大規模災害時のインターネット非接続時における XMPP に基づく情報配信手段の検討

YU　HUI†　大和田泰伯††　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
†† 情報通信研究機構　〒 980-0812 宮城県仙台市青葉区片平 2-1-3
E-mail: †{yu.hui,oguchi}@is.ocha.ac.jp, ††yowada@nict.go.jp

あらまし　近年インターネットは社会や経済の重要な役割を担当している．人間もスマートフォンで連絡することに慣れている．インターネットや通信端末機を組み合わせる通信システムは現代人の生活になくてはならないものになっている．しかし，地震等の災害によって，通信インフラが破壊される可能性があり，通信範囲が断片的になる状況があり得る．つまり，平常時に利用できる通信システムも利用不能になる．これに対し，非常時に必要な情報を伝送するという情報配信の大きな需給があり，相応の手段が必要である．本論文はインターネット非接続の条件下，断片化のネットワークを利用し，XMPP に基づく情報配信手段の検討である．

キーワード　XMPP，インターネット非接続，災害，Openfire，配信

1. はじめに

現代社会はネットワーク化がますます進んでいる．情報の交換でも，経済の発展でも，ネットワークを介して人間同士が緊密に繋がっている．つまり，インターネットは社会や経済の重要な役割を担当し，現代人の生活になくてはならないものになっている．

これに対し，地震等の自然災害により，通信インフラが断片化する可能性があり，伝送できなくなる可能性がある．例えば，支援者は被災者に余震や二次災害の予測情報及び飲用水や食料の配給情報を伝達する必要がある．又，被災者は避難所情報や道路等の生活インフラに関する情報を取得する必要がある．

以上の理由から，災害時の情報配信や通信が必要であることが分かる．

2. 平常時の通信システム

LINE や WeChat などの通信手段は，平時によく使用されている．

これらのシステムは，図 1 のように，故障回避や性能向上のためにクラウドシステム内でサーバ同士がクラスタを構成し冗長化され，全体としてサービスを提供している．

しかし，これらのシステムはインターネットに接続された時のみしか利用できないため，インターネットにつながらない場合には，たとえ同じ LAN 内につながっているユーザ同士であっても，これらの通信手段による通信は行えない．

災害時にはアクセスネットワークに障害が起きるなどでインターネットにつながらない場合には，ほとんどの平時に利用している通信アプリケーションは利用できなくなる．

よって，本論文では，この問題を注目し，非常時においてもサーバ間，ユーザ間で情報の交換ができる通信システムの構築について，検討を行う．

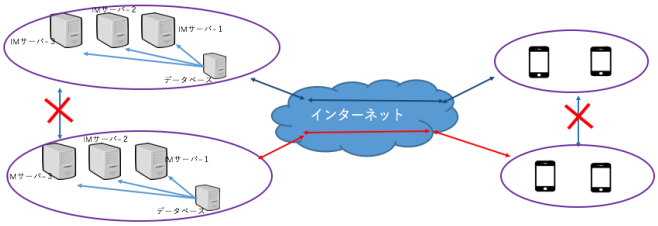

図 1　平常時の通信システム

3. XMPP プロトコルと Openfire サーバ

3.1 XMPP

XMPP とは，現在広く使用されているインスタントメッセージ (Instant message) システムの核心プロトコルの一つである．ユーザは JID というアカウントを取得し，ユーザ間，ユーザ・サーバ間で通信できる．

JID とは，「任意の名前@登録したサーバのドメイン名」という唯一のアカウントである．特に，名前が同じであっても，異なるサーバに登録したら，異なるユーザと推定する．つまり，JID を利用すれば，ユーザ名の重複は問題にはならず，異なるサーバ間での通信が可能になる．

XMPP でメッセージを伝達するため，全てのメッセージが以下のように XMLstanza を標準にして，パッケージされる．

1　Message stanze によって，ユーザ間，ユーザ・サーバ間で，メッセージを発信する．図 2 に示す．

2　Presence stanze によって，複数の受け手に状態を知らせる．（マルチキャスト的な一方向の通知）図 3 に示す．

3　IQ stanze によって，ユーザ間，ユーザ・サーバ間で，要求と応答をする．id は要求と応答の番号を取る．図 4 に示す．

```
<message
      to="test@192.168.111.174” id="tJhQ8-143"
      type="chat"
      from="admin@192.168.111.174”>
      <thread>tERD40</thread>
      <paused xmlns="http://jabber.org/protocol/chatstates"/>
</message>
```

図 2　Message stanze

```
<presence
      to='a@192.168.111.174'
      id='3NIsU-11'
      type='subscribed'>
</presence>
```

図 3　Presence stanze

```
<iq
    type="get"
    id="368-6"
    from="192.168.111.175”
    to="test@192.168.111.175/AndroidClient">
    <ping xmlns="urn:xmpp:ping”/>
</iq>
```

```
<iq
    to='192.168.111.175'
    id='368-6'
    type='result'>
</iq>
```

図 4　IQ stanze

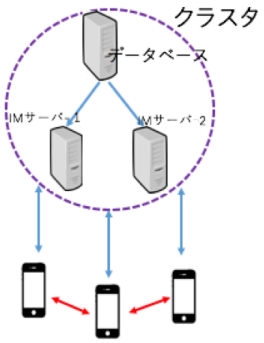

図 5　クラスタリング

3.2　Openfire

Openfire とは，XMPP に基づくインスタントメッセージシステムのサーバの一つである．使用方法が簡単で，基本的な通信システムの機能が付いており，拡張もできる．

Apache MINA を使用することにより，メッセージの受け取り，タイプの判断と処理が分かれ，開発者がソケット (socket) と NIO（Java new I/O）の複雑な操作を考えず，処理の機能やロジックに集中できる．

Openfire での，サーバ同士の 2 種類の接続方法はクラスタリングとフェデレーションである．

3.2.1　クラスタリング

クラスタリングとは、サーバ同士で同じデータベースを使用し，2 台以上の複数のサーバで構成し、全体が 1 つのシステムとしてサービスを提供する接続方法であり，現在広く使われている．これを図 5 に示す．

全てのサーバで同じサービスを提供できるようにし，ロードバランサを使い，リクエストを分散させることで，すべてのサーバで処理を分担する．また，もしあるサーバは故障を起きたら，障害の発生したサーバをシステムから切り離すことができるので，最大限にサービスの品質が保証できる．

よって，クラスタリングを使用すれば，ユーザは任意のサーバに登録しても，同じユーザ情報が取られる．

しかし，外部のアクセスネットワークに障害が起きるなどでインターネットにつながらない場合には，クラスタリングが応対不能になる可能性がある．

3.2.2　フェデレーション

Openfire のフェデレーションとは，別々のデータベースを利用し，認証機能を持つ独立のサーバ間で情報共有化する接続方法の一つである．

これを図 6 に示す．

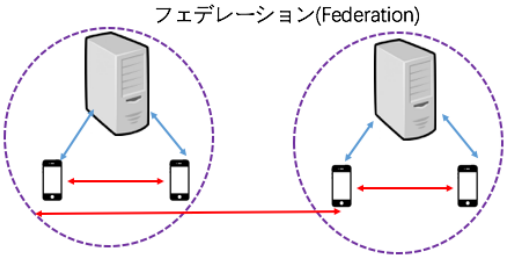

図 6　フェデレーション

信頼関係を締結したサーバ間で，信用するドメイン名を持つ発信者と受信者間の通信だけが伝送できる．

ユーザは登録したサーバだけにサインインできるため，異なるサーバにもう一度登録する必要がある．換言すれば，サーバとユーザアカウントが一つずつ対応する関係である．それでも，サーバ間でフェデレーションをすれば，ユーザ間の通信ができる．

クラスタリングは災害時の環境に適さない可能性があると考えるため，本論文はフェデレーションを使用する．

4.　サーバ間のネットワークトポロジ

ネットワークトポロジとは，通信ネットワークの接続形態を意味する．

代表的なネットワークトポロジには,「スター型」,「バス型」,「メッシュ型」などがある．各トポロジがそれぞれの利点や欠点を持っている．本論文は「スター型」と「メッシュ型」トポロジについて説明する．

4.1　スター型トポロジ

スター型とは，現在広く使用されているトポロジの一つであり，図 7 に示すように，メインサーバから放射状のようにサブサーバを接続される形態である．サーバ同士を繋げ，さらに台数を増やすことが可能で，拡張しやすい．

しかし，メインサーバが壊れた場合，すべての通信システムが途絶える可能性もある．

4.2　メッシュ型トポロジ

メッシュ型とは，図 8 のように，各サーバを直接繋ぐ接続形

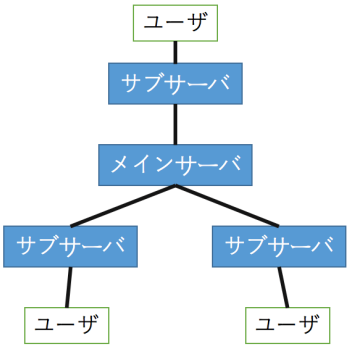

図 7　スター型トポロジ

態であることがわかる．このトポロジ型で，各サーバと相互接続する方式で，万一障害が発生しても，他のサーバのリンクを介して，通信を継続できるという利点を持っている．しかし，このトポロジ型は，ある程度の冗長性がある．それでも，災害時に適す可能があると考えている．

本論文はスター型やメッシュ型トポロジを利用して，通信システムを構築する実験を行う．

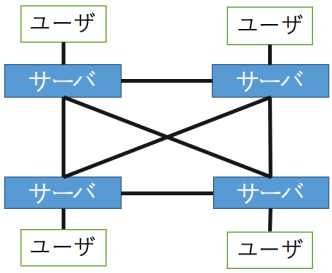

図 8　メッシュ型トポロジ

5.　提 案 手 法

5.1　システムの概要

本論文は以下二つの方法を提案する．

簡単にサーバを増やすし，ローカルに置くサーバの負荷を減らすめた，スター型トポロジを利用し，システムを構築する．概要を図 9 に示す．

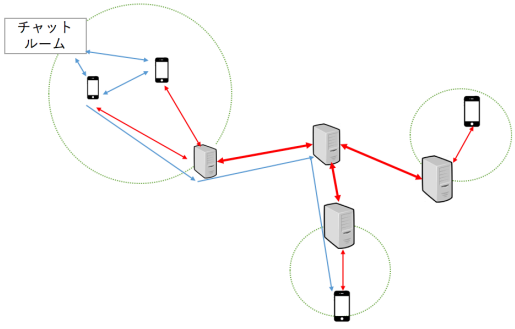

図 9　スター型トポロジでの提案システム

スター型トポロジで，メインサーバを防災本部に置き，各サブサーバは各避難所に置くと仮定している．各サブサーバはメインサーバと直接繋ぐ接続状態のようにする．そして，各サーバはフェデレーションを行なっている．

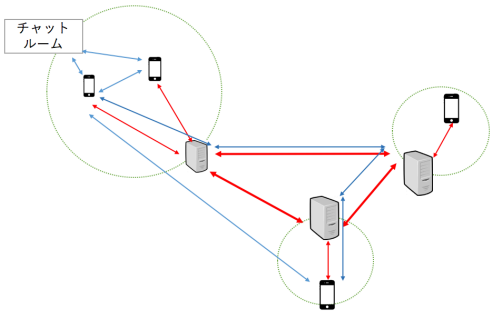

図 10　メッシュ型トポロジでの提案システム

しかし，災害時の複雑な通信環境を考えると，不安定な通信ネットを対応するため，もう一つのメッシュ型を利用するシステムを提案する．概要を図 10 に示す．

メッシュ型トポロジで，各サーバは各避難所に置き，フェデレーションで相互接続している．

本論文には，被災者が簡単に受信できるようにため，利用者を発信者と受信者にわける．

チャットルームは情報源を担当し，情報の発信と受信を行う場所にする．

ルームにはどのような情報が並ぶかという説明を付けているため，ルームに入らなくても情報の種類を確認できる．発信者は複数のチャットルームを作ることができ，このルームの中で情報を発信する．受信者は，対応するルームを選んで，情報を受け取る．

災害時，発信者間の協力が必要な場合，知り合いではない発信者同士があるので，対応手段が必要と考える．この問題を対し，本論文はユーザリストの同期を行う．

5.2　実 験 環 境

各（サブ）サーバに登録した発信者間で，通信できるようにするため，図 11 のように，VMware と Virtualbox を利用し，ローカル環境での実験環境を構築する．

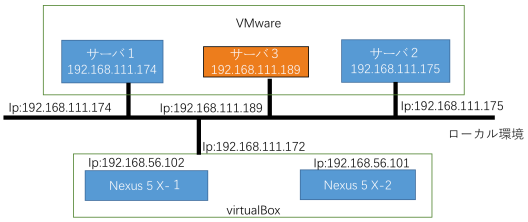

図 11　実 験 環 境

VMware で，サーバ 1，2 と 3 の OS は CentOS6.5 であり，全て openfire4.1.5 を実装している．データベースと IP アドレスを別々にするため，サーバ同士は独立のサーバと推定できる．サーバ名はそれぞれの IP アドレスにする．そして，サーバ同士はフェデレーションの方法で接続している．

Virtualbox で，アンドロイドのシミュレータの Nexus5x 二台をユーザ端末機にし，アプリケーションを実装している．サーバ同士とユーザ同士は同じローカル環境に置かれている．

詳しい OS とバーションの仕様は図 12 に示す．

	OS	バーション	データベース	ローカルのIP アドレス
サーバ 1	CentOS6.5	Openfire4.1.5	MySQL	192.168.111.174
サーバ3	CentOS6.5	Openfire4.1.5	MySQL	192.168.111.189
サーバ 2	CentOS6.5	Openfire4.1.5	MySQL	192.168.111.175
Nexus5x- 1	アンドロイド	7.1.1	-	192.168.111.172
Nexus5x- 2	アンドロイド	7.1.1	-	192.168.111.172

図 12　サーバとユーザの仕様

以下においては端末 1 と 2 がサブサーバ 1 と 2 にログインし，異なるサーバに登録したユーザ間の通信実験を行う．

5.3　スター型トポロジでの通信の流れ

発信者同士での通信ができるため、まずはサブサーバ同士をユーザリストに同期させる。次には、取得したリストから情報の交換を行う。詳しい流れを図 13 に示す．

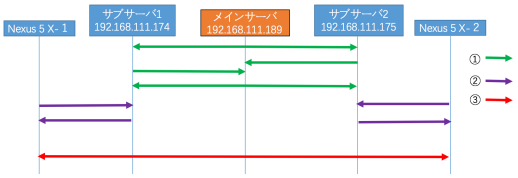

図 13　スター型トポロジでの通信の流れ

1)．メインサーバが全てのサブサーバにユーザリストを依頼し，サブサーバは自分が持っているユーザリストを送る．メインサーバは全てのユーザリストをまとめる．次に、メインサーバから、全てのサブサーバにユーザリストの同期を行う．これから，全てのサーバが全体のユーザリストを持っている．

2)．ユーザはそれぞれのサーバにログインし，それぞれの JID と同期したユーザリストを取得する．Nexus5x-1 の JID は a@192.168.111.174，Nexus5x- 2 の JID は a@192.168.111.175 である．

3)．ユーザは取得したリストから相手に自由に発信できるようになる．

5.4　メッシュ型トポロジでの通信の流れ

メッシュ形でのシステムで，各サーバは相直結しているサーバに自分を持っているユーザリストを送り，相手のユーザリストを受け取る通信網を構築する．詳しい流れを図 14 に示す。

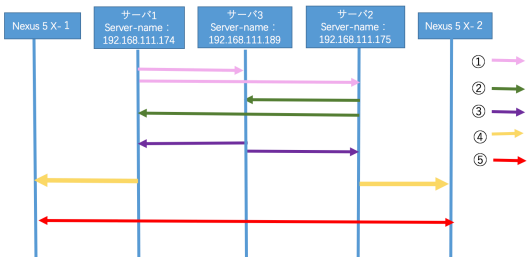

図 14　メッシュ型トポロジでの通信の流れ

1)．1 番目，2 番目と 3 番目でのは同じ働きである．各サーバは相直結しているサーバに自分を持っているユーザリストを送る．これと同時に，各サーバは受け取ったリストによって，持つユーザリストの更新を行う．

2)．4 番目とは，ユーザがそれぞれのサーバにログインし，サーバからそれぞれの JID とユーザリストを取得することである．

3)．5 番目で，ユーザは取得したリストから相手に自由に発信できるようになる．

5.5　スター型の通信結果

スター型での同期や通信実験は以下のように示す．Nexus5x-1 での取得したユーザリストは図 15 のように，192.168.111.189 はメーンサーバになり，サブサーバ 192.168.111.174 と 192.168.111.175 のユーザ情報を並べている．

図 15　スター型の同期したユーザリスト

スター型の通信ログを図 16 に示す。

図 16　スター型の通信ログ

5.6　メッシュ型の通信結果

メッシュ型での同期や通信実験は以下のように示す．Nexus5x-1 での取得したユーザリストは図 17 のように，サーバ 192.168.111.189,192.168.111.174 と 192.168.111.175 のユーザ情報を並べている．

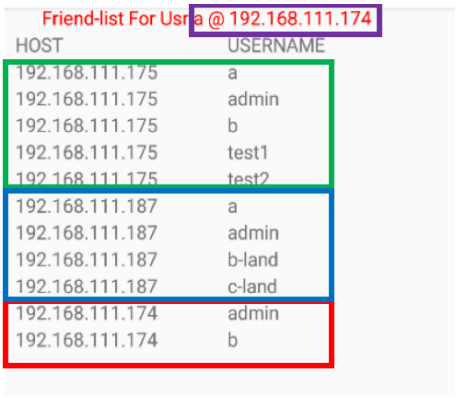

図 17　メッシュ型の同期したユーザリスト

メッシュ型の通信ログを図 18 に示す。

図 18　メッシュ型の通信ログ

5.7　発信者・受信者間の通信手段

本論文は利用者を発信者と受信者に分けている．個人によって，「有用な情報」の意味が異なるので，簡単に配信するため，チャットルームを作り，説明を付け，リストアップする．そこで，受信者は情報の選択権が与えられ，自分自身に適す複数以上の情報類が選ばれる．チャットルームを図 19 に示す．

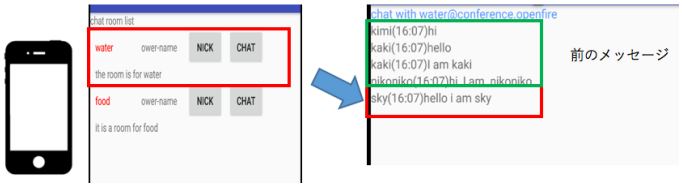

図 19　発信者・受信者間の通信手段

そして，重要な情報を受け取り損なわないために，前の１００枚のメッセージが自動的にサーバに保存され，任意の時点で入るユーザでも受け取ることができる．

6.　まとめと今後の課題

本論文では，まず，非常時の通信に向けた提案システムを説明した．次に，XMPP と Openfire サーバを利用し，情報共有化する実験を行った．

スター型やメッシュ型トポロジでの 2 種類接続の通信システムを構築した．次に，情提案手法の実行性を確認するために，実験を行なった．結果はサーバ同士で同期されるユーザリストによって，異なる (サブ) サーバにログインして，知り合いではない発信者同士でも通信できるようにした．

今後は XMPP プロトコルを拡張し，ルーム機能とサーバ同士の通信を改善する．現段階では，発信者・受信者間の通信のチャットルームは普通のルームと同じであるが，今後は災害時の配信環境により，適する発信手段を実現する．

7.　謝　　辞

本研究で UCLA の高井峰生先生から有用なアドバイスをいただきました．また，本研究は一部，JST CREST JPMJCR1503 の支援を受けたものです．ここに感謝の意を表します．

文　　献

[1] XMPP :
https://xmpp.org/， 2017 年 10 月参照.
[2] android-Developers:
https://www.igniterealtime.org/projects/openfire/ ， 2017 年 10 月参照.
[3] VirtualBox に CentOS6.5 をインストール ～ 初期設定
http://damepg.hatenablog.com/entry/2014/08/06/090217
， 2017 年 10 月参照.
[4] 仮想マシン IP 構成の概要
http://blog.51cto.com/igbugs/1679539
， 2017 年 10 月参照.
[5] Xmpp presentation https://www.slideshare.net/javaranger123/xmpp-presentation
，2017 年 10 月参照.
[6] 総務省，大規模災害時におけるインターネットの有効活用事例解説集
pp10-15 , 2017 年 12 月参照.
[7] cross-domain communication using an XMPP chat guard
，Raymond Haakseth ,Nils Agne Nordbotten，
Oddvar Brnstad， yvind Jonsson， Bengt Kristiansen
，2017 年 12 月参照.
[8] A Lightweight XMPP Publish/Subscribe Scheme for Resource-Constrained IoT Devices, HENG WANG, (Member, IEEE), DAIJIN XIONG, PING WANG, AND YUQIANG LIU ,2017 年 11 月参照
[9] Android-消息推送刻篇
http://www.imooc.com/learn/358
，2017 年 9 月参照.
[10] Android 消息推送刻
http://www.imooc.com/learn/223
，第二章 Socket 的使用和 Mina 框架解，2017 年 9 月参照.
[11] Connect to Genymotion Android emulator remotely
http://www.sarpex.co.uk/index.php/2016/10/02/connect-genymotion-emulator-remotely/
，2017 年 10 月参照.
[12] Linux 下 iptables 禁止端口和放端口
http://blog.csdn.net/zht666/article/details/17505789，2017 年 10 月参照.
[13] クラスタサービスとは？
https://enterprisezine.jp/iti/detail/12
，2017 年 10 月参照.
[14] クラスタサービスとは？
https://enterprisezine.jp/iti/detail/12
，2017 年 10 月参照.
[15] StanzaHandler
https://github.com/igniterealtime/Openfire/blob/master/src/java/org/jivesoftware/openfire/net/StanzaHandler.java
，017 年 10 月参照.
[16] StanzaHandler
https://github.com/igniterealtime/Openfire/blob/master/src/java/org/jivesoftware/openfire/net/StanzaHandler.java
，2017 年 10 月参照.
[17] XMPPService の一例
https://www.javatips.net/api/woc-master/WoChat-master/xmpp/src/main/java/com/wuxiaolong/ xmpp/XMPPService.java
，2017 年 10 月参照.
[18] https://enterprisezine.jp/iti/detail/12
2018 年 2 月参照.
[19] http://itdoc.hitachi.co.jp/manuals/3020/30203R3231/PCOP0228.HTM ，2018 年 2 月参照.
[20] https://www.hpe.com/jp/ja/japan/icewall/federation/overview.html， 2018 年 2 月参照.
[21] http://www.infraexpert.com/study/networking1.html
2018 年 2 月参照.