社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2005－MBL－32（13）
2005－UBI－　7（13）
2005／3／17

P2P 型情報共有システム -DirectShare-

千 田 陽 介[†]　佐 沢 真 一[†]　佐 藤 裕 一[†]

コンピュータ及びコンピュータネットワークの発達に伴い, 地理的に離れた複数のコンピュータで情報を共有する機会が増えている. 従来, このような目的には, どこかに一つ中心的な役割を担うコンピュータを構築し, 他のコンピュータはそれにアクセスする, いわゆる「クライアント・サーバ型」が一般的であった. しかし, 「クライアント・サーバ型」はその性格上中心的な役割を担うコンピュータ (サーバ) を構築する必要があるため, 情報共有を開始するまでのハードルが高い. そこで筆者らは, ピアツーピア型の情報共有システム “DirectShare” を構築した. ピアツーピアでは, 全てのコンピュータがクライアントであると同時にサーバ的な役割を担う. そのためユーザは気楽に情報共有を行う場を作成し, 情報共有を開始することができる. 本稿では DirectShare の概念と技術要素について述べるものである.

DirectShare: Infomation Shareing System by P2P

Yosuke Senta,[†] Shinichi Sazawa[†] and Yuichi Sato[†]

In this paper, we present the design of DirectShare, a peer-to-peer communication and collaboration tool. In the past technology for these purpose is used to a client-server model needs a server construction, and it is difficult to easily construction of virtual communication space. DirectShare provides fast and scalable data-sharing capabilities by taking full advantage of peer-to-peer technology.

1. はじめに

近年, パーソナルコンピュータ (PC) の高性能化・低価格化に伴い生産活動における PC の役割は益々重要になってきている. さらに磁気ディスクの高容量化により, 編集中の文章はもちろん, 過去に作成したデータ, メールのログなど, あらゆるデータをディスク内に保管し, 生産活動の資料にしている. しかし, PC に保存するデータが多くなればなるほど, 職場用, 自宅用, 出張用など PC を複数所有している場合, 各 PC 間でデータを複製し, 整合性を保つ作業が大変になる. これは一人が複数の PC を管理する時だけでなく, 複数の人間がそれぞれの PC でデータを共有する際や, 新しく PC を買い替えた際にも発生する.

このような問題を解決するため, 従来からデータおよびその処理はサーバで集中管理し, クライアントは表示・入力などの軽い処理を行う Sun Ray や Oracle NC などの “Thin Client” という概念がある[1]. また NFS や MS-Windows の「ネットワークコンピュータ」 (Samba) を用いて, ファイルサーバ上のディスクをあたかもローカル PC のディスクのように振舞わせ, データの一元管理を行う方法もある. しかし, このようなクライアント・サーバモデルでは, 原理的にネットワークに繋がないとデータを取り出すことができない. また, 例えばファイアーウォールに守られた LAN 内のファイルサーバに対し, 移動中の PHS 回線や自宅の ADSL 回線からデータを取り出すことも難しい. さらに, 近くの端末にデータを送る際にもサーバ経由となるため, 回線やサーバ性能を十分高くしておく必要がある.

そこで筆者らは, ピアツーピア (P2P) 型でデータを共有するシステム “DirectShare” を開発した[2),3)]. DirectShare では LAN に繋がっている際にバックグランドで必要なデータを各ローカルディスクにため込むため, ネットワークに繋げない時でも必要なデータを取り出すことができる. また, サーバを介さない P2P 接続のため, 回線に対する負荷も小さい. 一方, 各端末がデータを全てローカルディスクに入れるため,

- DHCP や出張先など接続のたびに IP アドレスが変わる端末に対し, どうやって P2P 接続を行うか.

† 富士通研究所
Fujitsu Laboratories LTD.

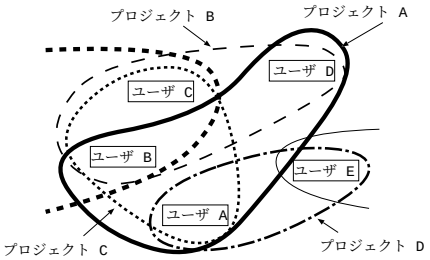

図 1 人の生産活動

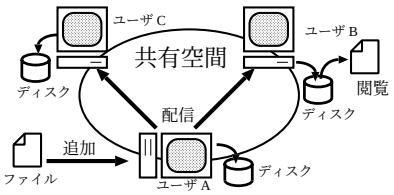

図 2 データの共有

- 勝手なタイミングで LAN に接続し切断する個々の端末の間で, どのようにデータの整合性を保つか.
- データを各ローカルディスクに入れることはセキュリティ上問題にならないか.
- 遠隔地など細い回線からの参加者にとって, 操作性が悪くならないか.

などのクライアント・サーバにはなかった P2P 固有の問題がある. 本稿では, DirectShare の構想とともに既存技術との違いを述べ, 続いてこれらの問題を解く技術的要素について述べていく.

2. 生産活動におけるデータ共有

一般的な人の生産活動では, 図 1 のように複数の人と関わりを持ち, 幾つもの生産活動 (プロジェクト) を並列に行っている. このプロジェクトは時間とともにメンバが入れ替わり, プロジェクトの達成と共に消滅する. DirectShare ではこの生産活動の単位を「共有空間」という概念でとらえ, 円滑に共有空間の管理を行い, 各共有空間の参加者の間で簡単にデータの共有や会議を行うことを目標としている.

データの共有とは, 例えば図 2 において, ユーザ A が必要なファイルを共有空間内に追加すると, ネットワークを介して自動的に他のユーザ (B, C) の端末に配信され, それを閲覧・編集することができるものである. 他のユーザがそのファイルを更新した場合, 同様に自動的に他の端末に配信され, 各ユーザが持つファイルは常に最新の情報に保たれる. この配信はユーザ単位でなく PC 単位で行われるため, 一人が複数の PC を持ち, 同じ共有空間に所属すると, データのバックアップを自動的に行うことができる. 共有されるデータは, 図 2 で示したファイルだけでなく, プロジェクトに関連する URL や写真, プレゼンテーション用のイメージ, 3 次元モデルなど生産活動の内容に応じ様々な形態を取ることができる.

一方, 共有空間のメンバ間で会議を開催する際には, 事前に配布されたデータを会議資料として遠隔地間で, 音声を介しながら同時に資料の参照やマーキング, 資料の編集などを行う. またこの会議において, 共有空間を先生と生徒で共有すれば E-Learning に利用することもできる.

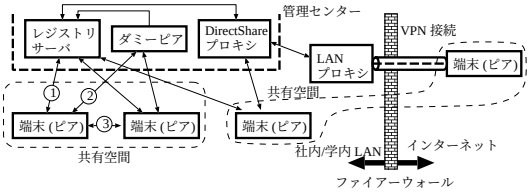

図 3 DirectShare システム全体図

3. DirectShare

3.1 既存技術

このように複数の端末間でデータを共有し, 遠隔会議を行うシステムとして, 既にいくつかの製品やプロジェクトがある. クライアント・サーバ型では Lotus Sametime[4] などが代表的である. しかし, 先ほど述べたように, クライアントサーバ型はネットワークに接続しなくては利用できない. 一方 P2P 接続によるシステムには Groove[5] や gnutella[6], SOBA[7] などがある. Groove については筆者らが 2002 年頃に利用してみた感覚では遠隔会議機能を開始するまでの手順が複雑で使いにくく, E-Learning に利用することが難しい. また gnutella は, サーバを全く使用しないため, ネット上で未知のユーザ (端末) との接続に対しスケーラビリティがない. SOBA は遠隔会議に重点を置いてるため非同期のデータ共有を行うことができない. また現段階ではフレームワークのみで具体的な実装は行われていない.

3.2 概　　要

以上の従来技術を鑑みて, DirectShare では P2P 技術をベースにして, 接続の容易性とデータの同期に重点をおき, データ共有と遠隔会議の両立をめざした. すなわち, ユーザ検索を容易にし, 相手がオフラインでもデータが届き, 必要ならばボタン一つでオンラインの者の間で遠隔会議を開始できるようにした.

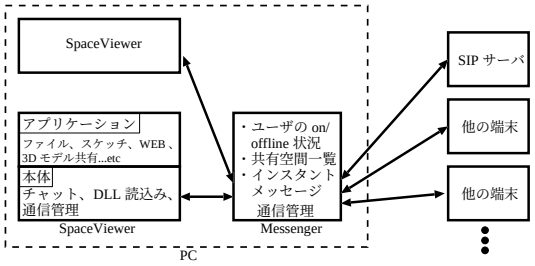

図 4 DirectShare 端末の内部構造

図 3 に DirectShare システムの全体図を示す. 図 3 のように DirectShare は, P2P 接続を行う各端末 (ピア) の他に, レジストリサーバ, ダミーピア, プロキシサーバの三つのサーバによって構成される Hybrid 型 P2P の一種である. このうちレジストリサーバは各端末の IP アドレスを集中管理する役割をなすもので, SIP (Session Initiation Protocol) で動作する. 各端末は起動時にまずレジストリサーバに接続し, 自分がネットワークに接続した (オンラインになった) ことを自分の IP アドレスと共に伝え, P2P 接続したい端末がオンラインかどうか問い合わせる (1). レジストリサーバへの問い合わせの直後, ダミーピアにも接続するが (2), この件に関しては 3.3 節にて詳しく述べるので, ここでは割愛する. レジストリサーバの回答により, 自分が接続したい端末の中でオンラインのものを, その IP アドレスと共に知ることができため, 端末に対し P2P 接続を行う (3). 一方, 自分がオンラインになった時点ではオフラインであったが, その後オンラインになった端末に関しては, 上述した手順によって相手側から P2P 接続をしてくるため, その時点で相手がオンラインになったことが分かる. このようにレジストリサーバはピアの位置情報を伝えるだけの非常に軽い処理を行っているため, クライアント数の増加に対するスケーラビリティが高い.

DirectShare ではレジストリサーバ接続用, P2P 接続用と様々なポートを使っており, そのままではファイアーウォールを越えることができない. 一般の社内/学内 LAN では, 外からのアクセスに対し, LAN プロキシを構築し, HTTP や SMTP, FTP など一般的なプロトコルに関して, VPN でトンネリングして通す. そこで図のように DirectShare 用のプロキシサーバを建て, LAN 外の端末とプロキシ間は HTTP でくるんだ通信を行う. これによりファイアーウォール外の端末とも共有空間の共有が可能になる.

図 4 に DirectShare 端末の内部構成を示す. 図のように DirectShare は Messenger と SpaceViewer という二つのプロセスから構成されている. このうち, Messenger は共有空間の有無に関わらず常に起動しているもので, 保有している共有空間の一覧表示や起動, 他のユーザのオン/オフライン状況の表示, インスタントメッセージや共有空間への招待メッセージの送受信などユーザとのインターフェイスを行う. しかし Messenger の一番重要な役割は他の端末との通信を司ることであり, すべての通信は Messenger を介して行わている.

表 1 DirectShare のアプリケーション

名称	機能
アプリ共有	VNC のように画面イメージの形で遠隔地間で任意のアプリケーションを共有する.
WEB	URL を共有, またスクロールやマーキングの共有も可能
Calendar	スケジュール帳を共有
VPS	VPS[8] を用いて, 遠隔地間で 3D モデルの動作の共有が可能
Sketch	画像イメージを共有, マーキングなどのドロー操作も可能
共有ファイル	ファイルを共有

一方 SpaceViewer は共有空間を表示・編集するもので, 共有空間を起動する度に 1 つのプロセスが動作する. 共有空間の種類に応じて, 共有するアプリケーションを選択できるように, 各アプリケーションは DLL の形をしている. そのため, 各 DLL を読み込み, Messenger 間との通信を行うとともに, チャットなど基底的処理を行う本体部分と, 各アプリケーション用の DLL に分かれている. 現在作成された DLL を表 1 に示す. DLL の形状を取っているため, 簡単に新しいアプリケーションを拡張することが可能である.

3.3 共有空間の同期

P2P 接続では端末 A が共有空間を操作すると, その情報が直接端末 B に送信され端末 B の共有空間に反映される. この際, 端末 A と B が同時に共有空間 (SpaceViewer) を立ち上げていた場合 P2P により直接情報が相手に届くが, 端末 B がその共有空間を立ち上げてなかったりオフラインだった場合, そのままでは端末 A の操作が端末 B に反映されず共有空間の不一致が生じてしまう.

そこで DirectShare では操作情報のバッファリングを行っている. 図 5 (1) は端末 B が Messenger のみ起動し, SpaceViewer を起動していない時の処理である. 図中 (a) において端末 A が共有空間を操作すると, その情報 (パケット) は端末 B に届く. しかし端

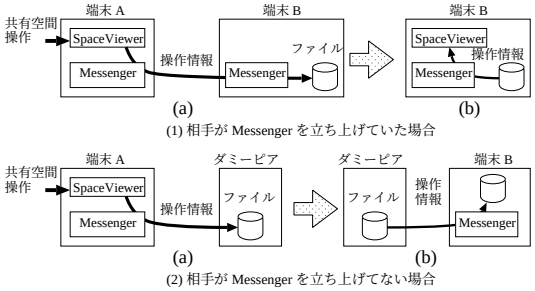

図 5 共有空間の同期

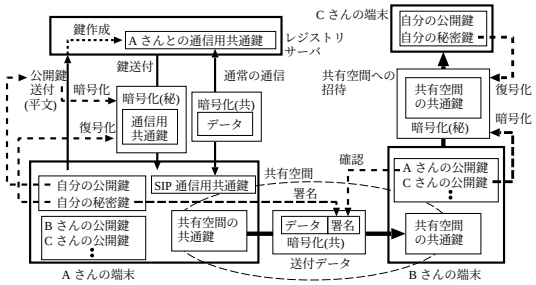

図 6 DirectShare におけるセキュリティ対策

末 B では SpaceViewer を起動してないため, その情報を解釈して共有空間に反映することができないため, そのままファイルに書き込む. 書き込まれた情報は図中 (b) において端末 B が共有空間を起動した際に読み込まれ, ファイルからパケットを一つずつ取りだし, 端末 A の状態に関わらず, それがあたかも今受信されたかのように処理を行い共有空間を更新する. ここでパケットの処理を Messenger でなく SpaceViewer で行うのは, パケットの処理は表 1 で示したアプリケーション依存であるため, Messenger で行うことは拡張性等の理由により難しいためである. この操作により端末 B が持つ共有空間は端末 A と一致することができる.

さらに端末 B がオフラインの場合, 図 5 (2) のように外部に 24 時間稼働し, 代理で情報を受けとる「ダミーピア」を用いて共有空間の同期を行っている. 図中 (a) において端末 A が共有空間を操作するとその情報は端末 B の代わりにダミーピアに送られ, 保存される. その後, 端末 B がオンラインになると, (b) のようにダミーピアから自分宛のデータをローカルディスクのファイルに転送する. その後は先ほどと同様に, SpaceViewer を起動した時点でパケットの処理を行い, 共有空間の同期を行う.

図 3 ではダミーピアは管理センターに一つ構築している. しかしすべての端末がこのダミーピアに接続すると負荷が集中してしまう恐れがあるため, 各自が自分専用もしくはグループ単位のダミーピアを簡単に構築できるようにしている. これは, 各端末はオフライン時に代理でパケットを受けとる端末をあらかじめレジストリサーバに登録しておき, 別の端末が該当端末の IP アドレスをレジストリサーバに問い合わせた際, オフラインであると同時に該当端末のダミーピアの IP アドレスを通知することで実現する.

このように自分専用のダミーピアを置くことで, ユーザはダミーピアとの通信が速くなる利点があり, またシステムにとっては管理センターのダミーピアの負荷を下げることができる.

3.4 セキュリティ

DirectShare はデータが各端末 (PC) にデータが蓄えられるため, PC が盗難にあった際, データを盗まれないようにする必要がある. さらに, 通信経路における盗聴や成り済ましなど対策もしなければならない. そのため DirectShare では図 6 のように公開鍵 (RSA 1024bit), 共通鍵 (Rijndael 128bit) の二つの暗号方式を用いてセキュリティ対策を行っている.

鍵を二種類使っているのは公開鍵方式の暗号化は暗号化・復号化に時間がかかるためで, 音声や共有空間の編集などの通信やディスクへの保存など普段の暗号化は高速な共通鍵方式で処理している. この共通鍵は共有空間毎に異なるため, 第三者を新しく共有空間に招待する時に共通鍵を安全に伝える必要があり, そこに公開鍵方式を利用している. また図のように公開鍵は, 成り済ましを防ぐための署名付けや, レジストリサーバとの通信に用いる共通鍵の交換などにも使われており, レジストリサーバの付随機能で管理されている.

一般にシステムの安全性と使い勝手はトレードオフの関係があり, 安全のためセキュリティを高めすぎると使い勝手が悪くなってしまう. そこで DirectShare では, 共有空間毎に選択的に 3 段階のレベルを設け, 共有する情報の重要度で使い分けができるようにした. 1 番低いセキュリティレベルでは, 単に送信データやディスクへの保存を暗号化するだけであり, 盗聴や成り済ましを防止しているが, DirectShare 内のデータをアプリケーションで開き「別名保存」を行うことで, 利用者は容易にデータを外に持ち出すことができる. そこで 2 番目のレベルでは, この別名保存を禁止し DirectShare 内のデータを外部デバイス等に保存し持ち出せないようにしている. さらに一番高いレベルでは, カットアンドペースト機能や印刷機能なども禁止しデータを完全に DirectShare の外に持ち出すことはできないようにしている. しかし, この方法でも例えばカメラを使ってモニタ画面を撮影することを防ぐこ

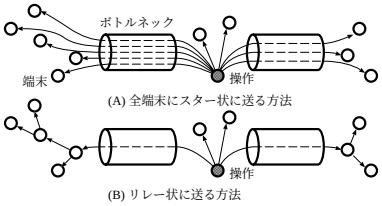

図 7 共有空間の編集情報を分配

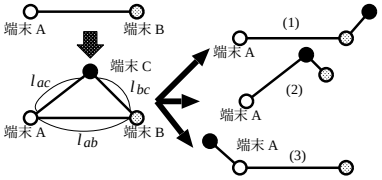

図 8 ハブ端末の検出

とはできない. さらに, ネットワークに繋がってなくても共有空間内の情報を参照できるように, 鍵は PC 本体の中に特殊な形でしまっているが, これは利便性を高める反面, 十分な実力と悪意を持った盗難者によって解読される恐れもある. このような問題を解決するため高機密な共有空間では, ユーザ情報を薄くモニタ画面に表示して撮影を牽制する, 指紋等の認証機能を入れ鍵の一部にするなどの対策を考えている.

4. データ配信

4.1 準最適ルートの検出

複数の端末が同一の共有空間に属している場合, その空間への編集情報は各端末に伝達される. その際, 図 7 (A) のように全端末にスター状に送るより (B) のように適当な端末をハブ化してリレー的に送る方が効率がよい. 特に図のように途中にボトルネックがある場合 (B) の方式は有効となる. この経路検索はセールスマン巡回問題の一種と考えることができる. しかし一般的な巡回問題は全端末 (全都市) の接続状況を把握し最適解を求めるものに対し, 本問題はハブ化している端末も, 誰かの端末のため途中でサインオフして消えることや, 誰かが新しくサインオンすることなどダイナミックに接続状況が変わる. そのため, 時間をかけて最適解を求める従来の巡回問題の解法は不向きとなり, 準最適解を高速に求める必要がある. また, 似た考えに Supernode[9] があるが, この場合 Supernode (ハブ端末) が予め固定されていることに違いがある.

その解法の概念を図 8 に示す. 端末の接続は必ず順次行われるため, 最初はまず二つの端末 A, B のみ接続している. この場合端末 A にとって B はハブ端末と考えることができる. そこに新たな端末 C が加わると, 既存の端末 A, B は C とのネットワーク上の距離を測定し互いに伝え合う. ここでネットワーク上の距離とは, 回線速度による指標値で ping の戻り値や, 適当な大きさのパケットを転送した際の時間などから算出したものである. AB, AC 間の距離をそれぞれ l_{ab}, l_{ac}, BC 間の距離を l_{bc} , $r = (l_{ab}^2 + l_{ac}^2 - l_{bc}^2)/2l_{ab}l_{ac}$ とすると, $r \geq R$ (R は適当な 1 以下の正数: 後述), かつ $l_{ac} > l_{ab}$ の時には端末 C をハブ端末 B の配下にし (1), $r \geq R$ かつ $l_{ab} > l_{ac}$ の時には端末 C を新たなハブ端末として端末 B は C の配下にする (2). 一方, $r < R$ の時には端末 C は B とは別の孤立したハブ端末にする (3).

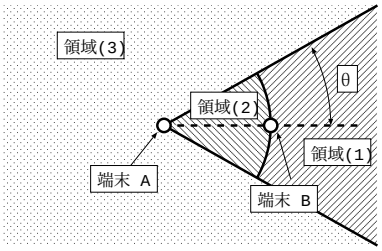

図 9 新しい端末の存在領域

ネットワーク距離を物理的な距離と考えた場合, これら (1),(2),(3) の領域と端末 A, B との位置関係を図 9 に示す. この図において $\cos\theta = R$ が成り立つ.

3 個目以降の端末が接続して来た場合, 既存のハブ端末の中から近い順に距離を比較し, 同様に既存のハブ端末の配下に入れる (1), 既存のハブ端末と入れ換える (2) かどうか判断し, 全てのハブ端末に対し条件が満たされなかった場合新しいハブ端末にする (3). このように距離の比較は全端末でなくハブ端末に対してのみ行うため, 計算コストが小さくてすむ.

各端末は全端末がそれぞれどのハブ端末に属しているかというルーティング情報を持ち, パケットを送付する際は, 各ハブ端末に対し宛先とともにパケットを送付する. パケットを受けとったハブ端末側でも同様のルーティング情報を持っているため, その情報に基づき自分宛以外のパケットは各ハブ端末に転送する. ただし, このルーティング情報は個々の端末が勝手に作るため必ずしも一致しない. そのためハブ端末でない端末への転送パケットが流れてくる可能性もある. その際はハブ端末でなく直接宛先に届けることでパケットのループを阻止している.

このアルゴリズムの効果を確認するためシミュレーションを行った. シミュレーションでは, x-y 平面上のランダムな位置に端末を 30 個発生させ, 端末間の距離をネットワーク上の距離とみなした. また, 各端末

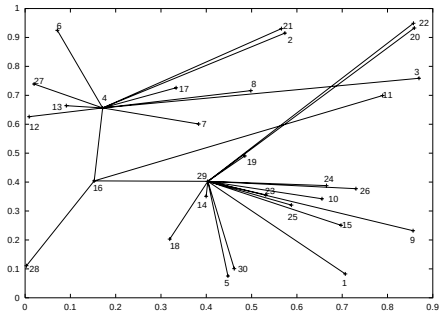

図 10 シミュレーション結果 (端末 16 のルーティング情報)

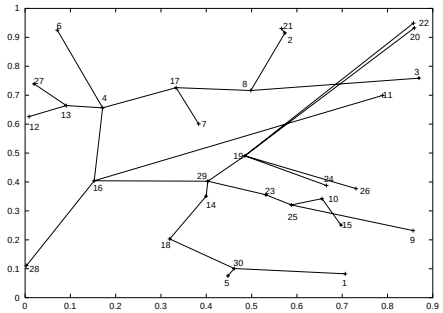

図 11 シミュレーション結果 (端末 16 が発信した情報の流れ)

は発生順に 1, 2, 3 と番号を振り, 接続は番号の若い順に行われていくものとした. R の値は 0.6 ($\theta = 53°$) を用いた.

図 10 に端末 16 が持つルーティング情報, 図 11 に端末 16 が情報を発信した時の経路図を示す. 図 10 において, 端末 3, 11, 20, 22 のルーティング情報が不自然な交差が発生していることが観察される. その影響で図 11 において最適な経路になっていない.

これは, 若い端末 2〜4 が図の上半分にあるため, 端末 3 はまず端末 2 続いて端末 4 の配下になったこと, 端末 11 は出現時, 右下の端末群 (1, 9, 10) が最下部の端末 5 をハブ端末にしており, 条件が満たされず独立したハブ端末になったこと, 一方端末 20,21 が出現した頃は, 端末 5 より上の方に位置する端末 14 がハブ端末となっていたため, 条件が満たされその配下になったことが原因である.

このように本アルゴリズムは完全な最適解は得られず, 接続した順番によって解が左右される. しかし本問題はアルゴリズムを改良しても, 例えば無線 LAN のローミング機能を使って端末が接続したまま移動しても発生する. そこで, アルゴリズムは改良せず, 各端末は負荷が大きくならない頻度で, 配下に端末を持つハブ端末以外の端末がランダムに再接続されたとみなし, 経路の再計算を行うようにした. また, ハブ端末がオフラインになった場合, 配下の端末がすべて再接続されたとみなし, 同様の処理を行う. このような処理を繰り返すことで, 次第に最適へと変移する.

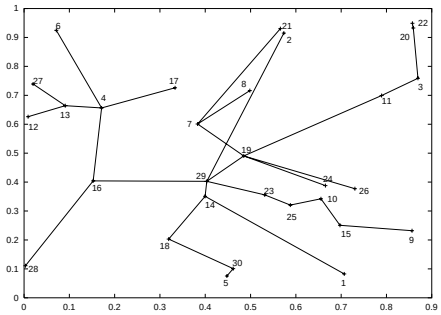

図 12 各端末が再接続した際の情報の流れ (端末 16 発信)

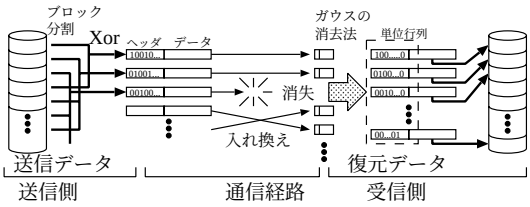

図 13 UDP + FEC による高速転送

図 12 に各端末が 1 から順に再接続された際の端末 16 の発信経路を示す.

4.2 高速転送アルゴリズム

図 7 において, 上述した経路の最適化のみならずボトルネック自体の高速化も望まれている. このようなボトルネックの原因は RTT (Round Trip Time) が大きい, パケット消失率が高いなどの回線品質によるものが多い.

一般的な TCP 通信はこのような低品質の回線では十分な速度を出すことができない. これは, 送信側は受信側からの応答 (ACK) が届かないまま送信できるデータ量がウィンドウサイズで制限されていることや, ACK が届かないことでパケット消失を検知し, 消失した所から再送を行う構造上の理由である. そこで ACK を用いない UDP でパケットを連続的に送付し, パケットロスを FEC (Forward Error Correction) で修復することにより, 高速にデータ転送を行うようにした. 同様の方法に Digital Fountain 社の Raptor Codes[10] があるが, これは繰り返し演算を行って FEC を実現している性質上ロジックが複雑でメモリ消費量が大きいという欠点があり, P2P 接続のため各端末に組み込むには適さない.

そこで, 筆者らは簡単な計算と少ないメモリ量で高速に符号化・復号化できる FEC を開発した. その概

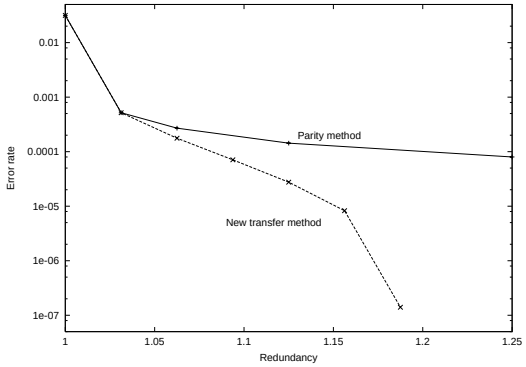

図 14 従来の方式と本方式の比較 ($n = 32$)

念を図 13 に示す. 図のように送信側ではデータを細かなブロックに分割し, 適当なブロックの間で XOR 演算を行う. そして XOR 演算したブロックを 1, 残りを 0 にしたヘッダを添えて送信する. 受信側ではパケットを蓄え, ヘッダに対しガウスの消去法を適用することで元データを復号する. この方法は数 % データを余分に送ることで, 経路上でパケットが幾つか喪失したとしても, 受信側で ヘッダの逆行列が得られると元データを修復することができる.

XOR を用いたエラー修正には n パケット毎に XOR によるチェックサムを 1 パケット添える方法があり, 音声や画像配信に使われている. この方法は $n+1$ パケット中に 2 個以上パケット消失が発生すると修復できない. パケット消失率 R とすると, m パケットが全て転送できる確率 P_1 は次式で表すことができる.

$$P_1 = \{(1-R)^{n+1} + (n+1)R(1-R)^n\}^{\frac{m}{n}} \quad (1)$$

一方本方式に対するエラー耐性 P_2 は, ブロック分割数を n, 送付パケット数を m とすると,

$$P_2 = \sum_{k=0}^{m-n} ({}_mC_k - G_{m,k})(1-R)^{m-k}R^k \quad (2)$$

となる. ここで ${}_mC_k$ は組合せの総数, $G_{m,k}$ は 0 と 1 で構成された $n \times m$ 行列の列を k 個を抜き取って作られた行列から $n \times n$ の単位行列を計算できない組合せの数である.

図 14 に $R = 0.1\%$, 分割数 $n = 32$ の時の従来の方式と本方式の冗長度とエラー対性の比較図を示す. 用いた行列は, $G_{m,k}$ の値が, k が小さい時できるだけ小さくなるものを選んだ. 表 2 に選出した行列の $G_{m,k}$ を示す. 図より冗長度が大きくなるにつれ本方式の優位性が観察できる.

このようにして, 送信側で本アルゴリズムにて符合

表 2 計算に用いた行列

m	k				
	0	1	2	3	4
32	0	-	-	-	-
33	0	0	-	-	-
34	0	0	176	-	-
35	0	0	70	3045	-
36	0	0	27	1386	31537
37	0	0	8	539	15321
38	0	0	0	139	6083

化したパケット群を UDP で送り受信側で復号する. 復号不可能な程パケットが消失した場合は再送を行うがその頻度は TCP と比べて少なく高速にデータを送ることができる. しかし UDP は経路の回線性能以上の速度でパケットを送り続けると, 回線性能以上の分はそのままパケット消失に繋がる. そのため, 回線性能以下の速度でパケットを送る必要がある.

一番簡単な方法は, あらかじめ何らかの方法で回線性能を知り, その性能以下の間隔でパケットを送付するようにチューニングすることである. しかしこの方法は, 受信先がどこか分からない P2P 接続に用いることができない. また, 送付前に一回テストパケットを送り回線性能を測定する方法もあるが, 転送中に同じ回線で別の通信が行われると, 輻輳が生じ復号不可能な程パケットが消失してしまう. 特にその通信が TCP だった場合, TCP の方は輻輳制御が働き, 転送速度を下げるのに対し, 本転送はまったく下げないため協調性に欠ける.

そこで UDP 転送にも輻輳制御を取り入れ, TCP をはじめとする他の通信との親和性を高める必要がある. TCP との親和性を高める “TCP-Friendly” な通信方法として様々な方法が提案されている[11]. それらはいずれも TCP の挙動を数式化しパケット消失が生じた際の転送レートの変化をシミュレーションする, TCP プロトコルをアプリケーション層で実装し TCP の動きそのものを実現するなど, TCP と同じ動きを行うことで TCP との親和性を高めている. すなわち, これらの方法は TCP と同じくパケット消失が発生した段階で転送速度を変えるが, その段階では既に多くのデータを送信した後のため, 送信を終えたデータの多くが消失してしまう.

そこで筆者らはパケット消失が始まる前にパケット消失が発生する兆候をつかみ転送速度を変える方法を構築した. 図 15 (A) はパケットを受けとる度に ACK を返す端末に対し, 1500byte のパケットをまず一つ送り, 以降 ACK を受けとる度に新しいパケットを二つ送る処理を行った際の, RTT の変化とパケット消失を

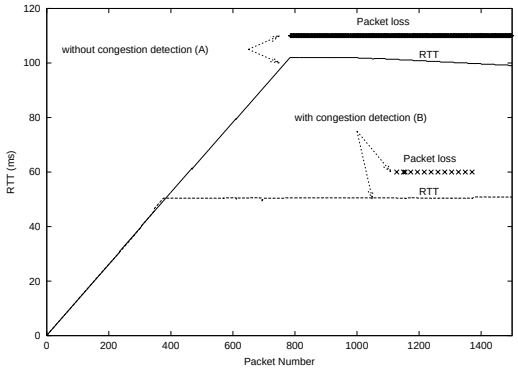

図 15 RTT とパケット消失の関係

測定したものである.

パケットを指数関数的に増加させているため, 転送速度はパケットを追う毎に速くなり, 後半パケットが大量に消失している. ところが, パケット消失が発生する前に RTT が増加していることも観測できる.

これは途中のルータやネットワークカードにバッファがあり, はじめ回線性能以上のデータを流しても, キューイングされ逐次処理したためだと考えられる. そこで ACK を受けとる度に RTT を測定し, その RTT がそれまでの通信によって得られた最小値とある閾値の和よりも小さければ, 新しいパケットを二つ, 大きければ一つ送るようにした. その時の結果を図 15 (B) に示す (閾値 40ms). 図よりパケット消失の発生がかなり押えられていることが分かる.

図 16 に RTT 及びパケット消失率に対する TCP と本方式の差を示す. 回線品質は帯域制御ソフト: Dummynet[12] を用い, 測定は二回の平均値を取った. この図より RTT が大きくなる/パケット消失率が高くなるなど, 回線品質が悪くなれば悪くなるほど TCP に比べ本方式が優位に立つことが分かる. また実際に日米間 (RTT: 約 200ms) で測定したところ最大で TCP の 7 倍という結果が得られた.

5. おわりに

P2P コミュニケーションシステム DirectShare について, その構想と技術要素について述べた. DirectShare では, 個人のコンピュータ環境に必要なのはコンピュータではなく, 付随するデータと人のネットワークであるという考えから, それを共有空間という概念で捉え, 簡単に端末間で共有空間のミラーリングを行い, データ共有と遠隔会議を行うことができる. 今後はサーバの多重化などによる堅牢性の増加, 異なる企業/学内など互いにファイヤーウォールに守られた別

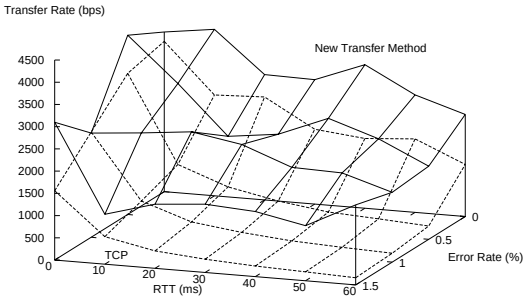

図 16 TCP との比較

の LAN 同士での共有などを行っていく. 特にファイヤーウォール越えは, 現状の HTTP にくるんだ通信は LAN のネットワークポリシーを変更しないで実現したものであるが, DirectShare 専用のプロキシを DMZ に設け, 異なる LAN の中の端末間でも P2P 通信を実現する方法を考えている.

また, 輻輳制御において現在, パケットを一つ送るか二つ送るかを判断する閾値は, 経験によって決めた固定値を用いているが, これを動的に変えるアルゴリズムを開発することも今後の課題である.

参 考 文 献

1) J.Golick, “Network computing in the new thin-client age”, netWorker: The Craft of Network Comptuting 3(1), pp30-40 (1999)
2) 千田ほか, “DirectShare によるコンピュータ作業環境の携帯とデータ共有”, 情報処理学会ユビキタスコンピューティングシステム研究会, pp159-164 (2003)
3) S.Sazawa et el. “Ubiquitous Working Enviroment by DirectShare”, IFAC 2004
4) `http://www.lotus.com/`
5) `http://www.groove.net/`
6) Andy Oram,“PEER-TO-PEER Harnessing the Power of Disruptive Technologies”, O’Reilly & Associates, INC (2001).
7) `http://www.soba-project.org/`
8) `http://salesgroup.fujitsu.com/plm/vps/`
9) Zhiyong Xu et el. “SBARC: A Supernode Based Peer-to-Peer File Sharing System”, the 8th IEEE Symp. onComputers and Communications, pp.1053-1058, 2003
10) A. Shokrollahi, “Raptor Codes”, Digital Fountain Technical Report, DF2003–06–001, 2003
11) Jörg Widmer et el. “A Survey on TCP-Friendly Congestion Control”, Special Issue of the IEEE Network Magazine “Control of Best Effort Traffic” 15(3), pp.28-37, May/June 2001
12) `http://info.iet.unipi.it/~lugi/ip_dummynet/`