平成25年度 修士論文

オンラインストレージを用いた
分散SNSの設計と実装

電気通信大学 大学院情報システム学研究科

情報システム基盤学専攻

1253022 若井 英之

指導教員　鶴岡 行雄　教授
多田 好克　教授
新谷 隆彦　准教授

提出日　平成26年1月27日

目 次

図 目 次

表 目 次

第 1 章

はじめに

近年，SNS(Social Networking Service) の普及によってコミュニケーションが活性化されている．Facebook[1] は 2012 年の利用者数が約 9 億人，日本でも 900 万人弱が利用しており [2]，社会に広く浸透している．SNS は情報拡散の速さから，メーカーから消費者へ商品情報の伝達への利用や，災害時において災害情報の共有に利用されている．SNS はユーザとユーザのつながりを管理し，つながっている相手に向けて情報の発信を行ってコミュニケーションを行う．例えば Facebook は「フレンド」のつながりでユーザ同士を結び付ける．

SNS の特徴として少数のサービスにユーザが集中するというものがある．ユーザが多いほどサービスの価値や知名度が高まり，新規ユーザの獲得性や情報発信者の利用意欲が大きくなる．反対に小さいサービスの価値は下がってゆきユーザは離れていくので，少数の大規模サービスにユーザが集中する．よってユーザデータを SNS 事業者が一極所持しているため，ユーザにとって必要以上の個人情報が一つの事業者に集まってしまう．SNS 事業者はユーザのつながりやデータを全て知ることができる．さらにボットにより SNS サーバへアクセスする研究では 250GB ものユーザデータを取得している [3]．このようにユーザデータを一極所持することはプライバシーの問題がある．

この問題を解決するため，コンテンツやソーシャルグラフ等のユーザデータを複数サーバに分散させる分散 SNS が提案されている [4][5][6]．たとえば VIS[5] では，ユーザが管理するサーバにユーザデータが保存されており，図 1.1 のようにサーバ

同士がデータを補完し合って SNS を構成する．また diaspora*[6] では，ユーザがデータの置き場所を公開されているサーバから選ぶことができ，またユーザのサーバを他ユーザのデータの置き場所として公開することもできる．VIS ではユーザがサーバを管理するため，サーバを用意する手間や保守を継続しなければならない問題がある．また diaspora* は他人にサーバ機能を貸すことができるが，ボランティアベースで運用されているため，VIS と同様にサービスの持続性の保証ができない．

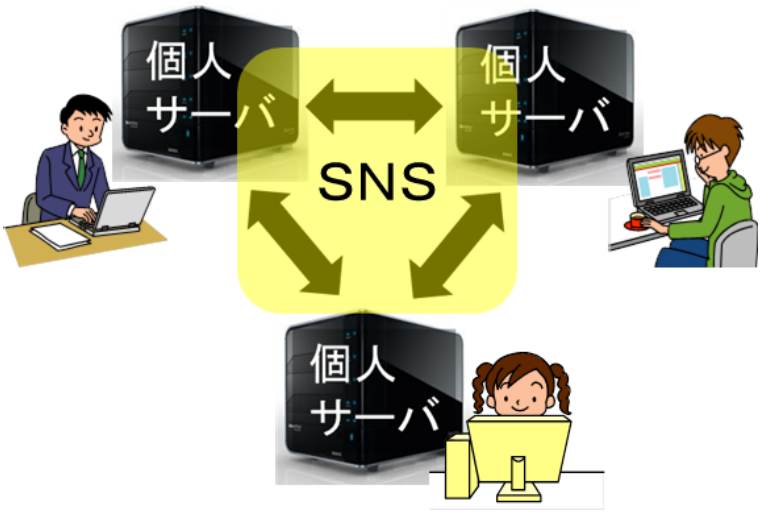


図 1.1: 分散 SNS

ところで，インターネット上のサーバを介して複数端末間のデータを同期・共有するオンラインストレージサービスが利用されている．オンラインストレージサービスはユーザの端末に専用アプリケーションをインストールすることで，サーバと端末間のフォルダやファイルを自動的に同期することができる．オンラインストレージサービスは，ユーザが利用するストレージ容量に応じて事業者にコストを支払っているため持続性がある．オンラインストレージを用いて端末間でファイル共有することは，SNS でコンテンツを共有することと似ており，オンライン

ストレージで共有したファイルを SNS コンテンツとして扱うことで持続性のある分散 SNS を構成することができる．なおオンラインストレージサービスを用いた分散 SNS として Frenzy[7] がある． Frenzy では，SNS でのコミュニケーション前にユーザ間で同期・共有設定が必要なため，やりとりできる範囲が固定的になる．よって，人の間のコミュニケーションを広げるという SNS の利点が失われている．

本研究では上記の問題を解決したオンラインストレージを用いた分散 SNS を提案する．ユーザがフレンドのフレンドの情報を得るために，SNS として扱うフォルダの構造を定義した．これによりオンラインストレージサービスで自動的にフレンドのフレンドのコンテンツを共有できる仕組みを実現した．また，SNS からフレンドを追加できるようにするため，フォルダにフレンド関係のセマンティクスを定義し，それに基づき共有設定を行うようオンラインストレージの動作を拡張した．提案システムによりプライバシー，持続性，拡張性を備えた SNS が実現できる．

本論文の構成を以下に示す．第 2 章では現在利用されている SNS や SNS を構成する既存の技術を紹介する．第 3 章では提案システムの設計について示す．そして提案システムによって新しいユーザの情報を得る手法とフレンド関係を追加する手法ついて述べる．第 4 章では提案システムの実装方法と実装したシステムについて紹介する．第 5 章では提案システムの評価について述べ，第 6 章では考察を述べる．第 7 章では本研究の結論を述べる．

第 2 章

関連研究

2.1 SNSに関する既存技術

SNSは人と人のアカウントのつながりを管理し，つながっている相手に向けて情報の発信を行ってコミュニケーションする．たとえばFacebookでは「フレンド」，mixi[8]は「マイミクシィ」というつながりでユーザ同士を結び付ける．Facebookやmixiの画面には，フレンドやマイミクシィが投稿した日記や写真などのコンテンツが表示される．またTwitter[9]ではつながりが「フォロー」と呼ばれ，ユーザはフォローしているユーザのコンテンツを閲覧でき，情報源を選ぶことができる．

SNSにおいて特徴的なのは，フレンドのフレンドの情報を得ることができるところである．たとえば図2.1のようにユーザA，B，Cがいたとする．AとB，BとCがフレンドであり両矢印で示されている．このときBが書いた日記をAが閲覧し，Bの日記に対してAがコメントをすると，CがAのコメントを見ることができる．Cから見てAはBのフレンドであり，フレンドのフレンドである．

Facebookやmixi，Twitterにおいてコンテンツを投稿するためには，ユーザの端末が投稿時にオンライン状態である必要がある．つまり，オフライン環境ではコンテンツを投稿できない．またコンテンツは事業者のサーバ上にあるため，オフライン環境では自分やフレンド等のコンテンツを取得できない．以上のようにオフライン環境では利用ができず，システム上の問題がある．

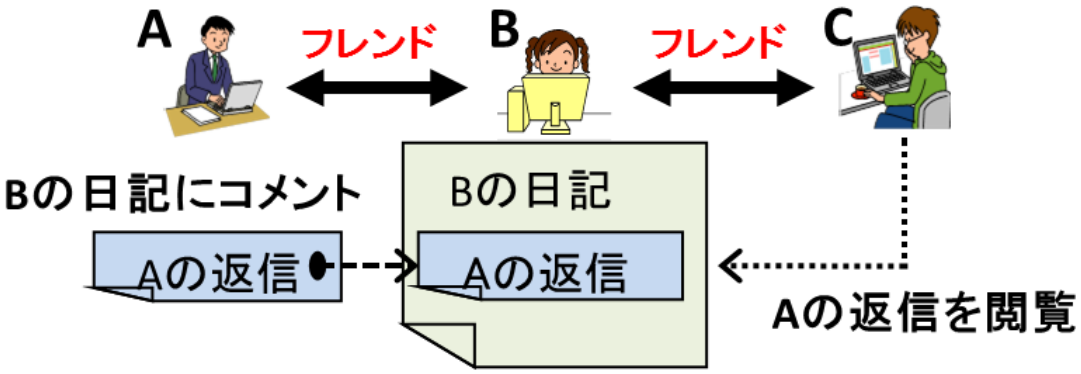


図 2.1: フレンドのフレンドの情報を得る

2.1.1 SNS の暗号化

FlyByNight[10] は Facebook のデータを暗号化するアプリケーションである．JavaScript で記述されており，ユーザのブラウザにダウンロードして使用する．クライアント側の JavaScript で秘密鍵/公開鍵のペアを生成する．ユーザが投稿や情報の更新を追加したとき，フレンドへ完全なアクセス権を付与する．JavaScript は、次にフレンドの公開鍵とフレンドの ID を使ってデータを暗号化し，データは Facebook を介して送信され，フレンドは秘密鍵を使用してデータを復号する．FlyByNight の問題点は画像の暗号化を行っていないため，テキストのコミュニケーションに限られる点である．さらに暗号化してコミュニケーションできる相手は既存のフレンドに限られ，新たなフレンドを追加できない点がある．また Facebook の規約 [16] によると，Facebook はユーザに実名と実際の情報の使用を求めている．さらに Facebook の利用規約 [15] には，Facebook 規約に違反するユーザへのサービス提供を停止すると示されている．よって暗号文をやりとりするユーザは Facebook からサービスを停止される可能性がある．

2.1.2 分散 SNS

VIS[5] は各ユーザが自身で管理するサーバを持つ．このサーバにユーザのソーシャルグラフやデータが保存されており，ユーザのサーバ同士がお互いのデータを補完し合って SNS を構成する．

Affelio[11] はユーザが自身の Web サーバにインストールして利用する形式の SNS ソフトで，個人は無料で利用でき，商用利用にはライセンス料が必要である．Affelio サーバ上で動く CGI のプログラムが他の Affelio サーバと連携して SNS を実現する．コンテンツのアクセス制御機能について,「友達」や「友達の友達」のほか、Affelio ユーザ以外の「ゲスト」を対象に制御できる．なお，Affelio は 2006 年以降活動を停止している．

diaspora*[6] では，VIS と同様に複数のサーバが互いのデータを補完し合って分散 SNS を構成する．さらに，ユーザが SNS へ参加する際に，自身のデータを置くサーバを公開されているサーバから選ぶことができる．また自身のサーバを他人のデータの置き場所として公開することができる．

Friendica[12] は、Friendica サーバ以外の SNS へのコネクタ機能を持つ SNS ソフトで，diaspora* のサーバともフォローやコメントができるようになっている．Friendica はスケーラビリティが小さく，数百人程度でサーバが一杯になってしまう．

これらの SNS における問題点は，ユーザがサーバを管理する必要があることである．diaspora* において他人のサーバを利用する際にはサーバを管理する手間が不要となる．しかし，サーバ公開はボランティアであるため，持続性の保証がない．

2.2 オンラインストレージサービス

オンラインストレージサービスはユーザにサーバマシンのディスクスペースを貸し出すサービスである．インターネット接続があればどこからでもデータの読み書きができる．例として Dropbox[13] がある．Dropbox では，ユーザが Dropbox に

ID とパスワードを入力してアカウントを登録し，Dropbox は登録したアカウントごとのデータを保存する．またユーザの端末に同期アプリケーションをインストールすることで，ユーザ端末のローカルフォルダと Dropbox サーバ上のフォルダを自動で同期することができる．たとえば，図 2.2 ではフォルダ”X”とフォルダ”Y”が Dropbox サーバと端末 T_A で同期されていることを示している．さらに，Dropbox 上でフォルダの共有設定をすることで，複数端末間でフォルダを共有することができる．たとえば図 2.2 で，ユーザ A がフォルダ”Y”をユーザ B と共有することを設定した場合，T_A と T_B 間でフォルダ”Y”が共有される．

同期アプリケーションがインストールされた端末はローカルにフォルダを持っているため，オフライン環境でもフォルダの内容を操作することができる．

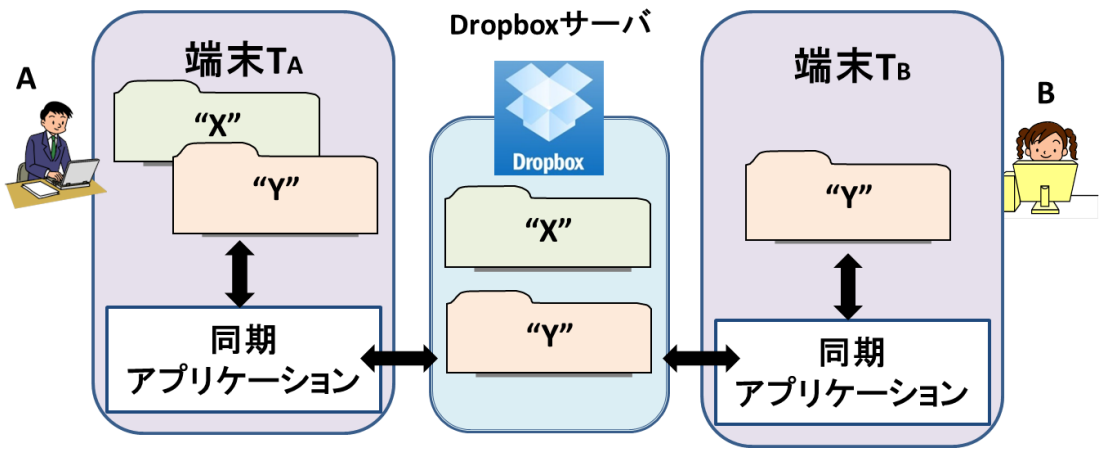


図 2.2: Dropbox の利用例

2.3 Frenzy

Frenzy[7] はDropboxを用いた分散SNSである．Dropbox内で共有したフォルダを利用することで，グループでコミュニケーションを行うことができる．なお，グループを作成して他ユーザとコミュニケーションするために，Dropboxと Frenzyで事前に設定が必要である．

まずDropbox上の設定を図2.3を用いて説明する．たとえば図2.4のようにユーザA，B，Cがグループ”G1”に属し，ユーザCとDがグループ”G2”に属しているものとする．BはDropbox上でフォルダ”G1”を作成し，”G1”をA，Cと共有するよう設定する．同様にDはDropbox上でフォルダ”G2”を作成し，”G2”をCと共有するよう設定する．

次にFrenzyの設定について説明する．ユーザA，B，C，Dはそれぞれ Frenzy上のGUIで，Frenzyで使用するフォルダをDropboxで共有されているフォルダの中から選択して設定する．具体的にはAは”G1”を，Bは”G1”を，Cは”G1”と”G2”を，Dは”G2”をそれぞれ選択して設定する．Aが”G1”を Frenzyで使用するフォルダとして選択したので，Frenzyは”G1”以下にフォルダ”jzuxmt8d42sk”を作成する．”jzuxmt8d42sk”はAの IDであり，Frenzyによりランダムな文字列として生成される．以下，簡単のため”jzuxmt8d42sk”をID_Aと表す．さらに，フォルダID_A以下にフォルダ”feeds”，ファイル”info.json”，ファイル”avatar.png”を生成する．”info.json”にはAのユーザ名が格納され，”avatar.png”にはAのアイコン画像が格納される．ユーザ名とアイコン画像は，Frenzyにより端末T_Aの個人設定から抽出される．同様に”G1”以下にはフォルダID_B，ID_Cが作成され，”G2”以下にはID_C，ID_Dが作成される．また，それぞれにフォルダ”feeds”，ファイル”info.json”，ファイル”avatar.png”が生成される．

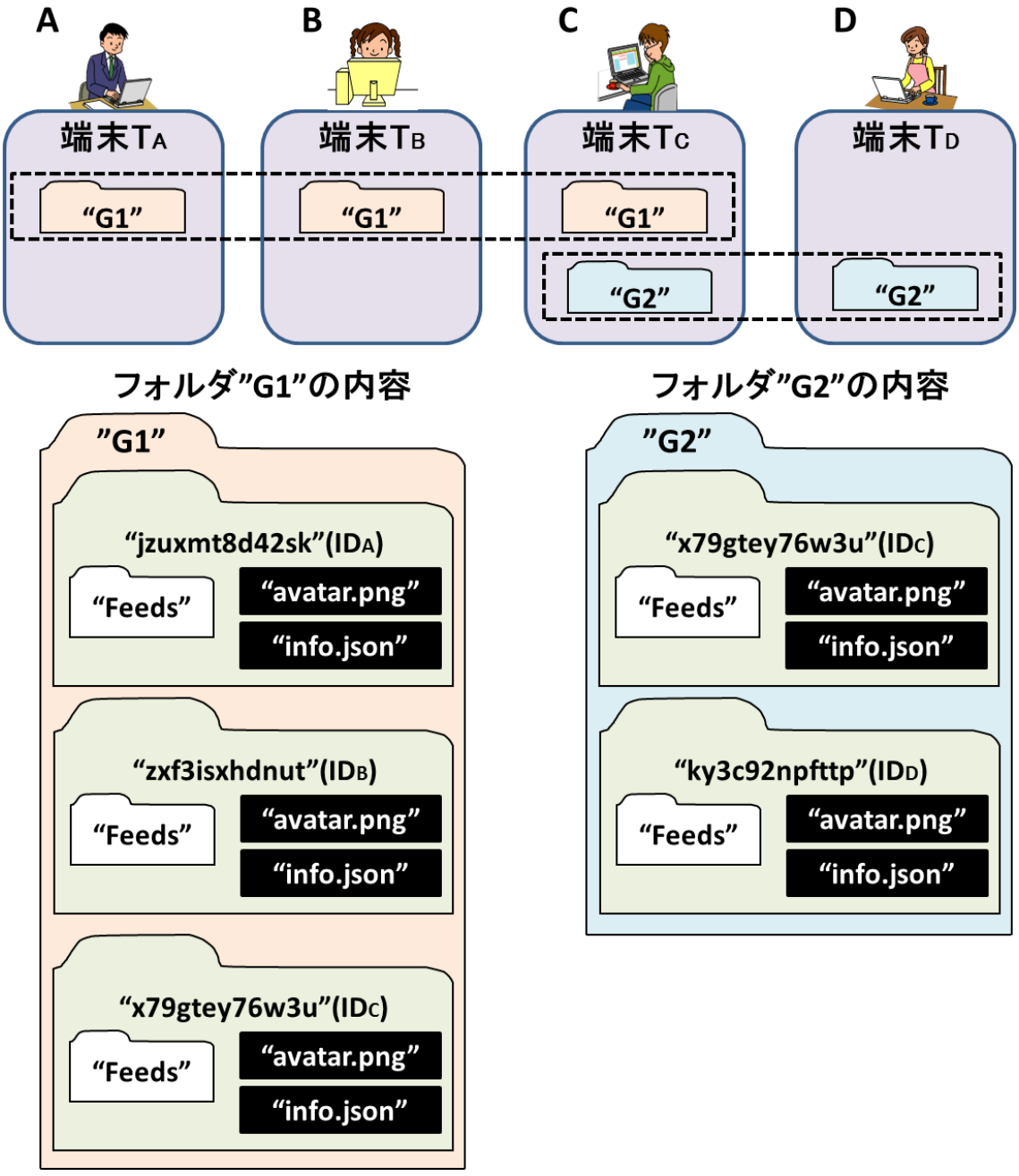


図 2.3: Frenzy の利用例

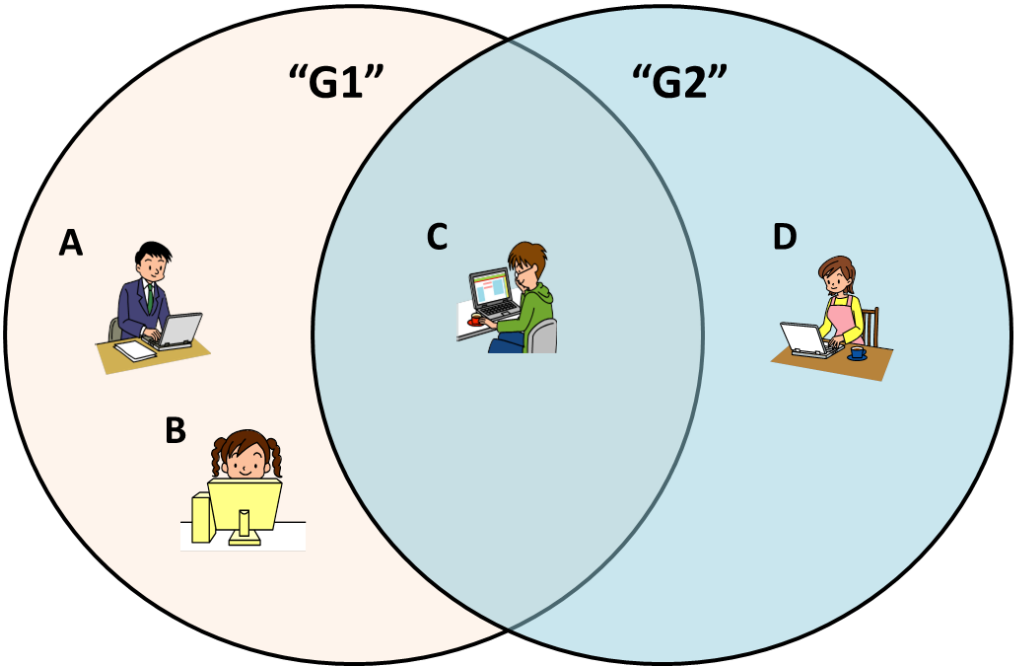


図 2.4: グループの例

図 2.3 の設定後にユーザが Frenzy を利用したときの動作について説明する．C が日記を投稿すると，日記はファイルとして”G1”以下の “ x79gtey76w3u ” (ID_C) 以下の”feeds” と，”G2”以下の “ x79gtey76w3u ” (ID_C) 以下の”feeds”に格納される．端末 T_D では，”G2”以下の “ x79gtey76w3u ” (ID_C) 以下の”feeds”を読み込んで C の日記を GUI に表示する．次に，D が C の日記を読んで返信を書き込むと，返信はファイルとして”G2”以下の “ ky3c92npfttp ” (ID_D) 以下の”feeds”に格納される．D の返信は，フォルダ”G2”を共有している C は見ることができる．しかしフォルダ”G2”を共有していない A と B は見ることができない．A と B が D の返信を見るためには，A と B をグループ”G2”に追加する必要があり，C または D が Dropbox 上でフォルダ”G2”を A と B と共有する設定をする必要がある．

以上のように，Frenzy は Dropbox で事前設定を行う必要があり，Frenzy からはグループの設定をすることができない．よって Frenzy ではコミュニケーション相手が固定となり，コミュニティを広げるという SNS としての機能が欠如している．

本研究では新たにフォルダの構造を定義し，フレンドのフレンドの情報を得る

仕組みとフレンドを追加する仕組みを提案し，上記の問題を解決したオンラインストレージベースの分散 SNS を提案する．

第 3 章

提案システム

本研究で提案する Online Storage-SNS(以後 OS-SNS) の設計を示す.

3.1 前提

OS-SNS を構成するための前提を示す.

- ユーザはグローバルユニークなユーザ ID を持つ.

  OS-SNS がユーザを識別するための ID である．たとえばメールアドレスである．本論文では，ユーザ A のユーザ ID を ID_A と表し，ID_A が”A@a.com” であることを ID_A = ”A@a.com ”と表す.

- ユーザの端末はインターネットに接続され，オンラインストレージサービスが提供する同期アプリケーションがインストールされている.

3.2 定義

OS-SNS を構成するために必要な概念や用語を定義する.

- コンテンツ

  テキスト，写真，動画，音楽，アプリケーションのファイルである.

- 投稿

  ユーザがコンテンツをSNSに入力して指定した相手が閲覧できるようになることを示す.

- コンテンツID

  コンテンツを識別するためのIDである．たとえば，ユーザIDと投稿時刻を組み合わせた値を使用する.

- ユーザ間の関係：フレンド関係

  お互いの情報を共有し閲覧することができる関係である．コンテンツを投稿する際に送信先として選ぶことができる.

- ユーザとコンテンツ間の関係：投稿者関係

  コンテンツとそのコンテンツを投稿したユーザの関係である.

- コンテンツ間の関係

  投稿に対する関係 (たとえば日記に対するコメントという関係) である.

3.3 要求

OS-SNSは以下の機能をそなえるものとする.

- コンテンツの投稿

  コンテンツをSNSに入力して投稿できる．たとえば画面に表示されたコンテンツ入力用のフォームにユーザがコンテンツを入力しボタンを押すなどのアクションを行うと，フレンドは投稿されたコンテンツを閲覧できる．また投稿する際に見せる相手を設定できる．たとえばフレンド全員や1人のフレンドなどである．特に見せる相手を設定しない場合はフレンド全員が閲覧できる.

- 投稿されたコンテンツの閲覧

  自分やフレンドが投稿したコンテンツを画面に表示できる．

- 投稿に対してコンテンツを投稿

  閲覧したコンテンツに関係するコンテンツを投稿できる．以降，この投稿を返信と呼ぶ．返信は返信対象の投稿がわかるように表示される．

- フレンドの追加

  コミュニティの幅を広げるために新しいフレンド関係を相手のユーザにリクエストすることができる．リクエストが承諾されるとフレンドになり，フレンド一覧を表すリストに追加される．

- フレンドの投稿への返信を閲覧することができる．

  自分や「フレンドのフレンド」が投稿したフレンドの投稿への返信を閲覧できる．これよりフレンドのフレンドの存在を知ることができ，フレンドを追加する機会を得る．

- 通知

  自分宛てのフレンド追加のリクエストを受け取ったときに，通知が表示される．

3.4 構成

OS-SNS は図 3.1 のように端末，インターネット上の同期・共有サーバで構成される．図 3.1 は同期・共有サーバと同期アプリケーションがオンラインストレージサービスとして動作し，フレンドの端末間でお互いの共有用フォルダが共有されている状況を示している．

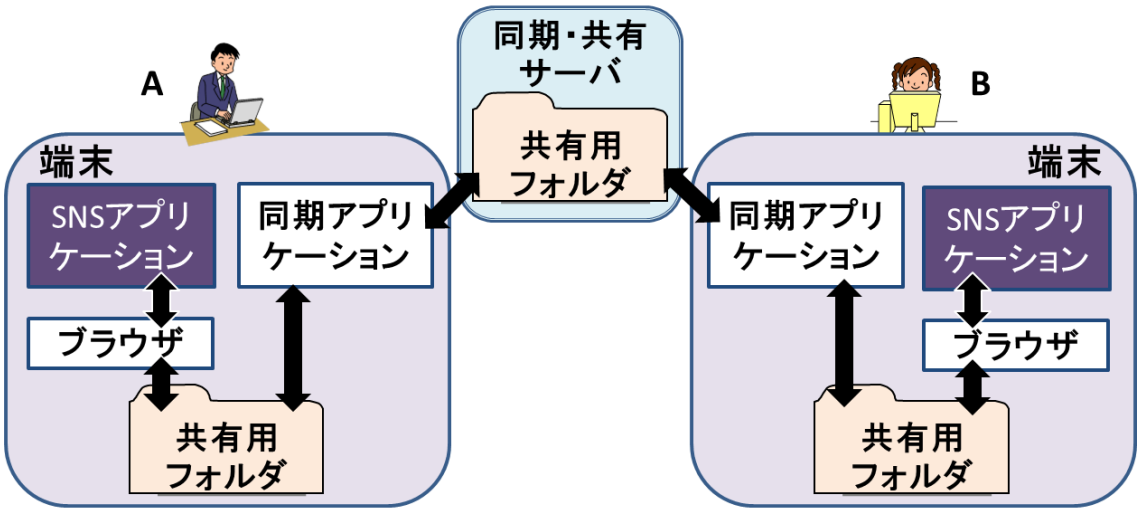


図 3.1: OS-SNS の構成

端末は SNS アプリケーション，同期アプリケーション (オンラインストレージが提供するアプリケーションや任意のアプリケーション)，共有用のフォルダ，ブラウザで構成される．

- ブラウザ

  SNS アプリケーションを実行することで，共有フォルダのファイルを SNS のコンテンツとして見せる UI の役割と，ユーザが入力したコンテンツをファイルとしてローカルフォルダへ格納する役割を持つ．

- SNS アプリケーション

  HTML と JavaScript で記述されたファイルである．共有用フォルダに格納されたファイルを読み込んでコンテンツとしてブラウザへ表示させる関数，またユーザから入力されたコンテンツをファイルとして共有用のローカルフォルダ内へ書き込む関数が記述されている．

- 同期アプリケーション

  同期・共有サーバの設定に従って，同期設定された端末の共有用フォルダと同名のサーバ上のフォルダを同期する．ユーザが端末の共有用フォルダを

更新すると，同期アプリケーションが同期・共有サーバに更新内容をアップロードする．また同期・共有サーバから更新を受け取って共有用フォルダの内容を同期する．

同期・共有サーバはユーザのフォルダを格納する．同期・共有設定された端末から共有用フォルダの更新を受け取り，サーバに格納されたフォルダの内容と同期する．サーバのフォルダが更新されると，サーバは同期・共有設定された端末に更新内容を送信する．

3.4.1 フォルダ構造

投稿するコンテンツを誰に見せるか設定するために，共有用フォルダの命名規則を定義する．たとえば図 3.2 は 3 人のユーザ A(以下 A)，ユーザ B(以下 B)，ユーザ C(以下 C) が OS-SNS を利用しており，それぞれの ID は ID_A=”A@a.com”，ID_B=”B@b.com”，ID_C=”C@c.com”である．実線矢印はフレンド関係を表す．A と B，B と C がフレンドである．また破線矢印は同名のフォルダが共有されていることを表す．A は自分の ID と同名のフォルダ ID_A と，フレンドの ID と同名のフォルダ ID_B と，自分の ID とフレンドの ID を組み合わせた名前のフォルダ $ID_A + ID_B$ を持つ．フォルダ $ID_A + ID_B$ = ”A@a.com+B@b.com”である．フォルダ ID_A，ID_B，$ID_A + ID_B$ は A と B で共有される．同様にフォルダ ID_B，ID_C，$ID_B + ID_C$ は B と C で共有される．フォルダ ID_B は A，B，C で共有される．

フォルダの用途

図 3.2 では，フォルダ ID_B は B がフレンドに公開する投稿が格納される．また B がフレンドに公開した投稿に対する A または C の返信が格納される．たとえば B がフレンド向けに日記を投稿すると，日記はファイルとしてフォルダ ID_B に格納される．このファイルは端末 T_A と T_C のフォルダ ID_B に同期される．A が SNS アプリケーションを開くと，フォルダ ID_B に格納された B の日記がブラウザに表

示される．ここでAがBの日記へ返信したとき，Aの返信はフォルダID_Bに格納される．CがSNSアプリケーションを開くと，ブラウザにID_Bに格納されたAの返信を閲覧できる．このフォルダ構成により，Cから見てフレンドのフレンドの関係にあたるAの返信を閲覧できるので，CがAをフレンドに追加する機会を得る．

フォルダ$ID_A + ID_B$はAまたはBがコンテンツを見せる相手にお互いを指定したときの投稿や，その投稿に対するAまたはBの返信が格納される．たとえばBが「Aに見せるがCに見せない」コンテンツを投稿したいとき，見せる相手をAに指定することでコンテンツがファイルとしてフォルダ$ID_A + ID_B$に格納される．

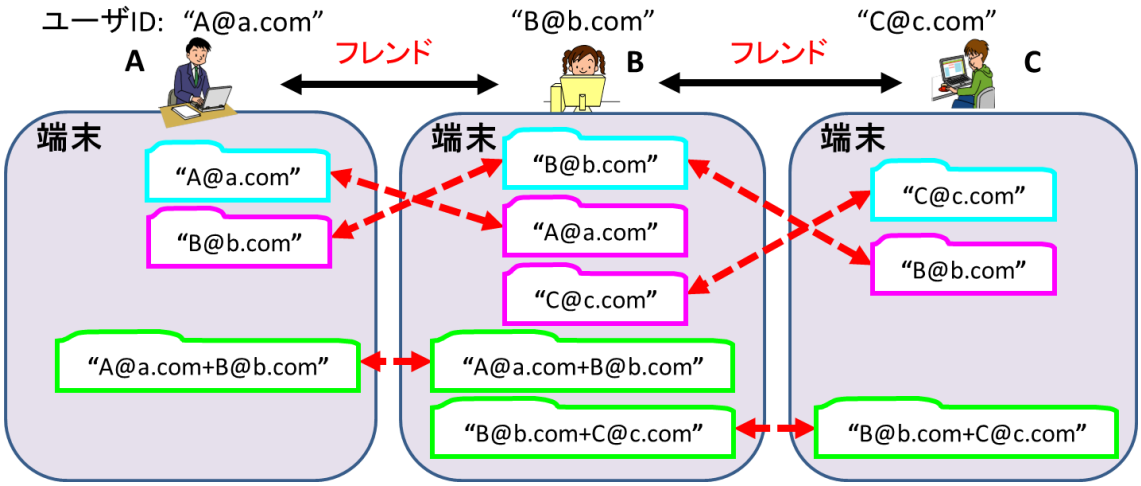


図 3.2: 3ユーザで共有されるフォルダ

投稿から閲覧の流れ

A が OS-SNS でコンテンツを投稿し，A のフレンド B がそれを閲覧するまでの設計を図 3.3 を例にして示す．図において A と B の端末には SNS アプリケーションがダウンロードされており，フォルダ ID_A が同期・共有サーバを通じて共有されていることを表している．以下の丸数字は図 3.3 と対応している．

① まずユーザ A は SNS アプリケーションをブラウザで開く．A の SNS アプリケーションは投稿フォーム表示関数を実行しブラウザに表示させる．

② A のブラウザ上に投稿フォームが表示される．

③ 次にユーザ A がコンテンツを入力して投稿ボタンを押す．

④ A の SNS アプリケーションは A が入力したコンテンツと，現在時刻の取得を行い，コンテンツ表現ファイルとして格納する関数をブラウザに実行させる．

⑤ ブラウザはファイルを A の端末の ID_A に格納する．

⑥ A の同期アプリケーションは A の ID_A の変更を見つける．

⑦ A の同期アプリケーションは同期・共有サーバに ID_A の変更を送信する．

⑧ ユーザ B の同期アプリケーションは同期・共有サーバから更新を受け取る．

⑨ ユーザ B の同期アプリケーションは受け取った更新を B の端末の ID_A に同期する．

⑩ ユーザ B が SNS アプリケーションをブラウザで開く．

⑪ B の SNS アプリケーションは画面表示用の関数を読み込んでブラウザに表示させる．

⑫ ブラウザが A の投稿を B の端末の ID_A からコンテンツ表現ファイルを読み込む．

⑬　ブラウザ上に表示する．結果 B は A の投稿を閲覧することができる．

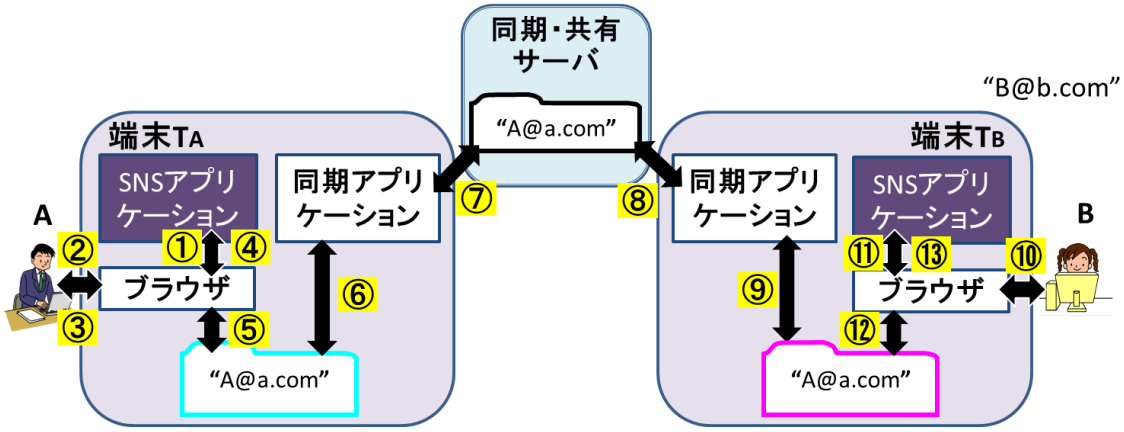


図 3.3: フレンドの投稿を閲覧する

3.5　フレンド追加の手順

図 3.2 のように 3 人のユーザ A，B，C が居て A と B，B と C がフレンドであったときに，B の日記に対する A のコメントを見た C が，A をフレンドに追加するための手順を図 3.4 に示す．SNS アプリケーションが A にフレンド追加のためのリクエスト (以下，フレンドリクエスト) を送信する手順と，同期・共有サーバがフレンドリクエストを処理する手順を以下に示す．丸数字は図 3.4 中と対応している．

① C は A にフレンドリクエストを行う．すると C の SNS アプリケーションは C の端末にフォルダ ID_A を作成する．

② 同期・共有サーバはフォルダ ID_A の作成を検知する．

③ 同期・共有サーバは A にフレンドリクエストの問い合わせを行う．

④ A は C からのフレンドリクエストを通知され，A がフレンドリクエストを承認する．

⑤ 同期・共有サーバは承認を受け取ってAとCの端末間のフォルダの共有設定を行う．

⑥ 同期・共有サーバはAの端末にフォルダ ID_C を作成し，AとCの端末にフォルダ $ID_A + ID_C$ を作成する．そしてAとCの端末のフォルダ ID_A，ID_C，$ID_A + ID_C$ を共有設定する．

SNSアプリケーションはフォルダを作成することでフレンドリクエストができる．また，オンラインストレージサービスがフォルダ構造からSNSのセマンティクスを理解し，上記の動作を行うことでSNSアプリケーションからのフレンド関係の拡張が可能になる．なおフレンド申請がAにより却下された場合は，⑤のタイミングで同期・共有サーバがCの端末のフォルダ ID_A を削除する．

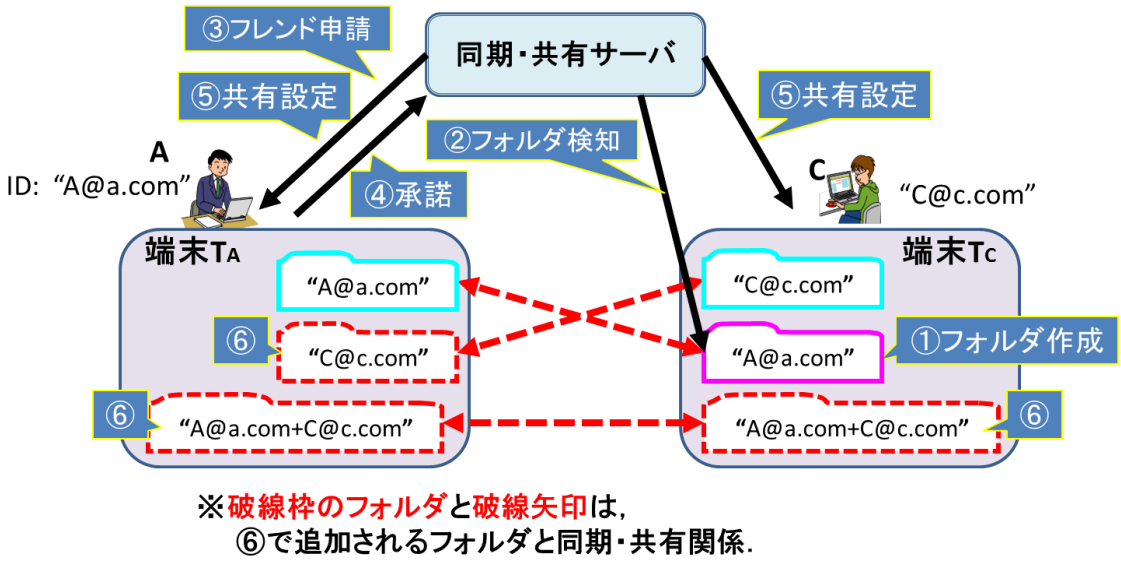


図 3.4: フレンドリクエストの手順

第 4 章

実装

SNS アプリケーションを HTML と JavaScript を用いて実装した．ブラウザからのローカルファイル操作のために，Internet Explorer の ActiveX 技術を利用した．

また同期・共有サーバとして Dropbox を用いた．Dropbox は，SNS アプリケーションのようなサードパーティシステムからのフォルダ生成や共有設定を行えないため，Dropbox を補助しフレンドリクエストの処理を行うシステムを Linux 仮想マシン上で実装した．OS-SNS ユーザのフォルダやファイルを監視するために inotify を用い，同期処理には Unison[19] を用いた．補助システムはシェルスクリプトで記述した．

4.1 SNS アプリケーション

ブラウザからのローカルファイル操作のための技術を紹介する．また実装した SNS 機能を紹介する．

4.1.1 ActiveX 技術

ブラウザより任意のローカルファイルを操作することはセキュリティの問題があるため，通常は機能が提供されていない．JavaScript において，Internet Explorer でサポートされる ActiveXObject オブジェクトである Scripting.FileSystemObject オブジェクトを利用することで，ブラウザからローカルファイル操作を行うことが

できる．利用方法は，まず ActiveXObject オブジェクトを FileSystemObject オブジェクト (変数 object) に割り当てる．

```
object = new ActiveXObject("Scripting.FileSystemObject");
```

次に object に対してテキストファイルの操作，ファイルやフォルダのコピーと移動に関するメソッドを用いる．

テキストファイルの操作

テキストファイルの操作について，OpenTextFile() メソッドを用いてファイルを開き，変数 file に開いたファイルを割り当てる．

```
file = object.OpenTextFile(filename, iomode, create, format);
```

ここで filename はファイルパスを指定，iomode はファイルの開き方について選択する (1:読み取り専用，2:書き込み専用，8:ファイルの最後に追記)．create は filename で指定されたファイルが存在しない場合にファイルを新規作成するかどうか選択する (true: 新規作成，false: 作成しない)．format は開くファイルの形式を示す (-2: OS デフォルト文字コード，-1: Unicode，0: ASC)．

次に ReadLine() メソッド，Write() メソッドを用いて OpenTextFile で開いたファイルの読み書きを行う．そして最後に Close() メソッドを用いてファイルを閉じる．例えば「C:user/Dropbox/new.txt」というパスのファイルに「新しいテキスト」という文字を書き込みたいとき，以下のような JavaScript コードとなる．

```
<script>
      var filename = “ C:user\\Dropbox\\new.txt ”
      var content = “ 新しいテキスト ”
      var object = new ActiveXObject("Scripting.FileSystemObject");
      var file = object.OpenTextFile( filename, 2, true, -1);
```

```
                    // 存在しなければ作成，Unicode で開く
        file.Write(content);  //content の内容を書き込む
        // file.ReadLine(); // ファイルから 1 行読み込む
        file.Close(); //閉じる
</script>
```

ファイルとフォルダの操作

ファイルとフォルダについて，コピーや移動を実装するためのメソッドを紹介する．

ファイルのコピーと移動は CopyFile() メソッド，MoveFile() メソッドを用いる．

```
// ファイルのコピー
object.CopyFile(source, destination, overwrite);
// ファイルの移動
object.MoveFile(source, destination, overwrite);
```

source はコピー (または移動) 元のファイルパス，destination はコピー (または移動) 先のフォルダパスを指定する．overwrite はコピー (または移動) 先に同名のファイルが存在する場合に上書きするか指定する (true: 上書き，false: エラーを返す)．

フォルダの生成、フォルダのコピーと移動に関するメソッドは CreateFolder(), CopyFolder() と MoveFolder() である．

```
// フォルダの生成
object.CreateFolder(source, destination);
// フォルダのコピー
object.CopyFolder(source, destination);
// フォルダの移動
object.MoveFolder(source, destination);
```

ここで source は作成，コピー (または移動) 元のフォルダパス，destination は作成，コピー (または移動) 先のフォルダパスを指定する．

4.1.2 実装した機能

SNS アプリケーションで実装した SNS 機能や要求のために作成したものを以下に示す．

- システムフォルダ

  OS-SNS を利用するユーザ自身のデータが格納され，誰とも共有されない．フォルダ名を”OS-SNS”として実装した．

- フォルダリスト

  OS-SNS で利用する共有フォルダの名前が記述され，システムフォルダ以下に格納される．SNS アプリケーションはこのリストを読み込んで SNS を構成する．フォルダリストはテキストファイル”folderlist.txt”として実装した．

- コンテンツ表現ファイル

  投稿者の ID，コンテンツ，コンテンツ ID，コンテンツ間の関係，コンテンツの格納場所を格納している．1 つのコンテンツ表現ファイルに別のコンテンツ ID を付加することにより 2 コンテンツを結びつけることができる．たとえば図 4.1 において，ユーザ A とユーザ B がフレンドであり，B が書いた日記が B のコンテンツ表現ファイルに格納されている．このとき B の日記は投稿者として B の ID，コンテンツ ID として B の ID と投稿時間を組み合わせたもの，コンテンツ間の関係はフレンド全員向けを指定するためフレンド，格納場所を B のコンテンツ表現ファイルとして格納されていることを表している．A が B の日記に対してコメントを書くと，投稿者として A の A の ID，コンテンツ ID として A の ID と投稿時間を組み合わせたもの，コンテンツ間

の関係は B の日記を指定するために B の日記のコンテンツ ID，格納場所を B のコンテンツ表現ファイルとして追記し格納される．

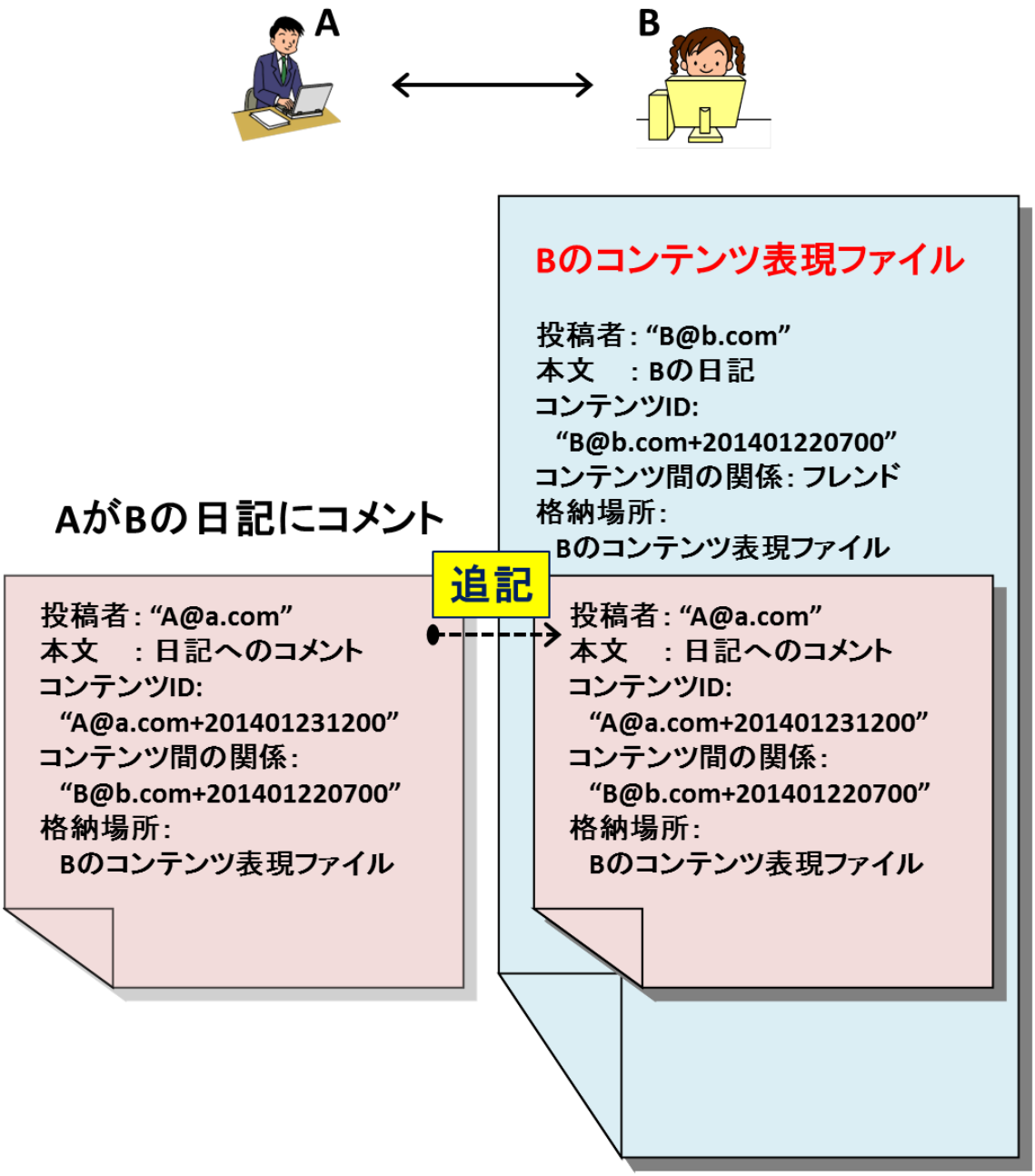


図 4.1: コンテンツ表現ファイル

コンテンツ表現ファイルはテキストファイル”timeline.txt”として実装し，3.5節で定義した3種類のフォルダ毎に1つ格納される．

コンテンツ表現ファイルをコンテンツ毎に作成してしまうと，コンテンツが削除された場合に，ファイルが削除されたことを同期アプリケーションがユーザの画面へポップアップ表示してしまう．オンラインストレージサービスはバージョン管理をしており，通知からどの投稿が削除されたのかすぐ知ることができる．しかし，削除する投稿内容はユーザにとって見られたくないものである．1つのファイルに追記する実装では，該当箇所の投稿内容を削除したとしても同期アプリケーションがポップアップ通知するのはファイルの更新であり，削除ではない．ユーザより削除した箇所を簡単に特定できなくなるため，1つのファイルに追記する手法を採用した．

- 初期設定

  SNSアプリケーションを利用するユーザを識別するためにユーザIDの登録をする．さらにシステムフォルダ，フォルダリストを作成する．ユーザがSNSアプリケーションを開くと，SNSアプリケーションはシステムフォルダの有無を確認し，システムフォルダが存在しない場合にはブラウザ上にユーザIDの入力フォームを表示する．ユーザがIDを入力すると，ユーザの端末にフォルダ名がユーザIDのフォルダが作成され，またシステムフォルダ”OS-SNS”を作成する．次に”OS-SNS”以下にフォルダリスト”folderlist.txt”を作成し，自分のユーザIDを書き込む．例えばユーザAが最初にSNSアプリケーションを開いたときに，Aのブラウザに ID 入力フォームが表示される．AはID_Aを入力して決定を押す．するとAの共有用ローカルフォルダにはフォルダF_A=”ID_A”と”OS-SNS”フォルダが作成され，”OS-SNS”以下のユーザリストにはF_Aが記述される．

- フレンド全員向けの投稿

たとえばA がSNS アプリケーションをブラウザで開くと，タイムラインの上に投稿フォームが表示される．コンテンツを入力して投稿ボタンを押すことによって投稿が完了する．ボタンが押されたとき，SNS アプリケーションは現在時刻の取得を行い，システムフォルダから自身のID を取得し，投稿内容をフォームから取得する．そして時刻，ID，内容，格納フォルダ名である F_A をフォルダ F_A 以下の”timeline.txt”に書き込む．

- 宛先を指定した投稿

  たとえばA がフレンドB にコンテンツを送りたいときは投稿フォームでユーザB を指定する．コンテンツを入力して投稿ボタンが押されたとき，SNS アプリケーションは現在時刻の取得を行い，システムフォルダから自身のID をコンテンツの著者として取得し，投稿内容をフォームから取得する．そして時刻，ID，内容，格納フォルダ名である F_{A+B} を F_{A+B} フォルダ以下の”timeline.txt ”に書き込む．

- 投稿への返信

  たとえばB の日記に対してA がコメントを投稿したいとき，B の日記の下に表示された投稿フォームにコメントを書き込んで投稿ボタンを押す．コメントを書き込む先は，B の日記が持っているコンテンツ格納場所情報であり，F_B となる．SNS アプリケーションは現在時刻の取得を行い，システムフォルダから自身のID を取得し，投稿内容をフォームから取得する．そして時刻，ID，内容，格納フォルダ名である F_B をフォルダ F_B 以下の”timeline.txt”に書き込む．

- リクエストファイル

  フレンドリクエストを受け取ったときの通知を行うために作成される．リクエストファイルは各ユーザのシステムフォルダに格納される．フレンドリクエストを受け取ると，誰からのリクエストであるか記述される．またユー

ザがリクエストに対して承諾や拒否するとリクエストの状態が更新される．リクエストの状態は以下の 2 種類用意した．

1... リクエスト状態

SNS アプリケーションがブラウザへリクエスト元 ID とリクエストへの返事ボタンを表示させる．返事ボタンは「承諾」,「拒否」を表示する．

2... リクエスト承諾

リクエストを受け取ったユーザが承諾を選択するとこの値に書き換わる．この値のフレンドリクエストはブラウザに表示されなくなる．

3... リクエスト拒否

リクエストを受け取ったユーザが拒否を選択するとこの値に書き換わる．この値のフレンドリクエストはブラウザに表示されなくなる．

リクエストファイルは，テキストファイル”req.txt”として実装した．

- フレンドリクエスト

フレンドリクエストを実行すると，リクエスト相手の ID と同名のフォルダが作成される．同期・共有サーバの補助システムはこのフォルダ作成を検知してフレンドリクエストの処理を行う．

- コンテンツ時間順の表示

コンテンツを時間順に表示するための機能である．SNS アプリケーションはシステムフォルダ”OS-SNS”以下の”folderlist.txt”に記述された利用フォルダ一覧を見て，共有用ローカルフォルダ内から SNS の構成のために開くフォルダを選択し，各フォルダ内の”timeline.txt”を開いていく．”timeline.txt”に記述されたコンテンツをクイックソートを用いて時刻が新しいものからブラウザに表示する．各コンテンツの下にはそのコンテンツに対する返信を投稿できるフォームを設けた． たとえばユーザ A がユーザ B とフレンドであ

る場合．A のフォルダリストには F_A，F_B，F_{A+B} が記述されている．よって A の SNS アプリケーションは A の共有用ローカルフォルダから F_A，F_B 内の”timeline.txt”をそれぞれ開き，コンテンツを時間順にソートしてブラウザに表示させる．これより，A は「A が全フレンド向けに投稿したコンテンツ」，「B が全フレンド向けに投稿したコンテンツ」,「A または B がお互いを公開範囲に指定して投稿したコンテンツ」を合わせて時間順に閲覧することができる．

4.2 同期・共有サーバ

ファイルやディレクトリの生成や更新を監視する技術について inotify と，2 つのファイルやディレクトリ内容の同期技術について Unison を紹介する．

また 3.6 節の設計について，Linux 仮想マシンでフレンドリクエストを処理する Dropbox 補助システムを実装した．

4.2.1 イベント監視と同期技術

inotify

inotify は Linux のファイルシステムイベントを監視するための機構を提供し，個々のファイルやディレクトリを監視する．ファイルシステム上でイベントの発生を待つときには，inotifywait コマンドを使用する．inotifywait はオプション-e の後にイベントを指定することでディレクトリやファイルを監視できる API である．監視できるイベントの種類を以下に示す．

- ACCESS

  ファイルがリードされた．

- ATTRIB

メタデータ (permission，タイムスタンプ，拡張属性，リンクカウント，UID，GID など) が変更された．

- CLOSE_WRITE

  書き込みのためにオープンされたファイルがクローズされた．

- CLOSE_NOWRITE

  書き込み以外のためにオープンされたファイルがクローズされた．

- CREATE

  監視対象ディレクトリ内でファイルやディレクトリが作成された．

- DELETE

  監視対象ディレクトリ内でファイルやディレクトリが削除された．

- DELETE_SELF

  監視対象のディレクトリまたはファイル自身が削除された。．

- MODIFY

  ファイルが修正された．

- MOVE_SELF

  監視対象のディレクトリまたはファイル自身が移動された．

- MOVED_FROM

  ファイル名の変更を行った際に変更前のファイル名が含まれるディレクトリに対して生成される．

- MOVED_TO

  ファイル名の変更を行った際に新しいファイル名が含まれるディレクトリに対して生成される．

- OPEN

  ファイルがオープンされた.

たとえば「test.txt」というファイルのオープンを監視したいときには，コマンド

```
inotifywait -e OPEN test.txt
```

のように使用する．test.txt がオープンされると，inotifywait は終了する．

Unison

Unison は 2 つのフォルダ (ディレクトリ) 間でファイルやディレクトリの内容を同期するオープンソースのプログラムで，Unix 系の OS や Windows で動作する．unison コマンドで 2 つのファイルを指定して同期を行う．unison のオプションとして以下のようなものがある．

- -force xxx，-prefer xxx

  xxx の部分に newer もしくは older を指定することで，更新日時が新しいファイルを基準に同期するか古いファ イルを基準に同期するか選択する．-force newer と-prefer newer は同じである．

- -batch

  ユーザに対して問い合わせを行なわない．更新は衝突しない限り自動的に行なわれ，衝突する更新はスキップされる．

たとえばファイル”test1.txt の内容が”テスト 1”であり，ファイル”test2.txt”というの内容が”テスト 2”であり，更新日時は”test2.txt”の方が新しいとき，コマンド

```
unison test1.txt test2.txt -prefer newer -batch
```

を実行すると，”-batch”オプションにより自動で”-prefer newer”に従い”test2.txt”の内容が”test1.txt”に同期される．結果的に”est1.txt”の内容が”テスト 2”となる．

4.2.2 フレンドリクエスト処理手順

ユーザCがユーザAにフレンドリクエストしたとき，フレンドリクエストの処理手順を図4.2に示す．前提として，補助システムはDropboxを用いてOS-SNSユーザのフォルダが同期・共有されている．以下の丸数字は図4.2と対になっている．

① 同期・共有サーバはCのDropboxフォルダに，フォルダの生成を監視するinotifyを設定する．

② 次にCがAにフレンドリクエストをすると，SNSアプリケーションがCのDropboxフォルダ以下にフォルダF_Aを作成する．

③ 同期・共有サーバのinotifyがフォルダ作成を察知し，F_Aよりフォルダ名を抽出する．同期・共有サーバはフォルダ名と一致するIDを持つ端末T_AのF_{OS-SNS}以下のreq.txtにリクエスト状態1とリクエスト元であるCのID_Cを記述する．さらに同期・共有サーバはAのreq.txtにinotifyを設定し，更新を監視する．

④ AがSNSアプリケーションを開くと，Cからフレンドリクエストが来ていることが通知される．Aがリクエストに承諾すると，req.txtのリクエスト状態を2に変更する．同期・共有サーバのinotifyがAのreq.txtの更新を検知し，req.txtを読み込んでリクエスト状態を抽出する．

⑤ 同期・共有サーバはリクエストの承諾を確認すると，Aの Dropboxフォルダ内にフォルダID_Cと$ID_A + ID_C$，Cの Dropboxフォルダにはフォルダ$ID_A + ID_C$を新規作成する．またAとBのF_{OS-SNS}以下のまた同期・共有サーバは同名のフォルダに対してinotifyを用いて新規作成と更新を監視し，更新があった場合にはUnisonで同期を行う．

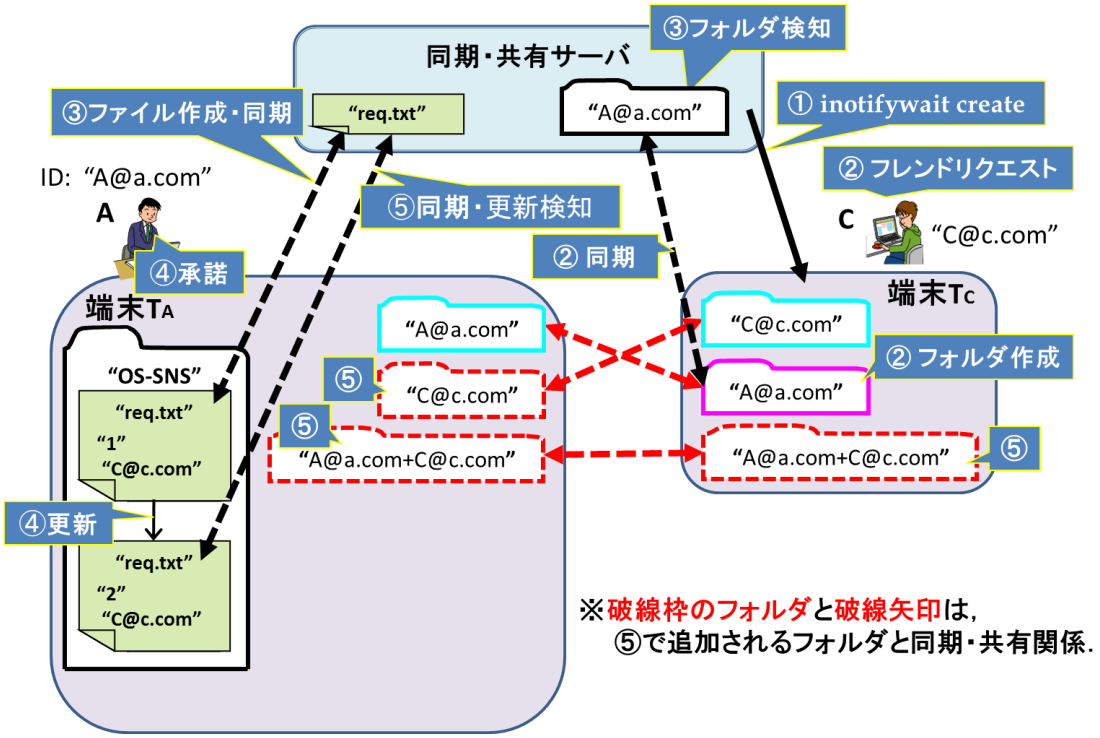


図 4.2: フレンドリクエスト処理手順

第 5 章

評価

5.1 プライバシー性，持続性，つながりの拡張性，オフライン利用の評価

SNS において持つべきであるプライバシー性，持続性，つながりの拡張性，オフラインの利用について，OS-SNS とその他の SNS を比較した．比較を表 5.1 にまとめた．

- Facebook

  一極集中型の SNS である Facebook について評価する．まずプライバシー性について，Facebook のポリシー [14] によると，Facebook は名前やメールアドレス，生年月日，性別などユーザが入力した情報の他に，ユーザのフレンドやコミュニケーションしたユーザから情報を集めている．そして集めた情報はユーザに関連性のある広告を配信することやデータ分析，テストに利用されると記述されており，プライバシー性が十分でない．持続性は広告の配信などにより収入を得ているため備えている．またコンテンツの公開先をインターネット全体などと設定できることや，フレンドを追加できるため，つながりの拡張性がある．オフライン利用について，ユーザ端末が Facebook サーバに接続している状態でなければコンテンツの投稿や取得を行えない．

- FlyByNight

  Facebook のコンテンツを暗号化する FlyByNight について評価する．プライバシー性は暗号化のため高い．持続性も Facebook を利用し備えている．しかしつながりの拡張性は，FlyByNight 内でフレンドを追加できないため備えていない．オフライン利用について，Facebook と同様に行えない．

- VIS，diaspora*

  VIS，diaspora* は個人サーバを利用している．そのためユーザがデータコントロールの権利を持ち，プライバシー性がある．しかし個人サーバはボランティアで管理されているため持続性が無い．つながりの拡張性について，SNS 内のユーザを検索できることやシステム内でフレンド関係を追加することができるため，拡張性がある．オフライン利用について，データを格納している個人サーバへアクセスできないため，コンテンツの投稿や取得を行えない．

- Frenzy

  Dropbox を用いているためプライバシー性と持続性を備えている．しかしシステム内でグループを設定する仕組みが無いためつながりの拡張性が無い．オフライン利用について，ユーザ端末のローカルでファイルの読み書きを行うため，コンテンツの投稿や閲覧を行うことができる．

- OS-SNS

  オンラインストレージを用いているためプライバシー性と持続性を備えている．またフレンドのフレンドとコンテンツを共有でき，OS-SNS よりフレンド関係を追加できるため，つながりの拡張性がある．オフライン利用について，Frenzy と同様に行うことができる．

表 5.1: SNS の評価

SNS	プライバシー	持続性	つながりの拡張性	オフライン利用
Facebook	×	○	○	×
FlyByNight	○	○	×	×
VIS，diaspora*	○	×	○	×
frenzy	○	○	×	○
OS-SNS	○	○	○	○

5.2 Frenzy と OS-SNS の比較

Frenzy におけるグループ設定と OS-SNS のフレンドリクエストについて，ユーザ側の手順を比較する．たとえば図 5.1 はユーザ A がユーザ B をグループやフレンドに追加するときの操作である．

Frenzy では A が Dropbox 上でフォルダを作成して B と共有設定をする．B が共有設定に承諾した後に A と B の両方が Frenzy 上で共有フォルダを使用する設定を行う．

一方 OS-SNS では，A がフレンドリクエストを行い，B がフレンドリクエストに承諾するだけで済むので，SNS 内でフレンドを追加することができる．

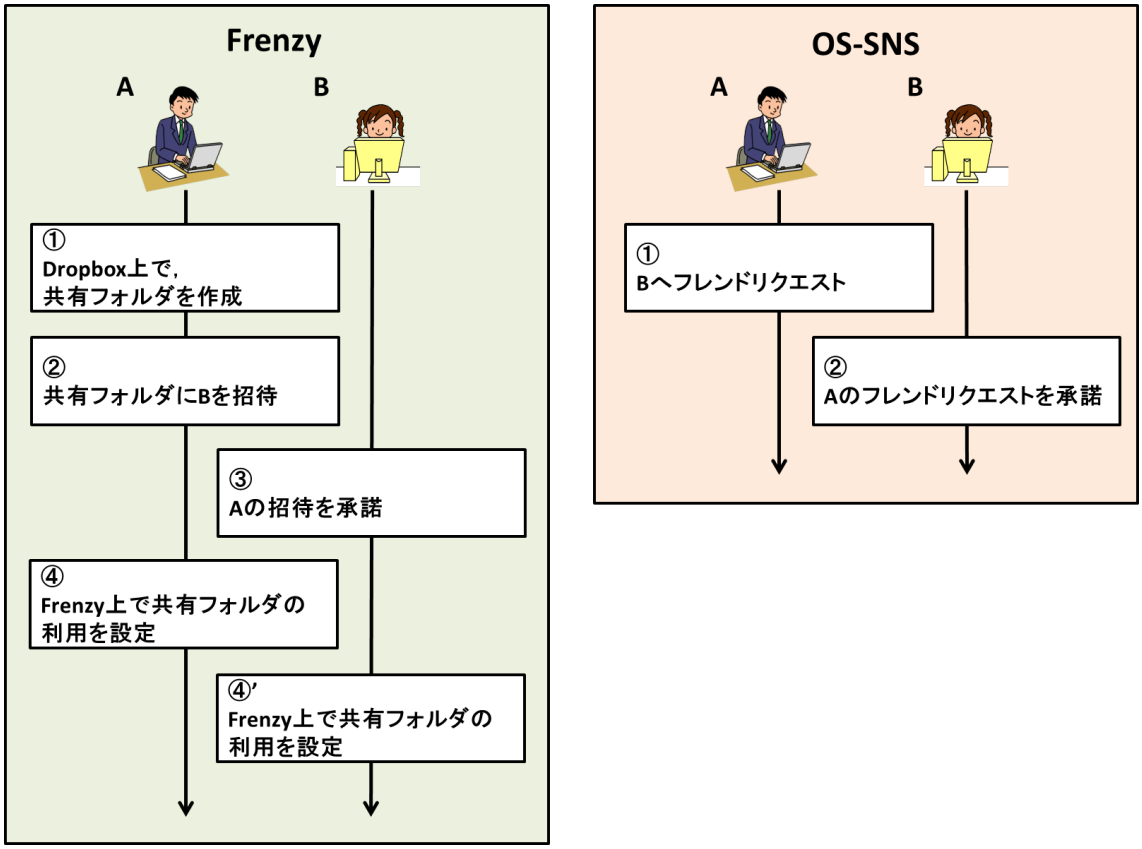


図 5.1: ユーザによるつながりの追加操作

5.3 コンテンツ投稿のレスポンス評価実験

OS-SNS はコンテンツを格納する共有用フォルダをローカルに持っているため，オフライン環境でコンテンツを投稿することができる．SNS アプリケーションのコンテンツ投稿レスポンスを評価するために，100 文字のテキストコンテンツを投稿する際にかかる時間を計測した．比較対象として，Facebook へのコンテンツ投稿時間も計測した．実験環境を表 5.2 に示す．

- OS-SNS

  投稿ボタンが押されてからファイルへの書き込みが完了するまでの時間を計測した．JavaScript 実行時間を計測するために，Date インスタンスを用いた．投稿を 100 回計測して，計測結果は平均 1.2[ms] となった．結果は 0〜

表 5.2: 計測環境

項目	仕様
OS	Windows 7 64bit
実装メモリ (RAM)	12.0GB
プロセッサ	Intel Xeon X5675
ブラウザ	Internet Explorer 11

3[ms] の間に納まった.

- Facebook

  投稿ボタンが押されて Facebook サーバへリクエストを送り，レスポンスが返ってくる時間を計測した．リクエスト・レスポンスの解析ツールとして Fiddler[21] を用いた．投稿を 100 回計測して，計測結果は平均 1477.2[ms] となった．結果は約 1253〜1904[ms] の間に納まった．このとき，Facebook サーバの ping(100[Byte]) 応答時間は平均 178[ms] であった.

OS-SNS のコンテンツ投稿時間は Facebook よりも圧倒的に速く，人間のまばたきが 100ms 前後である [20] ことと比較すると，OS-SNS のユーザは投稿にかかる時間を意識することなく利用できる.

第 6 章

考察

6.1 オンラインストレージサービスのプライバシーポリシー

オンラインストレージサービスのプライバシーポリシーはサービスにより差があるため，プライバシー性を保ちつつ OS-SNS を利用するためには注意する必要がある．たとえば Dropbox と GoogleDrive のポリシーは以下のようになっている．

- Dropbox

  Dropbox のポリシー [17] によると，Dropbox 社員がユーザのアップロードファイルを閲覧することを禁じている．法的に要求された場合には，少数の社員がファイルにアクセスできると記述されている．ファイルの中身を利用して分析や広告配信を行っていないため，一極集中型 SNS よりプライバシー性があると言える．

- GoogleDrive

  一方で GoogleDrive のポリシーは Google の利用規約 [18] にのっとっており，ユーザが Google にアップロードしたコンテンツは，Google が使用，複製，変更，派生または出版，上映，表示，および配布を行うための全世界的なライセンスが付与される．コンテンツの所有権が Google にも渡ってしまうことになるため，GoogleDrive はプライバシー性が低い．

6.2 暗号化

コンテンツはユーザ管理下にあるため，暗号化が可能である．コンテンツを暗号化することで，よりオンラインストレージサービスからのプライバシー性を高めることができる．

たとえばOS-SNSにおいて公開鍵暗号方式を用いて，秘密鍵は共有フォルダ以外の場所，公開鍵をフレンドのIDフォルダに格納する．フレンドのIDフォルダはフレンドのフレンドも共有しているため，公開鍵はフレンドのフレンドまで伝えることができる．

6.3 ストレージの分散

必要に応じて個人サーバを使うことも含め，複数のオンラインストレージを用いることでつながりの拡張性とコンテンツの分散が期待できる．

第 7 章

おわりに

本研究ではつながりの拡張性のある，オンラインストレージを用いた分散 SNS を提案して実装した．

ユーザがフレンドのフレンドの情報を得るために，SNS として扱うフォルダ構造を定義した．これによりオンラインストレージサービスで自動的にフレンドのフレンドのコンテンツを共有できる仕組みを提案した．また，SNS からフレンドを追加できるようにするため，フォルダにフレンド関係のセマンティクスを定義し，それに基づきフォルダの共有設定を行うようオンラインストレージの動作を拡張した．オンラインストレージサービスがフォルダ構造より SNS のセマンティクスを理解し，SNS アプリケーションと協調動作することでフレンドの追加を実現した．

実装において，フォルダの更新を発見する inotify の API と同期ツールである Unison を用いてオンラインストレージの動作を拡張した．そしてフレンドリクエストを処理する動作を確認した．

評価により，提案システムはプライバシー，持続性，つながりの拡張性を全て備えた SNS であることが示された．また，オフライン時にもコンテンツを読み書きでき，計測より書き込みのレスポンスが高いことが示された．

提案方式はコミュニケーションを行う SNS 以外のシステム，たとえばメールやメッセージングにも適用できる．

今後の課題として複数のオンラインストレージの利用が挙げられ，つながりの拡張性やコンテンツを置くストレージの分散が期待できる．

謝辞

本研究を遂行するにあたっては，いろいろな方々にお世話になりました．

まず指導教員の鶴岡行雄先生には日頃から熱心なご指導、そしてご鞭撻を賜わりました．またご多忙中にもかかわらず論文の草稿を丁寧に読んで下さり，大変貴重なご助言をいただきました．ここに厚く御礼申し上げます．

また，多田好克先生，小宮常康先生，荒堀喜貴先生，佐藤喬さんには研究を進める上で貴重な意見や助言をいただきました．ここに感謝の意を表します．

そして本研究が行なえたことは，研究方針や方法論について議論をし．共に研究生活をおくってきた，鶴岡研究室をはじめとした基盤ソフトウェア学講座の学生諸氏のおかげでもあります．最後に，これらの皆さんに感謝いたします．

参考文献

[1] Facebook, https://www.facebook.com/, 2014 年閲覧.

[2] Cereja Technology, Press Release, http://www.cereja.co.jp/press_release20120607.pdf, 2012

[3] Yazan Boshmaf, Ildar Muslukhov, Konstantin Beznosov, Matei Ripeanu, "The Socialbot Network: When Bots Socialize for Fame and Money," 27th Annual Computer Security Applications Conference (ACSAC'11), pp. 93–102, 2011.

[4] Sonja Buchegger and Anwitaman Datta, "A Case for P2P Infrastructure for Social Networks - Opportunities & Challenges," Sixth Wireless On-Demand Network Systems and Services (WONS) , pp. 161–168, 2009.

[5] Ramon Caceres, Landon P. Cox, Harold Lim, Amre Shakimov, Alexander Varshavsky , "Virtual Individual Servers as Privacy- Preserving Proxies for Mobile Devices," ACM SIGCOMM Conference - SIGCOMM, pp. 37–42, 2009.

[6] diaspora*, http://joindiaspora.com/, 2013 年閲覧.

[7] Frenzy, http://frenzyapp.com/, 2013 年閲覧.

[8] mixi, http://mixi.jp/, 2014 年閲覧

[9] Twitter, https://twitter.com/, 2014 年閲覧

[10] Lucas, M. M., N. Borisov, "FlyByNight: Mitigating the Privacy Risks of Social Networking," Workshop On Privacy In The Electronic Society - WPES, pp. 1–8, 2008.

[11] Affelio, http://affelio.jp/, 2014 年閲覧

[12] Friendica, http://friendica.com/, 2013 年閲覧.

[13] Dropbox, http://www.dropbox.com/, 2014 年閲覧.

[14] Facebook, データの使用に関するポリシー, http://www.facebook.com/about/privacy/, 2014 年閲覧.

[15] Facebook, 利用規約, http://www.facebook.com/legal/proposedSRR/, 2014 年閲覧.

[16] Facebook, コミュニティ規定, http://www.facebook.com/communitystandards/, 2014 年閲覧.

[17] Dropbox, Dropbox の安全性について, http://www.dropbox.com/help/27/, 2014 年閲覧.

[18] Google, Google利用規約, http://www.google.com/intl/ja/policies/terms/, 2014 年閲覧.

[19] Unison, http://www.cis.upenn.edu/ bcpierce/unison/, 2014 年閲覧.

[20] 田多英興, 山田冨美雄, 福田恭介, “まばたきの心理学,” pp.2–7, 北大路書房, 1991.

[21] Fiddler, http://www.telerik.com/fiddler, 2014 年閲覧.

付録 A

実装画面

OS-SNS の SNS 機能を動作させた例を紹介する．

- 初期設定

  初期設定ではユーザ ID を入力する．図 A.1 ではユーザ A が”A@a.com” と入力し，ID_A=”A@a.com” となった．

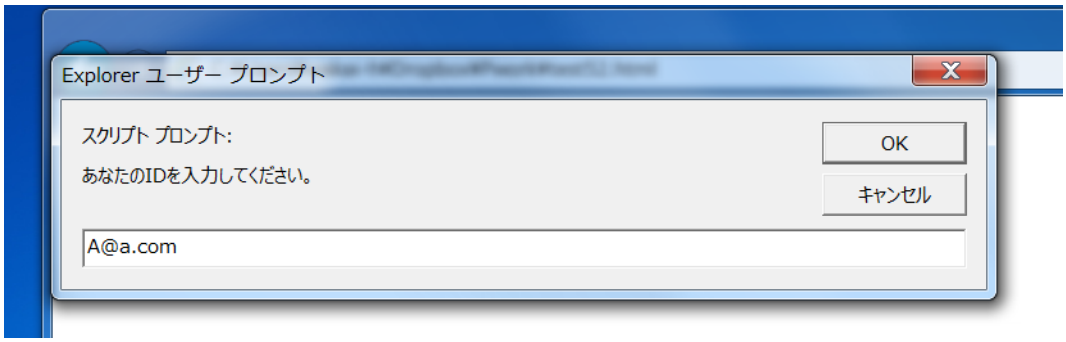


図 A.1: 初期設定画面

- 投稿

  画面に表示されたフォームにテキストを入力して書き込みボタンを押すことで投稿が完了する．図 A.2 では”はじめての投稿”と入力して書き込みボタンを押した．すると図 A.3 のように表示される．

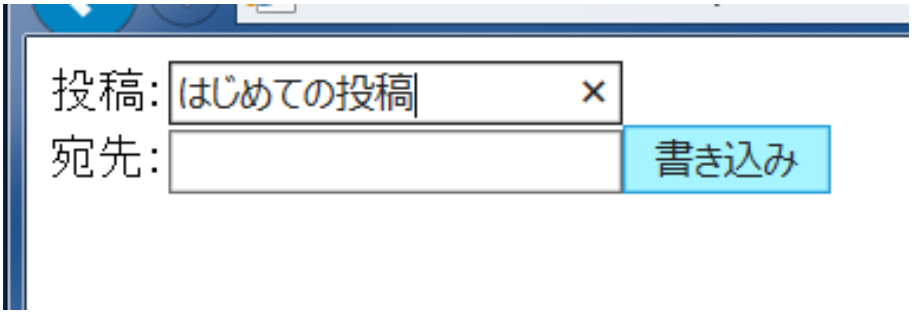
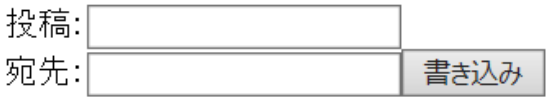


図 A.2: 投稿の例



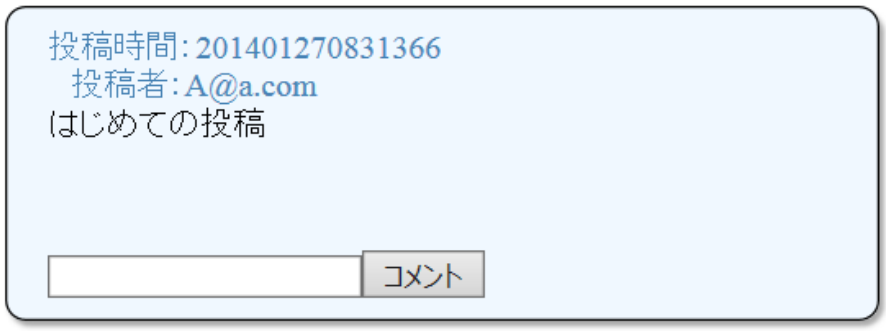


図 A.3: 閲覧画面 1

- 宛先を指定した投稿

  宛先のフォームに宛先を入力して投稿する．図 A.4 ではユーザ B(ID_B=”B@b.com”) がフレンドであったときに，宛先に ID_B を指定して投稿している．投稿後の画面は図 A.5 になる．

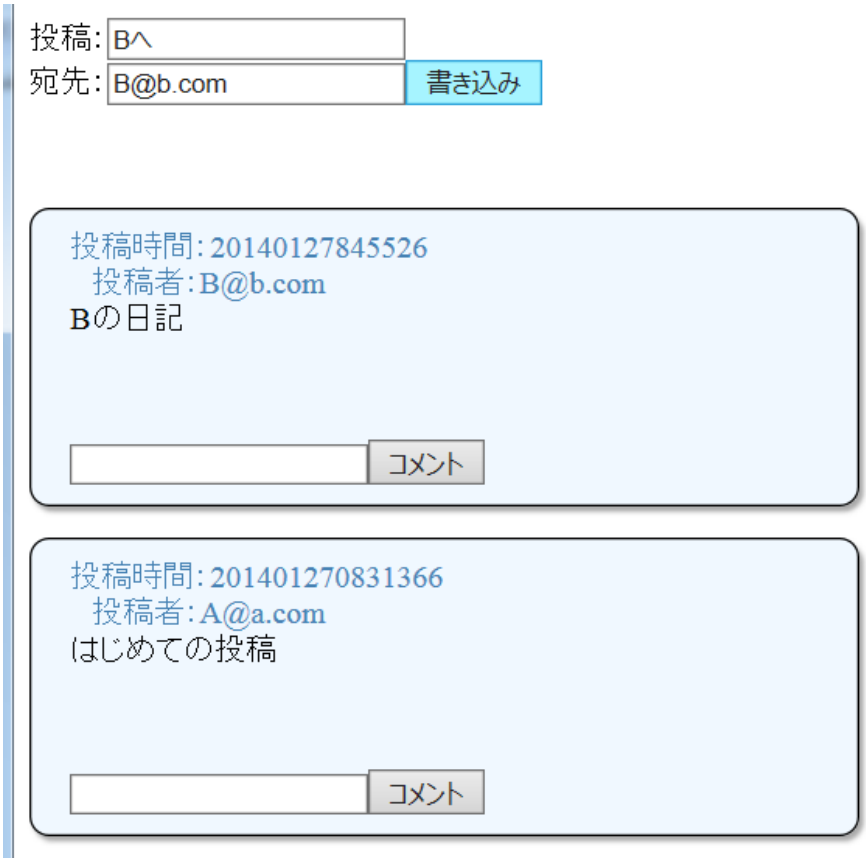


図 A.4: 宛先を指定した投稿例

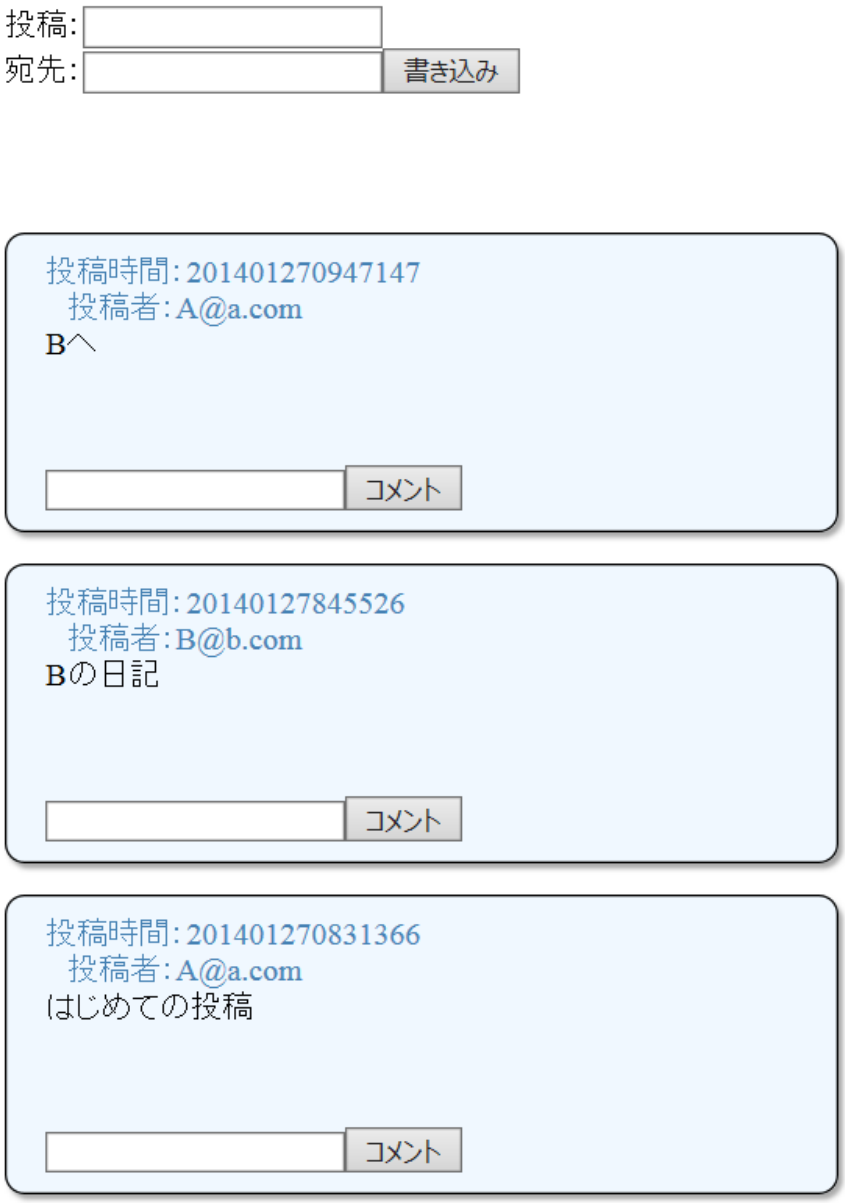


図 A.5: 閲覧画面 2

- 返信

  各コンテンツ下部の投稿フォームより返信を行う．図 A.6 では A が B の日記に返信する．フォームに”A の返信”と入力し，コメントボタンを押して返信を完了した．返信後の画面は図 A.7 になる．


投稿:
宛先:
書き込み
投稿時間: 20140127845526
投稿者: B@b.com
Bの日記
Aの返信
コメント
投稿時間: 201401270831366
投稿者: A@a.com
はじめての投稿
コメント

図 A.6: 返信の例

図 A.7: 閲覧画面 3

- フレンドリクエストの送信

  ユーザ B がユーザ C(ID_C="C@c.com") とフレンドであったとする．C も図 A.7 のように B の日記に書かれた A の返信を見て ID_A を知ることができた．次に C がフレンドリクエストを A に送信する．図 A.8 はユーザ C がフォームに ID_A を入力してリクエスト送信ボタンを押すことで，ユーザ C から A へのフレンドリクエストが完了したことを示している．

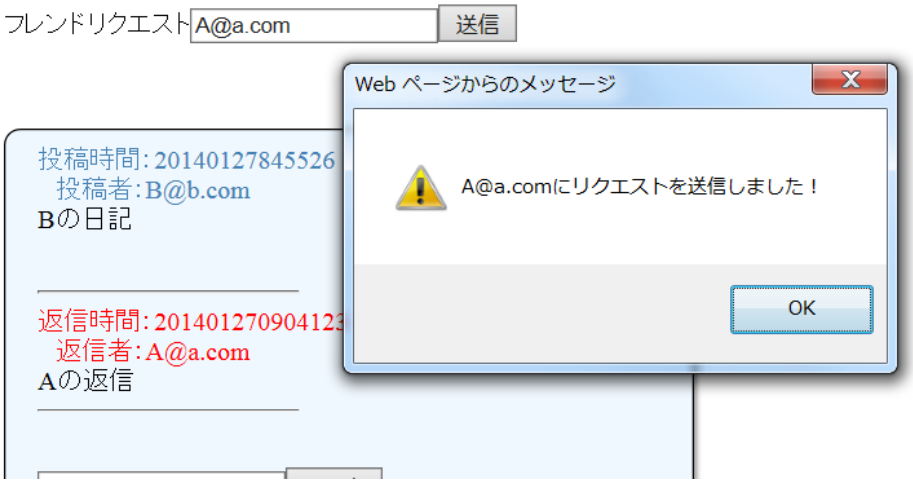


図 A.8: フレンドリクエストの送信例

- フレンドリクエストの承諾

  図 A.9 は，A の画面に C からのフレンドリクエストが来ている通知と承諾ボタン，拒否ボタンが表示され，A が承諾して A と C がフレンドになったことを示している．

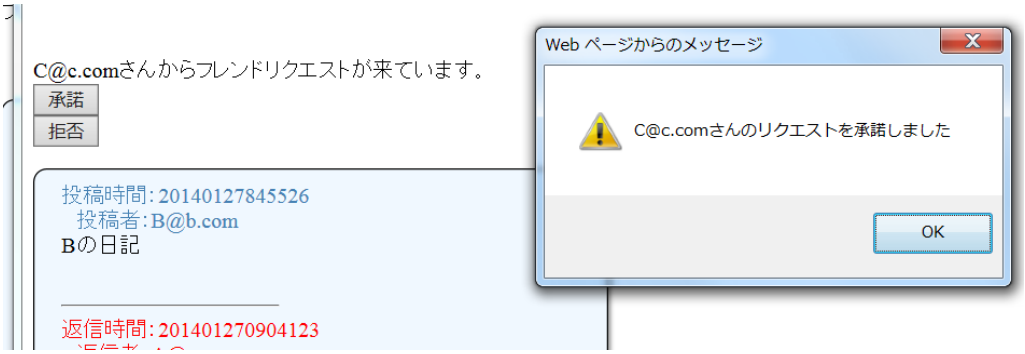


図 A.9: 承諾