# 広域災害時に可用な Webアプリケーションのための DTN フレームワーク

長谷川友香† 高井 峰生†† 小口 正人†

† お茶の水女子大学 〒 112–8610 東京都文京区大塚 2-1-1
†† UCLA Computer Science Department 3803 Boelter Hall, Los Angeles, CA 90095-1596, USA
E-mail: †yuka@ogl.is.ocha.ac.jp, ††mineo@cs.ucla.edu, †††oguchi@computer.org

**あらまし** 大規模災害時にはネットワークインフラが使用できなくなる地域が発生し，そのような地域では多くの人々が情報のやり取りに用いる Web アプリケーションが利用できなくなる．本研究では大規模災害時の特徴を考察し，広域的にインターネットが使用できない場合にどのような通信モデルが効率的かを検討する．広域災害時にはメッセージフェリーの存在を仮定できること，またリクエスト・レスポンス型ではない災害時特有の Web アプリケーション実装が必要であることを示し，それらの点を踏まえた Web アプリケーション実現のための DTN フレームワークを提案する．提案フレームワークを用いるための API を設計し，それを用いた Android 端末と汎用クラウドでの実機実装を紹介する．

**キーワード** 災害，Web アプリケーション，DTN，メッセージフェリー，Android

# A Proposal of a DTN Framework to Build up Disaster-tolerant Web Applications

Yuka HASEGAWA†, Mineo TAKAI††, and Masato OGUCHI†

† Ochanomizu University
2-1-1 Otsuka, Bunkyo, Tokyo, 112–8610 Japan
†† UCLA Computer Science Department 3803 Boelter Hall, Los Angeles, CA 90095-1596, USA
E-mail: †yuka@ogl.is.ocha.ac.jp, ††mineo@cs.ucla.edu, †††oguchi@computer.org

## 1. はじめに

近年，インターネットの普及に伴い，Web アプリケーションの利用が一般的となっている．家族や友人との連絡には主に LINE [1] のメッセージングサービスや Facebook [2]，Twitter [3] 等の Web アプリケーションを用いるというユーザも多いであろう．しかし緊急災害時にはネットワークインフラが断たれ，そのような方法で連絡が取れなくなる地域が生じる．そこで本研究ではそのような環境下で Web アプリケーションを利用可能にする手法について検討する．

この課題を解決するための技術として遅延耐性ネットワーク (DTN：Delay/Disruption Tolerant Networking) が広く研究されている [4]．DTN とは，継続的なネットワーク接続が不可能な状況に耐えうる通信技術を広くとらえたものである．DTN の一例として，蓄積転送通信と呼ばれる，通信路となる端末にデータの蓄積を行い，通信可能端末が出現した際にルーティングアルゴリズムに基づいてデータの複製転送を行う手法がある．この手法に関し，最適なルーティング方法が広く検討されてきた．

本研究では広域的な災害発生時に利用可能な Web アプリケーション実現のための DTN 通信モデルと，それを用いた Web アプリケーションの実装方針を提案する．なお本論文では，端末とクラウドが連携して動作するアプリケーション全般（ブラウザ上のアプリケーション，Android アプリケーション，iPhone アプリケーションなど）を指して,「Web アプリケーション」と呼ぶことにする．

## 2. 広域災害時に Web アプリケーションを利用するための手法の検討

### 2.1 想定環境

図 1 に本研究で想定する広域災害時における Web アプリケーション利用のための環境を示す．

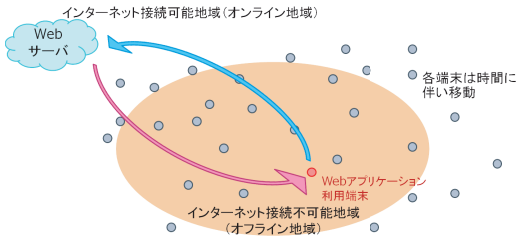


図 1　広域災害時における Web アプリケーション利用の想定環境

インターネット接続不可能地域（以下，オフライン地域）と接続可能地域（以下，オンライン地域）でのメッセージのやり取りが必須となる．オフライン地域の中は DTN 網を利用し，リクエストがオンライン地域に到達したら通常の通信網を用いてリクエストをサーバに届け，またサーバからのレスポンスを同様に通常の通信網，DTN 網を用いて端末に届ける必要がある．

### 2.2　災害時通信のための DTN ルーティング手法

DTN におけるルーティング手法は様々に研究されてきた．その中でも，Epidemic [5] や Spray and Wait [6] などの確率的転送方式が DTN 研究の対象として広く注目を集めている．確率的転送方式とは,「うまく中継してくれそうな」端末にメッセージをホップする方式である．また，災害時の利用に焦点を絞った研究として，インターネットに接続していない Android 端末間でバケツリレー方式でメッセージをルーティングする研究や，すれ違う端末間で経路表を交換することで確率的転送を行い，インターネットの接続できない地域からの Twitter の送受信を実現させる研究 [7] などがある．

図 2 に，確率的転送方式を用いて通信を実現した場合のモデルを示す．

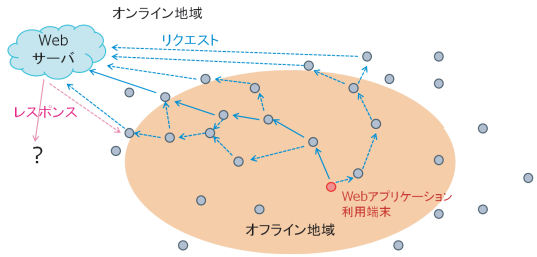


図 2　確率的転送方式を用いた通信モデル

確率的転送方式を用いた場合，まずリクエストをオフライン地域から DTN 端末をホップしてオンライン地域まで到達させる必要がある．この通信のホップ数を大まかに見積もるため，最もシンプルなモデルとして端末が移動せず，端末がオフライン地域内に均等に，互いに通信可能距離を空けて散らばっていると仮定する．オフライン地域を半径 10km，端末間の無線通信距離を半径 50m とし，オフライン地域の中心からオンライン地域までリクエストを転送することを考えると，単純計算で 200 ホップが必要となる．各端末間でのデータ転送成功確率（通信，端末が非故障状態であること，メッセージ複製が可能であること等）を 95%とすると，リクエストがオンライン地域に到達する確率は約 0.0035%となる．Epidemic などの複数の複製を許すルーティングを採用すると，オンライン地域に到達する経路は一つではないので冗長性があるとして端末間でのメッセージ転送成功確率を 99%としても，オンライン地域に到達する確率は約 13.4%である．これは最も理想的に端末が配置された場合であるので，もし端末が過疎である地域がオフライン地域内に存在すると，通信可能範囲内に端末がなくメッセージ転送が全く行えないという状況も考えられる．また，例えば東日本大震災のような大規模災害時にはオフライン地域は半径 10km どころではなく数倍から十数倍の規模になるため，さらにオンライン地域への到達可能性は低くなる．

各端末が移動すれば少ないホップ数でオンライン地域に到達すると考えられるが，確率的転送方式での移動端末は移動先が分からず，存在するかどうか自体の前提もないと仮定して設計されているので，移動端末の影響でリクエストがオンライン地域に到達するかどうかは状況依存になる．また，端末が移動すると端末間での通信可能時間は短くなるが，データは複製されてきているので転送すべきデータ量は膨大となっている可能性があり，全てのデータを転送できるとは限らない．

さらに，サーバからのレスポンスを端末まで返さなければならないが，この時まずサーバはどのノードにレスポンスを託すか選択しなければならない．端末が移動しないという環境下ではリクエストが転送されてきた経路を逆にたどればよいとしてもリクエストと同様に端末への到達確率は非常に低いものとなる．また，端末の移動を考慮に入れると，レスポンスをサーバに転送してきた端末がオフライン地域に戻るかどうか自体不確実なので，どの端末にレスポンスを転送するか決定するのは困難である．

電力消費量の問題もある．どの方式を用いるにしても共通のことであるが，通信量が多ければ多いほど端末の電力消費量は大きくなる．バッテリー切れは被災地において深刻な問題であり，できるだけ電力消費を抑えた通信をするべきである．確率的転送方式を用いると端末は通信路として用いられるため，自分の送りたいデータ以外のデータの通信にも電力消費をする必要がある．

これらの点から，メッセージ転送の目的を持った端末を仮定しないで広域なオフライン地域内を通信するのは実用的ではないと言える．

一方で，現実の災害を考えてみると，パケットを確実に中継してくれる端末を仮定するのは現実的であると思われる．具体的には，パケット中継を専門に行うトラックを被災地に巡回させられる可能性があり，また被災地をまわる物資輸送トラックや自衛隊などにパケット中継のための機器を載せるという方法も考えられる．そこで本研究ではメッセージフェリー方式のルーティングを採用することを提案する．メッセージフェリー方式は，ある経路を計画的に移動するノードをフェリーノードと呼び，これを利用して効率的なデータ転送を行う方式の DTN モデルである [8]．メッセージフェリー方式の特徴的な点として，純粋なメッセージフェリー方式では送信元ノードから受信ノードまでフェリーノードを介して 2 ホップでメッセージが届くこ

とを想定しているため，ルーティングアルゴリズムといったものは必要がない．また，複製を生成する必要がないのでノードのストレージや消費電力などを大幅に削減できるという利点がある．

本研究で提案するメッセージフェリー方式の通信モデルを図 3 に示す．フェリーノードはオンライン地域とオフライン地域の間を行き来するものとする．

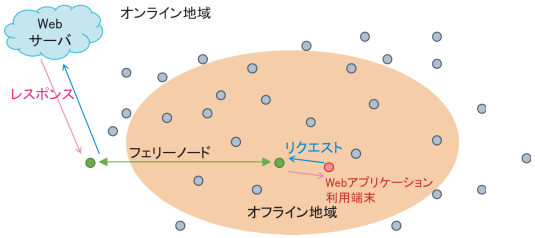


図 3　メッセージフェリー方式を用いた通信モデル

メッセージフェリー方式を用いればデータ転送は 1 ホップで済むのでオンライン地域へのリクエスト到達確率は飛躍的に高まる．また，レスポンスを返す際もサーバはフェリーノードにレスポンスを託せばよい．提案方式では端末がメッセージを送信するのはフェリーノードに限定されるため，各端末はメッセージの中継をする必要がなく，電力消費は大幅に抑えられる．

広域災害時においては，被災地域における通信支援という目的を持ったノードとしてのメッセージフェリーの導入が信頼性の高い通信を確立するために有効であると考える．

### 2.3　DTN フレームワークを用いた Web アプリケーション

広域災害時に利用可能な Web アプリケーションを実現するもっとも単純な方法は，既存の DTN フレームワーク上に Web アプリケーションを構築することである．DTN フレームワークが提供しているアプリケーション層の API を利用すれば，既存の Web アプリケーションにいっさいの変更を加えることなく動作させることが可能になる．具体的には，DTN2 [9] や IBR-DTN [10] などの DTN フレームワークが，そのような汎用的なアプリケーション層の API を提供している．また，これらの DTN フレームワーク上で Web サーバを動作させる研究も複数なされており，DTN において送受信されるバンドルの中に HTTP リクエストやレスポンスを包含させることで，あたかも通常であるかのような HTTP 通信を DTN 環境上で実現している [11] [12].

確かに，これら既存の DTN フレームワークを用いて既存の Web アプリケーションをそのまま DTN 環境で動作させることができれば，それが最善であると考えられる．しかし，広域災害時には通信帯域が非常に限定されており，必要な通信のみを最大限の効率で実現しなければならないことをふまえると，上記の手法は実用性に乏しいのではないかと考えられる．これは以下の二つの理由による．

第一の理由は，通信データ量の制限である．平常時には通信データ量が比較的大きくても問題にならない場合が多いが，災害時には通信帯域が非常に限定されており，さらに DTN 環境ではパケットを中継する端末に蓄積できるデータ量には限界があることをふまえると，必要最小限のデータのみを通信することが要求される．例えば，Web 広告やデザインのためのデータなどは通信されるべきではなく，本当にユーザにとって必要なデータのみが通信されるべきである．この事実は，災害時と平常時では Web アプリケーションの通信内容を大きく変更する必要があることを示唆している．つまり，平常時の Web アプリケーションをそのまま既存の DTN フレームワーク上で動作させるだけでは，災害時に要求される無駄のない通信は実現できないと考えられる．

第二の理由は，Web アプリケーションの基本となっているリクエスト・レスポンス型の通信モデルにある．Twitter や Facebook などのアプリケーションでは，クライアントがリクエストを送信し，それに対してサーバがレスポンスを返すことによって通信が成立する．

クライアントはレスポンスが一定時間返ってこないときにはリクエストかレスポンスが失われたと判断して，リクエストの再送を行う．しかし災害時には遅延の大きさが全く予想できず，最適な再送のタイミングを知ることはできない．最適なタイミングが分からないため，盲目的に多くのリクエストがクライアントから送られると予想される．すると，リクエストのみならずそれに対応してレスポンスも大量に発行されてしまう．しかし，実際にサーバやクライアントにとって必要なのは一つのリクエストと一つのレスポンスのみであり，その他の再送されたデータは限りある DTN ネットワーク資源を無駄に消費してしまうことになる．このように，遅延が大きく通信に信頼性のない環境下でリクエスト・レスポンス型の通信を行ってしまうと，「本当に必要なデータのみをネットワークに流す」通信を実現するのが難しくなる．つまり，災害時に Web アプリケーションを動作させる場合，平常時と同一のリクエスト・レスポンス型の通信モデルで動作させるのは効率的ではないと考えられる．

以上のような理由から，平常時の Web アプリケーションを，そのまま既存の DTN フレームワーク上で動作させるだけでは効率的な災害時通信は実現できないと考えられる．そこで本研究では，リクエストとレスポンスを分離した災害時向けの新たな通信プロトコルを提案し，それに基づく Web アプリケーションの設計法を提案する．

## 3.　提 案 手 法

広域災害時に実用的な Web アプリケーションを設計するための本研究での提案手法は次の二点にまとめられる．

- 災害時のルーティング手法としては，広域災害時には確率的転送方式では実用性のある通信を実現できないだろうことを踏まえて，メッセージ転送の目的を持ったフェリーノードの存在を仮定し，端末・フェリーノード間で通信を行うメッセージフェリー方式を採用する．
- Web アプリケーションの構築手法としては，既存の DTN フレームワーク上にそのまま Web アプリケーションを載せるだけでは効率的な災害時通信を実現できないことを踏まえて，リクエストとレスポンスを分離した新たな通信プロトコルに基づく DTN フレームワークを提案する．

具体的には，まず第 4 章で提案手法の有効性をオフライン地域における通信データ量の観点から検証する．次に第 5 章で提案手法の有効性についてシミュレータを用いて検証を行い，第 6 章で Web アプリケーションに提供する災害時用 API を提案し，第 7 章でリクエストとレスポンスを分離した通信プロトコルを設計し，第 8 章でこの DTN フレームワークを Android 端末に実装して検証する．

## 4. 提案手法の有効性の検証

本章では，フェリーノードと端末間のデータ通信をシミュレーションし，提案手法の有効性を検証する．本研究では，ネットワークシミュレータ Scenargie [13] を用いて，メッセージ転送を目的とした端末を導入した場合の各種ルーティング方式の有用性の検討を行った．Epidemic を比較対象として取り上げた．

### 4.1 シミュレーションシナリオ

今回，フェリーノードがオフライン地域でデータを回収・配達するシナリオを想定した．フェリーノードはオフライン地域ユーザへのメッセージを運んできており，それを配達した後，オンライン地域へのメッセージを各ユーザから回収する．

30m × 30m の範囲内に複数台の端末がランダムに配置されており，各端末は移動しないシナリオを用いた．これは，被災者が避難所に避難してとどまっている状況を想定している．この範囲内ではどの端末間でも無線通信が可能である．このシナリオの概要図を図 4 に示す．

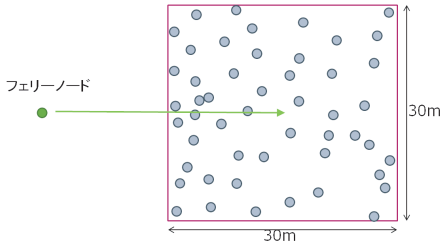


図 4　シミュレーションシナリオ概要

ルーティングアルゴリズムは以下の二種類を用いた．

（1） メッセージフェリーを宛先とした Direct Delivery（提案方式）

（2） Epidemic

提案方式を模擬するために，「メッセージフェリーを宛先とした Direct Delivery」を用いた．Direct Delivery とは，宛先ノードのみにメッセージを転送するルーティング手法である．普通ならば宛先ノードが DTN 網内になければこの手法は用いることができないが，今回はフェリーノードがメッセージをオンライン地域に届けてくれると仮定しているので，フェリーノードを宛先としてこのルーティング手法を用いた．

### 4.2 実験 1:端末台数の変化

実験 1 として，端末の台数を 10〜50 に変化させた実験を行った．シミュレーション時間は 1000 秒間で，10 秒経過時点で全ての端末のアプリケーションが 1KB のメッセージを送信する．120 秒経過時点でフェリーノードが通信可能範囲内に到着する．フェリーノードは到着すると同時に各端末への 10KB のメッセージの送信を開始する．以下に実験 1 の詳細を示す．

表 1　実験 1 の詳細

| | |
|---|---|
| シミュレーション時間 | 1000 秒間 |
| 端末数 | 10〜50 |
| フェリーノード数 | 1 |
| 端末送信メッセージサイズ | 1KB |
| 端末受信メッセージサイズ | 10KB |
| トランスポートプロトコル | TCP (NewReno) |
| 無線 LAN 規格 | IEEE802.11g |

本シナリオにおいて例えば端末数が 10 の場合，DTN のデータ送信単位であるバンドルは，端末からフェリーノード方向に 10，フェリーノードから端末方向に 10 送信されるはずなので，全体で 20 のバンドルが宛先に到達すればメッセージ転送が完了したこととなる．他の端末数の場合も同様に，(端末数)*2 のバンドルが宛先に到達すれば通信完了とみなせる．図 5 に宛先に到達したバンドル数の推移を示す．

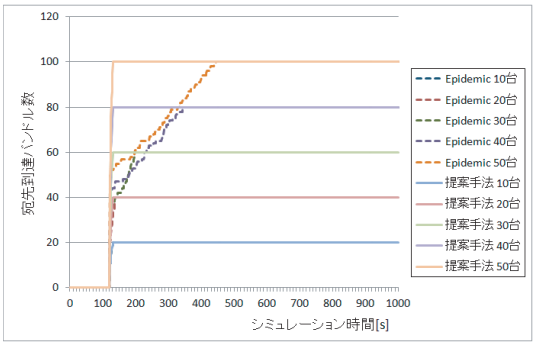


図 5　実験 1:宛先に到達したバンドル数の推移

端末数が 50 のとき，フェリーノードが通信可能範囲に到着する 120 秒から提案手法は 11 秒後，Epidemic は 324 秒後にメッセージ転送が完了した．提案手法は台数が増加してもメッセージ転送時間が大きく変化しないのに対し，Epidemic は台数によってメッセージ転送時間に大幅な増加が見られた．Epidemic ではあらかじめ避難所内端末間で共有されているバンドル分はフェリーノードが到着後すぐに宛先であるフェリーノードに到達するが，フェリーノードから端末方向のバンドル送信に長い時間がかかっていることが観察される．これは，Epidemic の特性である大きな通信量が影響するものと考えられる．端末数 50 の場合，フェリーノードが到着して最初に接続した端末に対し，提案方式ではその端末宛ての 1 バンドルのみ送信するのに対して，Epidemic ではその端末宛てのバンドルだけでなく，他の端末宛てのバンドルも合わせて 50 バンドル送信することになる．このように避難所内端末は系内の全てのバンドルを受信しようとするため，通信量が非常に大きくなる．提案方式においてバンドルを宛先に到達させるためには，フェリーノードはそれぞれの端末と個別に接続する必要があり接続のオーバヘッドはかかるが，それを補っても余りあるほど Epidemic の通信量や負荷が大きかったと推察される．

次に，ネットワーク層での各ルーティング手法の総受信データ量の推移を図 6 に示す．これは，端末とフェリーノードの受

信データ量を全て合計したものである．ネットワーク層での通

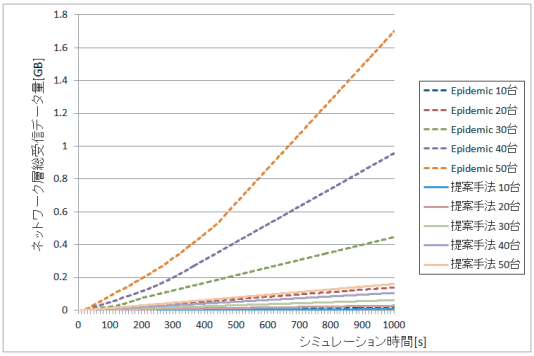


図 6　実験 1:ネットワーク層での総受信データ量の推移

信データ量には実データの通信量や DTN の制御パケットの通信量，上位層の TCP 制御パケットの通信量などが含まれる．そのため，バンドルの送信が終了しても制御パケット分の通信が引き続き行われている．ネットワーク層受信データ量は端末数 50 の場合，Epdemic は 1000 秒時点で 1.7GB，提案方式は 0.16GB となり，Epidemic の方が通信量が約 10 倍大きくなった．Epidemic の方が端末数増加による通信量への影響が大きく，提案手法で端末数を 10 から 50 に増加させたときに総受信データ量が 2.2 倍になるのに対し，Epidemic では 8.5 倍となった．

以上のことから，提案手法の方がメッセージ転送完了にかかる時間が短く，かつ端末台数の増加に影響を受けにくいモデルであることが示された．

### 4.3　実験 2:端末受信メッセージの大きさの変化

実験 2 として，端末の台数は 50 に固定し，各端末の受信するメッセージの大きさを 10〜100KB に変化させた実験を行った．シミュレーション時間は 5000 秒間で, 10 秒経過時点で全ての端末のアプリケーションが実験 1 と同様に 1KB のメッセージを送信する．120 秒経過時点でフェリーノードが通信可能範囲内に到着する．フェリーノードは到着すると同時に各端末へメッセージの送信を開始するが，この際のメッセージの大きさを変化させる．以下に実験 2 の詳細を示す．

表 2　実験 1 の詳細

| | |
|---|---|
| シミュレーション時間 | 5000 秒間 |
| 端末数 | 50 |
| フェリーノード数 | 1 |
| 端末送信メッセージサイズ | 1KB |
| 端末受信メッセージサイズ | 10KB〜100KB |
| トランスポートプロトコル | TCP (NewReno) |
| 無線 LAN 規格 | IEEE802.11g |

図 7 に宛先に到達したバンドル数の推移を示す．提案手法ではメッセージサイズが変化してもメッセージ転送完了にかかる時間にはほとんど変化が見られなかった．それに対し，Epidemic ではメッセージサイズがメッセージ転送時間に大きな影響を与え，100KB の場合はフェリーノード到着時点から 4880 秒経過しても転送が完了していない．

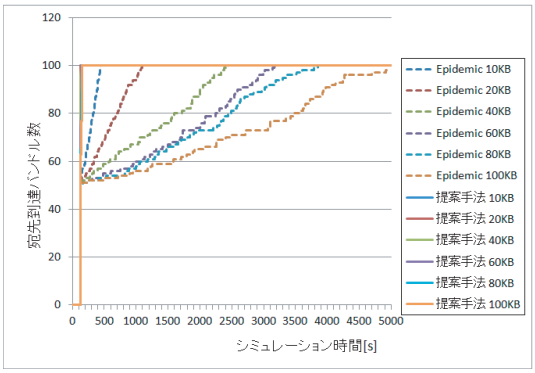


図 7　実験 2:宛先に到達したバンドル数の推移

次に，ネットワーク層での各ルーティング手法の総受信データ量の推移を図 8 に示す．これは，端末とフェリーノードの受信データ量を全て合計したものである．

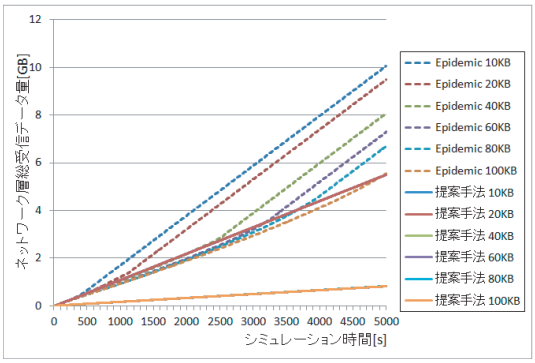


図 8　実験 2:ネットワーク層での総受信データ量の推移

どのメッセージサイズにおいても Epidemic の方が提案手法よりも受信データ量が大きくなるという結果となった．どちらのルーティング手法でもメッセージサイズが小さい方が受信データ量は大きくなっている．DTN 通信において制御パケットはブロードキャスト，データはユニキャストで送信されるが，データ送信が終わった後には制御パケットのみ送信され，全ての端末がそれを受信するため総受信データ量としては大きくなるためだと考えられる．

以上のことから，特にメッセージ転送時間について提案手法の方がメッセージの大きさに影響を受けにくいモデルであることが分かる．

### 4.4　シミュレーション結果に対する考察

想定環境である，フェリーノードが避難所とオンライン地域をつなぐという状況で，フェリーノードと端末が一対一通信を行うモデルである提案方式は有効に機能するということが示された．Epidemic を用いると通信完了に非常に長い時間がかかり，さらに通信量が大きいため限りあるバッテリー残量を大幅に消費してしまうという考察が得られた．また，メッセージ転送時間やデータ通信量において，提案手法は Epidemic と比較して端末数の変化や転送メッセージの大きさに影響を受けにくいモデルであることが示された．よって，移動端末の存在を仮定できる場合には確率的転送方式を用いるのではなく，提案手法のようにフェリーノードとのみ通信するルーティングを用いて効率的な通信を行うのが望ましいと考えられる．

しかし，今回の想定環境と異なりフェリーノードと通信できない端末が存在する場合には確率的転送方式を用いたほうが到達率が上がる可能性もあるため，さらに様々な環境を想定した調査が必要であると考えられる．

## 5. Web アプリケーションに提供する災害時用 API

本章では，メッセージフェリーを用いたモデルの通信を実現する，災害時に利用可能な Web アプリケーションを実装するときに用いる API の詳細を説明する．

### 5.1 API

Web アプリケーション実装の際は，端末側とクラウド側に分けて機能を実装することになる．提案 DTN フレームワークを用いることによってフェリーノードの存在を意識する必要はない．

端末側，クラウド側のアプリケーションに提供される API はそれぞれ次の通りである．

- 端末側
  - sendMessage(appID, message)
  - responseCallback(response)
- クラウド側
  - messageCallback(appID, message)
  - requestResponseCallback(appID)

図 9 に提供される API とアプリケーションの関係を示す．

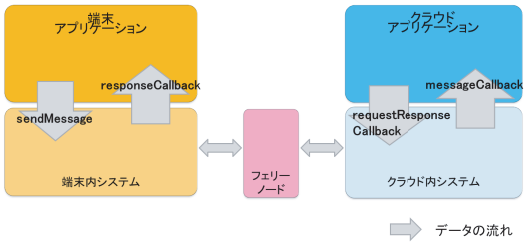


図 9　提供される API とアプリケーションの関係

それぞれの API について以下で詳しく説明する．

**sendMessage(appID, message)**

端末側で送信したい Message がある場合に，アプリケーション ID とメッセージを引数として取る sendMessage メソッドを呼び出す．アプリケーション ID とは，そのアプリケーションでのユーザ ID のことであり，例えば Twitter ならば Twitter アカウント ID，Facebook ならば Facebook アカウント ID などである．

**responseCallback(response)**

端末側で，フェリーノードから Response を受け取った時に呼び出される．アプリケーションは Response があった時の処理をこのメソッドの中で実装しておく．

**messageCallback(appID, message)**

クラウド側で，端末からの Message が到着した時に呼び出される．アプリケーション ID によってどのユーザが送った Message かを識別することができる．アプリケーションはその Message をデータベースに蓄積するなど，アプリケーション依存の処理を行う．

**requestResponseCallback(appID)**

クラウド側で，フェリーノードに各ユーザへの Response を託す際に呼び出される．アプリケーションは指定されたユーザへ送るべき Response を生成する．ここで，アプリケーションは緊急災害時ということを意識し，必要最小限のデータを送るように実装しておくべきである．

### 5.2 API 使用例

ここで一例として Twitter を取り上げ，どのように API を用いることが出来るのか紹介する．災害時に端末上の Twitter アプリケーションでユーザがツイートを行ったり，あるユーザへのダイレクトメッセージを送信したりすると，アプリケーションは Twitter アカウント ID とそれらのメッセージを引数として sendMessage メソッドを呼び出す．ここで，アプリケーションは災害時を意識して実装されるべきであるという方針から，あるツイートをお気に入りに登録する操作やフォローリクエストを送る操作などの場合にはこのメソッドを呼び出さないという様に実装することもできる．そうすればそれらの操作はただ単にキャンセルされ，通信資源やストレージなどを消費することがなくなる．

端末側で responseCallback が呼び出された際には到着したレスポンスをアプリケーション上で表示するよう実装する．ユーザにいち早く情報を伝えられるよう，ポップアップでレスポンスを表示する実装にすることもできる．

クラウド側で messageCallback が呼び出された際には到着したメッセージに応じて処理を行う．この場合はツイートをタイムラインに投稿することや，あるユーザにダイレクトメッセージを送ることなど，平常時に操作する場合と同じような処理を行えばよい．

クラウド側で requestResponseCallback が呼び出された際には，指定された Twitter アカウント ID のユーザに送信するデータを選択する．この時，そのユーザ宛てのダイレクトメッセージやそのユーザのツイートに対する返信などを中心に送信データを選択すべきであると考えられる．通信データ量を抑制すべきであるので，平常時のようにタイムラインに表示されるツイート全てを送信すべきではない．

## 6. 災害時用通信プロトコル

### 6.1 提案フレームワーク構成

提案フレームワークのシステム構成を図 10 に示す．フレームワークは端末，フェリーノード，クラウドに分かれている．フレームワークはそれぞれの中に内部ストレージを持つ．端末とフェリーノードは Wi-Fi で，フェリーノードとクラウドはインターネットを通じて接続することを想定している．

### 6.2 提案フレームワークにおけるデータフロー

提案フレームワークで使用するパケットの種類を表 3 に示し，提案フレームワークのクラウドデータフローの概要図を図 11 に示す．

提案フレームワークにおけるデータフローについて図 11 を用いて説明する．まず，フェリーノードがインターネットに接続不可能な地域（オフライン地域）から接続可能な地域（オ

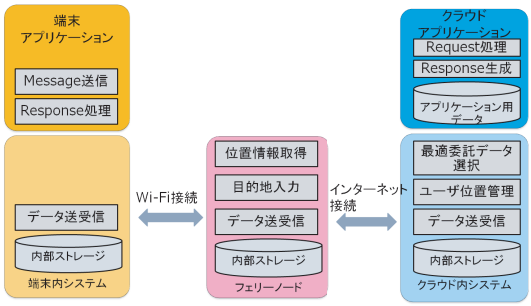


図 10　提案フレームワークのシステム構成図

表 3　提案フレームワークで使用するパケットの種類

| 名前 | 使用箇所 | 説明 |
|---|---|---|
| Message | 端末→フェリー<br>フェリー→クラウド | 端末からサーバへ送信するデータを格納 |
| RequestResponse | 端末→フェリー | 端末がサーバからの Response を要求していることをフェリーに伝達 |
| NotifyLocation | フェリー→クラウド | フェリーと通信をした際の端末ごとの位置情報を格納 |
| NotifyDestination | フェリー→クラウド | フェリーの行き先を指定 |
| Response | クラウド→フェリー<br>フェリー→端末 | サーバから端末へ送信するデータを格納 |

ンライン地域）へ移動する場合の流れを説明する．端末アプリケーションはサーバに対してデータの送信が必要になると，送信したいデータとアプリケーション ID を引数として取る sendMessage メソッドを呼び出す (1)．アプリケーション ID とは，そのアプリケーションでのユーザ ID のことであり，例えば Twitter ならば Twitter アカウント ID，Facebook ならば Facebook アカウント ID などである．すると端末内システムはそのデータとアプリケーション ID を Message パケットとして内部ストレージに蓄積する．端末とフェリーノードが通信可能距離に入ると端末とフェリーノードは Wi-Fi 接続を行い，端末内システムはストレージに蓄積しておいた Message パケットをフェリーノードに転送する (2)．加えて，自端末アプリケーション宛ての Response パケットがないかフェリーノードに問い合わせるための RequestResponse パケットを送信する (3)．フェリーノードは RequestResponse パケットを受信すると，該当 Response パケット有無の調査と現在地取得の二つの処理を行う．一つ目の処理である該当 Response パケット有無の調査では，RequestResponse パケットに含まれるアプリケーション ID を元に内部ストレージを調べ，その ID 宛ての Response パケットがあった場合は端末に転送する．二つ目の処理である現在地取得は GPS を用いて行う．現在地情報と RequestResponse を送ってきたアプリケーション ID を NotifyLocation パケットとして保存しておく．

フェリーノードがオンライン地域に到達すると，内部ストレージに蓄積しておいた Message パケットと NotifyLocation パケットをクラウドに送信する (4)(5)．クラウド内システムは

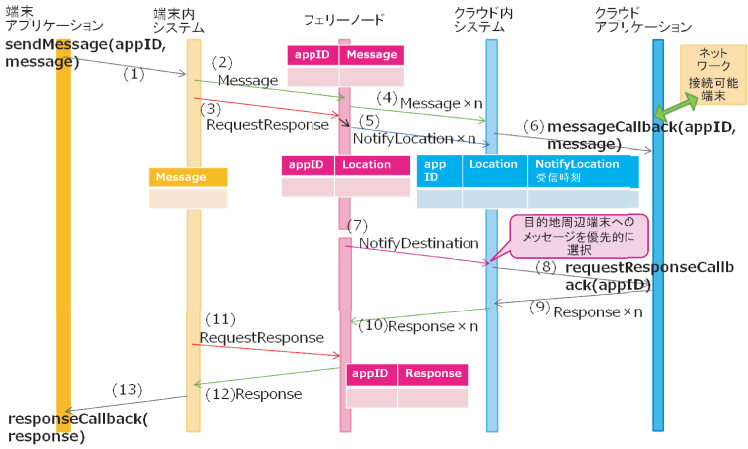


図 11　提案フレームワークのデータフロー

NotifyLocation パケットを受信すると，アプリケーション ID と位置情報，受信日時を保存する．Message パケットはアプリケーション ID とデータの形でクラウドアプリケーションに渡される (6)．この時，クラウドアプリケーションでは Message パケットのアプリケーション ID とデータを元に messageCallback が呼び出される．これでリクエストフェーズの流れが完了する．

次に，フェリーノードがオンライン地域からオフライン地域へ移動する場合の流れを説明する．まずフェリーノードはどこに行く予定かをクラウド内システムに NotifyDestination パケットを用いて通知する (7)．クラウド内システムは各端末アプリケーションの位置情報を管理しているので，フェリーノードの目的地の周辺に居る可能性の高いユーザを選択する．選択されたユーザ宛てのレスポンスをクラウドアプリケーションに対して要求する (8)．この時，クラウドアプリケーションでは requestResponseCallback が呼び出される．アプリケーションは通知されたアプリケーション ID 宛てのデータをクラウド内システムに返し (9)，クラウド内システムはそれらを Response パケットにして，フェリーノードに委託する (10)．フェリーノードはそれらの Response パケットを受信し，内部ストレージに蓄積する．

フェリーノードがオフライン地域に移動し，端末と Wi-Fi 接続すると，端末は RequestResponse パケットを送信する (11)．先ほど説明したようにフェリーノードは該当 Response パケット有無の調査と現在地取得の二つの処理を行う．端末内システムがフェリーノードから Response パケットを受信すると (12)，端末アプリケーションに Response パケット内のデータが渡される (13)．この時，端末アプリケーションでは responseCallback が呼び出される．

本研究では「リクエストとレスポンスを分離した通信プロトコル」を提案しているが，それはフェリーノードが各端末の現在位置を取得し，クラウド内システムが端末の位置を把握することで実現されている．クラウド内システムがフェリーノードの目的地に合わせて周辺にありそうな端末を選択するので，クラウドアプリケーションは指定された端末への Response を生成するだけでよい．

## 7. 実機実装

今回，提案フレームワークを使用したアプリケーションの例として伝言アプリケーションを実装した．このアプリケーションの機能は非常にシンプルであり，以下の通りである．

- 伝言を作成し，あらかじめ登録しておいたメンバーの中で宛先を指定（全員宛ても可能）して送信する
- 誰かの伝言を受信すると画面に表示する

図 12 に伝言アプリケーションの全体図を示す．端末とフェリーノードには Android 端末を，クラウドには汎用クラウドである Google App Engine を用いている．フレームワークの仕様上，実際にはフェリーノードには画面出力は必要ないが，今回は接続端末などの確認のために実装されている．

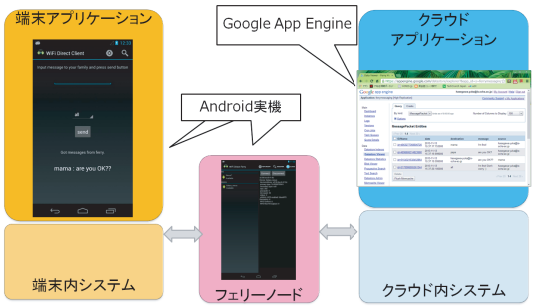


図 12　伝言アプリケーションの全体図

図 13 に伝言アプリケーションの Android 端末画面を示す．伝言の入力，宛先選択，送信ボタンまた受信した伝言の表示の機能がある．

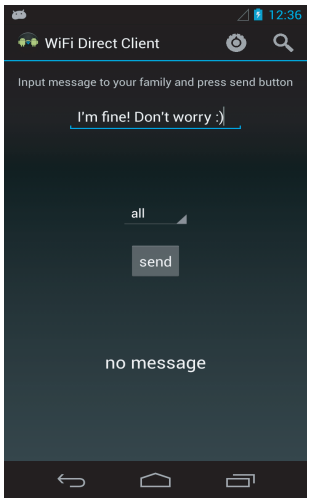


図 13　伝言アプリケーションの Android 端末画面

### 7.1 Android 端末側の実装

Android 端末では，ユーザが伝言を入力し，宛先を選択して「送信ボタン」を押すとアプリケーション ID また伝言と宛先をメッセージとして sendMessage メソッドを呼び出す．responseCallback メソッドでは，受信したレスポンスから伝言とその送信元ユーザを取得し，画面に表示する機能を実装した．

### 7.2 クラウド側の実装

messageCallback メソッドでは，受け取ったメッセージから伝言と宛先，送信元アプリケーション ID をデータベースに保存するよう実装した．requestResponseCallback メソッドでは，指定されたアプリケーション ID 宛ての伝言をレスポンスとして返すよう実装した．

## 8. まとめと今後の課題

大規模災害時にネットワークインフラが使用できなくなる地域において Web アプリケーションを実装するための DTN 通信モデルと Web アプリケーションの実装方針を提案した．大規模災害時にはメッセージフェリーの存在を仮定できることや従来のリクエスト・レスポンス型とは異なるモデルのアプリケーション実装が必要であることを論じた．アプリケーションに対して API を提供する DTN フレームワークを設計し，Android 端末と汎用クラウドを用いて実機実装を行った．

今後の課題として，今回提案した方式のメッセージフェリー型通信が適応できる環境をシミュレーションにより確認したい．また，メッセージフェリーとして機能するものにも，例えばトラックやヘリコプターなど様々な種類のものが考えられるので，それらの速度などの特性を考慮して委託する Response の複製数や種類を変更するなどの検討を行っていきたい．

## 文　献


[1] LINE, http://line.me/ja/
[2] Facebook, http://www.facebook.com/
[3] Twitter, http://twitter.com/
[4] Kevin Fall. "A delay-tolerant network architecture for challenged internets." Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, pp.27-34, ACM, 2003.
[5] Amin Vahdat and David Becker. "Epidemic routing for partially connected ad hoc networks." Technical Report CS-200006, pp.18, Duke University, 2000.
[6] Spyropoulos, Thrasyvoulos, Konstantinos Psounis, and Cauligi S. Raghavendra. "Efficient routing in intermittently connected mobile networks: the multiple-copy case." Networking, IEEE/ACM Transactions on 16.1, pp.77-90, 2008.
[7] 小山由, et al. "大規模災害時の安否確認システムと広域無線網利用可能エリアへの DTN に基づいたメッセージ中継法." 電子情報通信学会技術研究報告. MoMuC, モバイルマルチメディア通信 112.44, pp.171-177, 2012.
[8] Wenrui Zhao, Mostafa Ammar and Ellen Zegura. "A message ferrying approach for data delivery in sparse mobile ad hoc networks." Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing, pp.187-198, ACM, 2004.
[9] DTN2, http://www.dtnrg.org
[10] IBR-DTN, http://www.ibr.cs.tu-bs.de/projects/ibr-dtn/
[11] Jorg Ott and Dirk Kutscher. "Bundling the Web: HTTP over DTN." In Proc. Workshop on Networking in Public Transport, 2006.
[12] Zoebir Bong, Peng Boon Sng and Chai Kiat Yeo. "Web Access over Delay Tolerant Networks." International Journal of Future Computer and Communication, Vol.1, No.2, pp.202-205, 2012.
[13] Scenargie, http://www.spacetime-eng.com