A-009

Pure Java アプリケーションにおける 遠隔イベント通知のための汎用機構の提案

A Proposal of Generic Mechanism for Remote Event Notification in Pure Java Application

吉崎 順太† 中島 潤‡
Junta Yoshizaki Jun Nakajima

１．はじめに

VNCのようなデスクトップGUIを遠隔操作をするアプリケーション，あるいは異なるホスト間で画面を同期させるようなアプリケーションにおいては，それ専用のネットワークプロトコルや遠隔インターフェイスを実装する必要がある．しかし，GUIを構築するコードは大規模かつ複雑になる傾向にある．そこにネットワークを用いて同期する機能を加えることは開発の負担が大きく，コードの可読性や保守性の低下にもつながる．また，画面画像を一定間隔で転送する機構のものである場合はネットワークへの負荷が大きく，パフォーマンスの確保が困難である．

そこで本研究においては，通常は同一プロセス内のGUIコンポーネント間で行われるイベント通知をネットワーク経由により異なるホスト間で行うというアプローチを試みた．このアプローチでは既存のイベントディスパッチャおよびリスナインターフェイスをそのまま用いることが可能であり，他ホストへ伝送する情報も小さなイベントオブジェクトのみである．このアプローチの有効性を検証するために，ターゲットをPure Javaアプリケーションに絞り，リフレクションを用いた汎用機構を考案したので本稿において提案する．

２．遠隔イベント通知機構の概要

提案する遠隔イベント通知の汎用機構は以下の５つのロールにより成り立つ．（図１）

- イベント源コンポーネント
- イベント通知先リスナ
- 遠隔イベントオブジェクト
- 遠隔ディスパッチャ
- 遠隔リスナ

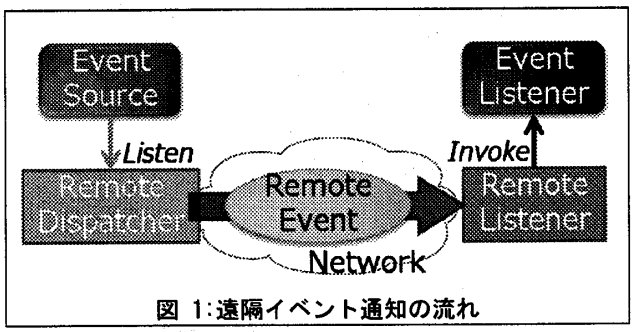

図 1:遠隔イベント通知の流れ

まず，イベント源コンポーネントにより発生したイベントを遠隔ディスパッチャが捕捉する．そのイベントからハンドラメソッドのシグネチャおよび引数のインスタンスを遠隔イベントオブジェクトとしてラップし，ネットワークを経由して遠隔リスナへ伝送する．遠隔リスナは受信した遠隔イベントオブジェクトからハンドラメソッドのシグネチャを取得し，通知先リスナのハンドラメソッドを検索する．ハンドラが見つかった場合は，それを呼び出す処理をGUIスレッドへ追加してイベント通知が完了する．

２－１．リフレクションによるハンドラ呼出し

遠隔リスナから通知先リスナのハンドラメソッドを呼び出す際に，メソッド名と引数リストから当該のハンドラメソッドを検索する．そのため通知先のリスナインターフェイスに依存せず，すべてのハンドラメソッドに対して呼び出しを行うことが可能である．

しかし，リフレクションによるメソッド検索は非常に計算コストが高い．整数の絶対値を求めるメソッドに対してリフレクションによる呼出しを100万回行い，それに要した時間を測定して10回の試行の平均を取ると以下の結果が得られた．

- 毎回検索した場合 … 2425.7ミリ秒
- キャッシュした場合 … 149.8ミリ秒

測定した時間は厳密なCPU時間ではなく環境に依存するが，比率としてはキャッシュした場合と比較して毎回検索すると約16倍の計算コストを要することがわかる．そこで，通知先リスナごとにハンドラメソッドのキャッシュを持つことで動作の最適化を図った．検索して見つかったメソッドはシグネチャをキーとしたマッパへ追加し，見つからなかった場合は無視リストとして保持するようにした．これによりハンドラメソッドごとに１度の検索で済むようにした．

２－２．イベントの伝送方法

遠隔ディスパッチャから遠隔リスナへイベントを通知する，すなわちネットワークを経由して遠隔イベントオブジェクトを伝送する実装方法について，本研究においてはソケットを用いた実装とRMI（遠隔メソッド呼出し）を用いた実装について動作検証を行った．

ソケットを用いる場合は実装が複雑になる反面，アプリケーションと同じJavaプロセス内で完結させることが可能である．オープン／クローズのタイミングや例外処理など細かい制御も可能である．

RMIを用いる場合は実装が容易になるが，イベント受信側でrmiregistryが実行され続ける必要がある．アプリケーション外のプロセスに依存することになるため制御が困難になる．また，イベント通知の遠隔化というアプローチは必然的にPeer-to-Peer型となる．そのためユーザが事前に自分でrmiregistryを起動しておくことが必要になり，利便性としては問題である．

いずれの実装方法にせよ，ソケットの場合はポート番号が，RMIの場合はバインド名が他のアプリケーションと重複する可能性がある．そういった接続時における問題は本

† 北海道情報大学大学院，Graduate School of Hokkaido Information University

‡ 北海道情報大学，Hokkaido Information University

研究の対象外とする．

２－３．遠隔ディスパッチヘルパ

既存のイベントディスパッチャに手を加えないという要件を満たすために，遠隔ディスパッチャがイベント源コンポーネントからのイベントを捕捉するためには，そのイベントに対応するリスナインターフェイスを実装する必要がある．これはJava言語の厳格な型付けによる制約である．そのため，遠隔ディスパッチャはイベントの種類ごとに実装する必要がある．そこで，遠隔イベントオブジェクトの生成と伝送の処理を再利用可能な形で抽出し，遠隔ディスパッチャの実装作業を軽減できるようにヘルパを用意した．

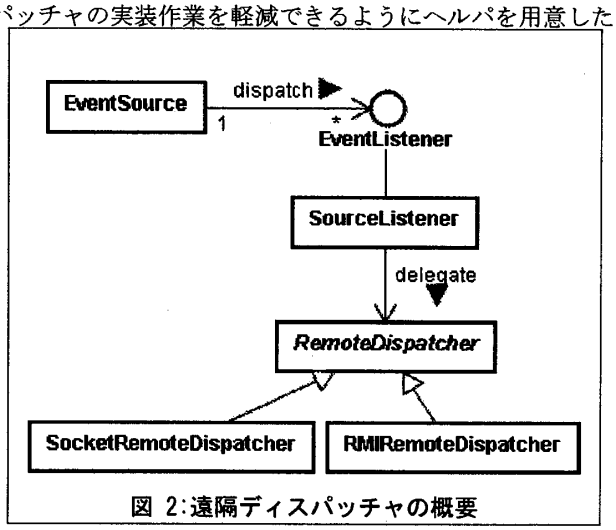

図 2:遠隔ディスパッチャの概要

遠隔イベントオブジェクトはハンドラメソッドのシグネチャおよび引数のラッパであり，通知先リスナのハンドラメソッドを呼び出すための情報である．ハンドラメソッドのシグネチャを抽出するには引数リストの扱いが問題となる．オブジェクトの場合は共通のメソッドにより型情報を取得することができるため，インスタンスのみで値と型の両方を与えることが可能である．しかし基本型の場合はそれが不可であるため，値とは別に型も与える必要がある．リフレクションによる呼出しの観点から，ハンドラメソッドの引数リストを以下の４タイプに分類できる．

- 引数なし
- 基本型１個
- オブジェクトのみ１個以上
- 基本型を含む１個以上

これらのタイプに対応したヘルパメソッドを用意することにより遠隔イベントオブジェクトの生成を容易にした．

イベントオブジェクトの伝送に関しては，伝送機能のみを備えたクラスを作成して処理を委譲できるようにした．また，インターフェイスを抽出して実装ごとにサブクラス化するとこにより，伝送方法の容易な切り替えを可能にした．本研究ではソケットによる実装とRMIによる実装を用意し，委譲先の切り替えが全体の動作に影響を及ぼさないこと確認した．

また，遠隔ディスパッチ処理にかかる時間はネットワーク環境に依存するため著しく遅くなる可能性がある．その場合にGUIスレッドをブロックしてしまうことを避けるために，遠隔ディスパッチ処理は別スレッドとして実行してGUIスレッドを即座に復帰させるようにした．

３．遠隔イベント通知による利点

遠隔イベント通知による最大の利点は通常のイベントリスナのインタフェイスをそのまま利用することができる点である．すなわち，ネットワークによる同期機能に対する依存関係を持たせずに，アプリケーション内のカスタムコンポーネントを構築することが可能ということである．これにより，コードの保守性や可読性といった品質を保ったまま同期機能を追加することが可能になる．

また，イベント駆動モデルは汎用的な仕組みであるためあらゆるアプリケーションに適用可能である．そのため，アプリケーションごとに固有のインターフェイスやプロトコルを定義および実装する必要がなくなり，開発における全体の工数を削減することにつながる．

４．遠隔イベント通知の問題点

一般的にGUI上におけるイベントはコンポーネントイベントかセマンティックイベントに区別される．コンポーネントイベントはコンポーネント上で発生するユーザアクションに応じて発行されるものである．これはマウスやキーによる入力などといったイベントを含み，マウスの座標や押下されたキーといったような具体的な情報を持つ．対してセマンティックイベントはコンポーネントイベントほど定義が明確ではなく，同種のイベントでも発生源のコンポーネントにより扱いが異なる．

遠隔イベント通知はイベント源とリスナが別プロセスで動作しているという特質上，イベントオブジェクトによりハンドラの動作が一意に定まる処理でなければ，両者のアプリケーションは同期しない．よって，遠隔イベント通知においてはコンポーネントイベントの通知に適しているが，セマンティックイベントを通知する場合には注意が必要である．例えばチャットアプリケーションにおけるメッセージ送信処理の場合，送信ボタンを押すと入力欄に入力されているメッセージが自分のメッセージ欄に追加される．その後他の参加者すべてのメッセージ欄に同じメッセージが追加されることが期待される．しかし，ここで“送信ボタンが押された”イベントを遠隔通知した場合，送信者のものではなく参加者が各々入力中のメッセージが参加者自身のメッセージ欄に追加されることになる．

また，イベントの種類に関係なくハンドラが破壊的な処理を行う場合にも注意が必要である．GUIアプリケーションにおいては，自分のプロセスを終了する処理をウィンドウイベントのリスナに記述することが一般的に行われる．この場合にウィンドウから通知されるイベントを他ホストへ遠隔通知すると，通知先のアプリケーションが突然終了することになる．

以上のように，遠隔通知するイベントは慎重に選定する必要がある．

５．おわりに

本研究において，アプリケーション間におけるGUIの同期方法へのアプローチとして，汎用的な遠隔イベント通知機構の考案およびJavaによる実装を行った．今後はこの機構を具体的なアプリケーションに組み込み，実用性の検証や問題点の洗い出しを行う必要がある．

また，この機構はイベント駆動・ネットワーク伝送・リフレクションの３要素で構成されており，これらを含む言語であれば実装可能である．この汎用性を実証するために，Java以外のいくつかの言語で実装して動作検証を行う必要がある．