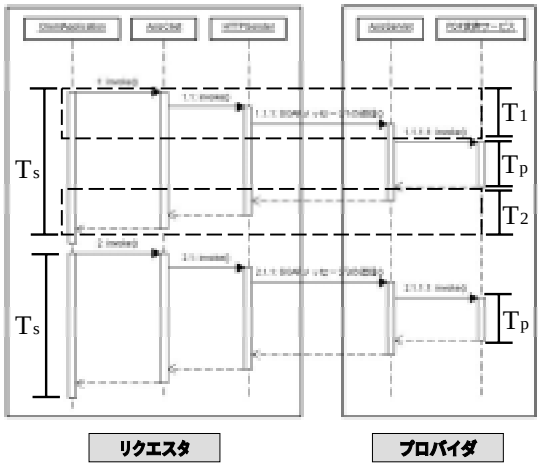


図 8:同期型 PDF 変換サービスのシーケンス図

次に, 非同期型 PDF 変換サービスのシーケンス図を図 9 に示す.

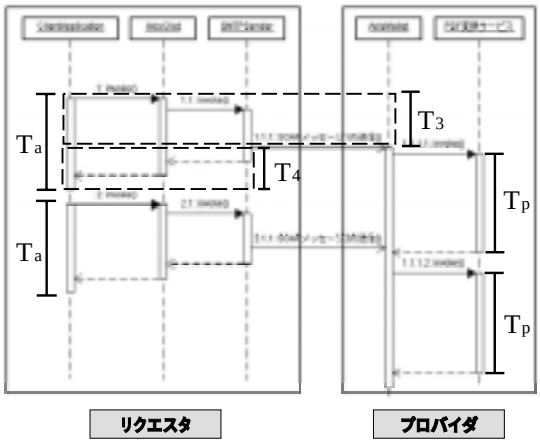


図 9:非同期型 PDF 変換サービスのシーケンス図

非同期型 PDF 変換サービスではクライアントアプリケーションから PS ファイルを送信すると SMTPSender が SOAP メッセージを送信する. ここで, 非同期型は送信が完了すると直ちにクライアントアプリケーションに処理が戻される. プロバイダ側で前のリクエストの PDF 変換処理が終了していなくても, 連続して PS ファイルを送信可能となった.

非同期型でクライアントがサービス利用に必要とする時間 T_a は式(2)で定義できる.

$$T_a = T_3 + T_4 \qquad (2)$$

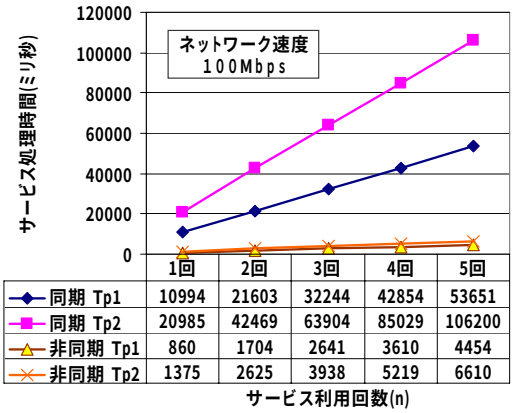
T_3 : SOAP over SMTP でﾘｸｴｽﾄ送信時の遅延時間
T_4 : ｸﾗｲｱﾝﾄｱﾌﾟﾘｹｰｼｮﾝへ処理を返す時間

以上のことから, 同期型と非同期型 PDF 変換サービスの処理時間の差は式(3)の⊿t の値で求められる.

$$\Delta t = T_s - T_a = (T_1 + T_p + T_2) - (T_3 + T_4) \qquad (3)$$

図 10 に, 異なるサイズの 2 つの PS ファイルを PDF ファイルに変換したときのクライアントのサービス処理時間を示す. 2 つの PS ファイルの PDF 変換処理時間(T_p)は, T_{p1} が約 7 秒, T_{p2} は約 15 秒である. 同期型のサービス処理時間は, 全体の 70%以上が T_p であるため, T_p に大きく影響を受ける. 一方, 非同期型は T_p の影響を受けないので, 利用回数が増加しても利用時間の増加量は少ない. このことから, サービスを利用する回数(n)と, T_p の増加により, 同期型と非同期型の処理時間差⊿t が増大する.

PDF 変換サービスの評価結果から, 非同期型 Web サービスを用いることは, ﾘｸｴｽﾀのサービス処理時間を短縮し, 効率の良いサービスの利用を実現できる有効な方法であることがわかる.



	1回	2回	3回	4回	5回
同期 Tp1	10994	21603	32244	42854	53651
同期 Tp2	20985	42469	63904	85029	106200
非同期 Tp1	860	1704	2641	3610	4454
非同期 Tp2	1375	2625	3938	5219	6610

図 10:PDF 変換サービス処理時間

6. 考察

Web サービス形態の多様化により, 同期型のみでは Web サービスの利便性を十分に生かせない. そこで, 本研究では, SOAP over SMTP による非同期型 Web サービスを提案した. この方法により, ﾘｸｴｽﾄﾒｯｾｰｼﾞを SMTP で送信する一方向型のﾒｯｾｰｼﾞﾝｸﾞが実現され, 長時間処理を行うサービスのクライアントの処理時間を短縮することが可能となる. 例えば, 試作した PDF 変換サービスでは, プロバイダ側の PDF 変換処理時間に影響されず, ﾘｸｴｽﾄの送信処理の時間間隔でサービスが利用可能となる.

さらに, 非同期型ﾒｯｾｰｼﾞに ID を付加し, ﾒｯｾｰｼﾞ間の関連を明らかにすることで, 複数のﾒｯｾｰｼﾞに対応関係を持つことができる. これにより, 複数のﾘｸｴｽﾄを集計し結果を返すような, 多重処理を行う Web サービスを実現することも可能である.

待ち状態を持たず, 一方向型ﾒｯｾｰｼﾞを行う非同期の特性を活かすことで, 新たな Web サービスの利用形態にも対応が可能になると考える.

7. 今後の課題

今後の課題としては, 以下のような項目が挙げられる.

(1) 応答ﾒｯｾｰｼﾞのﾎﾟｰﾘﾝｸﾞ

本研究では, 一方向型の非同期型 Web サービスを実現した. しかし, 非同期型Web サービスでも, レスポンスﾒｯｾｰｼﾞを期待する双方向型のﾒｯｾｰｼﾞ形態が考えられる.

今後は, 双方向型のﾒｯｾｰｼﾞ形態にも対応するため, ﾘｸｴｽﾀに応答ﾒｯｾｰｼﾞをﾎﾟｰﾘﾝｸﾞして, ﾒｯｾｰｼﾞを受信する機能を付加する必要があると考えられる.

(2) ﾘｸｴｽﾄの認証

実際に非同期型の Web サービスを提供するためには, 不正なサービスの利用を防ぐ方法が必要となる. 例えばﾒｯｾｰｼﾞ送信時にサービスを利用できるユーザであるかどうかをユーザアカウントやパスワードで認証する方法が考えられる.

8. まとめ

本研究では同期型Web サービスが持つ問題点を解決するために, SOAP over SMTP を利用した非同期型 Web サービスを実現する方法の提案と評価を行った. 非同期型 Web サービスを用いることで一方向型のﾒｯｾｰｼﾞﾝｸﾞが可能となり, プロバイダからのレスポンスを待機することなく, 効率の良いサービスの利用が可能になり, Web サービスの利便性の増大が期待できる.

参考文献

[1] 日本ユニテック, SOAP/UDDI/WSDL Web サービス技術基礎と実践 徹底解説, 技術評論社, 2002.
[2] 本 俊也, 最新 Web サービスマスタﾘﾝｸﾞハンドブック, 秀和システム, 2004.
[3] The Apache Software Foundation: Axis Documentation, http://ws.apache.org/axis/java/index.html.
[4] The Apache Software Foundation, James Documentation, http://james.apache.org/documentation_2_1.html.
[5] MSDN Online, Web Services Routing Protocol (WS-Routing), 2001, http://www.microsoft.com/japan/msdn/webservices/dnsrvspec/ws-routing.asp.