情報処理学会論文誌 Vol.55 No.7 1645–1654 (July 2014)

推薦論文

サーバプッシュにおけるモバイル端末のRRC状態を考慮したメッセージ配信遅延抑制方式

大西 健夫[1,a)] 城島 貴弘[1] 中島 一彰[1]

受付日 2013年11月12日, 採録日 2014年4月4日

概要：スマートフォンの利用拡大にともない，モバイル端末上のアプリケーションに対してイベント通知などのメッセージをクラウド上のサービスから任意の時点で配信するプッシュ配信が注目されている．3GやLTEなどのモバイル網では，モバイル端末の消費電力削減のため，無線のリソース確保状態（RRC状態）を遷移させており，RRC状態によってはプッシュメッセージの配信完了までに数秒単位の大きな遅延が発生する．また，プッシュ配信に利用するTCP接続を維持するためのKeep-Alive信号がモバイル網に輻輳を引き起こし，問題となっている．本稿では，Keep-Alive信号が不要となる配信時に接続を確立する方式のプッシュサーバに着目し，モバイル端末のRRC状態に応じた配信処理スケジューリングを実施することで，配信遅延の抑制を実現する手法を提案する．評価実験により，配信遅延を約13%抑制できたことを示す．

キーワード：モバイルネットワーク，配信遅延，RRC状態，プッシュ通知

Latency Reduction Method for Message Delivery of Server Push Based on RRC State of Mobile Terminal

Takeo Onishi[1,a)] Takahiro Shiroshima[1] Kazuaki Nakajima[1]

Received: November 12, 2013, Accepted: April 4, 2014

Abstract: Push Notification is event notice which is delivered anytime from services on the cloud to applications on mobile terminals. Push Notification attract rising attention according with expansion of smart phone. A transition of the radio resource control state (RRC state) aims at power consumption reduction in a mobile terminal using a mobile network such as 3G and LTE. But the transition of RRC state causes a delay with several seconds on delivering a push message. Moreover, Keep-Alive signals which keep a TCP connection used in Push Notification cause congestion in the mobile network. This paper proposes a scheduling method of delivery processing based on the RRC state for a push server which establishes the connection just before sending messages and does not need Keep-Alive signals. It is shown that our method can reduce delivery delays by 13%.

Keywords: mobile network, delivery delay, RRC state, push notification

1. はじめに

近年，移動体通信技術の進歩により，移動時にも高速通信が可能なスマートフォンの普及が拡大している．スマートフォンの特徴として，ユーザが任意のアプリケーションをインストールし，利用できることがあげられる．スマートフォンのアプリケーションに対してクラウド上のサーバから，SNSなどの更新通知，マルチメディア通信の着信や災害情報の通知などのメッセージをリアルタイムにプッシュ配信するプッシュサービスの利用が広まっている．従来，モバイル端末向けのプッシュサービスとしてSMSによるテキストメッセージが利用されてきた．しかし，SMS

[1] 日本電気株式会社
NEC Corporation, Kawasaki, Kanagawa 211–8666, Japan
[a)] t-onishi@cj.jp.nec.com

本稿の内容は2013年3月のマルチメディア通信と分散処理研究発表会にて報告され，同研究会主査により情報処理学会論文誌ジャーナルへの掲載が推薦された論文である．

図 1 IP プッシュのシステム
Fig. 1 System of IP push.

はモバイルキャリア独自のサービスであり，一般のアプリケーション開発者にはその利用が解放されておらず，また，モバイル網に接続された端末でしか利用できないという問題があった．そこで，プッシュサービスを IP ネットワーク上に構築することにより，アプリケーション開発者が容易に利用でき，かつ，IP 通信が可能な幅広い端末で利用可能な IP プッシュが提供されている（**図 1**）[1], [2], [3]．IP プッシュでは，アプリケーションサーバがコンテンツ更新通知などのメッセージ送信をプッシュサーバに依頼する．プッシュサーバは IP 通信を利用して，能動的に端末に対しメッセージを送信する．送信されたメッセージは端末上で動作するプッシュクライアントが受信し，宛先となるアプリケーションに対してメッセージを通知する．

IP プッシュでは，即時性の確保と通信負荷の抑制が重要となる．前述したマルチメディア通信の着信や災害情報の通知などに IP プッシュを利用する場合には，メッセージ配信の即時性が要求される．したがって，メッセージ配信遅延は IP プッシュにおけるサービス品質を表す重要な指標の 1 つとなる．また，IP プッシュは数百，数千万端末以上の多数の端末が利用することを想定しており，モバイル網に与える負荷を考慮したシステムとする必要がある．

メッセージ配信遅延の要因の 1 つとして，モバイル網における通信遅延があげられる．モバイル網を利用した IP 通信は，W-CDMA，LTE といった高速パケット無線アクセスの規格に則り実現されている．連続稼働時間が重視されるモバイル端末では，電力消費量を抑えるために，通信の必要がない場合は，通信時に確保した RAB（Radio Access Bearer）[4] を解放し無線通信を休止させている．この制御は RRC（Radio Resource Control）として規定されており [5], [6]，RAB の確保状態が RRC 状態として定義されている．RRC 状態間の遷移には，端末と無線局との間で RRC message の送受信が必要であり，この送受信に要する時間がサーバ・端末間での通信遅延という形で現れる．

また，プッシュサーバに負荷が集中し，同時に多数のプッシュ配信を実施しなければならない場合には，プッシュサーバ内の配信処理キューの中に待ち行列が発生することで，メッセージ配信に遅延が発生する．

次に，IP プッシュにおける通信負荷に視点を移す．通信負荷を左右する要因の 1 つとして，プッシュサーバと端末間の接続方式があげられる．IP プッシュにおけるプッシュサーバ・端末間の接続方式には，大きく分けて 2 つの方式が存在する．

1 つ目の方式は，端末からプッシュサーバに対して確立した TCP 接続を常時維持し，メッセージを配信する方式である（以下，TCP 接続維持方式と表記）．GCM（Google Cloud Messaging）[1] などがこの方式で，TCP 上の独自プロトコル，HTTP ロングポーリング [7]，Comet [8] や WebSocket [7], [9] などを用いて実装される．TCP 接続維持方式は，端末側からプッシュサーバに TCP 接続を確立するため，モバイル網とインターネット網との間に NAT（Network Address Translation）や Firewall 装置が介在する場合や，モバイル WiFi ルーターに接続された端末にもメッセージ配信が可能である．一方で，NAT や Firewall において接続を監視するテーブルにはタイムアウトが存在し，一定時間無通信時間が経過すると TCP 接続が切断されてしまうため，TCP 接続の維持には定期的な Keep-Alive 信号の交換が必要となる．その際，RRC 状態が休止状態になっていると，RRC 状態の遷移が発生する．このときに送受信される RRC message による負荷がモバイル網輻輳の一因となっている [10], [11]．特に，近年スマートフォンが普及したことで，SNS やインスタントメッセージなどのサーバ側から能動的に情報を通知する様々なアプリケーションの利用が広まり，Keep-Alive 信号によるモバイル網の負荷が急増している．そのため，モバイル網においては，Keep-Alive 信号による負荷への対処が重要な課題の 1 つとなっている [12]．また，定期的に通信を行うため，端末の電力消費量を増大させることにもなる [13]．

2 つ目の方式は，メッセージ配信時にプッシュサーバから端末に接続を確立して，メッセージを配信する方式である（以下，配信時接続確立方式と表記）[14]．TCP 接続維持方式と異なり，Keep-Alive 信号が不要なため，モバイル網に輻輳を引き起こさず，端末の電力消費量も抑えられる．一方で，プッシュサーバと端末の間に NAT が存在せず，同一のアドレス体系に属している必要があり，課題となる．しかし，通信事業者がプッシュサーバを導入する場合は，モバイル網に直接接続するネットワークインタフェースを設けることで課題を解消できる．そのため，配信時接続確立方式は，通信事業者がプッシュサービスを導入する場合には優れた方式であるといえる．

本稿では，通信負荷の点で優れた配信時接続確立方式において，モバイル端末の RRC 状態に基づくプッシュメッセージの配信制御を実施することで，多数のモバイル端末に同時にメッセージを配信する際の配信遅延（最大遅延および平均遅延）を抑制する方式を提案する．

2. モバイル端末における RRC 状態

本章では，モバイル端末の通信遅延に影響を及ぼす RRC 状態の詳細を，W-CDMA，LTE のそれぞれについて説明する．**図 2** は，W-CDMA における RRC 状態を示した図である．W-CDMA には，5 つの RRC 状態が定義されてお

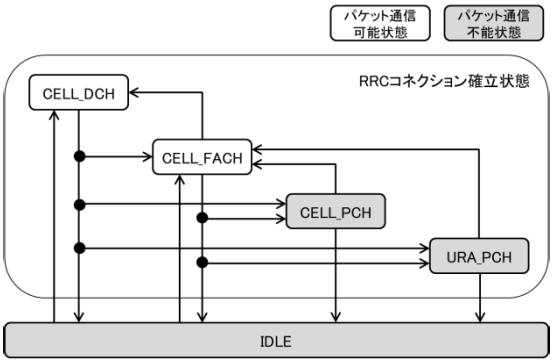


図 2 W-CDMA における RRC 状態
Fig. 2 RRC states in W-CDMA networks.

り，図中の上部に行くほど通信遅延が小さいが，消費電力が大きな状態となる [13]．以下，それぞれの状態について説明する．なお，説明文中示した *RTT*（Round Trip Time）は，インターネット上に配置したサーバとモバイル網に接続した Android 4.0 端末を利用して実測した値である．

- CELL_DCH
  IP 通信が可能な状態であり，5 つの状態のうちで最も通信遅延が小さい．サーバ・端末間の *RTT* を実測したところ，約 0.1 秒であった．消費電力は 5 つの状態のうちで最も大きい．
- CELL_FACH
  IP 通信が可能な状態で，通信量が少量の場合のみに用いられる．通信遅延は CELL_DCH に比べて大きくなり，*RTT* の実測値は約 0.5 秒であった．消費電力は，CELL_DCH の半分程度である [11]．
- CELL_PCH
  IP 通信は不可能な状態で，自端末宛ての IP パケットが無線基地局に届いているかを確認するための間欠受信（DRX）のみ行う．RRC message を交換するために無線基地局との間で確立される RRC コネクションは維持された状態である．IP 通信を行うためには，いったん RRC 状態を IP 通信可能な状態に遷移させる必要があり，通信遅延は CELL_FACH に比べて大きくなる．*RTT* の実測値は約 3 秒であった．消費電力は CELL_DCH の 2%程度である [11]．
- URA_PCH
  CELL_PCH とほぼ同じ状態であるが，端末が無線基地局単位ではなく，無線基地局をいくつか束ねたエリア単位で管理される．端末を持つユーザが高速移動している際に用いられる状態である．なお，URA_PCH 状態は観測されなかったため，*RTT* は実測できなかった．消費電力は，CELL_PCH とほぼ同程度である [11]．
- IDLE
  IP 通信は不可能な状態で，DRX のみ行う．CELL_PCH と異なり，RRC コネクションが開放された状態であり，CELL_PCH に比べて状態遷移に必要な RRC message 数が多いため，CELL_PCH に比べて通信遅延が大きくなる．*RTT* の実測値は約 4 秒であった．消費電力は，CELL_DCH の 1%程度となる [11]．

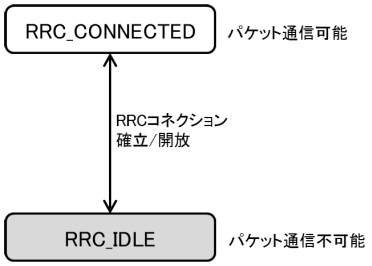


図 3 LTE における RRC 状態
Fig. 3 RRC states in LTE networks.

表 1 RRC 状態ごとのラウンドトリップタイム
Table 1 *RTT* value of each RRC state.

RRC 状態	*RTT*（秒）
CELL_DCH	0.11
CELL_FACH	0.45
CELL_PCH	2.6
IDLE	3.5
RRC_CONNECTED	0.047
RRC_IDLE	0.86

図 3 は，LTE における RRC 状態を示した図である．LTE においては 2 つの RRC 状態が定義されている．以下，それぞれの状態について説明する．

- RRC_CONNECTED
  IP 通信が可能な状態で，RRC コネクションが確立された状態である．通信遅延は CELL_DCH に比べて小さくなっている．*RTT* の実測値は約 0.05 秒であった．消費電力は，CELL_DCH に比べて大きい [15]．
- RRC_IDLE
  IP 通信は不可能な状態で，着信確認のための DRX のみ行う．RRC コネクションが開放された状態である．IP 通信を行う場合は，いったん RRC 状態を RRC_CONNECTED に遷移させる必要があるため，通信遅延は RRC_CONNECTED に比べて大きくなる．LTE は状態数を減らし遷移を簡略化することで，遷移時に必要な RRC message 数を削減し，RRC 状態遷移の高速化を図っている [16]．そのため，W-CDMA の IDLE 状態に比べると通信遅延は小さくなる．*RTT* の実測値は約 0.9 秒であった．消費電力は，RRC_CONNECTED の 2%程度である [15]．

以上のように，モバイル端末の通信遅延は，RRC 状態に応じて大きく異なる．**表 1** に，実測したサーバ・端末間の *RTT* をまとめた．なお，表中の *RTT* の値は，各状態につき 200 回以上の測定を実施し，得られた *RTT* の平均値となっている．

3. 配信時接続確立方式における配信処理

本章では，配信時接続確立方式の配信処理について説明する．

図 4 は，配信時接続確立方式におけるメッセージ配信時の処理の流れを示している．プッシュサーバはアプリケーションサーバからメッセージ配信要求を受け取ると，端末情報データベースにアクセスし，宛先となる端末の情報を取得する（①）．このとき，取得する情報は宛先となる端末の IP アドレス，ポートなどである．

次に，プッシュサーバは配信するメッセージをメッセージデータベースに保存する（②）．その後，端末に対して，配信するメッセージが存在することを示すトリガを送信する（③）．トリガを受信した端末はプッシュサーバにメッセージ取得要求を送信する（④）．メッセージ取得要求を受信したプッシュサーバは，メッセージデータベースを参照し，配信するメッセージを取得する（⑤）．最後に，プッシュサーバはメッセージ取得要求に対する応答として，メッセージを端末に送信する（⑥）．プッシュサーバは負荷分散のために複数台で構成されていてもよく，その場合，メッセージ取得要求は，いずれかのプッシュサーバに割り振られる．

なお，本稿では，トリガとして，TCP の接続確立時のパケットを利用し，TCP の 3way ハンドシェイク完了をもって，端末側でトリガを受信したと判断する．トリガ内にメッセージを入れず，端末からのメッセージ取得要求への応答でメッセージを送信するのは，端末認証を可能とし，セキュリティを確保するためである．そのほかにトリガに関するセキュリティの問題として，悪意のある第 3 者がトリガを大量に送信し，端末の電力消費増大を引き起こす攻撃があげられる．しかし，プッシュサーバを経由した不正なトリガ送信は，プッシュサーバの認証機構により防ぐことができる．また，モバイル網内に接続した第 3 者がプッシュサーバに偽装して，トリガを送信する可能性もあるが，偽装攻撃は，通信事業者がセキュリティ設定を施すことで防止できる．たとえば，プッシュサーバ以外からのトリガ受信ポート宛てのパケットをモバイル網内で破棄する，もしくは，モバイル網に接続する端末から同一モバイル網内の他の端末に直接パケットを送信できないようにする，などの対策がある．

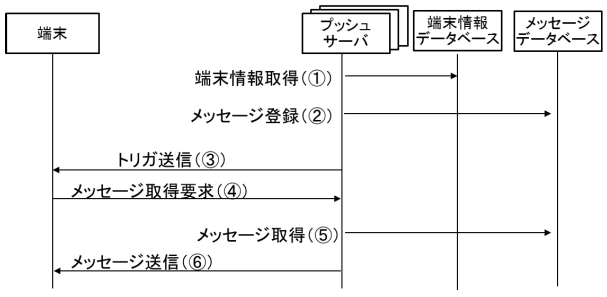


図 4 メッセージ配信処理
Fig. 4 Message delivery processes on IP push.

上述したメッセージ配信における遅延要因としては，プッシュサーバにおけるメッセージ配信処理遅延とプッシュサーバ・端末間の通信遅延があげられる．プッシュサーバにおけるメッセージ配信処理遅延はデータベースからの端末情報取得処理や，データベースへのメッセージ登録処理などに起因する遅延であり，個々の処理遅延は小さいが，メッセージ配信要求が短時間に集中した場合には，数秒程度以上の遅延となる．

プッシュサーバ・端末間の通信遅延は，プッシュサーバから端末に送信されるトリガの通信時間，端末からプッシュサーバに送信されるメッセージ取得要求の通信時間，および，プッシュサーバから端末へ送信されるメッセージの通信時間から構成される．このうち，トリガの送信時に発生する遅延は，端末の RRC 状態に応じて数百ミリ秒から数秒程度まで変化する．たとえば，IDLE 状態の端末に対してトリガを送信しようとすると，CELL_DCH への遷移が発生し，トリガが端末に到達するまでに 4 秒程度の時間を要することになる．その後のメッセージ取得要求送信時やメッセージ送信時には，端末の RRC 状態がすでに通信可能状態に遷移しているため，RRC 状態の遷移は発生せず，おおよそ数百ミリ秒程度の遅延となる．

以上のように，メッセージ配信における遅延として，主として，プッシュサーバにおけるメッセージ配信処理遅延と，端末の RRC 状態に依存したプッシュサーバ・端末間の通信遅延が発生する．

4. 提案方式

本章では，モバイル端末の RRC 状態に応じて通信遅延が異なることに着目し，配信時接続確立方式のプッシュサーバにおけるメッセージ配信処理をスケジュールすることで，多数のモバイル端末に同時にプッシュメッセージを配信する際の配信遅延を抑制する方式を提案する．

4.1 トリガ先行方式

ここでは，まず，RRC 状態遷移とメッセージ配信処理を並列に実施させることで，遅延の抑制を図るトリガ先行方式を提案する．

トリガ先行方式では，**図 5** のようにトリガ送信処理を先に実施する．トリガを送信後，端末からのメッセージ取得要求が到着するまでには，数百ミリ秒から数秒程度の時間的余裕が存在する．その間にメッセージ登録処理を行う．トリガの送信を先行して行うことで，端末の RRC 状態遷移が早期に誘発されるようになるため，メッセージの配信遅延が低減することが期待できる．

処理順序を後にしたメッセージ登録処理は，端末からの

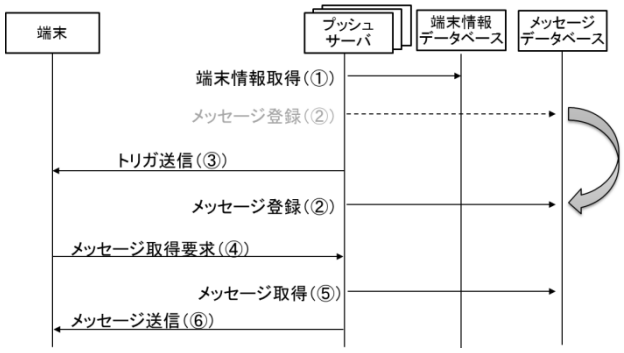


図 5 トリガ先行方式のシーケンス
Fig. 5 Sequence of method which prioritize trigger.

メッセージ取得要求をプッシュサーバが受信するまでには完了しなければならない．メッセージ登録処理が完了していない場合，メッセージデータベースを参照しても，配信すべきメッセージが登録されていない．そのため，メッセージが登録されるまでメッセージの配信を待たなければならず，メッセージの配信遅延が増大することとなる．

トリガ送信後，端末からのメッセージ取得要求が到達するまでの時間は最短で 0.1 秒程度（RRC_CONNECTED における通信時間）なので，約 0.1 秒でメッセージ登録処理が終了するようにすればよい．

4.2 RRC 状態スケジューリング方式

トリガ先行方式では，メッセージ登録処理をトリガ送信後，固定時間（約 0.1 秒）後までとした．しかし，端末の RRC 状態は IDLE の場合もあり，その場合は，メッセージ登録処理をさらに遅らせることができ，その分，ほかの端末に対するトリガ送信処理を早期に実施することが可能になる．

そこで，トリガ送信処理を優先的に実施し，かつ，RRC 状態を考慮して端末からのメッセージ取得要求を受信するまでにメッセージ登録処理が完了するように配信処理をスケジューリングする，RRC 状態スケジューリング方式を提案する．RRC 状態スケジューリング方式では，端末の RRC 状態に応じてメッセージ登録処理をデッドラインスケジューリング [17] によりスケジューリングする．メッセージ登録処理のデッドラインとなる時刻（D）は下記の式により決定する．

$$D = t + d_T + d_R \tag{1}$$

ここで，t はトリガを送信した時刻，d_T はトリガがプッシュサーバから端末に到達するのに要する時間，d_R は端末が送信したメッセージ取得要求がプッシュサーバに到達するのに要する時間である．d_T および d_R の値は，実測した RRC ごとのサーバ・端末間の RTT（表 1）に基づき算出する．RTT は，トリガ送信前の端末の RRC 状態（S_a），および，端末がトリガを受信することで遷移した後の RRC 状態（S_b）に依存する値である（S_a が CELL_DCH，CELL_FACH，RRC_CONNECTED の場合は状態遷移が発生しないため，$S_a = S_b$ となる）．まず，d_T は，トリガとして利用するプッシュサーバから端末への TCP の 3way ハンドシェイクが完了するのに要する時間，すなわち，状態 S_a の端末に対し SYN，SYN-ACK パケットを往復させる時間と，その後，状態 S_b に遷移した端末にプッシュサーバから送信した ACK パケットが到達するのに要する時間となる．$RTT(S)$ をプッシュサーバと状態 S の端末間の RTT とすると，

$$d_T = RTT(S_a) + 0.5RTT(S_b) \tag{2}$$

となる．また，d_R は端末が HTTP リクエストを行うための TCP 接続を確立し，その後送信する HTTP の GET リクエストがプッシュサーバに届くまでの時間となり，すなわち，状態 S_b の端末がプッシュサーバとの間で TCP の SYN，SYN-ACK パケットを往復させる時間と，その後送信した ACK パケットと GET リクエスト（ほぼ同時に送信）がプッシュサーバに到達するまでの時間となるので，

$$d_R = 1.5RTT(S_b) \tag{3}$$

となる．

式 (1) で与えられたデッドライン D に近いメッセージ登録処理を優先的に実行することで，トリガ送信を優先的に処理しつつ，メッセージ取得要求がプッシュサーバに届いたときには，メッセージ登録処理が完了した状態となるようにする．端末情報取得・トリガ送信処理の優先度は中程度とし，デッドラインに近いメッセージ登録処理が存在しない場合は，優先的にトリガの送信が行われるようにする．

なお，プッシュサーバはモバイル通信事業者の網内にあり，端末の RRC 状態はモバイル網内の装置から取得することを前提としている．また，1 つの無線基地局において，同時に通信可能な RRC 状態となることができる端末数（同時接続数）には上限が存在する [18]．同時接続数の上限を超えた場合には，数十秒単位の配信遅延が発生することとなる．しかしながら，一般的には，配信対象の端末は広域に分散しており，同時接続数の上限に達する可能性は低いため，本稿においては同時接続数の上限を考慮していない．

以上の方式により，プッシュサーバで多数のメッセージ配信処理が発生していても，早期に RRC 状態の遷移を誘発することができる．また，端末からのメッセージ取得要求が届いたときには，メッセージ登録処理が完了しており，即座にメッセージを送信することができるため，メッセージ配信遅延を低減することができると期待される．

5. 提案方式の評価

提案方式について，シミュレーションによる評価を実施

した．本章では，評価方法と結果について述べる．

5.1 評価方法

多数のモバイル端末を利用した評価が困難であったため，モバイル網での通信遅延を再現する基地局シミュレータと端末シミュレータを用いてシミュレーションによる評価を実施した．

図 6 に評価システムの構成を示す．端末シミュレータはプッシュメッセージを受信するモバイル端末の動作を模擬する．端末シミュレータは，計 1,500 個動作させ，2 台の PC 上でそれぞれ 750 個の端末シミュレータを動作させた．配信サーバは，メッセージのプッシュ配信を行うプッシュサーバである．端末情報（メッセージ配信対象端末の IP アドレスやポートなど），および，メッセージを保存するデータベースには，Key-Value Store（KVS）[19], [20] の 1 つである Couchbase Server 2.0 [21] を利用した．配信要求プログラムは，端末シミュレータへのプッシュメッセージ配信を要求するメッセージ配信要求を配信サーバに送信する．配信サーバ，KVS，および，配信要求プログラムは同一 PC 上に配置した．基地局シミュレータは，配信サーバと端末シミュレータの間に存在し，モバイル網の通信遅延を再現する．通信遅延の発生には，netem [22] を利用した．**表 2** に評価に用いたマシンの諸元を示した．

基地局シミュレータで発生させた通信遅延は，インターネット上に配置したサーバとモバイル網を利用した Android 4.0 端末で実測した値（表 1）に基づき決定した．端末シミュレータの RRC 状態は，**表 3** に示した比率で設定している．RRC 状態の比率は，Android 4.0 端末を使用して実測した RRC 状態ごとの総滞留時間の比率を採用した．測定は，2 ユーザに対して約 2 日間行い，2 ユーザの比率の平均値を採用した．測定した RRC 状態の比率を表 3 に示す．基地局シミュレータで発生させる遅延のうち，IDLE，CELL_PCH，RRC_IDLE の遅延に関しては，配信サーバから端末シミュレータに送信するトリガ（トリガには，TCP を利用しているため，より正確には，TCP の SYN パケット）のみに適用し，その後の通信については，遷移後の RRC 状態に基づく遅延を適用した．式 (1) のデッドライン時刻は，表 1 と式 (2)，(3) に基づき導出した値を使用した．**表 4** にトリガ送信時刻を起点としたデッドライン（式 (1) における $d_T + d_R$）を端末の RRC 状態ごとに示した．

評価では，配信要求プログラムが全端末シミュレータ宛て（1,500 端末）のメッセージ配信要求を配信サーバに送信し，メッセージ配信遅延を測定した．測定は，処理スケジューリングを実施しなかった場合（以下，FiFo 方式と表記），トリガ先行方式，RRC 状態スケジューリング方式の 3 種類について各々 100 試行ずつ実施した．各試行において，メッセージ配信遅延として，メッセージの受信を完了した端末数が i（$i = 1, 2, \cdots, 1{,}500$）となる時間（受信完了時間：$t(i)$），全端末に対する配信遅延の平均値（平均配信

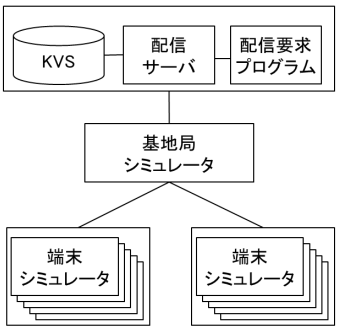


図 6 評価システムの構成
Fig. 6 System for evaluation.

表 2 評価マシンの諸元
Table 2 Specifications of machines for evaluation.

マシン	OS	CPU	メモリ
プッシュサーバ	Red Hat EL 6.1	Xeon 5160	10GB
ネットワークエミュレータ	Ubuntu 12.04	Pentium 4 3.6GHz	1GB
端末シミュレータ用 PC1	Ubuntu 12.04	Xeon 3.6GHz x2	5GB
端末シミュレータ用 PC2	Ubuntu 12.04	Xeon 3.6GHz x2	5GB

表 3 RRC 状態の比率
Table 3 RRC state rates in mobile terminals.

RRC 状態	ユーザ 1 (%)	ユーザ 2 (%)	平均比率 (%)
CELL_DCH	2.6	6.2	4.4
CELL_FACH	1.1	1.2	1.2
CELL_PCH	9.4	29.1	19.3
IDLE	55.1	42.2	48.7
RRC_CONNECTED	30.3	14.2	22.3
RRC_IDLE	1.5	7.0	4.3

表 4 RRC 状態ごとのトリガ送信時刻を起点としたデッドライン
Table 4 Deadline values of each RRC state.

	デッドライン（秒）
CELL_DCH	0.34
CELL_FACH	1.3
CELL_PCH	2.8
IDLE	3.7
RRC_CONNECTED	0.14
RRC_IDLE	0.95

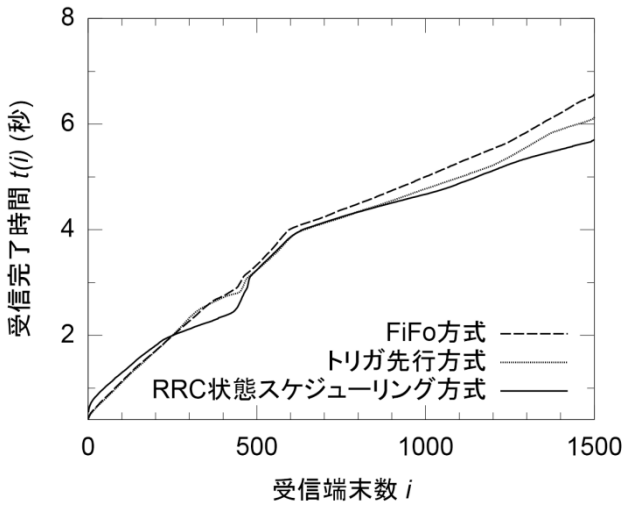


図 7 メッセージ受信時間

Fig. 7 Cumulative distribution of receiving times of push messages.

遅延：m），および，すべての端末がメッセージを受信し終えた時間（配信完了時間：M）を測定した．なお，m，M はそれぞれ下記の式により算出される．

$$m = \frac{1}{n}\sum_{i=1}^{n} t(i) \tag{4}$$

$$M = t(n) \tag{5}$$

なお，n を端末数（本評価では $n = 1{,}500$）とし，配信要求プログラムが配信要求を送信した時間を 0 とする．

5.2 評価結果

図 7 に評価によって得られたメッセージ配信遅延の結果を示す．図 7 は，各方式について，メッセージの受信が完了した端末数と受信完了時間（$t(i)$）の関係を示したグラフとなっている．グラフでは，各方式について測定した 100 試行の平均値を示している．グラフを見ると，RRC 状態スケジューリング方式，トリガ先行方式，FiFo 方式の順でメッセージ配信が完了していることが分かる．

以下，評価によって得られたメッセージ配信遅延（図 7）の特徴に関して，パケット通信可能な RRC 状態（RRC_CONNECTED，CELL_DCH，CELL_FACH）の端末への配信遅延と，パケット通信不可能な RRC 状態（RRC_IDLE，CELL_PCH，IDLE）の端末への配信遅延に分けて説明する．

まず，パケット通信可能な RRC 状態の端末に対するメッセージ配信は，RRC 状態スケジューリング方式が他の方式に比べて早期に完了しており，その違いが受信端末数 400 前後における受信完了時間の差となって現れている．

RRC 状態スケジューリング方式では，RRC 状態を考慮したデッドラインスケジューリングにより，メッセージを早期に受信することができるパケット通信可能な RRC 状態の端末に対する処理は先に，すぐにはメッセージを受信できないパケット通信不可能な RRC 状態の端末の処理は後に実施される．その結果，パケット通信可能な RRC 状態の端末に対するメッセージ配信処理が早期に完了し，配信遅延が小さくなる．一方で，FiFo 方式やトリガ先行方式では，パケット通信可能な RRC 状態の端末に対する処理が，すぐにはメッセージを受信できないパケット通信不可能な RRC 状態の端末に対する配信処理に待たされ，配信遅延が大きくなる．

また，パケット通信不可能な RRC 状態に対する配信遅延に関しても，RRC 状態スケジューリング方式が，他の方式に比べて優れており，図 7 のグラフ上の端末数 1,500 近辺の時間差となって現れている．RRC 状態スケジューリング方式では，トリガ送信を優先的に実施することで，パケット通信不可能な RRC 状態にある端末の状態遷移が早期に開始されるため，配信完了時間が小さくなる．トリガ先行方式においても，トリガ送信処理を優先的に行う．しかし，本来優先度を高くする必要のないパケット通信不可能な RRC 状態の端末に対するメッセージ登録処理の優先度を RRC 状態スケジューリング方式ほど下げられないため，トリガ送信処理が待たされてしまい，結果，配信遅延が大きくなっている．

表 5 メッセージ配信遅延

Table 5 Delay values of messages.

	FiFo	トリガ先行	RRC 状態スケジューリング
平均配信遅延（秒）	4.00±0.02	3.85±0.03	3.76±0.02
配信完了時間（秒）	6.57±0.05	6.13±0.06	5.71±0.05

表 5 に FiFo 方式と提案方式における平均配信遅延（m），配信完了時間（M）を示す．各方式に関し，m，M の 100 試行分の平均値と 95%信頼区間を示している．RRC 状態スケジューリング方式は，FiFo 方式に比べ，配信遅延平均値で 6%，配信完了時間で 13%改善している．

6. 関連研究

ここでは，本稿に関連する従来研究について述べる．

近年，TCP 接続維持のための Keep-Alive 信号のような定期通信による RRC 状態の遷移が，モバイル網やモバイル端末に与える影響についての研究がなされている．たとえば，Haverine ら [13] は 3G における定期通信がモバイル端末の電力消費に与える影響について分析しており，Puttone ら [12] は LTE における定期通信がモバイル網に与える RRC 状態遷移の負荷，および，モバイル端末の電力消費に関する分析を行っている．このような研究は，著

者らがKeep-Alive信号の不要な配信時接続確立方式のIPプッシュに関する研究を行う動機となっている．

Perälä ら [23] は，3GにおけるRRC状態に関する各種パラメータを分析するためのツール3G3Tを提案し，複数の通信事業者におけるRRC状態ごとの通信遅延を測定している．測定された通信遅延は，本稿において測定した実測値と近い値を示している．また，通信事業者によって同じRRC状態でも通信遅延の値は異なることを示しており，通信遅延値を変化させた場合の提案方式の評価は今後の課題である．

IPプッシュは，現在，様々なプラットフォーム上で提供されている．たとえば，AndroidにおけるGCM（Google Cloud Messaging）[1]，iOSにおけるAPNS（Apple Push Notification Service）[2]，FireFox OSにおけるSimplePush [3] などがあり，また，W3CにおいてPushAPI [24] の検討もなされている．これら様々なIPプッシュを統合するプラットフォームの研究 [25] もなされており，IPプッシュに関する需要の高さを示している．

Adya ら [26] は，IPプッシュサーバにおけるCPU負荷やメッセージ配信処理に要する時間などの評価を行っている．メッセージ配信処理の要因の分析を行い，システム内部の処理を見直すことで遅延低減が期待できるとしているが，モバイル網を介した通信による配信遅延を考慮したメッセージ配信遅延の評価は行っていない．

サーバから多数の端末にデータを配信する際のスケジューリングに関しても多数の研究がなされている．

たとえば，アプリケーションから与えられた制約条件によって，スケジューリングを行うEDF（Earliest Deadline First）やGPS（Generalized Processor Sharing）[27] などがある．EDFでは，アプリケーションから配信遅延のデッドラインがあたえられ，デッドラインに近い配信処理から実施する方式である．GPSは，各配信処理に優先度が与えられ，優先度の高い配信処理に対して多くの計算資源を割り当てる方式である．どちらも，アプリケーションの要請する配信遅延を達成するための方式であるが，同一種のアプリケーションからの配信要求が多数発生した場合には優先度に差を付けられないという課題がある．IPプッシュでは，多数のモバイル端末に同一内容の通知を一斉配信する用途にも使われ，その場合には適用できない．また，優先度をどのように与えるかという課題もある．

配信するデータサイズに基づくスケジューリング方式も提案されている [28], [29]．配信するデータサイズが小さい配信処理を優先させるSRPT（Shortest Remaining Processing Time）スケジューリングによって，配信遅延の平均を削減する効果が得られる．しかしながら，アプリケーションに対する通知を行うIPプッシュでは，1つの端末に配信するデータサイズは小さく，ほぼ同一サイズとなる．たとえば，GCMは配信データサイズに4 KBの制約を設けている．特に，多数のモバイル端末に同一内容の通知を一斉配信する場合は，配信サイズは同一となる．したがって，データサイズに基づく配信スケジューリングのIPプッシュへの適用は難しい．

7. おわりに

本稿は，モバイル網におけるRRC状態遷移による通信遅延を考慮することによって，多数のモバイル端末に対して同時にプッシュメッセージを配信する際の配信遅延を抑制する方式を新たに提案した．モバイル端末では，RRC状態遷移による遅延が大きいため，メッセージ配信時には早期にRRC状態の遷移を発生させる必要がある．提案方式では，モバイル端末のRRC状態を考慮し，配信処理スケジューリングを行うことで，RRC状態の遷移を発生させるトリガを早期に送信することで，メッセージの配信遅延を抑制する．シミュレーション評価を行った結果，メッセージ配信遅延に関して，平均配信遅延を6%，配信完了時間を13%抑制できることが示された．今後の課題としては，メッセージ配信要求が時間的に分散して到着する場合，モバイル端末のとるRRC状態の比率を変化させた場合の評価などがあげられる．また，無線基地局に同時接続可能な端末数には上限が存在する．同一無線基地局に在圏する多数のモバイル端末に同時にメッセージを配信する場合には，配信遅延が増大するだけでなく，同一無線基地局に在圏する他のモバイル端末の通信も妨げるなどの影響があるため，単位時間あたりの配信数を制限するなどの制御を検討する必要がある．

参考文献

[1] Google: Google Cloud Messaging for Android, Android Developers (online), available from ⟨http://developer.android.com/google/gcm/index.html⟩ (accessed 2013-09-13).

[2] Apple: Local and Push Notification Programming Guide: Apple Push Notification Service, iOS Developer Library (online), available from ⟨https://developer.apple.com/library/ios/documentation/NetworkingInteNetw/Conceptual/RemoteNotificationsPG/Chapters/ApplePushService.htht⟩ (accessed 2013-09-13).

[3] Mozilla: WebAPI/SimplePush, MozillaWiki (online), available from ⟨https://wiki.mozilla.org/WebAPI/SimplePush⟩ (accessed 2013-09-13).

[4] telecomHall: What is RRC and RAB?, telecomHall (online), available from ⟨http://www.telecomhall.com/what-is-rrc-and-rab.aspx⟩ (accessed 2013-09-17).

[5] 3GPP TS 25.331, Universal Terrestrial Radio Access (UTRA); Radio Resource Control (RRC); Protocol specification, V8.1.0 (2007).

[6] 3GPP TS 36.331, Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol specification, V8.17.0 (2012).

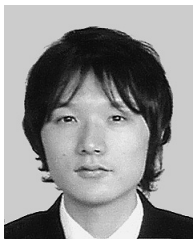
[7] Pimentel, V. and Nickerson, B.G.: Communicating and Displaying Real-Time Data with WebSocket, *Internet*

Computing, Vol.16, No.4, pp.45–53, IEEE (2012).
[8] Russell, A.: Comet: Low Latency Data for the Browser, Infrequently Noted (online), available from ⟨http://infrequently.org/2006/03/comet-low-latency-data-for-the-browseb/⟩ (accessed 2013-09-13).
[9] Fette, I. and Melnikov, A.: RFC 6455 - The WebSocket Protocol, IETF Documents (online), available from ⟨http://tools.ietf.org/html/rfc6455⟩ (accessed 2013-09-13).
[10] SiGNALS Research Group: SMARTPHONES AND A 3G NETWORK, SiGNALS Research Group (2010).
[11] GSM Association Official Document TS.18, Fast Dormancy Best Practises, Version 1.0 (2011).
[12] Puttonen, J., Virtej, E., Keskitalo, I., et al.: On LTE performance trade-off between connected and idle states with always-on type applications, *Proc. 2012 IEEE 23rd International Symposium on Personal Indoor and Mobile Radio Communications* (*PIMRC*), pp.981–985, IEEE (2012).
[13] Haverine, H., Siren, J. and Eronen, P.: Energy Consumption of Always-On Applications in WCDMA Networks, *Proc. IEEE 65th Vehicular Technology Conference*, pp.964–968, IEEE (2007).
[14] Sela, F.R., Leal, G.L. and Torregrosa, I.E.B.: Notification Server, La Cofa (online), available from ⟨http://www.lacofa.es/index.php/general/notification-server?lang=en⟩ (accessed 2013-09-13).
[15] Huang, J., Qian, F., Gerber, A., et al.: A close examination of performance and power characteristics of 4G LTE networks, *Proc. 10th International Conference on Mobile Systems, Applications, and Services* (*MobiSys '12*), pp.225–238, ACM (2012).
[16] 大久保尚人，ウメシュアニール，岩村幹夫ほか：高速・大容量・低遅延を実現する LTE の無線方式概要，NTT DOCOMO テクニカルジャーナル，Vol.19, No.1, pp.11–19 (2011).
[17] Liu, C.L. and Layland, J.W.: Scheduling algorithms for multiprogramming in a hard-real-time environment, *Journal of ACM*, Vol.20, pp.46–61 (1973).
[18] 渡辺裕明，平沢 哲，鈴木恭助ほか：NTT ドコモ様向け LTE 無線基地局装置，雑誌 FUJITSU，Vol.62, No.4, pp.388–393 (2011).
[19] DeCandia, G., Hastorun, D., Jampani, M., et al.: Dynamo: Amazon's Highly Available Key-Value Store, *Proc. 21st ACM SIGOPS Symp. Operating Systems Principles* (*SOSP '07*), pp.205–220, ACM (2007).
[20] Chang, F., Dean, J., Ghemawat, S., et al.: Bigtable: A Distributed Storage System for Structured Data, *Proc. 7th USENIX Symposium on Operating Systems Design and Implementation*, USENIX (2006).
[21] Couchbase: Couchbase Server, Document-Oriented NoSQL (online), available from ⟨http://www.couchbase.com/⟩ (accessed 2013-09-13).
[22] Linux Foundation: netem, The Linux Foundation (online), available from ⟨http://www.linuxfoundation.org/collaborate/workgroups/networking/nenet⟩ (accessed 2013-09-13).
[23] Perälä, P.H.J., Barbuzzi, A., Boggia, G., et al.: Theory and Practice of RRC State Transitions in UMTS Networks, *Proc. GLOBECOM Workshops*, pp.1–6, IEEE (2009).
[24] Sullivan, B. and Fullea, E.: Push API, W3C Working Draft (online), available from ⟨http://www.w3.org/TR/2013/WD-push-api-20130815/⟩ (accessed 2013-09-13).
[25] Lee, Y., Oh, J. and Lee, B.G.: Logical push framework for real-time SNS processing, *Proc. 2012 4th International Conference on Computational Aspects of Social Networks* (*CASoN*), pp.47–51, IEEE (2012).
[26] Adya, A., Cooper, G., Myers, D., et al.: Thialfi: A Client Notification Service for Internet-Scale Applications, *Proc. 23rd ACM Symposium on Operating Systems Principles* (*SOSP*), pp.129–142, ACM (2011).
[27] Andrews, M.: Probabilistic End-to-End Delay Bounds for Earliest Deadline First Scheduling, *Proc. 19th Annual Joint Conference of the IEEE Computer and Communications Societies* (*INFOCOM 2000*), Vol.2, pp.603–612, IEEE (2000).
[28] Rawat, M.: SWIFT: Scheduling in web servers for fast response time, *2nd IEEE International Symposium Network Computing and Applications* (*NCA 2003*), pp.51–58, IEEE (2003).
[29] Harchol-Balter, M., Schroeder, B., Bansal, N. and Agrawal, M.: Size-based scheduling to improve web performance, *ACM Trans. Computer Systems* (*TOCS*), Vol.21, No.2, pp.207–233 (2003).

推薦文

モバイル端末への配信時接続確立方式のサーバプッシュ型の配信において，端末の RRC 状態により遅延特性が異なることを利用したスケジューリングを提案している．提案方式について，シミュレーションにより，配信遅延抑制に約 13%の効果があることを示している．スケジューリングにも新規性があり，かつ，実用上に有用な知見を読者に与えることができるため，推薦論文に値する．

（マルチメディア通信と分散処理研究会主査 勝本道哲）



大西 健夫 （正会員）

2002 年東京大学理学部物理学科卒業．2008 年同大学大学院理学系研究科物理学専攻博士課程修了．博士（理学）．同年日本電気（株）入社．以来，リアルタイムコミュニケーションシステムの研究に従事．現在，日本電気（株）クラウドシステム研究所主任．2014 年情報処理学会 DPS 研究会運営委員就任．

城島 貴弘

1994 年大阪大学基礎工学部情報工学科卒業．1996 年大阪大学大学院基礎工学研究科物理系専攻情報工学分野博士前期課程修了．同年日本電気（株）入社．以来，リアルタイムコミュニケーションシステムの研究開発に従事．現在，日本電気（株）クラウドシステム研究所主任研究員．

中島 一彰 （正会員）

1993 年東京農工大学工学部電子情報工学科卒業．1998 年同大学大学院電子情報工学専攻博士後期課程修了．博士（工学）．同年日本電気（株）入社．以来，Web 会議システム等リッチコミュニケーションシステムの研究開発と標準化，DTN の研究開発に従事．現在，日本電気（株）クラウドシステム研究所主任研究員．電子情報通信学会会誌編集委員，情報処理学会 DPS 研究会運営委員を歴任．