2005－DSM－36（7）
2005／3／18

セッション層の設計に基づいたマルチメディア共有環境の構築

藤村　基†　藤村　真生†　久津輪　敏郎†

近年，パソコンとブロードバンド環境の普及により，様々なマルチメディア共有環境が出現している．これらは互いに違ったシステムを用いているため相互運用が不可能であることが多い．コミュニケーション手段としては相互運用やシステムの統一は必要不可欠である．また，複数のメディアを使用するために，同一ホストと通信するにも関わらず複数の接続を設ける必要がある場合がある．これはセッション層プロトコルが曖昧で，規格化されていないためである．そこで今回は，マルチメディア共有環境における基盤システムとして，ホスト間の接続を単一とする通信の一括管理を目的としたセッション層プロトコルを提案し，その有効性について検証した．

Construction of the multimedia communication system based on an effective protocol for the session layer protocol

Motoi FUJIMURA†　Masao FUJIMURA†　Toshiro KUTSUWA†

This research proposes an effective protocol for the session layer. It makes one and only connection between couples of hosts to manages various multimedia communications collectively. Recently, the various multimedia communication systems appeared by the spread of personal computers and broadband communication network. As the communication means, interoperating and the union of systems are necessary and indispensable. However, interoperating of these systems is impossible, because those systems are a mutually different. Moreover, these systems must have several connections for using several media, though these communicate between the same hosts. The reason for these problems is due to vague and not standardized the session layer protocol.

1.はじめに

近年，パソコンが世帯に1台から個人に1台へと急速に普及している．処理速度も向上しており，音声や動画をパソコン上で視聴することが一般的に行える高性能なパソコンが普及しつつある．また，インターネットが広く普及している．さらに，DSLや光ファイバーなどのブロードバンド通信網の普及が進んでおり，大容量のデータ通信も盛んに行われている．これにより容量の大きい音声や動画を用いたビデオ会議システムや，チャットツールが数多く出現[1)2)3)]している．ネットワークを介したゲームも多く出現し，社会現象的に普及している．

† 大阪工業大学大学院
Osaka Institute of Technology

ところが，これらのシステムはシステム同士の相互運用が考慮されていない．また，同一ホストと通信を行うにも関わらず，コミュニケーションに利用する音声，テキスト，ファイル転送といったメディアごとに通信の接続を設ける必要がある場合がある．これにより，不正アクセス防止のために設けているファイヤーウォールを解除しなければならない．危険な状態でシステムを利用することとなり，コンピュータウィルスなどの攻撃の対象となっている．

これらの問題はアプリケーションが直接利用するセッション層プロトコルが曖昧で，規格化されていないことにある．そこで今回は，マルチメディア共有環境における基盤システムとして，ホスト間の接続を単一とする通信の一括管理を目的としたセッション層プロトコルを提案し，その有効性につい

て検証した. これにより, 様々なアプリケーションが統一プロトコルを利用して通信を行うことが可能になるものと考えられる.

2.セッション層プロトコルの設計

2.1 設計

セッション層プロトコルを設計するさいに以下の3点を考慮し, 設計目標とした.

①今後のニーズの変化に対応できる拡張性
②アプリケーションが自由に選択できる柔軟性
③様々な環境に順応できる可搬性

この目標を基に設計したプロトコルを以下に説明する.

2.2 通信データ構造

データの表現はプレゼンテーション層で規約されるものである. しかし, セッション層の構造を説明するにあたっては予めデータ構造を想定しておかなければならない. ここでは, 以下に規約する構造を用いて説明する. また, パケットという言葉は第3層で使用するPDU(Protocol Data Unit)のことを意味する. しかし, 一般的にデータの固まりをパケットと呼んでいることから, わかりやすくするために通信データのことをパケットと呼ぶ.

通信に使用するデータは UTF-8 エンコーディングされた文字データとする. UTF-8 はファイルシステムに影響のない形式で1文字を1バイト～4バイトで表す [5].

文字列は今後の拡張を考慮し, 現在最も拡張性と可搬性があると考えられる XML(Extensible Markup Language)フォーマットとする. XML は, 汎用的なデータ記述言語として標準的なデータ記述フォーマットを提供するものである. [4]

パケットの構成は図 2.1 のように定義する. 尚,図中の改行及びインデントは見易さを考慮したものであり実際は1行で扱う. 1行目はXML宣言部であり, XML のバージョンや文字コードを記述する. パケット全体を<Packet>とする. セッション層でパケットを制御するために必要な情報を<Head>内に収め, 以下ではパケットヘッダ部と呼ぶ. また, アプリケーションが必要とするデータを<Data>内に収めパケットデータ部と呼ぶ. XML パーサが解釈できないパケットは違反とし破棄する.

パケットヘッダ部内は図 2.2 に示すように<From>,<To>,<Group>,<Data>の 4 つで構成することとした.

```
<?xml version="1.0" encoding="UTF-8"?>
<Packet>
        <Head>パケットヘッダ部</Head>
        <Data>パケットデータ部</Data>
</Packet>
```

図 2.1 パケットの構成

```
<Head>
        <From>送信元ID</From>
        <To>宛て先ID</To>
        <Group>グループID</Group>
        <App>アプリケーション名</App>
</Head>
```

図 2.2 パケットヘッダ部

送信元クライアントを識別する<From>は返信相手の特定に用いる. 宛て先クライアントを識別する<To>は配送するクライアントを特定するために必要である. 参加しているグループを識別する<Group>は, ブロードキャストするグループを指定する. <App>は, <Data>に収められたデータを使用するアプリケーションを指定し, 上位層に橋渡しするさいに用いられる.

2.3 ダイアログ制御

制御方法はアプリケーションの仕様に依存する. パケットを受信したホストはパケットヘッダの<App>の情報を元に配送先を決定する. 完全なサーバークライアントモデルの場合はサーバが判断を行い, クライアント同士が P2P 環境を構成している場合は各クライアントが判断することになる. 本研究ではメディアごとに独立してダイアログ制御ができるように設計した. また, ホスト同士をつなぐトポロ

ジはブロードキャスト型とリング型を用意し，このどちらを使用するかについても自由に組み合わせて利便性をはかることとした．

2.3.1 ブロードキャスト型(双方向同時通信)

図 2.3 は双方向同時通信を行うブロードキャスト型の概念図である．クライアントが送信したパケットをグループ内のすべてのクライアントに送信する．グループ内のすべてのクライアントが常に送信権を持ち，特に制限のないときはパケットを送信可能であるとする．

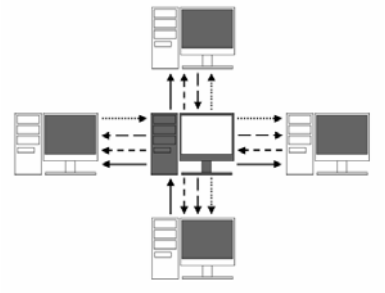

図 2.3 ブロードキャスト型

2.3.2 リング型(双方向交互通信)

図 2.4 は双方向交互通信を行うリング型の概念図である．ブロードキャストとは違い，送信権を持つクライアントのみがパケットを送信できる．パケットがグループ内の各クライアントを巡回することで全クライアントに情報を伝える．送信権を獲得したホストはヘッダ情報とアプリケーションの仕様によって次のホストへパケットを送信する．

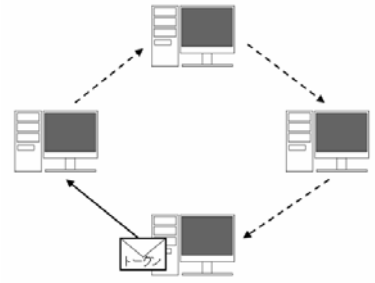

図 2.4 リング型

2.4 ダイアログ分割

2.4.1 開始シーケンス

開始シーケンスは通信開始の準備段階を規定する．アプリケーションに依存する部分が多く，ここではガイドラインを規定する．認証フェーズと初期化フェーズの2段階に分かれる．

ユーザの認証を行う認証フェーズでは，接続してきたホストに対してサーバから情報要求し，システムとの適合をチェックする．

初期化フェーズでは，接続してきたホストのアプリケーションのオープンや，コミュニケーション開始時の初期値を設定させる．

2.4.2 終了シーケンス

通信を切断する場合，切断を要求するホストが切断要求を送信する．その間，要求したホストは待機する．切断要求を受信したホストはアプリケーションを切断状態にする．切断の準備が完了次第切断応答を送信元に返信する．返信を受けると要求したホストが切断準備をする．その後，要求側がセッションを切断する．

3.マルチメディア共有環境

3.1 マルチメディア共有環境

2 で提案したプロトコルの有効性を確認するために，複数のメディアを使ってコミュニケーションを行えるマルチメディア共有環境を構築した．使用するメディアは，テキスト，キャンバス，3 次元形状，音声である．それぞれは独立したアプリケーションである．

すべてのアプリケーションは Java 言語を用いて実装した.Java 言語を用いた理由は，オブジェクト指向言語であり，後の拡張時に再利用を期待できることである．さらに，OS に依存しにくいアプリケーションを作成できることである．また，Java は XML との相性がよく，簡単に扱えるからである．よって，今回作成したシステムは Java に依存する．

3.2 XML 規格

4つのメディアは以下に規定する統一規格に則ったプレゼンテーション層プロトコルを使用する．Java 言語には，プリミティブ型とオブジェクト型がある．

プリミティブ型の XML 記述仕様を図 3.1 に示す. プリミティブ型とは基本データ型のことで, 整数を表す long(8 バイト), int(4 バイト), short(2 バイト), byte(1 バイト), 実数を表す double(8 バイト), float(4 バイト), 文字1字を表す char(2 バイト), 真偽を表す boolean(2 バイト)がある.

```
<プリミティブ型名>数値</プリミティブ型名>
```

図 3.1 プリミティブ型のXML仕様

オブジェクト型とは, 各クラスのインスタンスのことである. オブジェクト型は, 型名とパラメータ名で構成する. これを図 3.2 に示す. パラメータ名要素の内容として数値を挿入する. パラメータがオブジェクトの場合はオブジェクト名要素を挿入する.

この他に, プリミティブ型配列, オブジェクト型配列についての仕様を定めている. また, 例外となる Vector クラス, String クラスについても別途仕様を定めた.

また, この規格をもとに汎用性を考慮した解析機構を構築した.

3.3 サーバ

サーバアプリケーションは, セッション管理, クライアント管理, トポロジ管理, グループ管理, 開始シーケンスを行う.

セッション管理とは, 実際にパケットの送受信 [6] を実行すること, 及びクライアント情報を保持することである. また, パケットヘッダ部を解析し配送先を特定することを担う.

クライアント管理は, クライアント同士の連携を取ることである. クライアントを動的配列に格納してグループを形成する. 使用しているメディアの情報や所属しているクライアントの情報を参照できる. クライアントの追加・削除を行う.

トポロジ管理は, 様々なトポロジに対応しクライアントからのパケット配送方式を決定する. また, サーバを経由しないクライアント間の経路をクライアントに制御させる. 今回は3つのトポロジに対応させた. ブロードキャスト型スタートポロジ, サーバ経由型リングトポロジ, P2P型リングトポロジである. 以下で詳しく説明する.

グループ管理とは, グループの追加, 削除を行う. グループも動的配列に格納して管理する. また, グループ間の通信を可能にし, アクセスしているすべてのクライアントにパケットを送ることができる.

3.3.1 ブロードキャスト型スタートポロジ

各クライアントは, 図 3.3 のようにサーバにのみ

```
<オブジェクト名1>
        <パラメータ名1>
                <プリミティブ型名1>数値1</プリミティブ型名1>
        </パラメータ名1>
        <パラメータ名2>
                <オブジェクト名2>
                        <パラメータ名3>
                                <プリミティブ型名2>数値2</プリミティブ型名2>
                        </パラメータ名3>
                </オブジェクト名2>
        </パラメータ名2>
</オブジェクト名1>
```

図 3.2 オブジェクト型の XML 仕様

接続して通信を行う. パケットを受信したサーバは動的配列に格納されたクライアントのうち<From>の ID が一致しないすべてのクライアントにパケットを送信する.

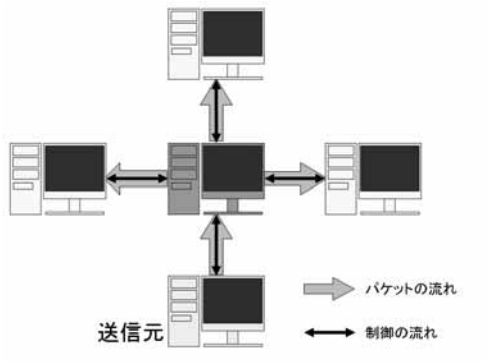

図 3.3 ブロードキャスト型スタートポロジ

3.3.2 サーバ経由型リングトポロジ

各クライアントはブロードキャスト型スタートポロジと同様にサーバに接続する. パケットはサーバが巡回させる構造を持ち, 図 3.4 のように巡回し仮想的にリング構造を構成する. パケットを受信したサーバは, 動的配列の中から<From>ID の一致するクライアントを探し, そのクライアントの次の要素のクライアントにパケットを送信する. 一致したクライアントが最後の要素に格納されている場合は, 先頭要素のクライアントに送信する.

1度サーバを経由しパケットヘッダ情報を読み込むため, 遅延を免れない. しかし, 構造が単純であるため管理しやすく, エラーを回避しやすいメリットがある.

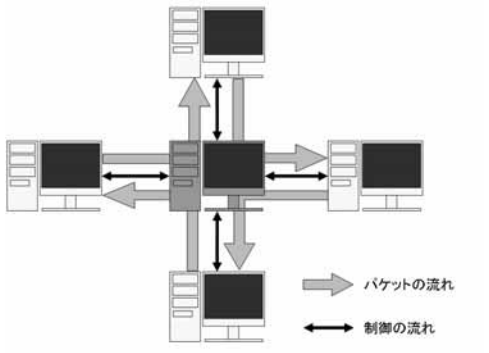

図 3.4 サーバ経由型リングトポロジ

3.3.3 P2P 型リングトポロジ

図 3.5 のように各クライアント同士が直接通信接続を行う. サーバは, クライアント間の通信接続の接続, 切断の指揮をとる. セッションが増えることで管理が難しく, 1度に多くの例外が発生する可能性を無視できない. 今回は, トポロジを形成することに重点を置いて構築した. クライアントが開始シーケンス終了後にリング接続シーケンスの開始を要求する. また, 切断する場合はリング切断シーケンスの開始を要求する. また, P2Pの経路が切断状態のときはサーバ経由でパケットを巡回させるものとする.

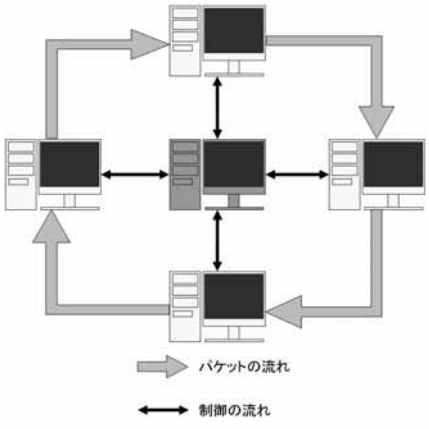

図 3.5 P2P 型リングトポロジ

3.4 メディア

3.4.1 テキスト

世界中で最も多く利用されていると考えられる形式である. データ量が小さいため狭帯域の環境でも対応できる.

入力された文字列に発言者の名前を付与することで発言者を特定し会話を進める. 発言はすべて双方向同時通信でブロードキャストする.

3.4.2 キャンバス

画面上にマウスを用いて絵を描くアプリケーションである. 単体で用いるのではなく会話の補助的な役目を持つと考えられる. また, 多人数でコラボレーションをして楽しむ使用法も考えられる.

ユーザが1つのオブジェクトを描き終わると描画情報を送信する. 描画情報はすべて双方向同時通信でブロードキャストする. このようにして2次元空間を共有する.

3.4.3 3 次元形状

このアプリケーションは, 予め作られた多角形をスイープして立体にする. この立体は, マウスクリックによって頂点を追加 [7]できる. これらの操作により立体作成のコラボレーションを行うものである.

3 次元空間を表現する Java3DAPI を用いた.

3.4.4 音声

我々が日常のコミュニケーションで自然に使っているメディアである. しかしながら, 音声のデータ量が大きいため高速な通信網を確保する必要がある. それぞれのクライアントがデータをブロードキャストすると帯域を大幅に消費する. さらに, 通信接続を1つに限定するための多人数の細切れ音声を1つずつ再生することになる. そのため, 音声の連続性や時間の同期を取ることができない. また, サーバが受信したデータをミキシングし配信する手法が考えられるが, サーバマシンに負荷が集中することになる.

そこで, リング型トポロジを採用し交互通信を行う. トークンを受信すると, 前の周回でトークンに加算した自分の音声を減算する. 減算した音声をスピーカに出力する. 自分の音声を加算する. このとき, 自分の音声をバッファしておきトークンを送信する. 退出者が発生すると問題が起こる. 退出者の音声は減算されることがなくなるため永久に巡回する. この問題を解決するために不正であると判断したトークンの音声をクリアして不必要な音声を取り除く. 必要な音声も消えてしまうことが欠点である. これは使用するミキシングアルゴリズムの限界であると考えられる.

4.考察

3 のように実装したシステムを用いて人間がコミュニケーションを行うテストを行った. テストユーザは8名で, ユーザに制限は設けなかった. セッション層に関する問題は発生せず, システムは正常に動作した. 提案するセッション層プロトコルが有効であることが検証できたと考える.

音声データの要素数 1,600 個の byte 型配列を処理するさいに問題が発生した. 1,600 個の要素を解析するのに時間がかかり音声が途切れる問題である. 音声データは 0.1 秒分であるにも関わらず解析に 0.2 秒以上要した. これについては XML の仕様を変更した. 図 4.1 のようにスペースで区切った数字とすることで 0.01 秒以下で解析可能となり解決できた.

```
<binary>0 0 0 2 0 -1 10 20</binary>
```

図 4.1 簡易byte型配列のXML仕様

これは, CPUのクロック周波数に依存する問題で, 今後の技術の向上により解決される問題であると考えられる. このように, 実際に作成したアプリケーションの変更により, 提案したセッション層プロトコルが変更されることはない.

セッション層プロトコルは, ダイアログ分割の設計や音声データの問題からわかるように上位層の要件と依存関係があることがわかった. しかしながら, 音声データの問題のように仕様変更が容易に行えるのは, セッション層の設計が頑強であるからである. また, 単一の通信接続を複数メディアで共有することを実現した. これにより, 通信接続セキュリティ面の問題を改善することができた. 以上のことから, 提案するセッション層プロトコルが有効であることが検証できたと考える.

5.まとめ

本研究で提案するプロトコルはコミュニケーションシステムのみに有効なだけではないと考える. すべてのプレゼンテーション層プロトコルを提案した統一規格に則って再設計することにより同一ホスト間の接続は唯一のものとなる. これにより, ポートの概念が払拭される. つまり, 現在抱えるセキュリティの問題がひとつ消滅することになる. また, アプリケーションの設計も簡素化され, 開発にかかる時間が大幅に短縮されるものと考えられる.

参考文献

1)飛田博章, 暦元純一 Flat3D:クリエーションとコミュニケーションを可能にする 3 次元共有仮想空間システム 情報処理学会論文誌 Vol. 44 No. 2 pp. 245～255 2003

2)伊藤英明・テーシューリン・中西英之・羽河利英 デジタルシティの 3 次元インタフェースの設計と実装 電子情報通信学会論文誌 D-I Vol.J86-D-I pp.592～599 2003

3)後藤真孝, 根山亮 Open RemoteGIG 遅延を考慮した不特定多数による遠隔セッションシステム 情報処理学会論文誌 Vol.43,No2,pp299～309 2002

4)Tony Graham Unicode 標準入門 翔泳社 2001

5)中山幹敏・奥井康弘 標準XML完全解説 技術評論者 2001

6)ElliotteRustyHarold Java ネットワークプログラミング オライリージャパン 2001

7)小堀研一, 春日久美子 基礎から学ぶ図形処理 工業審査会 1996