融合情報学輪講資料 2013/06/14

Web アクセススピード改善に関する研究動向
Recent Trend of Improving Web Access Speed

工学系研究科 電気系工学専攻 関谷研究室 修士課程 1 年 37-136482 藤居 翔吾

Abstract- **In accordance the explosive growth of smartphone users, Web access based on HTTP/TCP using a Web browser has become an indispensable service. The performance of web browsing depends on the aepect of hardware and software. In order to achieve the speeding up of web access, a lot pf paper are written. However, there are a few researches for reducing the number of HTTP transactions to achieve shorter response time.**

In this paper, the researches for reducing the response time are summarized and introduced. The response time of HTTP pipeline and Server Push in SPDY is compared by calculating the number of transactions. By the calculation, this paper concludes that SPDY achieve the fastest web access compared with HTTP pipeline. Some speeding up teqniques of Web access are introduced. The page-loading time can be shortened by about a third.

Index- **HTTP pipeline, Server Push, SPDY, Web access, Speeding-up, Round trips**

1 Introduction

ここ十年で, Web は私たちの生活に必要不可欠なものになった [1-6]. 特に近年では, スマートフォンの爆発的な普及によって, いつでも, どこでもインターネットを使って通信サービスを楽しめるようになった. 最近の報告によると, 2014 年にはスマートフォン契約数が過半数に達する, という予測もされている [7]. このことから, Web ブラウザを利用したインターネットアクセスが, 一つのインフラとして, 日常生活により密接に結びついていくと考えられる. ユーザーにとっての Web アクセスの満足度は, Web ページのリクエストを送ってから, 端末のディスプレイにページが表示されるまでの時間, すなわち, レスポンス時間に強く依存している [8-10]. 実際, Web ページのレスポンス時間の増加により, 利益の減少を引き起こすという報告もなされている.[12,13].

このような背景から, ネットワークにおける帯域幅の効率的な利用による, Web アクセス高速化の要求が高まり, 多くの研究が行われてきた [14-23]. 通信において, Web アクセスの性能は, レスポンス時間によって決定される. レスポンス時間に対する改善手法は, 対ハードウェアとソフトウェアの大きく二つに分ける事ができる. 本稿では, ソフトウェアの改善のみで, Web アクセス高速化を実現することを考える. また本稿では, 2 章で, レスポンス時間に寄与する要素と, それに基づく高速化に対するアプローチを示す. 3 章では, ページローディング時間改善に対する具体的な手法である, HTTP/1.1, SPDY で使われている高速化技術を示す. 4 章では, HTTP パイプライン, SPDY の評価結果を示す. 5 章では, これらの高速化技術における 1 度に要求できる最大リクエスト数をそれぞれ算出し, SPDY が最も高速化に寄与しうることを示す. 6 章では, SPDY を応用した Web アクセス高速化技術を紹介し, 今後の Web におけるレスポンス時間改善に対する道筋を示す. 最後に, 7 章では本稿で取り扱った Web アクセス高速化技術やその考え方をまとめる.

2 レスポンス時間

本章では, レスポンス時間に作用する要因や, 問題に対するアプローチ手法について述べる. 2006 年に発表された, WAN(Wide Area Network) アプリケーションのパフォーマンスに対するモデル式を以下に示す [5].

$$R \approx \frac{Payload}{Bandwidth} + RTT + \frac{APPturns(RTT)}{Concurrentrequests} + Cs + Cc \qquad (1)$$

上式において, 用いられているパラメータを以下にまとめる. Web アクセスの性能は主に以下の要素

からレスポンス時間を除いた, 7 個のパラメータによって示される.

- R : レスポンス時間
- $Payload$: ブラウザへ送られる総バイト数
- $Bandwidth$: 単位時間あたりに伝送できるバイト数
- $AppTurns$: 要求すべきリソースに対して送るリクエスト数
- RTT : コネクション確立のための信号を発してから応答が返って来るまでの時間
- $Concurrentrequests$: 同時に要求できるリクエストの数
- Cs : サーバでのデータ処理時間
- Cc : クライアントでのでの処理時間

実際に Web アクセスを改善するには, これら 7 つのパラメータのトレードオフを考慮しながら, 対策を行う. 現状, Web においては, クライアント・サーバシステムが主流であるので, クライアント, サーバ, クライアントサーバ間の通信の 3 つに対して, それぞれ適した改善が必要となる. 例えば, クライアント側で手軽に行える対策としては, パフォーマンスの良いメモリや CPU に換装し, 手持ちの端末の性能の向上により, データの処理時間を短縮することができる. しかし, 送信すべきデータの処理機構については, サーバ側の問題なので, 対策がとれない. 一方. クライアント側では, サービスを提供するバックボーンのネットワークにおいて, 広帯域の回線を使用する, などにより, レスポンス時間を改善することができる. しかし, 送ったデータの効率的な処理は, クライアント側で対策を行う必要がある.

このように, クライアント側, サーバ側, 通信機構でそれぞれ対策できる点, できない点があり, それぞれの特性を生かした対策を取る必要がある. 本稿では, 両者の通信機構に対して, ソフトウェアの改善による Web アクセス向上を調査対象としているので, 以下の条件のもと様々な高速化技術の比較を行う.

1. ペイロード, 帯域などの使用する回線の性能に依存するパラメータは考えない
2. サーバ側, クライアント側, のハードウェアの性能アップは考慮しない
3. サーバクライアント間の物理的距離や, ノイズに対する, RTT(Round Trip Time) のゆらぎはなく, 一定とする
4. 頻繁に接続が切れるような環境での, 再接続処理は考慮しない

このような条件のもと, 既存の HTTP/TCP の改善で, どのような高速化技術が最もレスポンス時間を小さくしうるのかを明らかにするのが本稿の目的の一つである.

3 HTTP/TCP による Web アクセス処理・高速化技術

本章では, 一般に広く行われている TCP 上で働く HTTP による Web アクセスの流れを示し, その高速化技術を紹介する.

3.1 Web アクセス

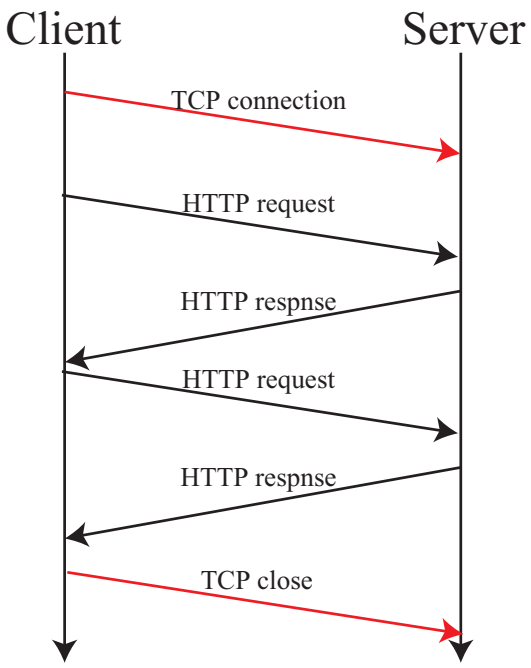

Fig.1 Processing flow of general HTTP over TCP request/response

Fig.1 に, HTTP と TCP による Web アクセスの処理動作を示す. Web アクセスは以下のような流れで行われる.

1. 3 ウェイ・ハンドシェイクによりクライアントサーバ間の TCP 接続を確立する
2. コネクションが確立すれば, クライアント側から HTTP リクエストを送る
3. クライアントからのリクエストに対し, サーバから HTTP レスポンスが送信される
4. クライアントが HTTP レスポンスを受け, データを受信する
5. クライアントが要求したデータを全て受信したら, クライアント側からコネクションの切断を要求し, 切断する

基本的に, HTTP では, 1 つのリクエストに対し, 1 つのレスポンスを返す. また, データの送受信は, 必ずクライアント側から開始されるのが, HTTP の特徴である. TCP の主な特徴は二つある. 一つは, コネクションを確立するのに計 3 回のトランザクションが必要であること. もう一つは, 1 つのサーバに対して, 複数のコネクションを確立する事ができ, 効率よく通信ができるという点にある. 同時接続数は, サーバとクライアントブラウザの設定によって決められるが, HTTP/1.1 においては 2 以下にする事が推奨されている [24].

3.2 HTTP パイプライン

Fig.2 に, HTTP パイプラインの動作を示す. HTTP パイプラインでは, クライアント側からの複数の HTTP リクエストが可能になる. これにより, 通常の HTTP で生じていた次のリクエストまでの待ち時間を短縮する事ができる. また, HTTP パイプラインでは, パイプライン化したフローは異なるプロセスでも単一のストリームとして論理的に解釈される.

3.3 SPDY

Fig.3 に, SPDY のサーバプッシュ機能の動作を示す [25]. SPDY は, Google が提唱した通信プロトコルであり, TLS 上で動くセッション層のプロトコルである. TLS は, TCP の上位に位置し, 暗号通信を実現する [26]. このため, TCP 接続のための, 3 ウェイ・ハンドシェイクの他に, TLS 接続のために, SSL ハンドシェイクを行う必要がある. これにより, 通常の HTTP 接続よりも, 通信開始時に多くトランザクションを必要とする.

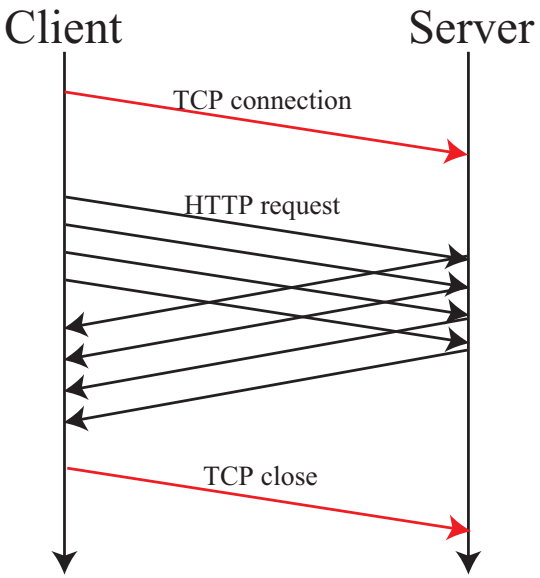

Fig.2 Processing flow of HTTP pipeline

SPDY の中の高速化技術の一つでサーバプッシュがある. サーバプッシュでは, クライアント側から HTTP リクエストを送信しなくても, サーバからデータを送信する事が可能になる. これにより, 不要な HTTP リクエストを減らす事ができ, 全体のトランザクションの数を減らす事ができる. さらに, SPDY ではマルチストリームでデータを要求する. このため, 複数のファイルをダウンロードする際, 複数のファイルに対し独立したストリームを生成するので, 一気にダウンロードを開始する事ができる.

4 HTTP パイプライン, SPDY の性能評価

本章では, HTTP パイプラインと SPDY の性能評価を示し, それらの比較を行う.

4.1 HTTP パイプライン

Fig.4 に, HTTP/1.1 のパイプライン化したリクエストを送ったときの 4 種類のサーバの性能を, 接

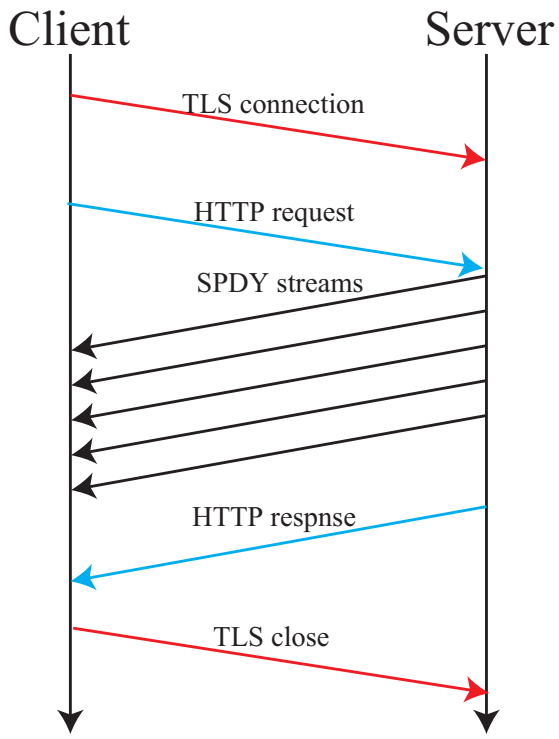

Fig.3　Processing flow of server push in SPDY

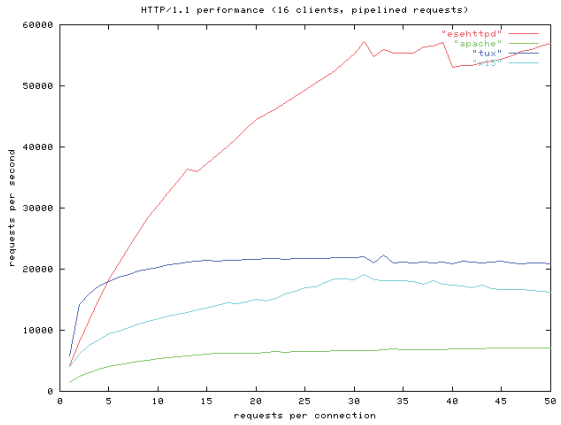

Fig.4　Performance of HTTP pipeline On/Off

続あたりのリクエスト数を変化させて測定した結果を示す [27]．4 種のサーバの内，“esehttpd” は HTTP パイプラインが On になっており，それ以外は Off になっている．また，この測定では，非常に小さいファイル (2 バイト) を転送している．この結果から，1 秒あたりに処理できるリクエスト数が増えればその分だけパイプライン化でき，処理できるリクエスト数が上がることが分かる．

4.2　サーバプッシュ-SPDY

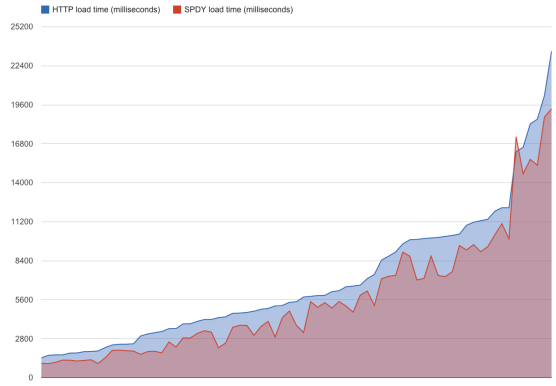

Fig.5　Comparison of SPDY vs. HTTP page load times

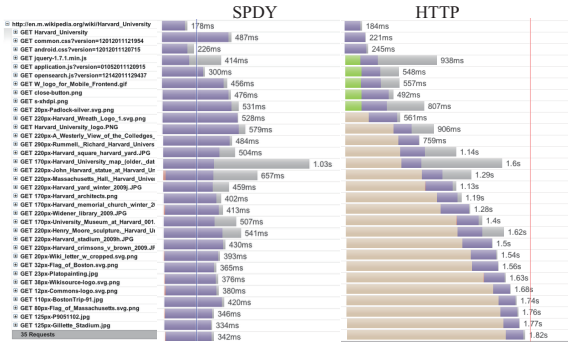

Fig.6　SPDY vs. HTTP load waterfall for web page of Harvard University

Fig.5 に，77 個の URL に対し，SPDY を適用した場合のページロード時間を測定した結果を示す [28]．この測定では，帯域を上り 1[Mbps]，下り 2[Mbps]，RTT150[ms]，パケットロス率を 1% 以下，という通信環境をエミュレートして行われた．通常の HTTP では，TCP コネクションを複数生成し，並列処理としてリソースをダウンロードする．今回の測定環境では，最大 6 本までコネクションを同時接続可能だった．その結果，Fig.6 に示すように，6 つのリソースに対しては，同時にレスポンスを受けられるが，残りのリソースに対しては，リソースのダウンロードが完了するまで待つ必要がある．一方 SPDY では，TCP 接続数は 1 本のみだが，マルチス

トリームにより全てのリソースに対して, リクエストを送信する事ができる. その結果オーバヘッドを削減でき, 平均 23% のロード時間を減少する事が可能となった.

5 HTTP リクエスト多重化

本章では, 1 度に要求できる最大リクエスト数をそれぞれ計算し, 比較を行う. 1 度に要求できる最大のリソース数は以下のように計算する事ができる.

$$MAX = TCPconnects \times HTTPrequests \quad (2)$$

本稿では, この定義を用いて HTTP パイプラインと SPDY の性能を比較する.

- HTTP パイプライン
  Firefox において, HTTP/1.1 に合致するサーバに対しては, TCP 同時接続数は 2 が規定値である [29]. パイプラインでの最大同時リクエスト数は 8 となっている [30]. 従って, 最大のリソース数は, 以下のようになる.

$$MAX = 2 \times 8 = 16$$

- SPDY
  SPDY では, TCP 接続数は 1 を推奨している. また, マルチストリーム数は, デフォルトでは無制限だが, 実装では 100 以上に設定するように推奨している. 従って, 最大のリソース数は以下のようになる.

$$MAX = 1 \times 100 = 100$$

上記の結果から一度に要求できる最大のリソース数は, HTTP パイプライン < SPDY となることがわかる. さらに SPDY では, 前述したサーバプッシュ機能により, 送るべきリクエスト数自体を減らす事ができるので, 総トランザクション数は, 上記の比較がより顕著に現れる事が期待できる. 例として, 200 個のリソースに対して, 総トランザクション数を計算すると, HTTP パイプラインでは 26 回, SPDY では, 3 回となり, HTTP パイプライン > SPDY となることが分かる。したがって, レスポンス時間は, HTTP パイプライン > SPDY となることが期待される.

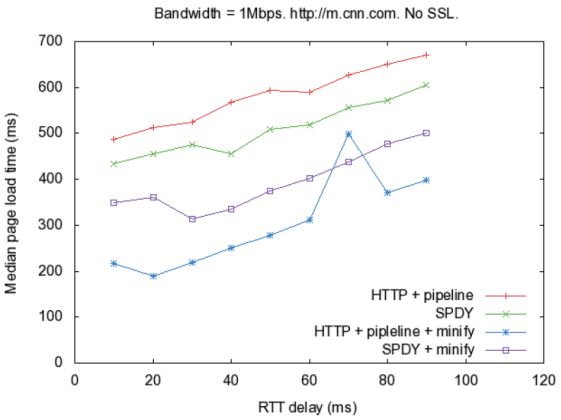

Fig.7 Transition and forecast of the number of subscribers of Smart-Phone

Fig.7 に, CNN のモバイルサイトにアクセスした時のページロード時間を測定した結果を示す [31]. このサイトには, 11 個のリソース (html ファイル 1 個, CSS ファイル 1 個, img ファイル 9 個) があり, 計 38.5[kB] のサイズである. 帯域は 1[Mbps] で, HTTP パイプラインの同時接続数は 11, SPDY は 1 という通信環境をエミュレートし, 測定された. 前述の計算よりも, HTTP パイプラインの接続数は多く設定してあるが, それでも SPDY による通信の方がページロード時間が短縮できていることがわかる.

これらの結果から, レスポンス時間を短縮するには, 総トランザクション数に依存し, 一度に要求できるリクエスト数が多い, またはトランザクション数を減らすことが Web アクセス高速化につながることがわかった.

6 先行研究

Table.1 に近年の Web ページ構成の変遷を示す [32]. この統計から, 現在の Web の構成が, より多くのサーバから, より多くの重いリソースのローディングを必要としており, そのような構成の Web ページへのアクセスに対し高速化を行うことが必要となっている.

Table.1 The trend of Web

	Nov 15 2010	Jun 1 2013
Avg. Page Size	702[kB]	1462[kB]
Resources/page	74	92
Domain	10	16

実際, SPDY では 1 つの TCP 接続のみ用いるため, 埋め込みデータなど, 複数のホストに対してリクエストを送る場合には, 十分効果を示さない事がある. Table.2 に近年報告された実際のアメリカの TOP500 の Web サイトに対する SPDY の効果を示す [33]. SPDY の仕様とこの結果から, 遅延が低い通信環境, サードパーティのドメインのような, 新たな TCP 接続が必要なリソースを含むページ, またそのリソースサイズが非常に小さいページにおいては, 効果が出にくいという事が分かる. これは, SPDY が, TCP と SSL により接続を確立するため, 通常の HTTP よりもトランザクション数が必要となり, 遅延が発生するである. この事実と昨今の Web の要求条件を踏まえて, 近年報告された SPDY を応用した Web アクセス高速化に対する先行研究を紹介する.

Table.2 Comparison with HTTP, HTTPS, and SPDY on mobile network

Network(Down/ Up[kBps], Latency[ms])	SPDY vs. HTTPS	SPDY vs.HTTP
Cable (5000/1000, 28)	SPDY 6.7%faster	SPDY 4.3%slower
DSL (1500/384, 50)	SPDY 4.4%faster	SPDY 0.7%slower
Low-latancy (780/330, 50)	SPDY 3%faster	SPDY 3.4%slower
High-latency (780/330, 200)	SPDY 3.7%faster	SPDY 4.8%slower

6.1 BoostEdge と SPDY によるデータとネットワークの最適化による Web アクセス高速化技術

Fig.4 に, 提案されたネットワーク構成を示す [34]. この研究では, 負荷分散のためにコンテンツ用 (Apache) とトランザクション用 (FlipServer)

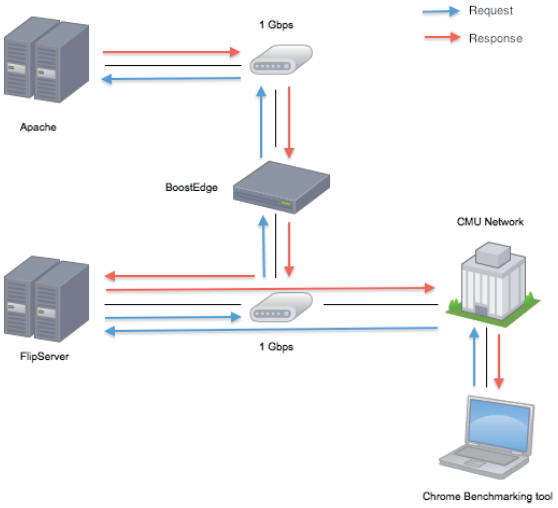

Fig.8 Dataflow for BoostEdge and SPDY combined

の 2 つのサーバを設けるような一般的な構成において, ページロード時間を短縮を目指している. FlipServer から Apache にリクエストを送る時に, SPDY を用いて, トランザクション数の短縮を行う. BoostEdge は, FlipServer と Apache 間のトランザクションに対して, データの最適化を行う. 最適化は主に, 頻繁にやり取りするデータのキャッシング, クライアントのスクリーンに適したサイズへの画像の非可逆圧縮を行う. すなわち, サイズが重いファイルには BoostEdge が, 比較的軽い複数のファイルには SPDY が作用する構成となっている. このようなテストベッドにおいて, 米国で最も訪れたサイトトップ 45 の内, 最も多くの HTTP リクエストを必要とするサイト上位 5 つに対して, そのコピーを二つのサーバに入れ, 測定を行った. その結果を Fig.9 に示す. この結果から, 単一のサーバに対する通常の HTTP と比較すると, トランザクション数が単純に増えるので, BoostEdge の効果が出ないが, SPDY を用いる事で, 不要なトランザクションを削減できるので, 二つのサーバをまたいで通信を行う事に対するトレードオフの影響を抑えることが可能になった.

しかしこのテストベッドでは, 一つのホストに対しリクエストを送るという点で, 現実の Web の状

況を想定しているとは言えない.

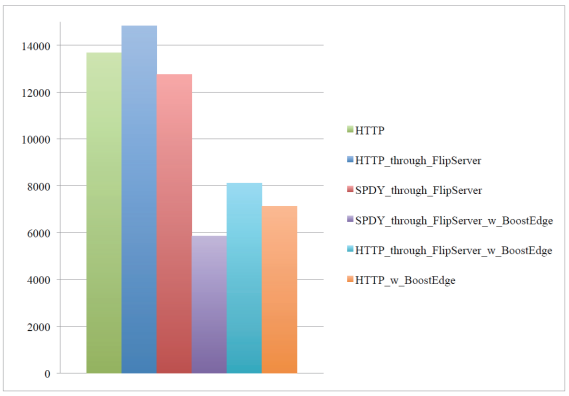

Fig.9　Results for five websites with the highest number of HTTP requests

6.2　キャッシュシステムと SPDY を応用した Web アクセス高速化技術

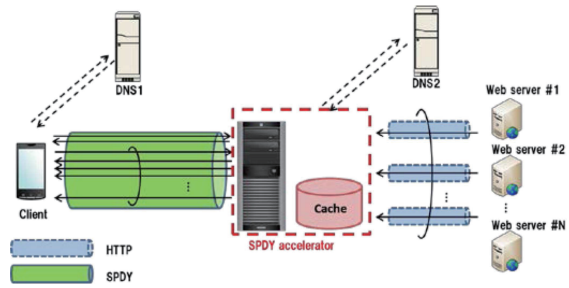

Fig.10　Transition and forecast of the number of subscribers of Smart-Phone

Fig.10 に, 提案されたネットワーク構成を示す [35]. この研究では, サードパーティからのリソースが埋め込まれたページに対して, SPDY に適したネットワークへ最適化し, ページロード時間の短縮を目指している. クライアントから, SPDY accelerator に対し SPDY でリクエストを送信し, SPDY accelerator は対象の HTML ファイルのあるサーバへリクエストを送り, データを受信する. SPDY accelerator が HTML データを受信したら, サードパーティリソースが埋め込まれていないかチェックし, 埋め込まれていた場合は, SPDY accelerator がリクエストを送り, キャッシュにそのデータを保存する. SPDY accelerator では, 送信可能なリソースから順次, クライアント側へレスポンスを返していく. このシステムでは, 本来複数のサーバに対し, 複数の TCP 接続が必要なため, SPDY の効果が現れにくかった構成を, クライアント側から見れば, 単一のサーバ (SPDY accelerator) に SPDY で通信を行う, と擬似的に構成することで, Web アクセスを高速化しようという試みである. このシステムを用いて, 7 つのサーバから 44 のリソース (HTML ファイル 1, CSS ファイル 1, img ファイル 42), 計 320[kB] に対して, RTT150[ms] でパケットロス 0% の通信環境で, Web アクセスを行った. その結果を Fig.11 に示す. この結果

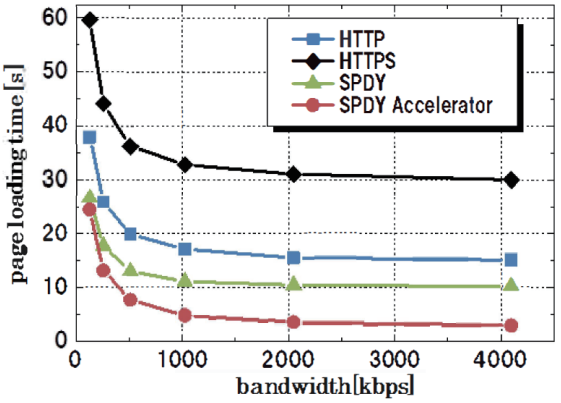

Fig.11　Transition and forecast of the number of subscribers of Smart-Phone

から, 通常の HTTP を用いたものと比べ, SPDY accerelator システムではページロード時間を 1/3 に短縮する事ができた. これにより提案手法は, マルチドメインで構成されるページ対して, SPDY を有効に作用させる事が確認できる. 今回, エミュレートした通信環境は, モバイル回線を想定したものであるが, さらにレイテンシが高い回線や, パケットロスの多い回線に対しても有効に働くのか検討する必要がある.

これらの報告から, Web アクセス高速化のためには, 総トランザクション数を少なくすることが必要であることが確認できた.

7 まとめ

本稿では Web アクセスの高速化技術を紹介した. 要求したデータに対して行われる全てのトランザクション数を算出することにより, HTTP パイプラインと SPDY のレスポンス時間を見積もり, Web アクセスをより速くするためには, 総トランザクション数の少ない手法を使用し, ページロード時間を短縮させることが必要であることを示した.

また, 先行研究で使用されている技術がこの原則に基づいていることを確認し, 現状の Web に対するアクセス高速化のためには, クライアント側で生成する TCP コネクションを集約することが必須であることを示した. 今後の研究動向では, レイテンシが高く, パケットロス率の高いようなモバイル通信環境において, Web アクセス高速化をどう行うべきか, という問題に対して, いかにトランザクション数を少なくできるかという方向へ研究が進んでいくと考えられる.

参考文献

[1] Karl Andersson and Dan Johansson “Mobile e-Service Using HTML5”, 37th IEEE Conference on Local Computer Networks (LCN), 2012.

[2] Ngu Phuc Huy and Do vanThanh, “Evaluation of mobile app paradigms”, MoMM’12 10th International Conference on Advances in Mobile Computing & Multimedia pp25-30, 2012.

[3] Eugen Pop and Victor Croitoru, “WEB Service Based Platform for Mobile Business”, Communications (COMM), 2012 9th International Conference pp293 - 296, June. 2012.

[4] Probir Ghosh, Andrew Rau-Chaplin, “Performance of Dynamic Web Page Generation for Database-driven Web Sites”, IEEE NWeSP 2006. International Conference, pp. 56-63, Sept. 2006.

[5] Peter Sevcik and Rebecca Wetzel , “Field Guide to Application Delivery Systems” , MSDN Magazine, 2006

[6] Z. Wang, F. X. Lin, L. Zhong, and M. Chishti. “Why are Web Browsers Slow on Smartphones?”, HotMobile ’11, pp91-96, Mar. 2011.

[7] Yo Nakajima, ” スマートフォン市場規模の推移・予測 (2013 年 3 月)”, http://www.m2ri.jp/newsreleases/main.php?id=010120130328500, Mar. 2013.

[8] Touch, Joe, John Heidemann, and Katia Obraczka. ”Analysis of HTTP performance.”, ISI Research Report ISI/RR-98-463, Dec.1998.

[9] Raj, R. Jeberson Retna, and T. Sasipraba. ”Web service selection based on qos constraints.” Trendz in Information Sciences & Computing (TISC) IEEE, 2010.

[10] Raj, R. Jeberson Retna, and T. Sasipraba. ”Web service selection based on qos constraints.” Trendz in Information Sciences & Computing (TISC) IEEE, 2010.

[11] Bouch, Anna, M. Angela Sasse, and Hermann DeMeer. ”Of packets and people: a user-centered approach to quality of service.” 8th International Workshop on. IEEE, 2000.

[12] Zona Research, ”The Need for Speed II,” Zona Market Bulletin, no. 5, April 2001.

[13] OnlineGraduatePrograms, Instant America http://www.onlinegraduateprograms.com/instant-america/, March 2012.

[14] Junichi Funasaka, “高ロス率ネットワークにおける分割ダウンロードの性能向上を目指すトランスポートプロトコルについての一検討”, 201th IEICE Technical Report pp.253-258, 2008.

[15] Huang, Junxian, et al. ”Anatomizing application performance differences on smartphones.” Proceedings of the 8th international conference on Mobile systems, applications, and services. ACM, 2010.

[16] Sevani, Vishal, and Bhaskaran Raman. ”Understanding HTTP traffic performance in TDMA mesh networks.” Communication Systems and Networks (COMSNETS), 2013 Fifth International Conference on. IEEE, 2013.

[17] Natarajan, Preethi, et al. ”SCTP: an innovative transport layer protocol for the web.” Proceedings of the 15th international conference on World Wide Web. ACM, 2006.

[18] Radhakrishnan, Sivasankar, et al. ”TCP fast open.” Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies. ACM, 2011.

[19] Dukkipati, Nandita, et al. ”An argument for increasing TCP’ s initial congestion window.” ACM SIGCOMM Computer Communication Review 40.3 (2010): 27-33.

[20] Agarwal, Sachin. ”Real-time web application roadblock: Performance penalty of HTML sockets.” Communications (ICC), 2012 IEEE International Conference on. IEEE, 2012.

[21] Xinogalos, Stelios, Kostas E. Psannis, and Angelo Sifaleras. ”Recent advances delivered by HTML 5 in mobile cloud computing applications: a survey.” Proceedings of the Fifth Balkan Conference in Informatics. ACM, 2012.

[22] Cohen, Edith, and Haim Kaplan. ”Prefetching the means for document transfer: A new approach for reducing Web latency.” Computer Networks 39.4 (2002): 437-455.

[23] Al-Fares, Mohammad, et al. ”Overclocking the Yahoo!: CDN for faster web page loads.” Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference. ACM, 2011.

[24] Fielding, R., et al. ”RFC 2616: Hypertext Transfer Protocol—HTTP/1.1, 1999.” URL http://www. rfc. net/rfc2616. html (2009).

[25] Belshe, Mike, and Roberto Peon. ”SPDY Protocol.” (2012).

[26] Dierks, T., and E. Rescorla. ”Rfc 5246: The transport layer security (tls) protocol.” The Internet Engineering Task Force (2008).

[27] Akira Higuchi, “Getting High Performance With Esehttpd”, http://esehttpd.sourceforge.jp/doc/ja/high-performance.html

[28] Matt Welsh, Ben Greenstein, and Michael Piatek, “SPDY Performance on Mobile Networks”, https://developers.google.com/speed/articles/spdy-for-mobile, April. 2012.

[29] Darin Fisher, “HTTP Pipelining FAQ”, https://developer.mozilla.org/en-US/docs/HTTP_Pipelining_FAQ, 2010.

[30] Firefox and Hacks, “The Truth About the Firefox “Pipelining” Trick”, http://egonitron.com/2007/05/25/the-truth-about-the-firefox-pipelining-trick/, 2007.

[31] Padhye, Jitu. ”A comparison of SPDY and HTTP performance.” (2012).

[32] “HTTP Archive”, http://httparchive.org/, June. 2013.

[33] Guy Podjarmy, “Not as SPDY as You Thought”, http://www.guypo.com/technical/not-as-spdy-as-you-thought/, June. 2012.

[34] Rosenberg, Steven, Surbhi Dangi, and Isuru Warnakulasooriya. ”Data and Network Optimization Effect on Web Performance.” (2012).

[35] Gen Mineki, Satoshi Uemura, and Teruyuki Hasegawa, “SPDY Accelerator for Improving Web Access Speed”, ICACT 2013, Jan. 2013.