L_071

インタフェースレス志向によるJiniシステムの構築

Design of Interface-less Oriented Jini Systems

門脇 恒平 † 早川 裕志 ‡ 小板 隆浩 † 佐藤 健哉 †
Kohei Kadowaki Hiroshi Hayakawa Takahiro Koita Kenya Sato

1 はじめに

デジタル機器をネットワークに接続するだけで，機器間での協調動作を容易に実現するためのJava分散オブジェクトの応用技術がJini[1]である．Jiniにおいては，サービスごとにあらかじめ定義されたインタフェースを利用側のクライアントにおいて保持しておく必要があるため，現時点で未定義な(将来登場する)機器を直接サポートすることは不可能である．

本研究では，未定義な機器を含むすべてのサービスのインタフェースが汎用的なインタフェースを継承し，クライアントにおいてサービスごとにあらかじめ定義されたインタフェースを持つ必要のないシステム構築手法を提案する．

2 Jiniシステムの問題点

2.1 Jiniの概要

Jiniとは，様々な機器をネットワークを通じて接続し，相互にハードウェアまたはソフトウェアの機能をサービスとして提供しあうための技術仕様[2]である．“Simply Connect:ただ機器同士をつなぐだけで利用できる環境を構築すること”を目指している．

図1に示すように，Jiniを構成する主な要素としてサービスの提供側であるサービスプロバイダ，サービスの利用側であるクライアント，ネットワーク内のすべてのサービスを管理するLookupサービス，クライアントがサービスを利用するために必要なクラスファイルを格納するcodebaseが挙げられる．これらの要素はそれぞれが互いに独立したサービスとして振舞う．サービスプロバイダがサービスを提供してからクライアントがサービスを利用できるようになるまでの手順を以下に示す．

1. サービスプロバイダはサービスプロバイダとクライアントの両方において既知のインタフェースを実装する形でサービスオブジェクトを生成する．
2. サービスプロバイダはLookupサービスを検索し，発見したLookupサービスにサービスオブジェクトを登録する．
3. クライアントはサービスオブジェクトが実装しているインタフェースを用いてLookupサービスから利用したいサービスを検索する．
4. クライアントはLookupサービスからサービスオブジェクトをダウンロードする．
5. 保持しているインタフェースとcodabaseからダウンロードしたクラスファイルを用いてサービスオブジェクトを再構築し，サービスを実行する．

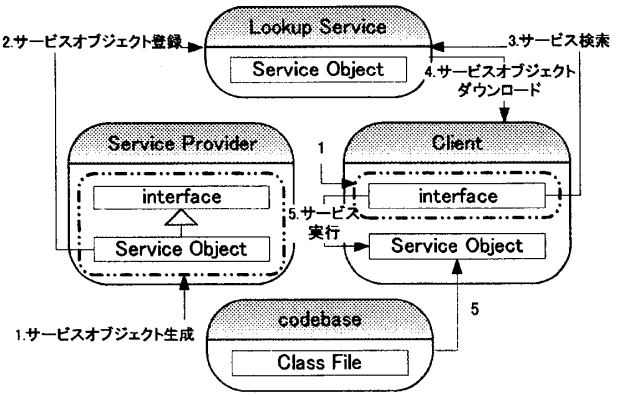

図1 Jiniシステムの構成要素と動作手順

2.2 問題点

Jiniシステムでは，クライアントにおいて目的のサービスが実装しているインタフェースのクラスファイルを保持していなければ，クライアントは目的のサービスを検索することができない．また，何らかの方法で目的のサービスを検索できたとしても，サービスから受信したオブジェクトを目的のサービスが実装しているインタフェースを用いて再構築しなければ，クライアントはサービスで定義されているメソッドを利用することができない．

以上の理由により，クライアントが任意のサービスを利用するためには，利用する可能性のある個々のサービスごとにあらかじめ定義されたインタフェースのクラスファイルをすべて保持しておく必要がある．そのため，メモリ制約のある環境においては，クライアントは未定義な機器をサポートすることができない．例えば，あるクライアントが新しくホームネットワークに参加しようとするとき，ホームネットワークに接続されたすべての機器を利用するためには，それぞれの機器が提供しているサービスごとに定義されたインタフェースのクラスファイルを保持しなければならない．

3 提案システム

本研究では上記の問題点を解決するために，クライアントにおいてサービスごとにあらかじめ定義されたインタフェースのクラスファイルを持つ必要のないJiniシステムの構築手法を提案する．提案システムの構成を図2に示す．

提案システムにおいてサービスが利用可能となるまでの手順を以下に追記する．

1. サービスプロバイダとクライアントはサービスの本質的な性質を表現した汎用的なインタフェースを保持し，サービスプロバイダは汎用的なインタフェースを継承したインタフェースを実装することにより

† 同志社大学 工学部 情報システムデザイン学科
‡ 同志社大学大学院 工学研究科 知識工学専攻

サービスオブジェクトを生成する．
2. クライアントは汎用的なインタフェースを用いてLookupサービスから利用したいサービスを検索する．
3. クライアントはダウンロードしたサービスオブジェクトからcodebaseの情報を取得する．
4. codebaseからサービスオブジェクトを再構築するためにクラスファイルをダウンロードするのではなく，実行可能なjarファイルをダウンロードする．
5. ダウンロードしたjarファイルを実行する．必要ならばRMIを用いてサービスプロバイダ上のサービスオブジェクトのメソッドを呼び出す．

上記の手法を用いることで，クライアントはサービスごとにあらかじめ定義されたインタフェースのクラスファイルを保持していなくともサービスを検索することが可能となる．また，クライアントはサービスオブジェクトを再構築せずにサービスを利用することが可能になるため，サービスごとにあらかじめ定義されたインタフェースのクラスファイルを保持していなくともサービスを利用することが可能となる．

例えば，あるクライアントがプリンタを利用する目的で新しくホームネットワークに参加するとき，個々のプリンタ独自のインタフェースを持たなくても，プリンタであることを表現するインタフェースのクラスファイルを持つだけで任意のプリンタを利用することが可能となる．

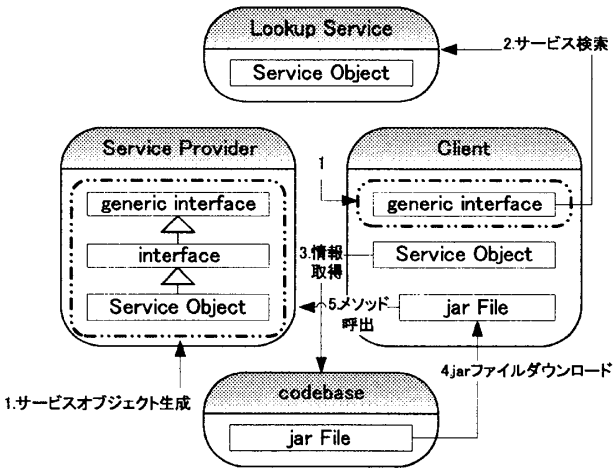

図2　提案システムの構成要素と動作手順

4　実装

4.1　プリンタサービス

提案手法の有効性を確認するため，クライアントが任意の文字列をサービスプロバイダ上の仮想的なプリンタで印刷することができるプリンタサービスシステムを実装した．動作例を図3に示す．

実装したプリンタサービスシステムを構成する主な要素としてGenericPrinterインタフェース，DoshishaPrinterインタフェース，DoshishaPrinterImplクラスが挙げられる．GenericPrinterインタフェースは一般的なプリンタであることを表す汎用的なインタフェースであり，図2におけるgeneric interfaceに対応している．全てのプリンタサービスのインタフェースはGenericPrinterインタフェースを継承している．DoshishaPrinterインタフェースは具体的なプリンタサービスを表すインタフェースの例であり，図2におけるinterfaceに対応している．DoshishaPrinterImplクラスはDoshishaPrinterインタフェースを実装しており，図2におけるService Objectのクラスに対応している．

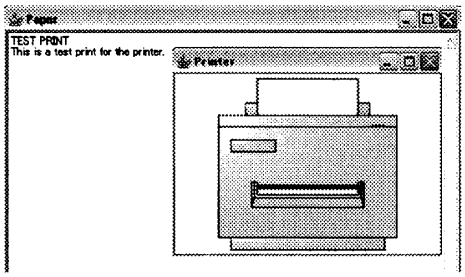

図3　実装したプリンタサービスシステムの外観

4.2　評価

実装したプリンタサービスシステムでは，jarファイルに提案システムにおいてサービスを実行するのに必要なクラスファイルだけでなく，Jiniシステムに基づいたサービスの利用者が必要とするクラスファイル群も格納するため，Jiniシステムと提案システムのどちらのシステムを採用しているクライアントからでもサービスを利用することが可能となる．

新機能が実現できる一方，実行可能なjarファイルを作成する際，クラスパスを相対パスで指定しなければならないため，Jiniのライブラリなどをjarファイルに含めなければならず，codebaseに格納するjarファイルのサイズが大きくなる．Jiniシステムと提案システムの比較結果を表1に示す．

表1　Jiniシステムと提案システムの比較

	未定義サービスのサポート	codebaseの格納ファイルサイズ	インタフェースのファイルサイズ
Jiniシステム	×	19 KB	354 B
提案システム	○	1170 KB	125 B

5　まとめと今後の課題

本研究では，サービスの利用者がサービスごとにあらかじめ定義されたインタフェースを持つ必要のないインタフェースレス志向によるJiniシステムの検討を行った．また，本手法に基づいた実装を行い有効性を確認した．今後の課題として，サービスプロバイダとクライアントが汎用的なインタフェースを持つ必要のないJiniシステムの構築手法および，サービスごとにJiniライブラリをカスタマイズすることによりクライアントがcodebaseからダウンロードするjarファイルのサイズの縮小化の検討を行う予定である．

参考文献

[1] Sun microsystems: “Jini Network Technology - White Papers” (2001). http://www.sun.com/software/jini/whitepapers/
[2] Jan Newmarch: “A Programmer’s Guide to Jini Technology”, Apress (2000).