

HTTP を利用したデータ要求タイミングの動的決定方法

岡原 弘典[†]

三菱電機(株) 情報技術総合研究所[†]

1. はじめに

組込み機器の遠隔監視を行うシステムにおいて、機器を管理するユーザからは、遠隔地にある機器の動作状態の監視、チェックを事務所から行いたいという要求がある。また機器を納入するセットメーカでは、電話相談業務やメンテナンス業務において、客先での機器の運転状況や画面を遠隔地から直接確認、操作したいといった要求もある。このように、パソコンや携帯電話等の一般的な情報端末を使用して、遠隔地にある組込み機器の状況や操作画面をインターネット経由で監視・操作するシステムが求められている。

本論文では HTTP(Hypertext Transfer Protocol) を利用して監視データ等の継続的な送受信を行う場合に、受信側である Web ブラウザからのデータ更新要求リクエスト間隔を動的に変化させることで、受信側でのデータ更新間隔時間と、データ更新遅延時間を短縮させることを目的とする。

2. Web ブラウザを用いた遠隔監視システム

Windows の画面操作を遠隔地から行える機能としては Windows 標準のリモートデスクトップ機能^[1]がある。しかしリアルタイム OS で動作し、ウィンドウシステムでない組み込み機器を遠隔監視させるのには、この機能は適さない。また、組み込み機器を含めた操作画面の遠隔操作に広く用いられている RealVNC^{[2][3]}のような専用ツールもあるが、遠隔監視を行う全端末に専用ツールをインストールする手間が生じること、また監視対象と端末間のプロトコルを拡張したい場合には、全ユーザに専用ツールのアップデートを強いることになる。

現在は様々なプラットフォーム上で動作する汎用表示ツールとして、Web ブラウザが普及している。このため上記課題を解決するものとして HTTP を利用することとした(図 1)。

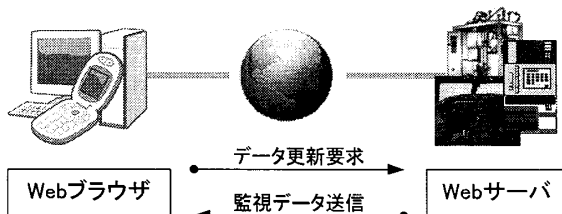


図 1. HTTP を利用した監視システム

A dynamic decision method of data request timing for HTTP, [†] Hironori Okahara, Information Technology R&D Center, Mitsubishi Electric Corporation

3. HTTP 利用における課題

遠隔監視等のアプリケーションにおいて「データの伝送効率を最大にして、受信側での遅延時間を最小にすること」を目標として考えると、技術課題は、Web ブラウザ上でのデータ更新間隔時間の短縮と、データ更新遅延時間の短縮の 2 点になる。以降では、これらについて考える。

図 2 は Web ブラウザと Web サーバ間の HTTP 通信例を示したものである。Web ブラウザではベースページを表示しながら、Ajax(Asynchronous JavaScript + Extensible Markup Language)技術を使用して、データ更新要求間隔 D_{req} おきに①、②、③の 3 回のリクエストを発信している。Web サーバ側では要求受信のたびにデータを作成して、レスポンスとしてデータを送信している。ブラウザ側では受信データによって表示内容を更新する。

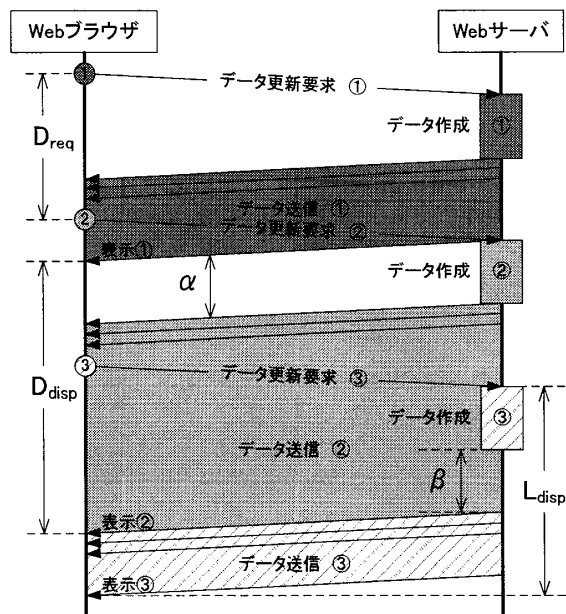


図 2. HTTP 通信例 (D_{req} 一定)

データ更新間隔時間 D_{disp} は、Web ブラウザにおいて、前回のリクエストによって表示内容が更新されてから、次に更新されるまでの時間である。②のケースでは、データ更新要求が遅すぎるため、前回リクエストのデータ送信完了後に Web サーバの送信バッファが空になり、伝送路が使用されていない期間 α が発生している。この結果 D_{disp} が増加している。データ更新間隔時間を短くするため、 α をできるだけ小さくする必要がある。

一方、データ更新遅延時間 L_{disp} は、Web サーバ

側でのデータの作成開始から、そのデータを Web ブラウザへ送信し、ブラウザ側で表示されるまでの時間である。③のケースでは、データ更新要求が早すぎるため、データ送信を開始する時点で前回リクエスト分のデータ送信が完了しておらず、送信バッファでの送信待ち時間 β が発生している。この結果 L_{disp} が増加している。遅延時間を短くするため、 β をできるだけ小さくする必要がある。

4. 提案方式

(1) 送信データサイズとデータ更新要求間隔

Web サーバ側で作成されたデータに対しては、送信データ量削減のために、既に Web ブラウザへ送信済みの情報からの変更点（差分）のみを送信データとすることが多い。あるいは更に可逆圧縮を行って送信データを作成することもある。いずれにせよ送信データサイズは毎回大きく異なる。

通常の Ajax 技術を使用した Web ページでは、定期的に更新要求を行うのが一般的であるが、上記のようなサイズが毎回異なるデータを送信する場合は、 α 、 β のような無駄な時間が大きくなる。

α 、 β を小さくする方法としては、毎回の送受信データ量のばらつきを少なくする方法と、送受信タイミングを変化させる方法が考えられるが、本論文では後者の方法について提案する。

(2) データ更新要求間隔時間の定式化

図 2 に対して、データ更新要求タイミングを動的に決定した場合の Web ブラウザ-Web サーバ間の通信の例を図 3 に示す。なお、データ更新要求間隔を D_{reqi} ($i \geq 2$)、Web サーバでのデータ作成処理時間を CR_i 、データ送信時間を RR_i 、通信遅延時間を L としている。

$\alpha_i + RR_i$ で表されるデータ更新間隔時間 D_{dispi} と、 $CR_i + \beta_i + RR_i + L$ で表されるデータ更新遅延時間 L_{dispi} を最小化するため、 $\alpha_i = 0$ 、 $\beta_i = 0$ となるように D_{reqi} を定式化すると、以下(A)式が得られる。

$$D_{reqi} = (CR_{i-1} + \beta_{i-1} + RR_{i-1}) - CR_i \quad \text{--- (A)}$$

注意点としては、Web ブラウザにおいて、Web サーバからの送信データの最初の packets を受信し、次のデータ更新要求間隔 D_{reqi} を算出する時点では、(A)式の項 RR_{i-1} と CR_i が確定していない点がある。例えば図 3 の②の時点において D_{req2} を計算する場合、 CR_1 と $\beta_1 (=0)$ は確定しており、データ送信によってこの値を Web ブラウザ側へ渡せるが、 RR_1 と CR_2 はまだ確定していない。

しかし、データ送信時間 RR_{i-1} については、伝送路のトラフィックが安定していれば送信するデータサイズからほぼ正確に予測可能であるため、データサイズで代用することができる。また次回データ作成時間 CR_i についても、毎回のリクエストに対しての変動が少ない場合は、前回リクエスト

時の値 CR_{i-1} で代用する方法や過去の CR の平均値を採る方法などが考えられる。

他の制約条件としては、Web ブラウザに RR_{i-1} の最初の packets が到着して初めて、次のデータ更新要求間隔 D_{reqi} が算出可能となる点がある（図 3 の②の時点）このことは以下(B)の不等式で表すことができる。なお、通信遅延 L を求めるために、予め Web ブラウザとサーバ間で通信テストを行っておく必要がある。

$$D_{reqi} \geq CR_{i-1} + 2L \quad \text{--- (B)}$$

以上の(A)、(B)2式より、遅延時間 L 、 β と、処理時間 CR 、 RR をもとにして、データ更新要求間隔 D_{req} を求めることができる。

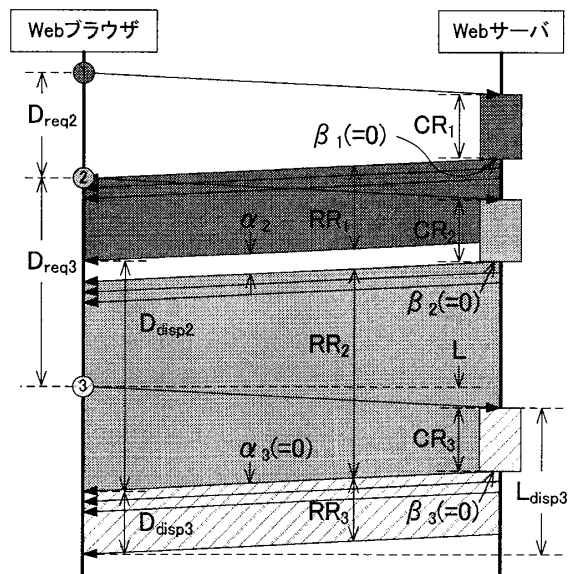


図 3. HTTP 通信例 (提案方式)

5. まとめと今後の課題

提案した方法を採用すれば、遠隔監視システムのようにデータを継続的に送信しながら、毎回のデータ量に大きな変動があるような HTTP 通信においても、Web ブラウザからのデータ更新要求のタイミングを動的に変化させることで、通信遅延による遅れ時間と、受信側でのデータ更新間隔時間を短縮させることができると考える。

今後は、定期的にデータ更新要求を行う方式と、提案方式を実システムに実装して、遅延時間 L_{disp} とデータ更新間隔時間 D_{disp} を測定して比較を行い、効果について検証を行う予定である。

【参考文献】

- [1] Microsoft Corporation, "WindowsServer2003 ターミナルサービスの技術概要", Jul 2002.
- [2] RealVNC : <http://www.realvnc.com/>.
- [3] T.Richardson, "The RFB Protocol", Aug 2008.