Java応用によるリモート監視システム

古谷雅年*、山内豊彦**

*　（株）日立製作所システム開発研究所
川崎市麻生区王禅寺 1099

**　日立東北ソフトウェア（株）
川崎市幸区鹿島田 1119-23

概要

上下水道などのプラントは、リアルタイム監視制御系を構築し、その上の監視制御室にて、専任オペレータによって運用・運転されるのが一般的である。このプラントデータを多目的用途で有効活用するために、インターネット技術（特にJava）を適用した遠隔監視システムを提案した。実装にあたり、分散オブジェクト技術にはJava RMI、データベースアクセスにはJDBC-ODBCブリッジを用い、監視の応答性を向上するために、Javaアプレットにデータキャッシュを持たせ、Javaアプレット－Javaサーバアプリケーション間のデータ転送量を抑える方式を提案した。また、システムの応用として遠隔保守・点検に対し、不正アクセス防止、排他処理などの課題に着目し、暗号技術を応用した方式を提案した。

キーワード
Java、遠隔監視、分散オブジェクト、データベース、暗号

1　はじめに

安全でおいしい水の安定した供給、豪雨時の浸水・洪水被害の防止、生活排水・工場排水などの浄化による放流先河川などの環境保全など、上下水プラントの運用・運転は市民の生活にとって欠かせないものとなっている。一般に、プラントは、リアルタイム監視制御系上に（遠隔）監視制御室を設置し、そこに配置された専任オペレータによって運用・運転される。このとき、プラントデータはデータベースに蓄積される。このプラントデータをオペレータ以外の関係者がしばしば活用する機会がある。例えば、調査・分析、維持管理、点検･保守、環境管理、設計、苦情対策などである。活用の際、現状では、監視制御室に赴くか、オペレータに問い合わせるか、後日資料を閲覧するなどといった措置がとられている。

一方、最近のインターネット技術の発展により、多種多様な情報が、手元で簡単に得られるようになった。これは、時々必要となるような情報を入手する手段としては、非常に有効である[5][6]。

本稿では、運用・運転によるプラントデータを多目的な用途に容易に利用するために、インターネット技術（特にJava[1]）による遠隔プラント監視システムの実装、及び、その応答性を向上するための方式について報告する。また、本システムの発展の１つとして、遠隔保守・点検への応用も期待されている。本稿では、それに関する課題と対策として、排他処理、不正防止などについても言及する。

2　プラント監視システム

2．1　システム構成

図１に全体システムを示す。プラントの運用・運転は、監視制御系の遠隔監視制御室で行われる。この監視制御系と情報系とを接続し、監視制御系からプラントデータを情報系にあるデータベースに逐次転送し、格納する。データベースは、ODBCドライバを有するRDBを用い、SQL言語でアクセスする。

一方、プラントデータを必要とする利用者は、手元にあるWebブラウザを有する端末から、Webサーバにアクセスして必要なデータ、及び、プログラムを入手する。本稿では、図2のように入手するプログラムをJavaアプレットとし、Webブラウザは JDK1.1.5をサポートするものを用いる。また、JavaアプレットとJavaサーバアプリケーションとの間は、分散オブジェクト技術の1つであるJava RMI(リモートメソッド呼び出し)[3][4]を用い、データベースへのアクセスは JDBC-ODBCブリッジドライバ[2]を用いる。

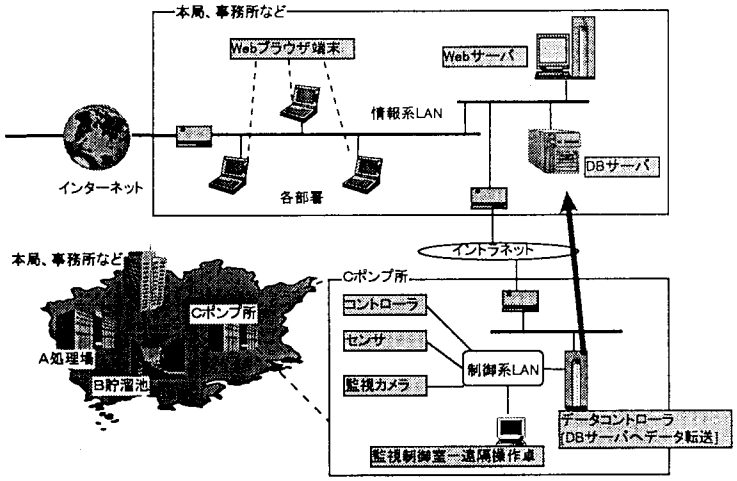

図1　全体システム

2．2　システムへの要求事項

(1)データ規模

一般に、上下水事業者は、地理的に分散した複数のプラントを管理している。事業者にもよるが、蓄積される（計測、または、加工による）データ項目数の規模は、数百～数万に及ぶ。さらに、データは最近の1週間～1ヶ月くらいは、数分単位間隔の時系列データとして保有し、それ以前のものも特徴日は同じように保有し、特徴日以外でも、統計的に整理した形で保有される。多目的な用途での利用を考慮すると、システム側はこれら大量のデータを扱うためのデータベースは必須である。

(2)リアルタイム応答性

プラントデータが数分単位で更新されるので、利用者側の端末もそれ以内での更新が要求される。また、気象やプラント故障などの警報データは速やかに入手すべきで、初期のロード時（Java VMの起動時やJavaアプレットのダウンロード時）を除いて、全ての監視画面において、数十秒程度の応答性は必要となる。

2．3　プロトシステム

2．3．1　RMIのパフォーマンス評価

クライアントとサーバ、または、サーバと監視制御系サーバとの間で連携を行なう技術の1つとして、分散オブジェクト技術があり、Java RMI[3][4]、CORBA、HORBなどがその代表である。本稿では、クライアントはJavaアプレット、サーバはJavaアプリケーションとして実装し、Java RMIを利用する。RMI(Remote Method Invocation)は、サーバマシンのJava VM上のオブジェクトのメソッドを呼び出すことができる。

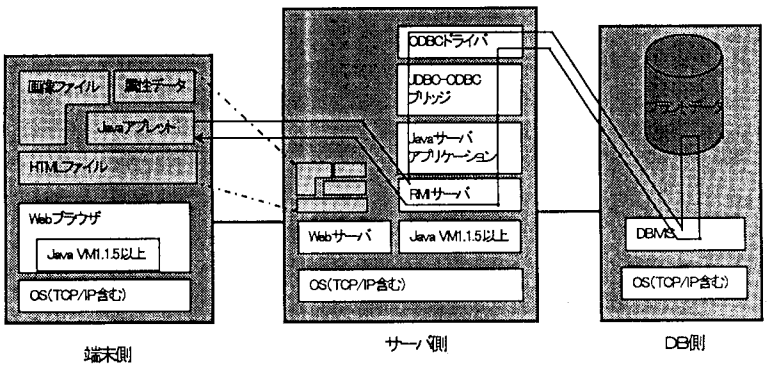

図2　システム構成

機器など	性能
サーバ	Pentium90MHz * 2、メモリ 64MB、WindowsNT4.0、JavaVM1.1.5
クライアント	Pentium166MHz、メモリ 144MB、Windows95、Netscape Navigator4.05
ネットワーク	10BaseT、同セグメント、プロキシなし

表１　実験環境

return の型	vsize	
	10	500
Object[vsize][2]	200～300	7000～8000
double[vsize][2]	50～120	1700～1900
float[vsize][2]	50～120	1700～1900

表２　実験結果(msec)

RMIでは、メソッドの引数、及び、もどり値として、Java基本型(int,double,boolean など)、及び、Serializable インターフェースを実装した参照型(Object,String,Vector など)が利用できる。そこで、RMI の応答性能（メソッド呼出しの前後で時刻を計測）について、戻り値の型と配列サイズをパラメータとして応答速度について比較検討した。表２に実験結果を示す。また、このときの実験環境を表１に示す。

表の通り、Java 基本型(double,float)は参照型(Object)に比べて数倍速いが、基本型の double と float とでは差はなかった。本稿では RMI のメソッドの引数、戻り値には、可能な限り基本型を用いる。

２．３．２　実装方法

本稿では図３に示す動作フローを持つプラントデータ監視システムを提案する。その特徴を以下に示す。

(1)サーバ側では、RMI サーバを常時起動してクライアント側からのメソッド呼出しの待機をするが、その際、ドライバ(JDBC-ODBC ブリッジ)の登録、RDB との接続をした状態にしておく。

(2)クライアント側の Java アプレットの初期化時に、プラントデータをキャッシュするオブジェクトを生成する。プラントデータは Vector に格納する。このキャッシュはアプレット終了時まで保持する。

(3)更新時、または、ユーザ要求時に、クライアント側で表示等に必要とするプラントデータの範囲に対して、キャッシュ状況を確認し、表３のように不足範囲にデータの検索範囲を絞り、RMI する。

(4)サーバ側では要求範囲に基づきデータを検索・加工し、できるだけデータを Java 基本型(int,double など)に変換して値を返す。

(5)戻り値とキャッシュデータを使って、クライアント側で必要とする範囲のプラントデータにマージし、それを加工、表示する。

(6)表３に示したキャッシュルールに従って、クライアント側のキャッシュを更新する。

このルールでは、要求範囲とキャッシュ済み範囲との間で時間的隙間があるときは、時間的に新しい方だけをキャッシュに保存し、連続する、または、重なる場合には、or 結合でキャッシュに保存する。

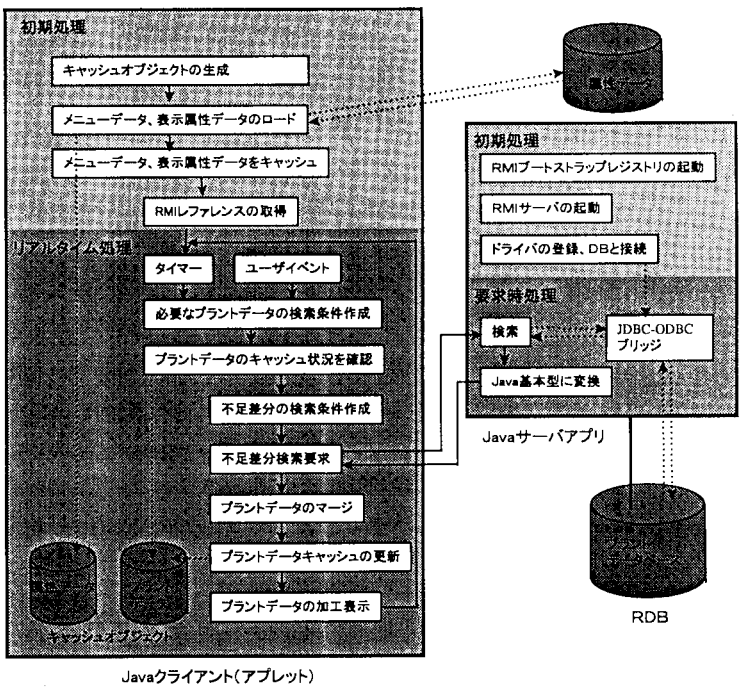

図３　動作フロー

No.	要求とキャッシュの関係	差分検索範囲		キャッシュ範囲		検索範囲／キャッシュルール
		初期時刻	最終時刻	初期時刻	最終時刻	
1	CI<I<CF<F	CF+1	F	CI	F	差分／OR結合
2	CI<CF<I<F	CF+1	F	CI	F	すきま補間／OR結合
3	CI<CF<<I<F	I	F	I	F	要求通り／新しい要求範囲に更新
4	I<CI<F<CF	I	CI−1	I	CF	差分／OR結合
5	I<F<CI<CF	I	CI−1	I	CF	すきま補間／OR結合
6	I<F<<CI<CF	I	F	CI	CF	要求通り／もとのキャッシュを維持
7	I<CI<CF<F	I	F	I	F	要求通り／新しい要求範囲に更新
8	CI<I<F<CF	null	null	CI	CF	検索不要／もとのキャッシュを維持

CI:キャッシュの初期時刻 CF:キャッシュの最終時刻
I:要求範囲の初期時刻　F:要求範囲の最終時刻

表３　差分検索とキャッシュ

２．３．３　実装結果

本稿で提案する方法により実装した結果を表４に示す。実験環境は表１と同一であり、DBMS には市販のものを利用した。

表に示す通り、プラントデータ監視において、応答性能に影響があるのは、サーバ側の JDBC-ODBC ブリッジドライバによる DB のデータ検索時間、サーバ･クライアント間の RMI によるデータ転送時間、クライアント側のデータの加工･表示時間である。初めてデータを見るとき(新しいデータ項目を選択したとき)、及び、かなり過去のデータを見るときなどの未キャッシュ時 (vsize=400～500) では、応答時間にして約５秒、少ない差分検索で済む更新時 (vsize=1～2) には、約１秒という結果を得た。

計測項目	未ｷｬｯｼｭ時	更新時
DB検索時間	1800	200
データ転送時間	2000～3000	300～500
表示時間	400	400
応答時間	4200～5200	900～1100

表４　実験結果(msec)

図４が本方法によるトレンド監視画面例である。更新時には、図の差分範囲だけを要求し、その他はキャッシュデータを用いる。キャッシュ＆マージにより、一度表示させたことがある項目・範囲の再表示や、表示方法の切り替え(例えば、トレンド→表形式など)表示時などデータの再ロードが不要となる。

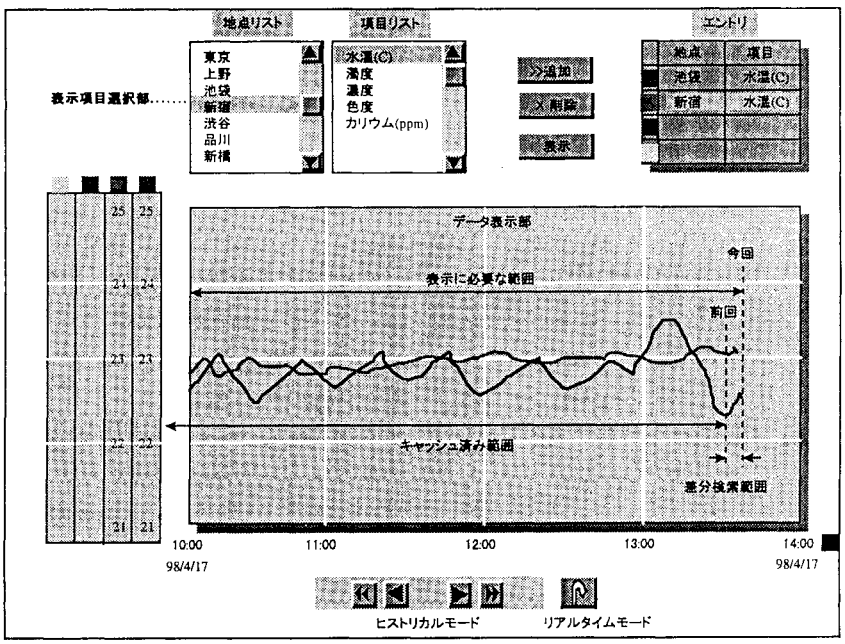

図４　プラントデータのトレンド監視画面とキャッシュを利用した差分検索

３．システムの展開

３．１　遠隔保守・点検への応用

２章で述べたとおり、プラントの日常の運用・運転は既存の基幹監視制御系によって行われる。一方、保守・点検などは保守員により別作業として行われ、無人プラントを含め１日に数箇所巡回する。保守・点検作業の中には、ごく簡単な操作を施して動作を確認するだけのものもあり、プラントに巡回することなく遠隔保守・点検を実施することで、業務効率（保守員の移動箇所・時間の削減、点検・保守機会の増加）の改善が期待されている。

３．２　システムへの要求事項

(1)排他処理

Ｗｅｂシステムは、一般に不特定多数が利用するが、機能ごとに一部の者だけ利用できるようにアクセス制限をかけることも可能である。しかし、アクセス権を持つ複数の保守員が同時に操作を行うと、作業に支障をきたす場合があり、一連の保守・点検作業中は、１人の保守員だけが操作権を持つようにする。

(2)応答性と進捗状況確認

保守・点検作業は、１つの操作で完了することはまれで、操作に対する結果を確認しながら、次の操作に移行していく。また、１つの操作に対する実行時間（数秒～数分）がかかる場合もある。従って、操作に対するリアルタイム応答性を確保するとともに、数秒間隔で１つの操作に対する進捗状況を操作中の保守・点検員にレポートするようにする。

(3)不正アクセスの防止

単なるデータの閲覧とは異なり、ここでは、機器の操作を伴う保守・点検を想定しているので、不正利用者による不正アクセスを防止しなければならない。

(4)アクセスログ

保守員、及び、作業内容（操作コマンド、時間、結果）の記録をとる。

３．３　プロトシステム

３．３．１　提案方法

本稿では図５に示す動作フローを持つ遠隔操作（制御）システムを提案する。その特徴を以下に示す。

(1)保守員の端末からは、Web サーバのみにアクセスできるようにし、アクセス権、操作権、操作コマンドが適正な場合のみ、Web サーバは保守・点検対象の機器（または、機器を管理するサーバ）に操作コマンドを転送する（３層構造）。

(2)クライアント側の Java アプレットで操作開始時に、暗号化[7]のための共通鍵をランダムに生成し、サーバ側の公開鍵を使って、共通鍵を暗号化し、ユーザＩＤを添付してサーバに転送する。サーバ側はユーザＩ

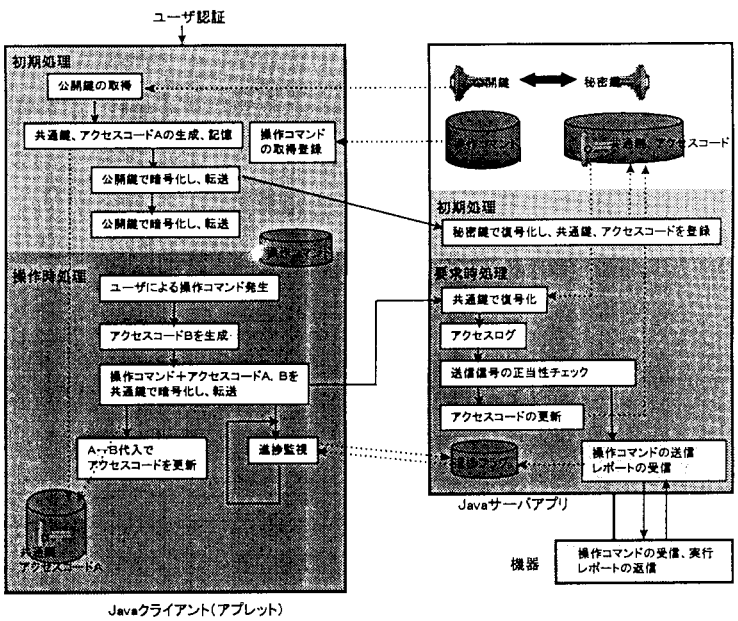

図５　動作フロー

Ｄと共通鍵を登録する。操作権が使用中であれば、登録を不成立にする。
(3)クライアント側で、操作時毎にアクセスコードをランダムに生成し、
「操作コマンド＋今回生成したアクセスコード＋前回生成したアクセスコード」
の３つを含むものを共通鍵で暗号化し、ユーザＩＤを添付してサーバに転送する。サーバ側は登録してある前回のアクセスコードが一致すれば、さらに、操作コマンドを機器へ転送する。さらに、新しいアクセスコードを登録する（不正操作フィルタリング）。
(4)操作コマンドを出した後、クライアント側の Java アプレットで、操作進捗確認用スレッドを生成し、周期的（数秒サイクル）にサーバ側の進捗状況レポートを確認しにいく。操作完了結果がレポートされていれば、スレッドを終了する（トランザクションモニタリング）。

３．３．２　効果

(1)リアルタイム応答性を考慮すると、Web 端末から直接的に保守・点検対象機器にアクセスする２層構造も可能だが、不正アクセスフィルタリング、排他処理など機器側のオーバーヘッドが大きくなるため、機器は、Web サーバのみとしか通信できない３層構造とした。
(2)毎回変化するランダムコードを生成し、操作コマンドに付加することで、同じ操作コマンドの転送でも通信上を流れる暗号は毎回異なるため、暗号解読のきっかけ（平文と暗号文の対）を与えにくい。
(3)操作端末毎にランダムコードを生成しているので、たとえ、同じＩＤとパスワードで別の保守員が別の端末から操作に介入しようとしても、ランダムコードが一致しないので、操作に介入できない。なお、別の操作端末からできるように（引き継ぎたい時など）するためには、操作権を持っている保守員が操作権を解放すれば可能である。
(4)暗号に必要な共通鍵は、操作時に動的に生成されて利用されるので、鍵管理数が最小限に抑えられる。また、共通鍵は端末ごとに異なるので、通信の双方向で暗号化が必要な場合にも利用できる。

4　まとめ

プラントデータを多目的な用途で利用するために、インターネット技術（特に Java）を適用した遠隔監視・制御システムを提案した。特に、分散オブジェクト技術には Java RMI、データベースアクセスには JDBC-ODBC ブリッジを適用し、応答性を確保するために、キャッシュ方式を提案した。本方式を実装した結果、キャッシュ利用時で１秒程度の応答性だった。また、点検・保守などの遠隔操作時の課題として、不正アクセス防止、排他処理をとりあげ、ランダムコードを付加した操作コマンドの暗号化送信を提案した。

今後は、RMI のファイヤーウォール(http プロトコルのみ通過)を超えての利用、または、ダイヤルアップ接続での利用、別タイプの JDBC ドライバの利用などについて評価する予定である。

参考文献、ＵＲＬ

[1]http://java.sun.com/products/jdk/1.1/docs/
[2]http://www02.so-net.or.jp/~kikuta/jdbcnote/jdbcnote.html
[3]http://www.javacats.com/JP/articles/Qusay/RMI.html
[4]http://www.intlab.soka.ac.jp/~matsumi/material/java/RMITutorial/RMITutorial.html
[5]古谷ら：「水公共事業向けイントラネット応用情報システムの開発」、
情報処理学会第５６回全国大会講演論文集(3)、p329-330、1998
[6]電気学会：「公共施設研究会」、1998/4
[7]「インターネットセキュリティ 基礎と対策技術」、オーム社、1996