Vol.2014-MBL-72 No.7
Vol.2014-CDS-11 No.3
2014/8/28

家電遠隔制御のための 大規模リアルタイム通信システムの設計と評価

南圭祐[†1]　　川添博史[†1]　安次富大介[†1]

本稿では家電遠隔制御のための大規模リアルタイム通信システムについて述べる．典型的なホームネットワークにおける実現性，スケーラビリティ，応答性といった観点に基づいてシステムを設計する．アーキテクチャを検討するに当たって，ランデブー方式とフォワーディング方式の二方式が考えられる．実験により二方式の性能評価を行い，ランデブー方式がメッセージ転送のスループットにおいて優れていることが分かった．

Design and Evaluation of Large-scale and Real-time Remote Control Architecture for Home Appliances

KEISUKE MINAMI[†1]　HIROSHI KAWAZOE[†1]
DAISUKE AJITOMI[†1]

This paper presents the system architecture of bidirectional communication for remote control of home devices. The system architecture is designed in view of the requirements such as feasibility for typical home users, scalability, and responsiveness of interactive operations. Considering these requirements, we have two approaches: Rendezvous architecture and Forwarding architecture. The experimental results in a testbed environment indicate that the Rendezvous architecture provides higher throughput of message transferring than Forwarding architecture.

1. はじめに

近年，インターネットに接続された家電機器の増加に伴い，これらを遠隔制御する利用形態が普及しつつある（図1）．例えば，宅外に居るユーザがスマートフォン上のアプリ（ユーザアプリ）を操作し，テレビ番組の録画予約や，照明のオンオフ，帰宅前の室温調節，家庭エネルギー管理システム（HEMS）の様子の確認などを行うサービスが既に実現されている．

家電遠隔制御に関しては多くの研究がなされており，近年では，Web 技術とクラウドコンピューティング技術を用いてサービスの柔軟性や可用性を向上させる取り組みが複数存在する[1][2][3]．その中で Masuo ら[4]は，家電を低遅延かつインタラクティブに扱うために WebSocket を採用したシステムを提案しているが，大規模な数のクライアントを収容するための性能やスケーラビリティが考慮されていない．

そこで本稿では，大規模化を想定した家電遠隔制御のための双方向通信システムのアーキテクチャについて述べる．ユーザとサービス提供者の双方の観点から生じる課題を整理し（2 節），ランデブー方式とフォワーディング方式について検討する（3 節）．これらは典型的なホームネットワークに適用可能かつスケーラブルで，特にランデブー方式はインタラクティブな操作に対する応答性を最適化している．実験環境を構築してこれらの方式を評価した（4 節）．最後に，結論を述べる（5 節）．

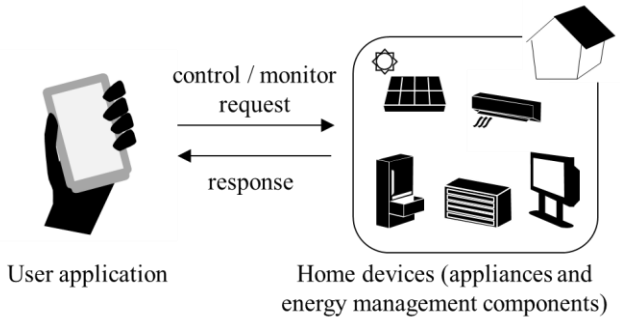

図 1. 遠隔制御の概要

2. 課題

本節では，ユーザとサービス提供者の双方の観点から生じる三つの課題について述べる．

2.1 典型的なホームネットワークにおける実現性

システムは, 典型的なホームネットワークに適用可能で，ユーザの重荷となるコストや手間を無くすことが望ましい．ユーザの中にはグローバル IP アドレスが利用できない場合があるが，システムはユーザに ISP や契約内容の変更を要求しないことが望ましい．また，専用ハードウェアの購入やホームルータの複雑な設定（パケットフィルタリングやポートフォワーディング等）が不要なことが望ましい．

2.2 スケーラビリティ

システム性能はクライアント数に従って拡張可能である

†1 (株)東芝　研究開発センター
Corporate R&D Center, Toshiba Corp.

べきである．クライアント数が増加すると，トータルのメモリ消費とメッセージレートが増加するため，それに対応できるだけのメモリ搭載量とメッセージ処理性能がシステム側に要求される．

一般的なサービスでは，クライアント数はサービスインの時点から徐々に増加していく．予め十分なシステムリソースを用意してサービスを開始させることも出来るが，不要な運用コストがかかる場合がある．

2.3 応答性

家電機器とユーザアプリの間で低遅延の双方向通信が行えるべきである．近年の家電機器はしばしば多くの操作の組み合わせを必要とする（例えば，現状の値を取得し，それに基づいて新しい値を設定し，設定後の値を確認するなど）．もしそれぞれの操作に時間がかかってしまうと，トータルの使い勝手に大きな影響を与えることになる．

3. アーキテクチャ設計

3.1 WebSocket

典型的なホームネットワークにおける実現性を考慮すると，WebSocket[5]は有効な技術である．クラウドの WebSocket サーバが家電機器とユーザアプリの WebSocket コネクションをそれぞれ収容し，通信の中継ポイントとして振舞う．これにはいくつかのメリットがある．(1) コネクションの確立は常に家庭内から行われるため，ファイアウォールでフィルタされることがない．(2) グローバル IP アドレスや専用機器，ホームルータの設定が必要ない．(3) コネクションが一度確立されれば，データ交換の際のオーバヘッドが小さい．(4) HTTP Proxy を挟んでも通信することが出来る．

3.2 メッセージ転送アーキテクチャ

運用中の単一サーバをスケールアップさせることは困難であり，スケールアップ自体にも限界が存在する．従ってスケーラビリティを達成するためには，システムを複数の WebSocket サーバを使って構成出来，またその台数を任意の数まで増加させられることが望ましい．しかし，単一サーバのアーキテクチャと比較して，クライアント間でやりとりされるメッセージを複数のサーバ間で適切に処理する必要が生じる．一方で，家電遠隔制御を想定した場合，相互に通信を行うのはユーザのスマートフォン等とそのユーザが所有する家電機器となることに着目する．これにより，以下の二つの方式 (a) ランデブー方式と (b) フォワーディング方式が考えられる．

(a) ランデブー方式（図 2）は，通信を行う可能性のあるクライアントのペアを同一の WebSocket サーバに収容する．そのために，新たなサーバ（Rendezvous サーバ）を導入する．クライアントはコネクション確立前に Rendezvous サーバに問い合わせ，接続すべき WebSocket サーバの URI を取得する．

(b) フォワーディング方式（図 3）は，通信を行う可能性のあるクライアントのペアを別々の WebSocket サーバに収容することを許容する．即ち，クライアントが接続する WebSocket サーバは任意の方法によって選択することが出来る．クライアントからメッセージを受信した WebSocket サーバは，宛先クライアントが収容されている WebSocket サーバへ任意の Forwarding mechanism を使ってこれを転送する．

二つの方式の主な差は応答性であると考えられる．図 4 に示したように，フォワーディング方式ではデータが二つの WebSocket サーバを経由するが，ランデブー方式では一つの WebSocket サーバ内で転送が完結することが保証されるため，転送に要する時間が短くなると予想される．

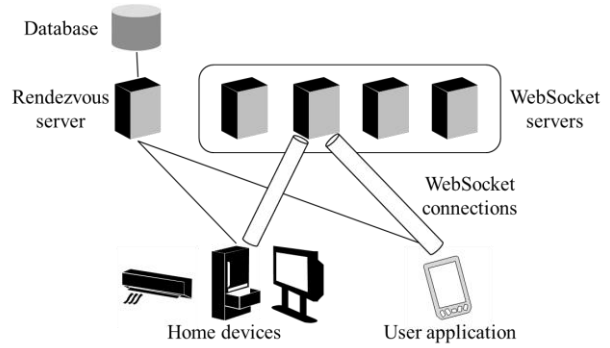

図 2. ランデブー方式

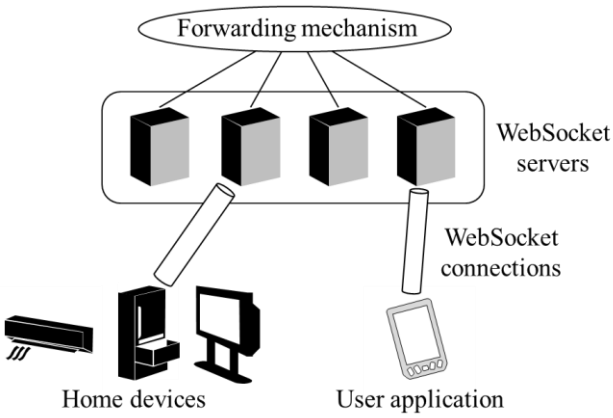

図 3. フォワーディング方式

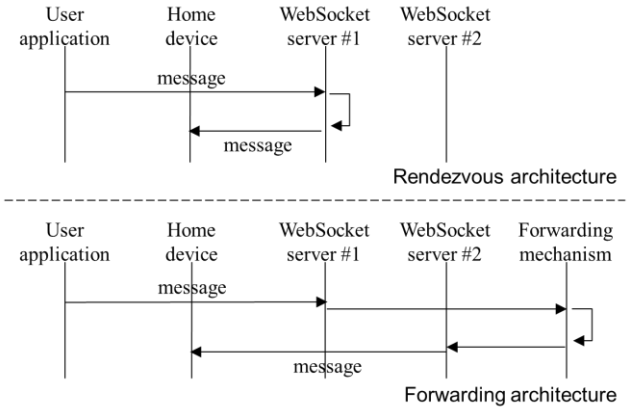

図 4. メッセージ転送のシーケンス

3.3 コネクション確立シーケンス

コネクション確立の手順について述べる．我々のユースケースでは，家電機器は常にユーザアプリに先駆けて WebSocket コネクションを確立し，そのコネクションを維持しつつ待機する．それから，ユーザアプリが起動され，遠隔制御を行うタイミングでコネクションを確立する．各 WebSocket サーバには固有の URI が割り当てられており，クライアントはその URI に基づいて WebSocket サーバに接続することが出来る．

3.3.1 ランデブー方式の場合

ランデブー方式のシーケンスは以下のとおりである．家電機器はリクエストを Rendezvous サーバへ送信する．Rendezvous サーバは WebSocket サーバを決定し，応答としてその URI を返す. 家電機器は指定された URI に接続する．次に，ユーザアプリが制御したい家電機器を記したリクエストを Rendezvous サーバへ送信する．Rendezvous サーバは該当する家電機器が接続している WebSocket サーバの URI を応答として返す．ユーザアプリは指定された URI に接続する．

3.3.2 フォワーディング方式の場合

フォワーディング方式のシーケンスは以下のとおりである．家電機器とユーザアプリは利用可能な WebSocket サーバの URI を任意の方法（例えばサーバに問い合わせるなど）によって取得する．次に，その中から任意の URI を選択して接続する．

4. 実装と評価

ランデブー方式とフォワーディング方式に基づく実験システムを Amazon Web Services (AWS) を使って実装した（表 1）．さらに，設計時の性能予測がどの程度の有意差を伴うかを確認するために両方式の評価を行った．

4.1 ランデブー方式の実装

ランデブー方式では，二台の WebSocket サーバと一台の Rendezvous サーバで構成した（図 5）．家電機器が接続している WebSocket サーバの URI を保存するためのデータベースとして，memcached[6]を使用した．

4.2 フォワーディング方式の実装

フォワーディング方式では，WebSocket サーバ二台と，利用可能な WebSocket サーバの URI をクライアントへ提供するサーバ（Rendezvous サーバと同等スペック）一台で構成した（図 6）. WebSocket サーバ間の Forwarding mechanism として，Redis[7]の Pub/Sub 機能を使用した．Pub/Sub 機能とは，あるキーに対して SUBSCRIBE しておくと，そのキーに対して PUBLISH されたデータを受け取ることができる機能である．フォワーディング方式の場合，クライアントが接続した WebSocket サーバはそのクライアントのユニークな ID（client ID）に対する SUBSCRIBE コマンドを Redis サーバへ発行する．クライアントからメッセージを受信した WebSocket サーバは宛先 client ID に対する PUBLISH コマンドを Redis サーバへ送信する．これにより，宛先クライアントを収容している WebSocket サーバがメッセージを受け取ることができ，WebSocket サーバはそれを宛先クライアントへ転送する．

4.3 測定

負荷クライアント用のインスタンスをサーバインスタンスと同じリージョンに設置し，1 万～10 万 TLS コネクション（家電機器とユーザアプリが 9:1 の割合）を確立する．その後，各ユーザアプリから家電機器に対して合計 100 万メッセージを送信した．

各メッセージの転送にかかった時間を図 7 に，システム全体のメッセージ転送のスループットを図 8 に示す．

4.4 評価

図 7 より，クライアント数に関わらず，ランデブー方式のほうがフォワーディング方式よりも 0.5ms ほど遅延時間が短いことが分かる. しかし, RTT が十分大きい場合（10ms～）にはこの差はほとんど意味を為さないため，3.2 節で述べた予想に反して，アプリの操作に与える影響は小さいと考える．

図 8 より，クライアント数に関わらず，ランデブー方式がフォワーディング方式よりもスループットが 15%程度優れていることが分かる．2.2 節で述べたように，メッセージ処理性能はクライアント収容台数に影響するため，ランデブー方式のほうが少ないサーバ台数で多くのクライアントを収容でき，運用コストの点で有利であると考える．

表 1. 実験環境

Cloud infrastructure	AWS (Tokyo region)
Load balancer	ELB (only for Rendezvous server)
Database (Rendezvous arch.)	ElastiCache cache.m1.small Engine: memcached 1.4.14
Forwarding mechanism (Forwarding arch.)	ElastiCache cache.m1.small Engine: redis 2.8.6
Rendezvous server	EC2 m1.medium x 1
WebSocket server	EC2 m1.medium x 2
Virtual clients	EC2 m1.medium x 5
RTT between EC2s	About 1ms

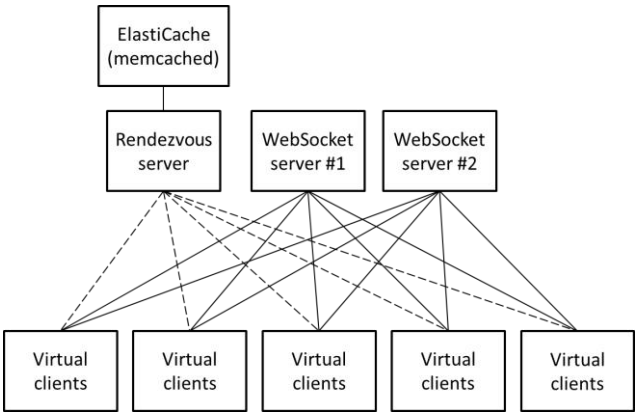

図 5. ランデブー方式のインスタンス構成

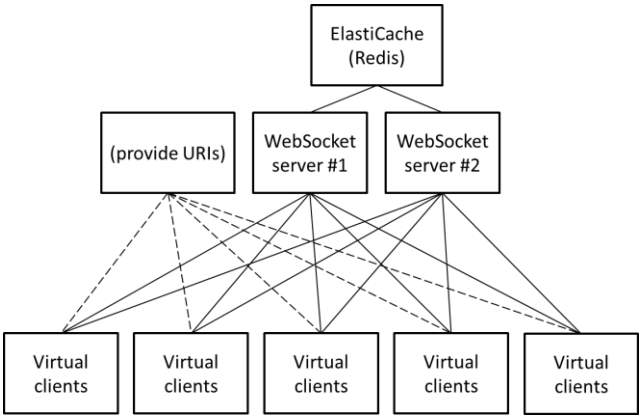

図 6. フォワーディング方式のインスタンス構成

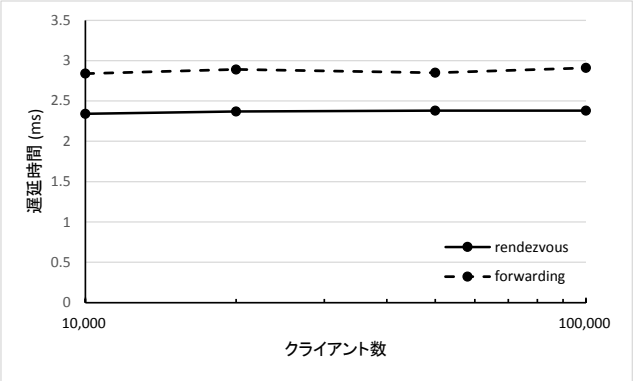

図 7. メッセージ転送時間

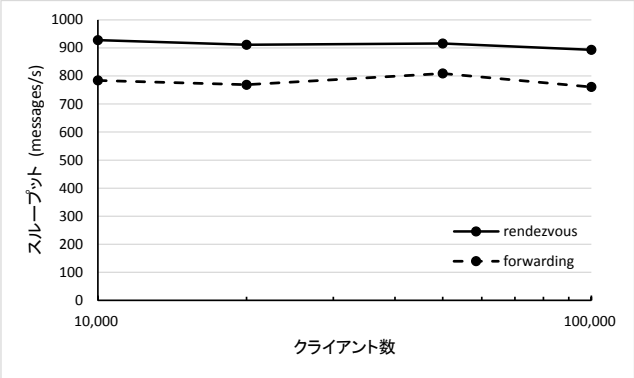

図 8. メッセージ転送スループット

5. おわりに

本稿では，家電遠隔制御のための双方向通信システムのアーキテクチャについて，実現性とスケーラビリティ，応答性を考慮した二つの方式を検討した．実験結果より，メッセージ転送の遅延時間において両方式は実用上の差は無いと考えるが，メッセージ処理性能においてランデブー方式のほうがフォワーディング方式よりも優れていることを示した．今後はより多くのクライアント数（100 万～）においても同様の傾向が得られるか検証を進めたい．

参考文献

1) M. Amarnath, Home-Appliance Control using Mobile Cloud Technology in Web2.0 Platform, Procedia Engineering, vol. 38, 2012, pp. 3587-3595.
2) Soliman, M. et al., "Smart Home: Integrating Internet of Things with Web Services and Cloud Computing," CloudCom, 2013 IEEE 5th International Conference on, vol.2, pp.317,320, 2-5 Dec. 2013.
3) Lih-Jen Kau et al., "A cloud network-based power management technology for smart home systems," Systems, Man, and Cybernetics, 2012 IEEE International Conference on, pp.2527,2532, 14-17 Oct. 2012.
4) T. Masuo et. al., “Study on HEMS over Cloud System utilizing Realtime Web Technologies,” IEICE Tech. Rep., vol. 112, no. 350, NS2012-117, pp. 1-6, Dec. 2012.
5) Fette, I. et al., “The WebSocket Protocol,” Internet Engineering Task Force (IETF), December 2011, http://tools.ietf.org/html/rfc6455.
6) memcached, http://memcached.org/
7) Redis, http://redis.io/