講演番号　0147
セッション 8 ソフトウェア工学

Cometを利用したProxyサーバの提案

Proxy Server Based on Comet

中野一樹　　奥野拓
Kazuki Nakano　　Taku Okuno

はこだて未来大学
Future University Hakodate

1．はじめに

Web 上には，多数の Web アプリケーションが存在している．ユーザは，それらの Web アプリケーションを利用する上で常に最新の情報を得たいというニーズを持っている．このニーズを満たすために，近年 Comet といわれる技術が登場した[1]．しかしながら，Comet を用いて作成されていない既存のアプリケーションに対しては同期性を確保することが難しい．また，既存の Web アプリケーションに直接 Comet を適用することは容易ではない．本研究では，このような問題を解決する手法を提案する．具体的には，クライアントとサーバ間で直接通信するのではなく，Proxy を利用した HTTP 通信に着目する．この Proxy に Comet を適用することで，ユーザにクライアントとサーバ間での同期性が確保された通信を提供する．

本研究では，クライアントとサーバ間で直接通信するのではなく，Proxy を利用した HTTP 通信に着目する．この Proxy に Comet を適用することで，ユーザにクライアントとサーバ間での同期性が確保された通信を提供する．

2．Webアプリケーションにおける同期性

従来の Web アプリケーションは，サーバがクライアントの HTTP リクエストに応じて HTTP レスポンスを返す．すなわち，通信を開始するためのトリガーはクライアント側にある[2]．このため，サーバ側で発生した変化をクライアント側に反映させることができない．

例として Web ブラウザ上で利用できるチャットアプリケーションを取り上げる．チャットアプリケーションのサーバには複数のクライアントから不定期にリクエストが送られてくる．自分以外のユーザがメッセージを送信したタイミングで，サーバに繋がっているすべてのクライアントにメッセージを反映させる必要がある．

そのために HTTP 通信を用いたチャットアプリケーションでは，クライアントからサーバに対して定期的にリクエストを送るポーリング方式を使用している．これを図 1 に示す．図 1 におけるイベントは他のクライアントからメッセージが送信されたことを示している．

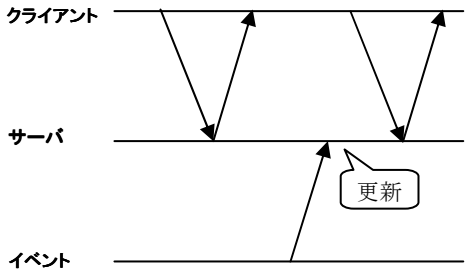

図 1 Web アプリケーションの通信

しかしながら，ポーリング方式の場合，リクエストの頻度を上げることで，同期性を向上させる反面，サーバへの負荷，クライアントとサーバ間のネットワークトラフィックを増加させることに繋がる．

3．Cometを用いたWebアプリケーション

Comet では，クライアント側が Web サーバに対して HTTP リクエストを送信し，サーバ側はそのリクエストに対し直ちにレスポンスを返さず，セッションを保持する．その後，Web サーバでイベントが起きると，保持していたリクエストに対してレスポンスを送信する．ブラウザ側はふたたび応答待ちの状態に戻る．この方式を取ることで，擬似的に Web サーバからのデータ送信を可能にし，リアルタイムにクライアントと通信を行える[]．すなわち，HTTP によるサーバプッシュ型の通信である．Comet を利用した場合のデータの流れは図 2 の通りである．定期的なリクエストによるアクセスの集中を回避でき，必要なときだけ通信を行うのでネットワークへの負荷が少ないというメリットもある．しかし，HTTP リクエストのセッションを保持しておくことで，サーバ側のプロセスやスレッドが開放されない状態になる．これは，メモリの浪費に繋がり，サーバに負荷がかかってしまうが，様々な Comet サーバ開発プロジェクトにおいて解決策が取られている．

Comet サーバの代表例として，Jetty6[4]，ShootingStar[5]，Cometd[6]が公開されている．ShootingStar は大規模運用にフォーカスされて開発されており，Cometd は，Python，Java，Perl でそれぞれ実装されている．

4．Cometを用いたProxyサーバ

Web サーバが Comet を用いて実装されていない限りサーバプッシュ型の通信を行うことはできない．既存の Web アプリケーションに Comet を適用することは，Comet

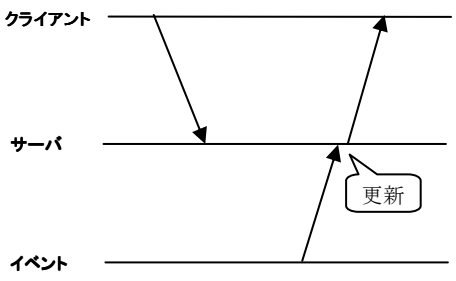

図 2 Comet を利用した場合の通信

を用いることで，イベントの定義といったアプリケーションの仕様から検討を行った上で，実装する必要があるため，容易ではない．

しかし，Proxy を利用する場合，Proxy が Comet を用いて実装されていればサーバプッシュ型の通信を実現することが可能である．本研究では，Comet の機能を持った Proxy サーバを構築する．ここでの Proxy サーバはクライアントと Proxy サーバの間では Comet を利用し，Proxy サーバと Web サーバの間でポーリングを行う．ポーリングを行うことによって，ネットワークの負荷を高めてしまうが，複数ユーザが Proxy サーバを使うことを想定した場合，各々のユーザがポーリングを用い Web サーバにアクセスを行うより，Proxy サーバが一括してポーリングを行うことで，ネットワークの負荷を軽減することできると考えられる．システムの利用イメージを図 3 に示す．

本研究で構築する Proxy サーバを用いて，Web サイトの閲覧を行った際の通信は以下の通りである．

1. クライアントは Proxy サーバに対して HTTP 通信を行う．
2. Proxy サーバが，Web サーバへの通信を代行する．その際，Proxy サーバはクライアントからのリクエストを保持する．
3. Proxy サーバは Web サーバに対して定期的に更新の確認を行う
4. 更新があった場合，クライアントに対して，Proxy サーバがレスポンスを返し，また応答待ちの状態に戻る．

以上を繰り返すことにより，閲覧している Web ページに更新があった場合，その内容をクライアントに反映することができる．上記の流れを図 4 に示す．

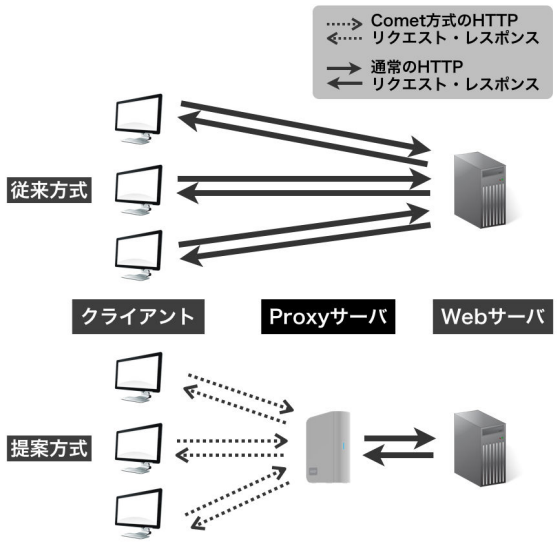

図 3　Proxy サーバを用いることによる通信の比較

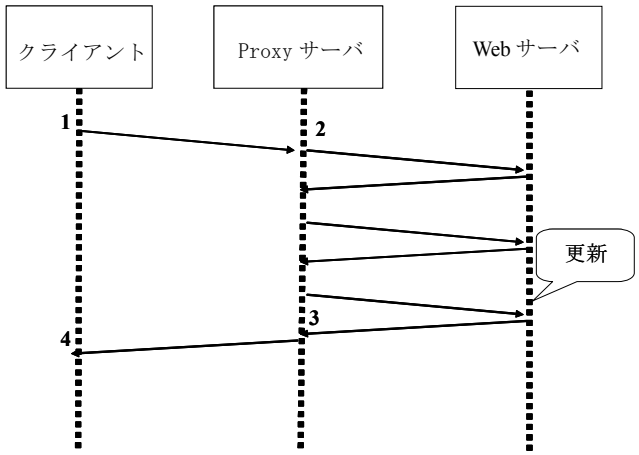

図 4　Proxy サーバを介したデータ通信

5. 実装

実装には，Comet サーバの機能を持つ Jetty6 を用い，Proxy サーバの機能を提供する ProxyServlet を用いて実装を行うことを検討している．Jetty6 は Java で実装された Web サーバである．ProxyServlet を用いることで，Jetty6 に HTTP リクエスト/レスポンスを代行する Proxy サーバとしての機能を持たせることができる．本研究では，Proxy サーバとしての機能と Comet サーバとしての機能を繋ぐ部分を新たに実装する．

6. まとめ

ユーザに対して，最新の Web ページを閲覧可能にすることを目的として Comet を利用した Proxy サーバの提案を行った．

複数のユーザがこの Proxy サーバを用いて Web サイトの閲覧を行った場合と，直接 Web サイトに接続して定期的にリロードを行った場合と，直接 Web サイトに接続して定期的にリロードを行った場合とのトラフィックの差を比較することで提案手法の有効性を検証する．

参考文献

[1] Comet(Programming), http://en.wikipedia.org/wiki/Comet_(programming).
[2] Lingr and Comet-技術解説編, http://blog.japan.cnet.com/kenn/archives/003149.htm.
[3] 瀧内元気，すぐわかる Comet，WEB+DB PRESS vol.41，p91‐p113，技術評論社，2007.
[4] Jetty6，http://www.mortbay.org/jetty-6/
[5] ShootingStar，http://rubyforge.org/projects/shooting-star/.
[6] Cometd，http://cometd.com/.