M-084

メタデータを利用したJava分散システムアーキテクチャの提案

Proposal of Java Distributed System Architecture Using Meta Data

石田 匠平 †　門脇 恒平 †　早川 裕志 ††　小板 隆浩 †††　佐藤 健哉 †
Shohei Ishida　Kohei Kadowaki　Hiroshi Hayakawa　Takahiro Koita　Kenya Sato

1 はじめに

近年，さまざまな機器をネットワークに接続し，それぞれの機器が連携してユーザにサービスを提供する分散システムが注目されている．既存の分散システムのアーキテクチャとして，Jini[1] や UPnP[2] が存在する．しかし，これらの仕様では，サービス提供者 (サーバ) が用意するサービス検索用の情報をサービス利用者 (クライアント) が知っている必要がある，という問題がある．この問題の解決方法として，サービスを複数の視点から表現するメタデータを用いることで，クライアントがサービス検索用の情報を予想する，という方法が考えられる．本稿では，複数のメタデータによってサービスを表現することで，この問題を解決する分散システムのアーキテクチャを提案する．

2 既存のアーキテクチャの問題点

既存の分散システムアーキテクチャは，サービスを検索するための情報として，サーバがサービス固有の情報を設定している．この情報とは，Jini の場合はインタフェースであり，UPnP の場合は XML ファイルの Service Type の記述である．そして，サービスを利用するためには，クライアントはサーバに登録されている情報を予め知っていることを必要としている．

しかし，このように事前に共通の情報を持たせることは，将来的に多数の機器がネットワークに接続された場合，機器間の連携を困難にしている．具体的には，未定義機器や新サービスへの対応などである．例としてオーディオ機器を考える．音声のファイル形式には，WAVE, MP3, WMA などがある．現在のオーディオ機器はこれらのファイル形式をサポートしているとしても，後に新しい圧縮形式が登場した場合，既存の機器では再生できない．しかし，Jini や UPnP の場合，サービスを “オーディオ機能をもつもの” といったことを表す単一の情報 (例えば，Jini なら AudioPlayer インタフェース，UPnP なら Audio という Servicce Type の記述) で検索するため，再生可能なファイル形式によるサービスの判別ができない．

3 提案アーキテクチャの概要

3.1 コンセプト

本稿では，サーバとクライアントの間で事前の情報共有が不要となる分散システムアーキテクチャを提案する．クライアントがサーバに登録されている情報を知らずにサービスを発見するには，サーバの設定するサービス発見用の情報を，クライアントが予想できなければならない．つまり，サーバはクライアントの予想に備えてサービス発見用の情報を設定しなければならない．そのため，サービス発見用の情報は複数設定する．これらの情報は，サービスの機能を端的に表す文字列とする．これらは，サービスに対するメタデータと呼ぶ．十分な数のメタデータを用意することが出来れば，クライアントがサービスを発見することが可能となる．また，未定義機器や新サービスに関しても，従来のサービスとの共通点を持っていれば，クライアントがサービス検索のための情報を更新することなく，サービスを利用できる．前章の例では，予め再生可能なファイル形式をメタデータとして登録しておけば，音声の新しい圧縮形式が登場しても，再生可能なファイル形式の判別が可能となる．つまり，あるサービスの再生可能なファイル形式として WAVE, MP3 がメタデータとして登録されているとき，クライアントが WMA を再生可能なサービスを探そうとして “WMA” という文字列を送信しても，そのサービスはメタデータが一致しないので，利用不可能だと分かる．

サービスを実行するには，リフレクションを用いる．リフレクションとは，クラスのフィールドやメソッドの情報を取得したり，クラスやメソッドなどの名前を与えることでメソッドを実行したりする Java の機能である．リフレクションを用いると，サーバがクライアントに，サービスを構成するファイル，実行すべきメソッドとメソッドの属するクラスの情報を与えることで，クライアントはサービスを実行できる．具体的には，ファイルに記述されている，サービス起動のためのクラスのオブジェクトをクラス名から生成し，そのオブジェクトからクラスと同様にメソッドのオブジェクトを生成する．生成したメソッドのオブジェクトから，そのメソッドを実行することで，サービスを起動する．以上の手法を用いることで，クライアントは事前にサービスの発見や起動に必要となる情報を一切知ることなく，サービスを利用することが可能となる．

3.2 構成と処理手順

本アーキテクチャは，クライアント，サーバ，ネットワーク上のサービスを管理する機器 (以下マネージャ) によって構成される．この構成図を図 1 に示す．

図中の 1〜5 は，アーキテクチャが行う処理の順番を示している．それぞれの処理については以下に述べる．

1. サーバがサービス名，サービスに対するメタデータ，クライアント側でサービス実行に必要となるクラス名，メソッド名の情報を用意する．

† 同志社大学大学院工学研究科
†† 九州大学大学院システム情報科学府
††† 同志社大学工学部情報システムデザイン学科

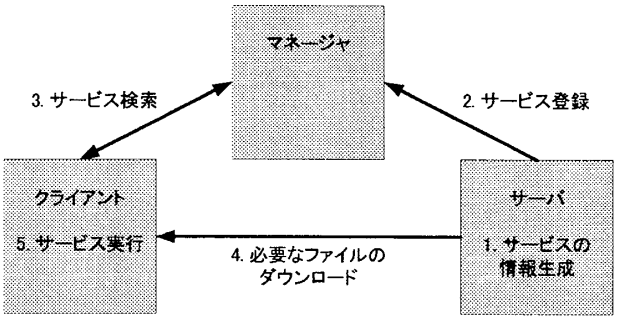

図1　提案アーキテクチャの構成と処理の手順

2. サーバがマネージャにサービス登録を要請する．この時マネージャはサーバのIPアドレスを取得し，サービス名，メタデータ，クラス名，メソッド名の情報とともに保持する．
3. クライアントが連携したいサービスについての文字列を作成し，マネージャに送信する．マネージャはクライアントから受け取った文字列に一致するメタデータを持っているサービスを探し，一致したサービスのリストをクライアントに返す．
4. クライアントが，マネージャから渡されたリストから実行したいサービスを選択する．そして，IPアドレスをもとに，選択したサービスのサーバにアクセスし，必要なファイルをダウンロードする．
5. サービスを構成するファイル，サービスを起動するためのクラス名とメソッド名を取得できたので，クラスオブジェクト，メソッドオブジェクトを作成し，これを実行することでサービスを起動する．

上記の手法を用いることで，クライアントはサービス発見用の情報や実行に必要な情報を予め知ることなく，サービスを利用することができる．

4　システムの実装

提案アーキテクチャが実際に機能することを確かめるために，クライアント機器上で“Hello World”という文字列を表示するサービスを実装した．このサービスについてサーバが設定したメタデータは“hello world”，“Hello World”，“HelloWorld”，“test”，“print”の5つで，クライアントが用意した検索用文字列は“Hello World”，“HelloWorld”，“HELLO WORLD”の3つである．なお，サービスのメタデータは筆者が設定し，クライアントの検索用文字列は協力者が設定した．この際，協力者に“Hello World という文字列を表示するサービスがあり，それを検索する場合に自身がキーワードとする文字列は”と質問し，その回答を検索用文字列として採用した．

次に，このシステムで実際に行われた処理を述べる．まずサーバ機器上で“hello world”，“Hello World”，“HelloWorld”，“test”，“print”の5つのメタデータと，サービス名，サービスを実行するクラス名とメソッド名を準備する．そして，これらの情報をマネージャ機器に送信し，マネージャ機器がサービスを登録する．この後，クライアントがサービスを検索するために，“Hello World”，“HelloWorld”，“HELLO WORLD”の3つの情報をマネージャ機器に送信する．マネージャ機器はサービスを検索し，結果として“Hello World”と“HelloWorld”の情報に一致するサービスを発見する．そしてマネージャ機器がクライアントに，サービスに関する情報とサーバ機器のIPアドレスを伝える．クライアントは，サービスを選択し，そのサービスの持ち主であるサーバ機器にアクセスし，必要なファイルを取得する．最後に，クライアントがサービスを起動する．結果として，“Hello World”の文字列が表示される．

5　提案アーキテクチャに対する考察

本アーキテクチャでは，サーバとクライアントの間で，事前にサービス検索用の情報に共有することなく，サービスを発見し利用できることが確認できた．JiniやUPnPはサービス発見の仕組みによって，検索用の情報の記述の仕方が定められているが，提案アーキテクチャは，サービス発見の仕様にとらわれず，サービス内容についての複数の候補を設定することができる．

6　まとめと今後の課題

既存の分散システムアーキテクチャには，サーバとクライアントの間で，サービス検索に利用する情報についてコンセンサスが必要となる問題点がある．本稿では，サービスを多角的に表現するメタデータを用いてサービスを発見し，リフレクションを用いてサービスを起動することで，クライアントが事前に持つ情報の量を最小化したアーキテクチャを提案した．

本アーキテクチャのサービスの発見のメカニズムは，クライアントとサーバがサービス発見に使用する情報をお互いに予想することによって成り立っている．そのため，サービスを確実に発見することが保証されない．この具体的な原因は以下の通りである．

1. 半角文字と全角文字，大文字や小文字，スペースの有無などの，文字列の細かな違い
2. 同義語の扱い
3. 検索する言語の違い

これらの違い全てに対応するメタデータを記述することは，大きな作業量となる上に，リソースも多く必要となる．解決手法としては，1は文字列の操作，2は意味解析を利用する方法[3]，3は機械翻訳が考えられる．

今後の課題として，サービス発見の精度の定量的な評価とその向上および，上記問題点への対応，グローバルネットワークを介しての機器連携を考えている．

参考文献

[1] Arnold, K., Wollrath, A., O’ Sulliban, B., Scheifler, R., and Waldo, J.: The Jini Specification - Second Edition, Addison-Wesley (2001).
[2] UPnP: UPnP Device Architecture, (2003). http://www.upnp.org/resources/documents/
[3] Steller, L., Krishnaswamy, S., and Newmarch, J.: Discovering Relevant Services in Pervasive Environments Using Semantics and Context, ICEIS Ubiquitous workshop (2006).