学位論文

ユーザ主導型ネットワークサービスと その構成に関する研究

指導教員　　森川 博之 助教授

東京大学大学院工学系研究科電気工学専攻

学籍番号 37-37085

三村 和

目 次

図目次

表 目 次

第1章

/序論

1.1 背景

インターネットと計算機の発展に伴い，ユーザとそれらの関係も常に変遷を遂げてきた．メインフレーム時代には複数人がネットワークを通じて 1 台の大型計算機を共有していたが，1980 年代半ばにはいわゆるパーソナルコンピュータ（パソコン）が登場し，個人が 1 台の計算機を利用するようになった．またインターネットの普及により，パソコン同士を結びつけて協調的な作業をするようにもなった．その後現在に至るまで，インターネットの高速化，計算機の小型化，低廉化によって個人が複数の計算機を所有するようになった．そして，今後訪れるであろうユビキタス環境 [1] の下では身の回りのデバイスが高機能化，遍在化していき，ネットワークを通じて複数人が複数台の計算機を所有，あるいは共有するようになる．つまり，ユーザと計算機のマッピングがより複雑化し，ユーザ中心，サービス（アプリケーション）中心なネットワーキングが必要となる．

一方，インターネットがグローバル規模で目覚しい発展を遂げた要因は，インターネットプロトコル（IP: Internet Protocol） の設計指針としてシンプルさとロバストさを優先させ，ネットワークそのものにおいて必要不可欠となるメカニズムだけを組み合わせたことにある [2]．すなわち，現在のインターネットアーキテクチャは，任意の IP アドレスへの到達性の効率化を第一指針として構築されており，ネットワークトポロジに強く依存した形で管理されている．これにより，ユーザはインターネットに接続された端末に対して “いつでも，どこでも” アクセス可能となる．もちろん NAT [3] などの影響によって必ずしもエンド・ツー・エンドの通信が実現されているわけではないが，IPv6 [4]，あるいは Network Pointers [5] や UIP [6] などの取り組みにより，近い将来この問題も解決されるであろう．しかしながら，現状のインターネットにはいわゆる管理者主導の枠組みしか存在せず，ユーザはネットワークリソースを利用する際に，管理者の管理する IP アドレスなどのネットワーク情報を用いることしかできない．これまでのインターネットに関する研究においても，如何に効率よく，障害に強く相手にデータを届けるのかについて注力されてきたと言え，その上でのユーザ中心，サービス中心のネットワーキングについてはそれほど考えられていない．これらの要因が相まって，ユーザの要求を満たすようなインターネットの利用が妨げられている．

現状の IP ネットワークの中でユーザの要求を満たすことの難しさは，ファイアウォールを設定することが困難であることからも見てとれる．ユーザの要求として，たとえば同一ユーザが所有している携帯端末だけに対して，提供するサービスへの通信を認めるということが挙げられる．ところが，そのような管理ポリシを表現する枠組みは存在せず，IP アドレスなどの情報を用いることしかできないところに様々な不都合が生じる．たとえば，

DHCP [7] を用いて IP アドレスを管理している場合，ユーザの取得する IP アドレスが一意ではないため，ユーザの要求を実現するようにファイアウォールを設定することができない．

このような不都合さは，管理者とユーザとの間で異なる要求を前述した管理者主導の枠組みで実現しようとすることに起因する．つまり，ユーザの利便性を向上させるためにはユーザと管理者の要求を切り離し，管理者が構築している既存のインターネット上にユーザ指向の枠組みを再構築することが求められる．このような試みは Layerd Naming Architecture（LNA）[8] や Regions [9]，Virtual Internet [10] などいくつか提案されている．たとえば LNA では，User-level Descriptor，Service Identifier，Endpoint Identifier という 3 層の新たなネーミング機構を導入することで IP にかかる負荷を減らし，結果としてユーザの利便性を高めている．

インターネット以外の分野を俯瞰してみると，管理者とユーザ間で要求が異なり，それを適切に分離したことによって大きな成果を挙げた例として，オペレーティングシステムの基本機能の 1 つであるファイルシステムが挙げられる．ファイルシステムは記録装置に管理されているデータを管理するための方式であり，ここでは管理者とはカーネルに相当する．データは記録媒体上の物理アドレスに格納されており，カーネルは効率的かつ高速なデータへのアクセスを達成する．このとき，そのアドレスに実際にどのような意味のデータが格納されているのかは意識しない．他方で，ユーザは自分が使い易いように種類や重要度などによってファイルを分類する．もしユーザがカーネルの管理ポリシに従ってファイルアクセスを行っていたとすると，その利便性の悪さは明らかである．逆にユーザの管理ポリシに従ってカーネルが格納位置などを行うと，非常に効率が悪くなるだろう．ユーザと管理者の管理要求を適切に分離し，ファイルシステムがその間を繋ぐことによって最適な作業を行うことを可能にしている．このような管理者とユーザとの関係は，将来のインターネットのあり方にも当てはまることではないだろうか．

管理者とユーザの管理要求を分離してユーザ指向なネットワーキングを実現するには，当然新たな機構をインターネットアーキテクチャに導入する必要がある．その際，新たな機構をルータやスイッチを含んだインターネット全体に組み込んでいくのか，あるいはインターネットのエンド側に位置する端末だけに導入していくのかが議論となる．より高機能・高信頼なネットワークを実現しようという試みであるアクティブネットワーク [11,12] などでは前者のスタンスを採った．しかし，実装の複雑さ，インターネット全体への導入の困難さから未だ実用には至っていない．一方で，ここ数年隆盛したオーバレイネットワークに関する研究 [13–17] では，IP ネットワーク上ではなかなか実用に至っていない技術を後者のスタンス，すなわちエンド側の端末上に実装することで実現する．たとえば，

OverQoS [14] ではエンド側に専用のノードを導入することで，そのノード間で構築されるオーバレイネットワーク上にてパケットロス率に関する QoS（Quality of Service）を保証する．このようにエンド側の端末上に導入する方が，遅延やスループットなどの単純な性能では劣ることになるが，実用化への閾値は遙かに低くなると言える．したがって，本論文においてもインターネットのエンド側に位置する端末だけに新たな機構を組み込むというスタンスを採用し，ユーザ主導型ネットワークサービスを実現することが妥当であると考える．

1.2 本論文の目的

本論文は，ユーザ主導なサービス指向ネットワーキングを実現するということを目標とし，ユーザの要求に応じたネットワークの個人化と，それに基づいた端末・サービス間の動的な通信制御を支援するためのネットワークミドルウェア，およびソフトウェア構成を論じる．ユーザにしてみれば，インターネット全体を管理したいわけではなく，自分の管理すべきデバイスやサービスといったリソースを，利用する状況に応じてリソース間の通信を柔軟に管理したいのである．そこで，管理の複雑さをなるべく排除しつつユーザの多様な要求に応えられる枠組みを構築することを目指す．

具体的には，ユーザの要求に応じた適応的なアクセス制御を目的とし，物理ネットワークトポロジに依存しないサービス指向な端末グルーピング機構について示す．また，構築されたグループ内における効率的かつ適応的なブロードキャスト配信機能の提供を目的とし，アプリケーション層マルチキャスト（ALM: Application-level Multicast）を用いた配信サービスを支援するミドルウェアのアーキテクチャについて示す．さらには，複数ユーザがネットワークを共有する場合にはユーザ同士の信頼関係を常時から醸成しておくことも重要な課題の 1 つであると考え，インフォーマルなコミュニケーションの活性化を支援するためのプレゼンス機構について示す．

以上の検討を通じて，来るべきユビキタス環境におけるユーザ主導型ネットワークサービスとその構成方法のあり方についての議論を行う．

1.3 本論文の構成

本論文の構成を以下に示す．

第 1 章　　序論

第 2 章　柔軟なアクセス制御のためのサービス指向グルーピング機構
第 3 章　適応的な多地点配信サービスのための機能ユニット指向ミドルウェア
第 4 章　インフォーマルコミュニケーション支援プレゼンス機構
第 5 章　結論

第 2 章では，ユーザの要求に応じた適応的なアクセス制御を実現するためのサービス指向な端末グルーピング機構について論じる．様々な端末やサービス（アプリケーション）がネットワーク上に遍在する環境では，サービス単位での柔軟なアクセス制御を実現する共通基盤をネットワークレベルで提供することが必須となる．そこでまず，サービスを単位としたアクセス制御を行うための端末グルーピング機構である MyNetSpace（MNS）システムの設計指針について示す．MNS は端末群の仮想的な閉域グループであり，グループ内での安全な通信機能と端末管理機能をネットワークレベルで提供する．端末は属する MNS ごとに専用の仮想ネットワークインタフェース（VNI：Virtual Network Interface）を作成する．それぞれのサービスはこの VNI を通して MNS 内部で安全な通信を行うことができる．また，サービスが適切な VNI を指定し，通信を許可する MNS を明示的に選択することによって，MNS システムはネットワークレベルでありながら IP 層からは分離した “第 3.5 層” で動作する．端末は複数の MNS に属することが可能であり，柔軟なグルーピングを実現する．

次に，このような MNS システムを実現するために，MNS アナライザ，サーバ，マネージャという 3 要素から構成されることを示し，それらの実装と動作検証を述べる．MNS アナライザは各端末内において，MNS ごとに固有な識別子をパケットに付加し，それを解析することで適切な VNI へ通信を振り分ける．MNS サーバは MNS ごとに 1 つ起動され，MNS 内の参加端末を管理する．参加認証では，たとえば端末が部屋にいることを検査するために外部の ID タグ監視基地局などとも連携する．また，参加端末だけに共通鍵を配布することで安全性を高める．MNS マネージャは各端末に 1 つ起動され，MNS への参加や，MNS アナライザの設定を行う．

第 3 章では，特定のグループ内における効率的かつ適応的なブロードキャスト配信機能をアプリケーションに対して提供するための ALM ミドルウェアアーキテクチャについて論じる．第 2 章の MNS システムを導入する上で，既存のブロードキャストの仕組みが利用できないという問題が生じるため，それに代わる機構を提供することが要求される．このような枠組みの中でのブロードキャストチャネルの利用法を考えると，端末やサービス

の発見・検出と，比較的大容量なデータを扱うリアルタイム多地点配信が挙げられる．ここでは，後者の多地点配信機能を提供するための機構について論じることにする．

ALM はグループ管理やパケット複製などといったマルチキャスト機能をアプリケーション層，すなわち端末側において実現するものであり，IP マルチキャストに比べてストリーミング配信やビデオ会議などのアプリケーション，あるいは CPU 速度やネットワーク帯域などの端末リソースに応じてその構成を変更しやすいという利点をもつ．これまで ALM のための仕組みやアルゴリズムが多数提案されているが，マルチキャストに関するすべての機能は個々のアプリケーションごとに独立に開発して組み込まれている状況は冗長であり，時間の浪費を生み出すことになる．そこでまず，開発上の冗長性を軽減し，最小限の労力で様々な ALM 機能をアプリケーションに組み込めるようにすることで開発効率を改善することを目指し，RelayCast と呼ぶ機能ユニット指向 ALM ミドルウェアを示す．RelayCast は，最小かつ基本的な ALM 機能を 1 つのユニットとして提供し，また新たなアルゴリズムを柔軟に追加できる枠組みをもつ．具体的な設計を行うにあたり，まず既存の ALM システムを鳥瞰した際に主としてオーバレイネットワーク構築機能とマルチキャストルーティング機能という 2 つの機能を抽象化できるということに着目する．また，通信機能，データベース機能，評価機能に関して共有モジュール機能としてそれぞれ独立化する．すなわち，RelayCast は ALM に関する抽象化された基本機能と共有モジュール機能の，5 つの機能ユニットから構成される．機能ユニットはコンポーネントというそれぞれが異なるアルゴリムで動作するブロックをもち，適切なコンポーネントを組み合わせることで，RelayCast は様々なサービスからの要求を満たすことが可能である．

さらに，上記の設計指針に基づいたプロトタイプ実装を行い，実用性の検証として実験において性能上のボトルネックが生じていないことを示す．

第 4 章では，インフォーマルコミュニケーションを支援するためのプレゼンス機構について論じる．物理的に離れた拠点のユーザ同士が MNS システムなどの仕組みを用いてネットワークを個人化し，それぞれがサービスを共有するような場合，お互いに顔の見えない所にいる相手と信頼関係を常時から醸成しておくことで円滑な運営が行えるであろう．一般に，日常生活の中で雑談のようなインフォーマルなコミュニケーションを積み重ねることが信頼関係の構築に有益であると言われている．インフォーマルなコミュニケーションは会議などフォーマルなコミュニケーションと比較すると，偶発的で相手も話題も不特定という特徴があり，プレゼンス情報によりユーザの状況を適切に伝達することがインフォーマルなコミュニケーションの支援には重要となる．そこでまず，インフォーマルなコミュニケーションを支援するためのプレゼンス技術に求められる要素を検討し，それ

に基づくシステム設計を示す．実空間に存在するユーザをネットワーク空間に投影するために，ユーザごとにユーザエージェントを備え，プレゼンス情報の（外部情報源からの）取得，伝達，合成という 3 段階のプロセスで処理する．このときサーバでプレゼンス情報を処理，蓄積しないことで，情報処理の柔軟性とユーザのシステムに対する安心感を得ることができる．

次に，いくつかのプレゼンス情報を利用出来るテストベッドとして実装を行ったアプリケーションを示す．究極的にはある 1 種のプレゼンス情報を伝えることで目標とするようなコミュニケーション支援をできることが理想であるが，実際にはプレゼンス情報の種類は極めて多岐に渡り，現時点ではその解を見いだすのは非常に困難である．更なる検討を有意義なものとするためには，どのような情報をどのような形で他のユーザへと伝達することがコミュニケーション支援に効果的であるかを，実際に試用していく中で見極める必要がある．

最後に，テストベッドを用いて行った，プレゼンス情報のコミュニケーションに与える影響に関する評価実験の結果について報告する．

第 5 章は結論であり，本論文における成果をまとめるとともに，今後残された課題について議論を行っている．

第2章

/柔軟なアクセス制御のための サービス指向グルーピング機構

2.1 はじめに

インターネットはグローバル規模で任意の IP アドレスへの到達性を確保することを第一目的とし，それを達成した [2]．しかしながら，ユーザにしてみれば全世界の端末と平等に通信するだけでなく，その中の限られた端末とだけ通信したいという要求も存在する．たとえば，家の共有ファイルを公開する端末に世界中どこからでも，しかしいくつかの自分の携帯端末だけからアクセスを許可したい．このような要求は，個人による Virtual Private Network（VPN）ツール（ [18–21] など）やパーソナルファイアウォール（ [22] など）の利用の隆盛からも窺い知ることができる．

今後，身の回りのあらゆるものが計算能力をもち，ネットワーク上に遍在化するに伴い，ユーザはそれらの上で動作する多種多様なサービス（アプリケーション）をインターネットを介して利用するようになる．さらには，Web サービスのようにサービスのモジュール化が進めば，サービス同士の繋がりがより重要になるだろう．このような環境では，ユーザの要求を満たす制御機構を個別に実装するのではなく，共通の基盤として提供することが必要である．

そのような共通基盤を提供するにあたり，筆者らは端末のグルーピングに着目する．サービスを公開する，またはサービス間で連携する場合，自分が所有する端末，特定の部屋にある端末などのようにその通信範囲は限定される．また，通信範囲に含まれる端末が動的に変化したり，サービスが通信範囲そのものを変更したりする．そのため，図 2.1 に示すように端末の閉域グループを作り出し，サービスの通信範囲をその閉域グループ内に限定することが効率的であると考える．このとき，端末上では多くのサービスが動作することから，端末は複数のグループに参加できることが求められる．このようにサービス単位で端末のグルーピングを行うことをここでは “サービス指向グルーピング” と呼ぶ．

本章での目的は，ユーザが柔軟に端末のグループを作り出し，それに基づいてサービス間における通信制御を行うための共通基盤を提供することである．これに向けて，筆者らは MyNetSpace（MNS）システムの検討を進めている．MNS は各々のユーザが定義する閉域グループで，MNS に参加しない端末との通信を許可しない．また参加認証機能をもち，ユーザが所有する端末や特定の部屋にある端末といった条件を満たさなければ端末は参加することができない．一方で，端末は条件を満たせばこれら複数の MNS に同時に参加できる．サービスと各 MNS を結びつけることによって，サービスは閉域グループ内で安全な通信を行うことができる．

MNS システムの設計指針の 1 つは，現状のインターネットアーキテクチャとの親和性を考慮し，サービス指向グルーピング機構をネットワークレベルで提供することである．

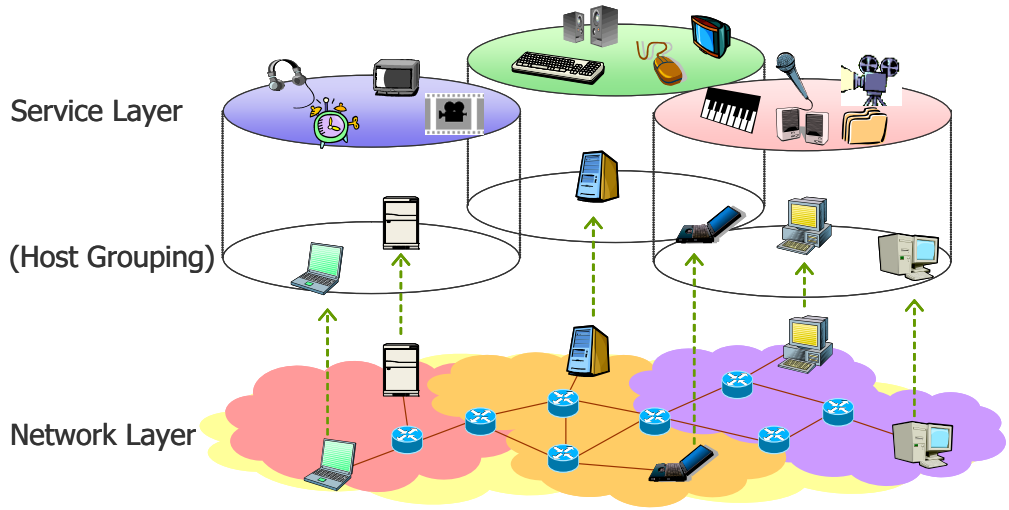


図 2.1: 端末のグルーピングによるサービス管理

サービスは，MNS ごとに専用の仮想ネットワークインタフェース（VNI：Virtual Network Interface）を介して MNS 内部との通信を行う．VNI を通ったパケットには各 MNS に固有な識別子（MNSID）がラベル付けされ，それを解析することによって適切な VNI へと転送される．このようにネットワークレベルで実現することにより，サービスは TCP などのトランスポート層でのプロトコルを利用することができる．

もう 1 つの設計指針は，通信を許可する MNS をサービスが明示的に選択することである．既存手法 [23, 24] ではルーティングによって暗黙的にグループを選択する．グループごとに新たに仮想 IP アドレス空間を割り当て，サービスからは相手の仮想 IP アドレスを指定する．これは既存のサービスに対して変更を要さない反面，いくつかのインターネットアーキテクチャ上の制約を生む．端末が複数のグループに属する際に，割り当てたアドレス空間が衝突したり，サービスが意図しないパケットを受信したりする可能性がある．また，ユーザが設定ミスを行った際に他の実通信に影響を与える可能性もある．将来的にユーザが銘々に多数のサービスを扱うようになるならば，ユーザ主導なグルーピング機構と管理者主導な IP ネットワークの機構とは完全に分離するべきであろう．本手法ではサービスが通常の物理インタフェース間での通信に加えて，自端末内の VNI を明示的に指定することによって，ネットワークレベルでありながら IP 層からは分離した “第 3.5 層” において MNS 内での閉域な通信を実現する．

MNS システムに関連した研究として Regions プロジェクト [9] が挙げられる．Regions

プロジェクトでは，将来の大規模，広分散，かつヘテロなネットワークのためのアーキテクチャ設計で鍵となるのはグルーピング機構であると主張する．ネットワーク上のエンティティ（ルータや端末）は必ず，Region と呼ぶ何かしらのメタ的な意味をもつグループに属する．Region はその内部のエンティティが同一の技術や管理ポリシをもつものとして定義される．その区分には関しては Autonomous System（AS）やサブネットなどの到達性を確保するためものだけでなく，たとえば “課金モデル”，“セキュリティポリシ” などより抽象的なものを汎用的に扱うとしている．Regions プロジェクトでは，グローバル規模で上記のように汎用的にネットワークを区切ることにより，柔軟な IP ネットワークの管理を達成することを目的とする．しかしながら具体的な機構の構築や運用には至っていない．一方，MNS システムではエンド側の端末の構成に着目し，ユーザがサービスごとに端末の閉域グループを作り出し，その特性をサービスから利用できる枠組みを，IP 層から分離した形で実現することを目指す．

以上を踏まえ，MNS システムを MNS アナライザ，MNS サーバ，MNS マネージャという 3 つの要素から構成した．MNS アナライザは MNS 内におけるサービス間の通信を実現する．VNI を通過したパケットに対応する MNSID を付加することで通信の識別を行う．MNS サーバと MNS マネージャは MNS の参加端末の管理を実現する．MNS サーバは MNS ごとに 1 つ存在して参加認証などを行う．MNS マネージャは個々の端末に 1 つずつ存在して MNS への参加要求を出し，また VNI や MNS アナライザの設定などを行う．

本章の残りの構成は以下の通りである．2.2 節では，サービス指向グルーピングと設計指針について説明する．2.3 節では，MNS システムの構成について述べる．2.4 節では，MNS システムのプロトタイプ実装と，その動作検証について述べる．2.5 節では，MNS システムの適用例を示し，最後に 2.6 節でまとめとする．

2.2 サービス指向グルーピング

2.2.1 グルーピングに基づくサービス管理

1 台の端末上で複数のサービスを動かしているときに，サービスごとに端末をグルーピングを行う様子を図 2.2 に示す．ここでは，たとえば以下のようなシナリオを想定している．

シナリオ： A 君は研究室からラップトップを渡されており，研究室や自宅，またホットスポットなどよりインターネットに接続している．そのような状況下で A 君は以下のようなことを実現したいと考えている．

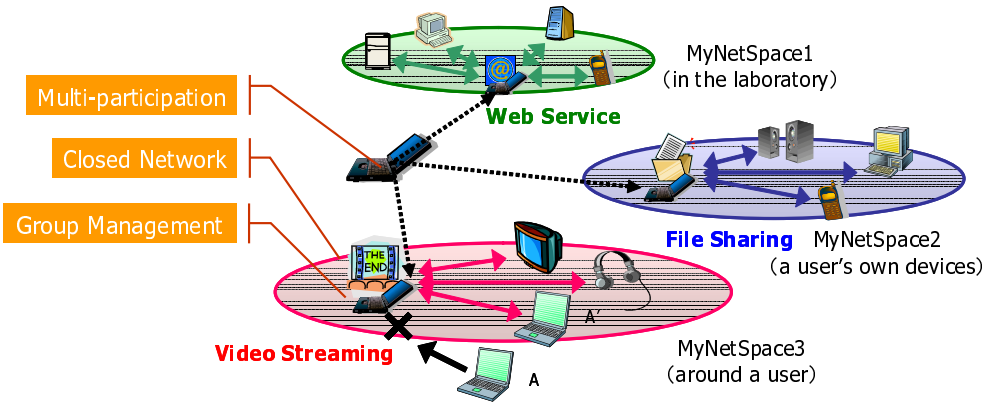


図 2.2: 複数の MyNetSpace へのサービスの公開

1. A 君は Web サービスとして研究室の空調を操作するサービスを開発し，ラップトップ上で実行している．研究室にいない人間が自由に空調を操作できては困るので，研究室内部にある端末だけから Web サービスへアクセス可能としたい．

2. A 君は渡されたラップトップを含めて複数の端末を所持しており，用途によって使い分けている．その際にデータが複数の端末に分散してしまい，必要なときに利用できない状態に陥るのは困る．そこで，自分が所持している端末同士ではお互いのデータへ常にフルアクセス可能としたい．

3. A 君はラップトップを用いてビデオストリーミングサービスを行いたいが，その際にデータへのアクセスは身近にある端末だけに許すようにしたい．つまり，大学にいるときは大学のスピーカやディスプレイからはアクセス可能であるが，自宅へ戻った際には自宅のスピーカやディスプレイからアクセス可能としたい．

ここで重要なのは，全てのシナリオにおいてラップトップが提供するサービスはユーザが望むグループ，すなわち MNS へしか提供されないということである．シナリオ 1 では “研究室内部にある端末”，シナリオ 2 においては “ユーザが所有している端末”，シナリオ 3 では “ユーザの身近にある端末” というのをを MNS として定義し，サービスの公開先を選択するという操作を行う．公開されたサービスはそれぞれの MNS に参加した端末だけからアクセス可能であり，その MNS 外部にある端末からのアクセスは制限されなければならない．たとえば，端末 A がサービスを公開する端末の周辺に存在する場合，端末 A は MNS3 に参加してビデオストリーミングを受信することができる．逆に，端末 A が離

れていった場合には MNS3 への参加権を失うため，ビデオストリーミングを受信できなくなる．

2.2.2 機能要件

2.2.1 節にて述べたシナリオを満たすサービス指向グルーピング機構を実現するためには，(a) 端末が複数の MNS に同時に参加できること，(b) MNS 内部では閉域な通信が保たれること，(c) 端末の MNS への参加条件を管理できることが要求される．(a) に関して，端末が既に独立な MNS に参加していたとしても，原理上は他の MNS への参加が制限されることがあってはならない．ただし，ある端末に物理距離が ‘遠い’・‘近い’ など各々の MNS の定義が排他的である場合はこの限りではない．この要求はシステムアーキテクチャの設計指針に関わる部分である．

(b) に関して，MNS 内部における通信は許可し，それ以外の通信は破棄されなければならない．つまり，各通信パケットに対してそれが MNS 内部の端末から送信されたものであることを証明し，かつ受信した際にはそのパケットを検査できなければならない．また (a) の要求があるので，端末が複数の MNS に属する場合に，物理的には同一端末宛の通信であってもそれぞれの通信がどの MNS におけるものなのかを判別しなければならない．この要求に関する機能をどうサービスに提供するべきかもまた，システムアーキテクチャの設計指針に関わる部分である．また，これに関する機能を実現する機構を以下では内部通信機構と呼ぶことにする．

(c) に関して，ユーザが定義した MNS の参加条件に基づいた認証や監視を行い，端末の参加・脱退を管理しなければならない．参加条件とは，具体的には ‘研究室にある端末’ や ‘ユーザが所有するデバイス’ などのメタ的情報である．しかし，実際にはユーザ ID の入力や，RFID を用いた物理位置情報取得といった外部機構との連携などを設定することで実現されるものである．この要求に関する機能は，基本的にはサービスの外側で実現されるべきである．これに関する機能を実現する機構を以下では端末管理機構と呼ぶことにする．

2.2.3 ネットワークレベルでの提供

サービスが MNS 内部の端末との通信を行う手段の提供方法，すなわち (b) に関する機能をどうサービスに提供するかについて議論する．これには 2 つの方法が考えられる．1 つは，ライブラリの中で新たなアプリケーションプログラムインタフェース（API：Application

Program Interface）を定義する方法である．サービスはこの API を通して特定の MNS 内部での通信を行う．JXTA [25] で提供される PeerGroup では，オーバレイネットワークで論理グループを形成し，JXTA 上で提供される専用インタフェースと API を通してその内部のリソースやサービスへのアクセスを許可する．しかし，JTXA プラットフォーム上で作成されたプログラムでしか動作せず，既存のものに対してはプログラミングフレームワーク自体の変更を強いることになる．つまり，この方法では既存のサービスに対して大幅な変更を要し，またプログラミング言語を限定してしまう．

もう 1 つの提供方法は，MNS へのアクセスを仮想的なネットワークインタフェース（VNI）の形でサービスに見せて，ネットワークレベルで提供する方法である．サービスは，MNS ごとに用意される VNI を介して MNS 内部との通信を行う．VNI としてサービスに見せるので Socket API をそのまま利用でき，サービスの作成において既存のプログラミングフレームワークを崩すことなく，グルーピング機構を導入できる．また当然のことながら，TCP など既存の上位層プロトコルをそのまま用いることができる．筆者らはなるべく現在のインターネットアーキテクチャと親和するように新たな機構を導入することが重要であると考え，後者の方法を採用する．

2.2.4 グループの明示的な選択

本章で想定するのは，サービス提供者が通信を許可するグループを自ら選択するような利用環境である．MNS システム上では，具体的にはサービスが VNI に対してバインドすることによってグループの明示的選択を行う．これにより，グループ分けというユーザ主導な枠組みと管理者主導な IP 層とを分離して考えることができる．

ネットワークレベルでグルーピング機構を提供するには，既存手法 [23, 24] のように VPN 技術を用いて閉域グループを構築する手法が考えられる．この手法では，それぞれのグループに対して新たに仮想 IP アドレス空間を割り当てる．サービスは端末上のルーティングテーブルにしたがって暗黙的にグループを選択することになる．これらは既存のサービスに対して変更を要さない反面，いくつかのインターネットアーキテクチャ上の制約を生む．端末が複数のグループに属する際に，割り当てた仮想 IP アドレス空間が衝突する，あるいはサービスが意図しないパケットを受信する可能性がある．また，ユーザが設定ミスを行った際に他の実通信に影響を与える可能性もある．このような理由から，本手法ではサービスが通常の物理インタフェース間での通信に加えて，自端末内の VNI を明示的に指定することによって，ネットワークレベルでありながら IP 層からは分離した“第 3.5 層” において MNS 内での閉域な通信を実現する．

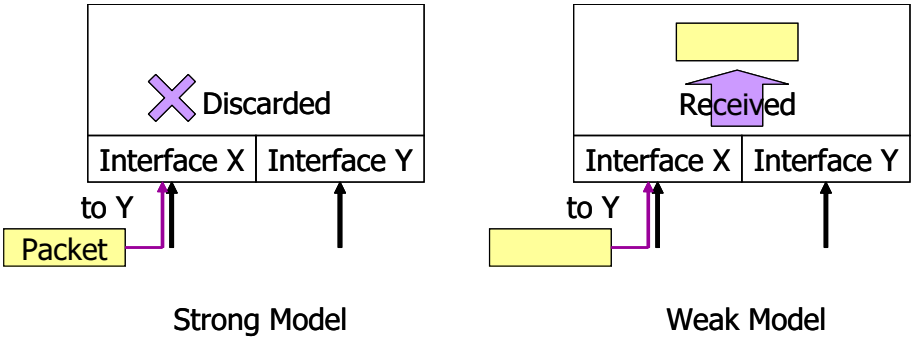


図 2.3: 複数インタフェースをもつ際の強モデルと弱モデル

そもそも，サービスがネットワークインタフェースを明示的に指定する（本手法の場合にはバインドを行う）ことが正しいのかということが議論になる．端末が複数のネットワークに属する場合，図 2.3 に示すようにインターネットには強モデルと弱モデルという 2 つの考え方がある [26]．強モデルはインタフェースについたアドレスで待ち受けを行い，弱モデルではホストについたアドレス（すなわち *INADDR_ANY*）で待ち受けを行う．一般にリンク層は強モデル，IP 層は弱モデルで設計されている．後者の理由は，マルチホーム環境においても任意の IP アドレスへの到達性を柔軟に確保したいからであった．そのため，現在のサービスも基本的には弱モデルで設計されるものが多い．しかし，ここで想定しているのはサービス自身が通信範囲を指定する環境である．すなわち，強モデルに基づいたサービスの構築が推奨されるような環境である．環境の変遷とともに強モデルが再び推奨されているのならば，サービスが通信範囲となる閉域グループ（すなわち MNS）を明示的に選択するという方針は妥当であると考える．

2.3 MyNetSpace システム

2.3.1 システム構成

MNS システムは，内部通信機構と端末管理機構の 2 つから構成される．前者は，TCP などの既存通信機構に影響を与えずに VNI 間で正しくパケット送受信を行う機構であり，MNS アナライザによって実現される．後者は，参加認証や監視を行うことによって端末が MNS に所属していることを保証する機構であり，MNS マネージャと MNS サーバによって実現される．

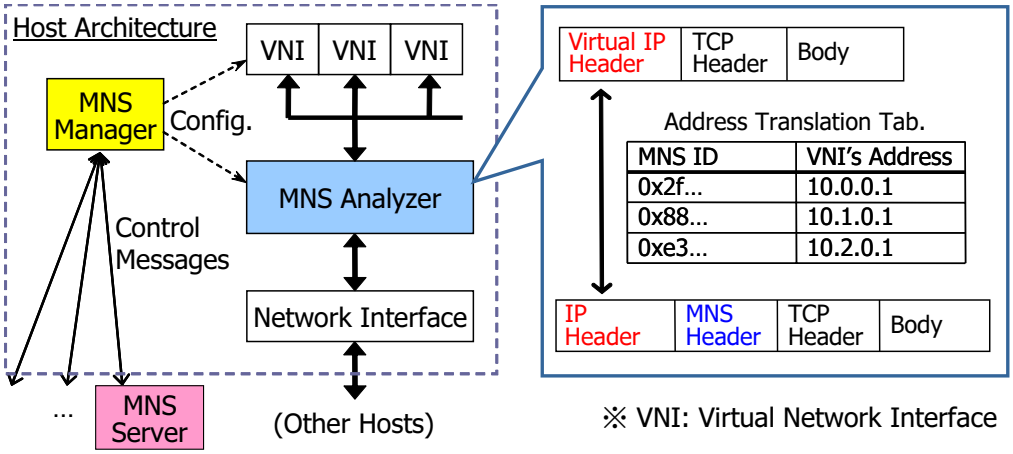


図 2.4: システム構成要素の関係

これら 3 つの要素の関係を図 2.4 に示す．MNS アナライザは，VNI と物理ネットワークインタフェースの間を仲介し，適切なパケットの受け渡しを行う．MNS サーバは MNS ごとに 1 つ存在し，参加の認証や MNSID の配布などを行う．MNS マネージャは各端末ごとに 1 つ存在し，MNS サーバと制御メッセージのやり取りを行って端末を MNS へ参加させる．また，各端末内における VNI や MNS アナライザの設定を行う．

2.3.2 内部通信機構

図 2.5 に同一の MNS に参加する端末間におけるパケット送受信の動作詳細を示す．簡単に説明すると，VNI を通ったパケットには MNS ごとに固有な MNSID を含むヘッダが挿入され，それを解析することによって適切な VNI へと転送されることになる．それぞれの端末は MNSID が ‘MNS3’ であるという情報だけを共有し，お互いの VNI に付けられる仮想アドレスの情報は共有しない．

MNS 内部の通信は以下の要領で実行される．MNS アナライザにはあらかじめ，自端末の VNI の仮想アドレスと MNSID との対応表が設定されている．送信端末では，VNI に割り当てた仮想アドレス（10.0.0.1）を送信元に設定し，通信相手の実 IP アドレス（133.11.100.1）を宛先に設定する．MNS アナライザは，VNI を通過したパケットがインターネットへ送出される前に，送信元を仮想アドレスから自端末の実 IP アドレス（133.11.90.1）へと変換し，パケットに MNSID（MNS3）を付加する．したがって，インターネット上では通信を行う双方の実 IP アドレスと MNSID がパケットに記載されることになる．受信端末

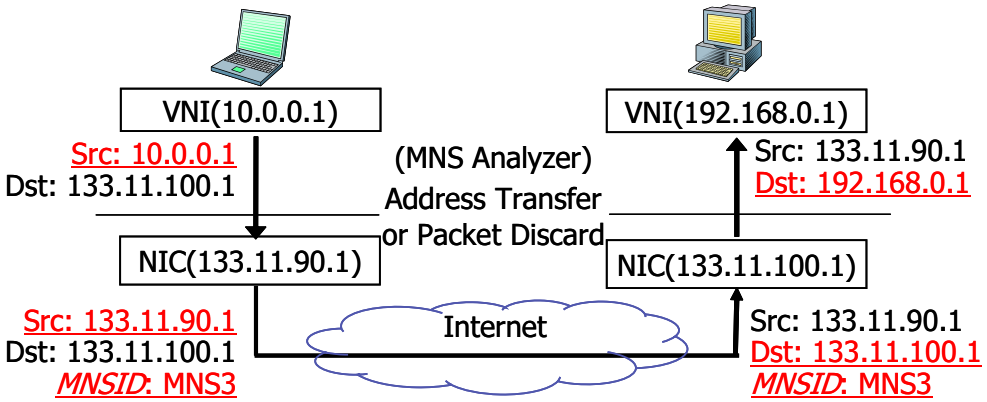


図 2.5: MyNetSpace 内部におけるパケット送受信の動作詳細

では，MNS アナライザが MNSID をもとに対応表を参照し，宛先を該当する VNI の仮想アドレス（192.168.0.1）に変換する．その後，MNSID を削除して該当の VNI へとパケットを転送する．MNSID がない，あるいは MNSID が対応表と一致しないパケットは VNI へと転送されないので，仮想的に閉域ネットワークが実現される．

送信元は通信相手の VNI の仮想アドレスを直接指定するのではなく，自端末の VNI を指定することによってどの MNS における通信かを示す MNSID を明示的に指定し，相手の実 IP アドレス宛に通信を行う．仮想アドレスは端末内だけで使われるので，端末は他の端末への影響を考えることなく VNI に対して自由なアドレス付けが可能となる．

本章では MNS システムを実現するのに IP 層からの分離を主張しているので，本来であれば VNI に付けるべき仮想アドレスは IP アドレス（*AF_INET*）であってはならず，新たなアドレスファミリを定義するなどしなければならない．しかし，現状において特定の IP アドレスにバインドできるサービスも多数存在することから，これらを変更なく利用できるようにするためにあえて VNI には仮想 IP アドレスを割り振ることにする．

2.3.3 端末管理機構

図 2.6 に，端末が MNS への参加手続きを開始してから内部通信を行うまでのシステム全体での動作を示す．ユーザが端末を MNS に参加させる場合，端末上の MNS マネージャを通して MNS サーバとの間で制御メッセージを交換して認証を行う．認証後，MNS サーバから MNSID と共通鍵が配られる．この共通鍵は，MNS から脱退した端末が既知の MNSID

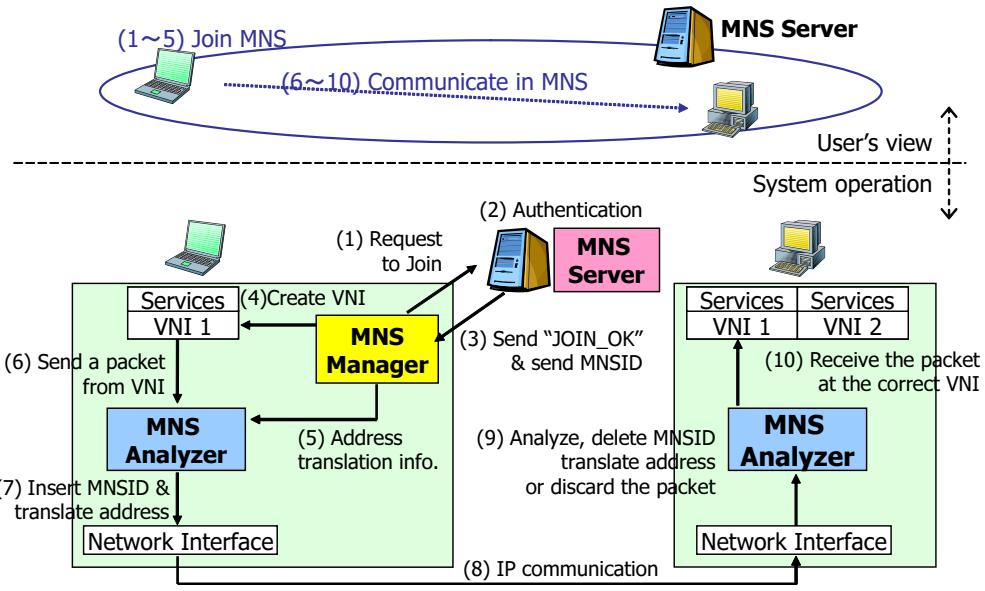


図 2.6: MyNetSpace システム全体の動作

を使って通信を行うことを防いだり，通信の暗号化を行うのに用いられる．厳密には端末の脱退が生じる度に鍵を変更することが望ましいが，MNS への出入りが激しい環境でも適用することを考慮し，定期的に鍵を変更するという方法を採る．MNS マネージャは参加した MNS に対応した VNI を作成して仮想アドレスを割り振り，MNS サーバから受信した MNSID と鍵を VNI との対応情報とともに MNS アナライザへ通知する．その後，2.3.2 で述べた手段を用いて端末間で MNS 内部の通信を開始する．

MNS サーバは，ユーザが MNS を作成するごとに 1 つ起動されるプロセスである．その構成は図 2.7 に示すように，ポリシ管理部（Policy Management），計算部（Computation），通信部（Communication）の 3 つに分かれる．ポリシ管理部は，ユーザが定義した MNS への参加条件（ポリシ）と現在の状態を照合しながら参加の許否を決定し，計算部に対して通知する．ユーザが MNS を作成する際には，このポリシ管理部の動作を決めることになる．たとえば，ある部屋にいる端末が MNS へ参加できるようにするには，外部の端末監視機器から情報を取得して現在の状態を記録し，その情報がない端末からの参加要求を受け付けない，あるいは脱退させるという動作を決定する．ここで，ポリシ判断情報に関しては MNS サーバ内部で所持することも可能だが，別の端末がもち，それを参照する形態も十分に考えられる．たとえば “サービスに課金を済ました端末” という参加であれば，

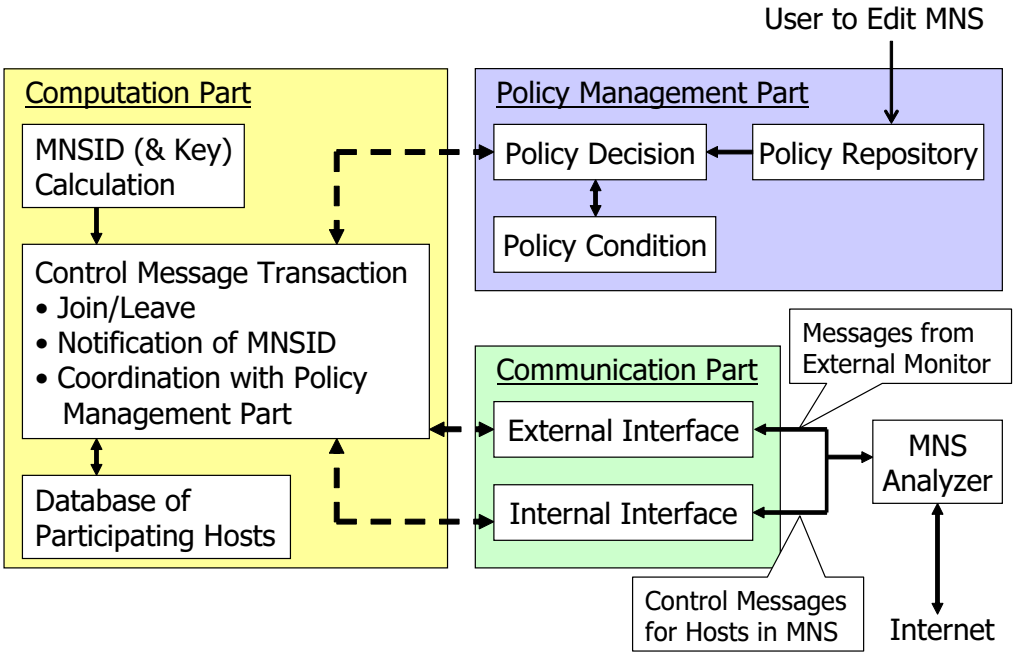


図 2.7: MNS サーバのフレームワーク

実際にサービスを提供している端末，組織が課金を済ました端末のデータベースをもっており，MNS サーバはこれをポリシ判断情報として参照することになるであろう．さらに，ユーザのポリシに関しては 2.2.1 節で挙げた例だけでなく，多種多様なものが考えられ，またユーザが自由に追加できなくては柔軟なシステムということはできない．それらの多様なポリシを充たすかの判断を下すためのポリシ判断情報には，端末の物理位置監視やユーザ情報データベースの利用など様々な機構が必要になると考えられる．そのため，MNS サーバのシステムとしては特定のポリシに特化した認証を組み込むのではなく，なるべく汎用的に判断できるフレームワークを定め，各々の実際の認証に関しては別個組み込むことができる形態をとる．

計算部は，MNS を構築・維持するための基本的な処理を担当する．この計算部は MNS の設定に関わらず共通の処理を行う．具体的には，MNS マネージャとの間でやり取りされる参加／脱退の制御メッセージの処理を行う．参加要求メッセージを受信した場合にはポリシ管理部に伺いを立てる．このときメッセージには認証に関する情報が付加されてきているので，その情報も一緒にポリシ管理部に渡す．ポリシ管理部で参加が受理されると，計算部は MNS マネージャに対してその MNS の MNSID と現在の共通鍵を通知する．また，参加ポリシの判断に用いられる外部監視機構から取得される情報も，一度計算部を経由してポリシ管理部に渡される．さらには，MNS に参加する端末の情報保持や，MNS 作成時にユニークな MNSID の生成と共通鍵の計算も行う．

通信部は，外部と内部のインタフェースをもち，それぞれで通信の待ち受けを行う．内部インタフェースを通しては，MNS 内部の端末と制御メッセージの交換を行う．外部インタフェースを通しては，ポリシ管理のための外部監視機構から情報を取得する．この内部インタフェースに特定の VNI を指定することで，ある MNS に参加する端末だけから参加要求を受け付けるという動作を容易に実現することができるようにする．

2.3.4 ブロードキャストチャネル

MNS システムを実現するにあたり，IP 層からの分離を図ったためにブロードキャストアドレスという概念がなくなり，その結果既存のブロードキャストチャネルの仕組みが利用できない．そのため，既存のブロードキャストチャネルの機能に代わる仕組みを提供しなければならない．

MNS という閉じたグループ内におけるブロードキャストチャネルの利用法としては，端末・サービス発見，MNS 内全員へのリアルタイムデータ配信の 2 つが挙げられる．そこで，これらのための機能を別個に提供することを考える．前者に関しては，やり取りに要

するパケット量は少ないため，MNS サーバと MNS マネージャ間での制御メッセージを拡張すれば実現できるであろう．具体的には，MNS サーバをディレクトリサーバに見立てて端末・サービスを登録し，MNS マネージャを通じてそれらのリストを取得する．しかしながら，サービスの記述やインタフェースの策定など多数の問題を解決しなければならないため，端末・サービス発見のための機能は今後の課題とする．

後者に関しては，いわゆるマルチキャストによって代用可能である．しかし，IP マルチキャストのインターネットへの展開がなかなか進んでいない状況を考慮すると，それ以外の手段も用意するべきであろう．そこで，もう 1 つの手段としてアプリケーション層マルチキャスト機能を提供することを提案する．これについては，第 3 章にて詳しく述べることにする．

2.4 実装と動作検証

2.4.1 実装の概要

MyNetSpace システムのプロトタイプの実装を，多様化する端末環境を考慮して Windows XP SP1 と Linux カーネル 2.4 系の 2 つのプラットフォーム上において行った．MNS アナライザは Windows 上では，ネットワークスタックの最下層に位置する NDIS（Network Driver Interface Specification）中間ドライバ [27] に実装され，カーネルモードで動作する．Linux 上では，Linux カーネル内でパケットフィルタリングのために一般的に用いられる Netfilter [28] のモジュールとして実装される．MNS サーバと MNS マネージャは，両プラットフォーム上ともユーザモードで動作するプログラムとして実装される．

2.4.2 内部通信機構の実装

MNS ヘッダの定義

MNS 内部通信に用いる MNS ヘッダを図 2.8 に示す．MNS ヘッダは MNSID と認証データを含む．Length にはヘッダ長のバイト数を格納する．Version には MNS システムのバージョンを格納する．現在は 0x1 となっている．Hash には認証データの計算に用いるハッシュ関数を指定する．現在は 0x0 ならば認証データなし，0x1 ならば共通鍵をそのまま認証データとして利用，0x2 ならばペーロードと共通鍵を使って MD5 [29] を計算，0x2 ならばペーロードと共通鍵を使って SHA-1 [30] を計算する仕様になっている．Encrypt にはペイロード部分の暗号化に用いるアルゴリズムを指定する．現在は 0x0 ならば暗号化なし，

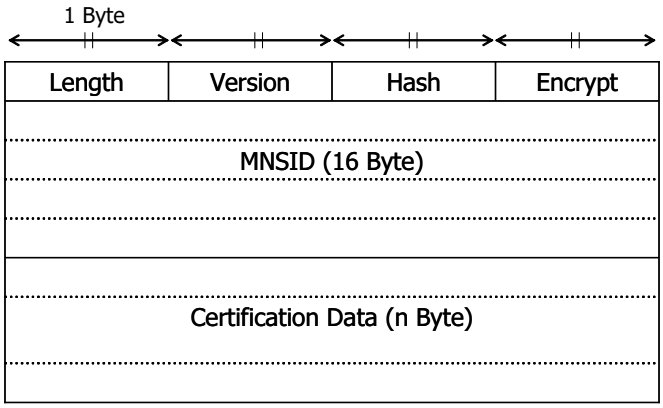


図 2.8: MNS ヘッダの定義

0x1 ならば blowfish [31] を用いて暗号化する仕様になっている．

Windows 上における実装

図 2.9 に実装概念図を示す．MNS アナライザは OS の TCP スタックよりも下位層でパケットの書き換えを行う必要があるため，NDIS 中間ドライバとして実装した．NDIS とは Microsoft 社が定めたネットワークドライバの標準仕様であり，その中間ドライバは図 2.10 に示すようにネットワークドライバにミドルウェアとして機能を追加できるように準備されている．NDIS 中間ドライバは，上位層にあるトランスポートドライバからは NIC ドライバのようにみえ，逆に下位層に存在する NIC ドライバからはトランスポートドライバのように見える．そのため，NDIS 中間ドライバを作成する場合には本来の NIC ドライバ，トランスポートドライバが提供していた Miniport / ProtocolXxx 関数全てを提供する必要がある．

MNS アナライザは，ドライバレベルでパケットを捕まえて 2.3.2 で述べた操作を行い，加えて IP ヘッダの書き換え，および TCP/UDP チェックサムの再計算を行う．パケットを送信する際，上位のトランスポートドライバから MiniportSendPackets 関数が呼ばれる．そこで，中間ドライバにおいて MiniportSendPackets 関数に処理を追加する．まず，MiniportSendPackets 関数から送信するパケットを取り出し，それに MNSID を含む MNS ヘッダを加えたデータ長のデータメモリを NdisAllocateMemoryWithTag 関数を用いて確保する．次に，元の送信データを新しく確保した領域にコピーし，送信元アドレスの変換と MNS ヘッダの挿入を行う．その後，新しく作成した送信パケットの IP チェックサムと

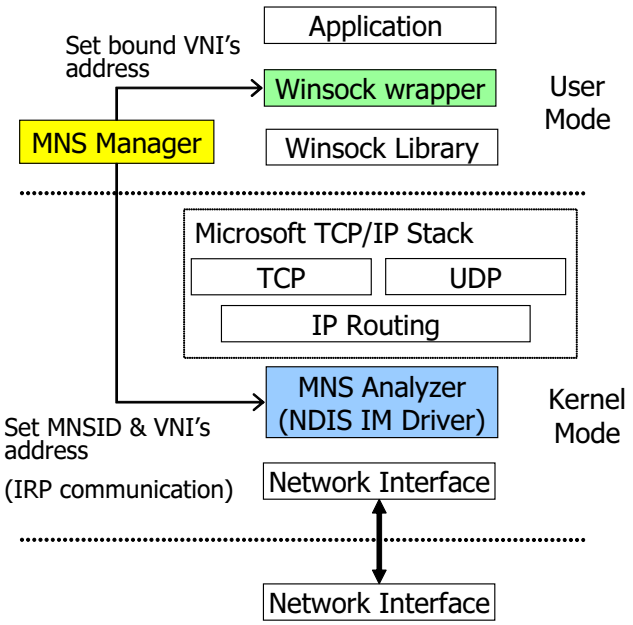


図 2.9: Windows における実装概念図

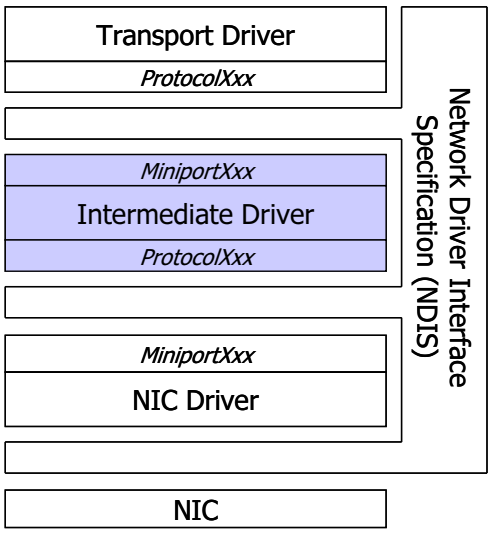


図 2.10: NDIS（Network Driver Interface Specification）の概要

```
struct MNSEntry{
    MNSVNIType  MNSVNIAddress;  // VNI's IP address
    MNSKeyType  MNSKey;         // MNS-key
    MNSIdType   MNSId;          // MNSID
};
```

図 2.11: MNSEntry 構造体の定義

TCP/UDP チェックサムを再計算し代入する．最後に，元の送信パケットの領域を開放し，NdisSend 関数を用いて新しいパケットを NIC ドライバへと転送する．

パケットを受信する際にはまず，ProtocolRecvPackt 関数から受信したパケットを取り出し，MNS ヘッダが挿入されているか判断する．MNS ヘッダが挿入されていた場合には，パケットから MNS ヘッダ長だけ減らしたデータ長のデータメモリを確保する．MNS ヘッダの削除と宛先アドレスの変換を行った受信パケットをコピーした後，新しく作成した受信パケットの IP チェックサムと TCP/UDP チェックサムを再計算し代入する．最後に，新しいデータを NdisMIndicateReceivePacket 関数を用いてトランスポートドライバへと転送する．

また，MNS マネージャから MNS アナライザに対して VNI と対応する MNSID，鍵の情報を通知するために，ドライバ間の通信のために規定された I/O Request Packet を利用した API を用意した．カーネルモードのデータへのアクセスは Win32 シンボリックリンクという形で提供されるので，それに対して write 命令を呼び出して情報を更新する．実際には参加 MNS のエントリの追加・変更・削除に関して，IOCTL_PTUSERIO_SETNEWMNSENTRY・IOCTL_PTUSERIO_CHANGEMNSENTRY・IOCTL_PTUSERIO_DELETEMNSENTRY という I/O リクエストを DeviceIOControl の第 2 引数として渡し，MNSEntry 構造体のポインタを第 3 引数に渡すことで実現する．MNSEntry 構造体の定義は図 2.11 の通りである．

Linux 上における実装

Linux カーネル 2.4 においても MNS システムの実装を行った．図 2.12 にその概念図を示す．MNS アナライザは，VNI を実現する仮想デバイス vnic.o と IP パケットを書き換える nf_mns.o という 2 つのカーネルモジュールから構成される．

vnic.o は実際のデバイスをもたないので外部との通信はできないが，通常のネットワークインタフェースと同様に ifconfig コマンドで IP アドレスを割り振ることができる．また，

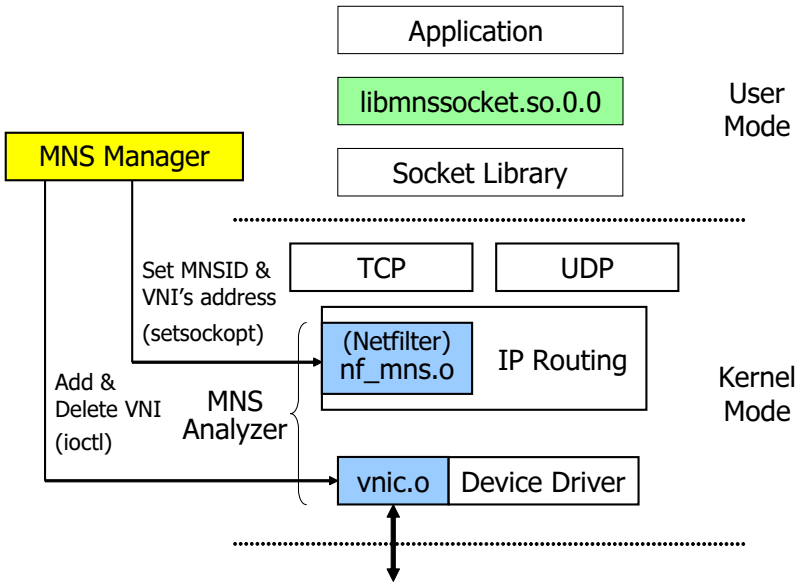


図 2.12: Linux カーネル 2.4 における実装概念図

loopback インタフェースと同様の構造をもつので端末内部での通信が可能である．vnic.o には任意の名前の VNI をメモリが許す限り登録できる．

vni.o はロードに成功すると vni インタフェースを生成する．ただし，vni は VNI を生成・削除するためだけのインタフェースであり，VNI としては動作しない．vni に raw ソケットを使用した ioctl を発行することで，VNI の追加・削除を行うことができる．VNI の追加・削除するにはそれぞれ，VNIC_ADDDEVICE・VNIC_DELDEVICE という ioctl リクエストを ioctl の第 3 引数として渡し，第 4 引数に if_req 構造体のポインタを渡す．これを用いて，実際に VNI を追加する際のプログラミング例を図 2.13 に示す．

MNS アナライザを形成するもう 1 つのモジュールである nf_mns.o は，IP パケットのアドレス変換と MNSID の解析するモジュールであり，Netfilter [28] のモジュールとして実装される．Linux カーネル内でパケットを管理するデータ構造である sk_buff 構造体に対して MNS ヘッダの追加と IP ヘッダの書き換えを行う．また，MNSID と鍵の対応のチェックも nf_mns モジュールで行う raw ソケットに対して setsockopt を実行することで，VNI 毎に MNSID と鍵を設定することができる．setsockopt の第 3 引数には以下を指定する．

- MNS_SET_ADDR： setsockopt の第 4 引数で指定した mns_opt 構造体の領域に書きこまれた MNSID と鍵を設定する．

```
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <string.h>
#include "vnic.h"
#include <linux/if.h>
#include <linux/if_ether.h>
   ～
int sk;
struct if_req ifr;

// Create raw socket
sk = socket(PF_PACKET, SOCK_RAW, htons(ETH_P_IP));

// Specify a kind of interface handled by 'ioctl'
strncpy(ifr.ifr_name, "vni", IFNAMSIZ);

// Name the interface
ifr.ifr_data = "vni1";

// Call 'ioctl'
ioctl(sock, VNIC_ADDDEVICE, &ifr);
   ～
```

図 2.13: 仮想ネットワークインタフェースを追加するプログラミング例

```
struct mns_opt {
    __u32  mnsid[5];       // MNSID
    __u32  seckey[5];      // Key
    __u32  vniaddr[1] ;    // VNI's IP address
    char   vni[IFNAMSIZ];  // Network interface name
};
```

図 2.14: mns_opt 構造体の定義

- MNS_GET_SIZE： VNI の数を取得する．取得した値は setsockopt の第 4 引数で指定した int 型の領域に保持される．
- MNS_GET_INFO： 全ての VNI に関する MNSID と鍵の値を取得する．取得した値は setsockopt の第 4 引数で指定した mns_opt の配列に書き込まれる．このオプションを実行する場合は，前もって VNI の個数を MNS_GET_SIZE で取得しておかなければならない．

なお，MNSID，鍵の値は図 2.14 に示す mns_opt 構造体を介してやり取りされる．

ソケットラッパライブラリ

既存サービスから MNS の内部通信を利用できるようにするため，VNI へはローカル IP アドレスを割り振る．サービスは VNI の IP アドレスに対してバインドし，通信する MNS を選択する．しかし，バインドのオプションをもたない既存サービスも多く存在するので，図 2.9 や図 2.12 に示すように Winsock ラッパライブラリと libmnssocket.so.0.0 モジュールを用意した．これは，特定のネットワーク API の実行時に bind を強制的に呼び出すことで bind を実行しないサービスでも MNS システムの機能を利用することができる．bind を呼び出すタイミングは以下である．

- socket： UDP のソケット生成時に bind を呼び出す．
- connect： connect の実行前に bind を呼び出す．

Winsock ラッパライブラリの場合にはバインドさせたい MNS を MNS マネージャからあらかじめ選んでおく．libmnssocket.so.0.0 モジュールの場合には以下ように VNI のアドレスを指定して，モジュールをプレロードする．

```
$ > VNIADDR=10.0.0.1 LD_PRELOAD=/usr/lib/mns/libmnssocket.so.0.0 ¥
firefox http://192.168.0.161/
```

2.4.3 端末管理機構の実装

端末管理機構の実装では，MNSID の計算，鍵の変更，MNS 作成時の設定項目の 3 つを特に考慮する．MNSID の計算については，ユニークな値を算出する必要がある．本実装では 128 ビットの乱数として計算し，確率的に MNSID の衝突を回避するようにする．もし衝突した場合には，両方の MNS に参加した端末の MNS マネージャにて検出し，MNS サーバに対して MNSID を変更するように要求を出す．

鍵の変更については，厳密には端末の脱退が生じる度に更新することが要求される．しかし，この方法は端末の MNS への出入りが激しい環境においてはあまり効率的ではない．そのため，本実装ではより汎用的な利用を意図し，定期的に鍵を更新する方法を採った．また，鍵の長さは現在は 160 ビットと生成している．

MNS 作成時の設定項目については，作成者は名前や MNS サーバのアドレスなど以外に，参加認証のために MNS サーバのポリシ管理部の設定を行う必要がある．しかし，ポリシ管理において用いる情報は位置情報やユーザ情報など多種に渡り，さらには端末の物理位置監視やユーザ情報データベースなどの外部機構との連携も必要になる．基本的な動作は 2.3.3 節で述べたフレームワークに従うことになるが，汎用的なルールやメッセージフォーマットの記述方式は課題として残る．これに対する一検討として，RFID 基地局，およびタグリーダ付携帯電話と連携するシステムを実装した．これに関しては 2.5 節にて詳しく述べる．今後はより詳細な検討を行う予定である．

最後に，MNS サーバと MNS マネージャとの間においてやり取りされる制御メッセージについて表 2.1 に示す．ここではもっとも基本となるものだけを定めている．メッセージの種類は参加，脱退，鍵変更に関するものであり，それぞれリクエストとそのレスポンスで構成される．LOGIN には参加認証で必要な情報が付加される．LOGIN_OK には MNSID と鍵が付加される．また，KEYCHANGE には変更された鍵と有効期限が付加される．

2.4.4 動作検証

Linux 上で実装した MNS アナライザに対して初期評価を行った．評価実験では，MTU を変えながら，Pen4 2 GHz，DDR266 512 MB を積んだ 2 台の PC 間で 30 MB のファイ

表 2.1: 制御メッセージ

Message	Payload	Src.	Dst.	Description
LOGIN	Authentication Info. (User-ID, etc.)	Manager	Server	Login request
LOGIN_OK	MNSID, Key	Server	Manager	Login success (LOGIN response)
LOGIN_FAIL		Server	Manager	Login failure (LOGIN response)
LOGOUT		Manager	Server	Logout request
LOGOUT_OK		Server	Manager	(LOGOUT response)
KEYCHANGE	Key, Expiration date	Server	Manager	Key change
KEYCHANGE_OK		Manager	Server	(KEYCHANGE response)

ルを受信したときのスループットを測定した．実験は初期評価のため，ヘッダの Hash を 0x1，Encrypt を 0x0 として測定した．

結果を図 2.15 に示す．横軸を MTU，縦軸を受信スループットとし，比較のために MNS アナライザを通さない場合を載せた．MNS アナライザを通した場合，MTU 1500 バイト付近ではスループットが 95%程度に低下した．MTU を小さくすると，アドレス変換と MNSID 挿入の操作回数が増え，結果として 87%程度まで低下した．若干の性能低下は確認されたが，実用の範囲内と考えている．

次に，実際に VNI を通して通信を行っている様子を図 2.16 と図 2.17 に示す． 前者は Windows におけるパケットキャプチャの表示，後者は Linux における netstat コマンドの表示である．動作確認には Web サーバと Web ブラウザを用いて HTTP にて通信を行った．図から，それぞれ VNI に割り振ったアドレス（10.0.0.1）を使用していることが分かる．このとき正常に通信が行われ，Web ブラウザには Web ページが表示された．

また，Linux において ifconfig コマンドによってネットワークインタフェースを表示させた画面を図 2.18 に示す．これから 1 つの端末上に複数の VNI が作成さていることが分かる．さらに，図 2.19 にはこのとき参加中である MNS に関する情報を示す．

2.5 MyNetSpace システムの適用例

2.5.1 位置情報に基づく MyNetSpace の構築

本節では，実装した MNS システムを用いて構築した，実空間上の位置情報に基づくサービスアクセス制御システムについて報告する．ここでのシナリオとして，MNS のサービス提供条件は「ある部屋にいる間だけアクセスを許可する」というものになる．具体的な

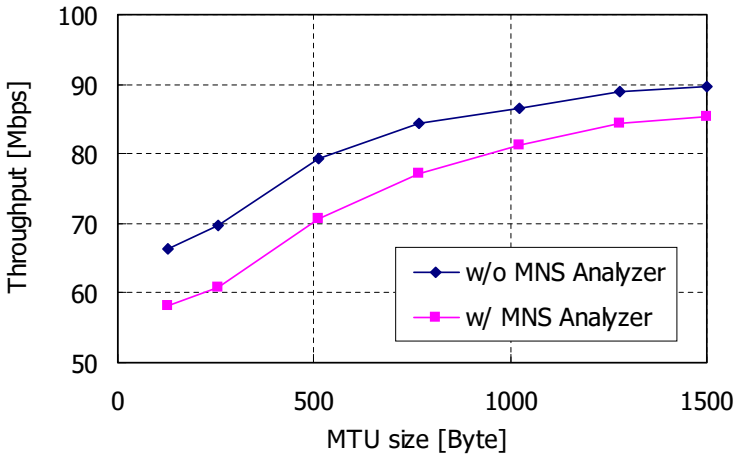


図 2.15: MNS アナライザを導入した際の MTU と受信スループットの関係

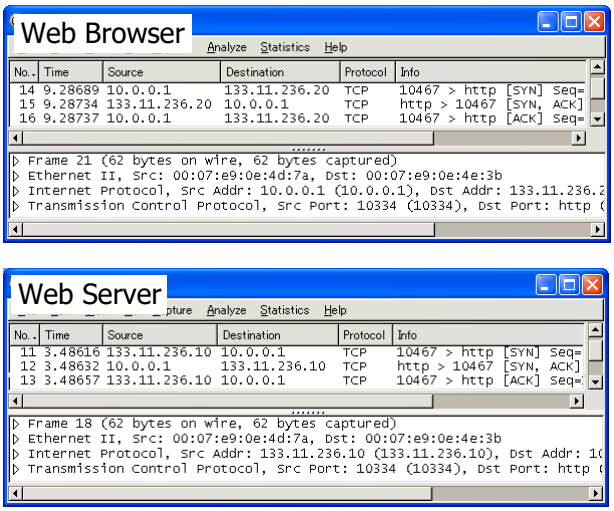


図 2.16: パケットキャプチャによる内部通信の様子の表示（Windows 版）

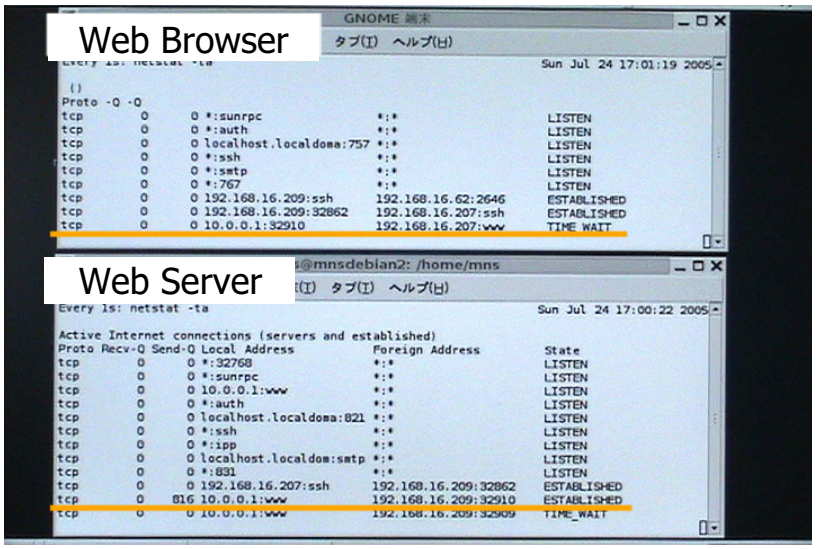


図 2.17: netstat コマンドによる内部通信の様子の表示（Linux 版）

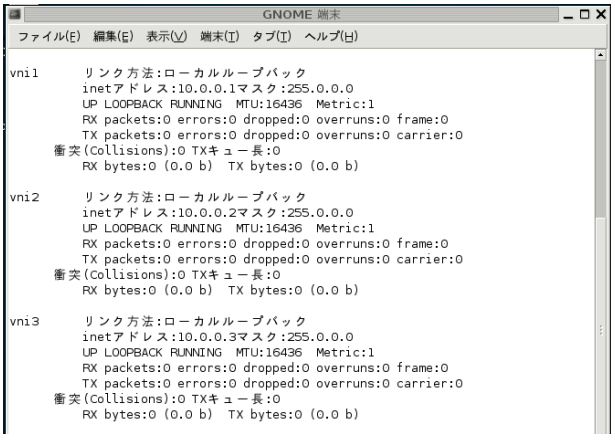


図 2.18: 複数の仮想ネットワークインタフェースが作成されている様子（Linux 版）

図 2.19: 参加中の MyNetSpace の表示（Linux 版）

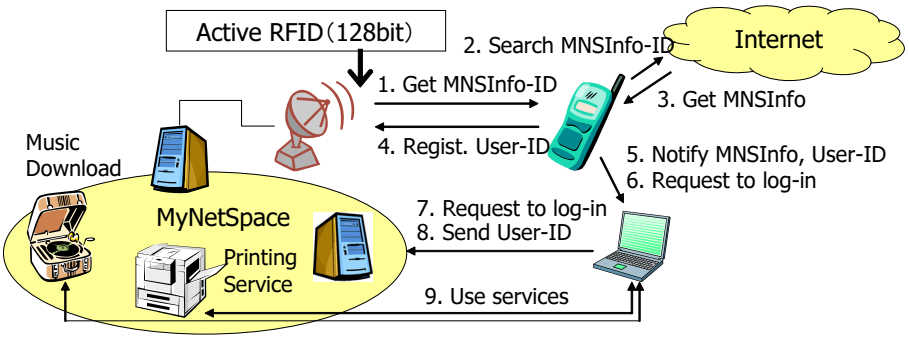


図 2.20: タグリーダ付携帯電話を用いた近傍 MyNetSpace の検出と参加

サービスとしては，特定の部屋から閲覧可能なファイルやデータを自分の端末で受信できるサービスや，逆に自分が所有しているデータをその場にあるプリンタやスピーカに出力できるサービスが考えられる．このようなサービスを提供するための MNS がネットワーク上に数多く設置されており，ユーザはその中から必要なものを発見し，ログインしてサービスを受ける．このようなシステムを実現するためには，ユーザの周辺に存在する MNS を発見し，発見した MNS に自分の端末からアクセスしてログイン認証手続きを行うことが必要となる．一度自分の端末を MNS にログインさせた後は個々の端末に対して認証手続きを必要とせず，同じ MNS に参加する全ての端末と内部通信できるようにする．これにより手続きの冗長性を防ぎ，シングルサインオンを実現する．

今回は図 2.20 のように，タグリーダを搭載した携帯電話とアクティブ RFID タグを用いた．携帯電話はユーザが日常生活の中で常に身につける通信端末であり，ユーザの近傍にあるネットワークを検出するデバイスとして相応しい．また，アクティブ RFID は可変なデータを発信機から半径数メートル〜十数メートルの範囲に限定して発信することが可能で，その到達範囲はユーザが発信機の近傍にいることを表すのに適当である．それぞれの

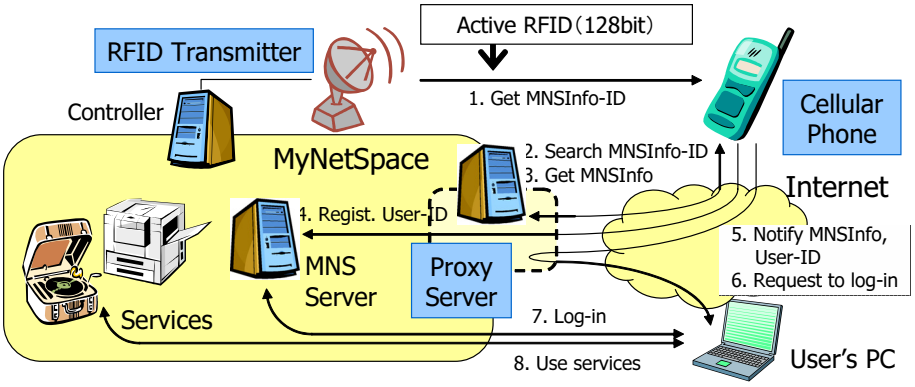


図 2.21: 近傍 MyNetSpace の検出の実装の様子

MNS はアクティブ RFID タグ発信機を通じて，自身を識別するための情報を周囲に対して無線発信している．ユーザは携帯電話のタグリーダでそれを受信し，インターネットを経由してその MNS へのログインに必要な情報を入手したうえで，自分の指定した端末に対しログイン手続きを行ってサービスの授受を行う．ログインを継続できる時間には上限を設ける．

2.5.2 実現形態

実際の実装システムを図 2.21 に示す．ここで本実装では，携帯端末として図 2.22 に示した KDDI が試作開発した RFID タグリーダ搭載携帯端末 [32] を使用した．MNS 検出・参加機構の中核となるのは RFID タグ発信機，タグリーダ付携帯電話，そしてプロキシサーバである．本開発で利用した携帯電話では，通信する IP アドレスがアプリケーションごとに固定であるため，任意の IP アドレスをもつ端末を通信相手に指定することができない．これを解決するために，携帯電話からの通信を仲介するプロキシサーバを設置している．

RFID タグ発信機は操作用端末の COM ポートにシリアル接続されており，この端末上の専用アプリケーションから設定される．128 ビット長のアクティブ RFID には MNSInfo を検索するためのキーとなる MNSInfo-ID が含まれている．MNSInfo とは MNS の名前，提供されるサービスの説明，MNS サーバのアドレスなど MNS へのログインに必要となる情報である．1 台の発信機からは最大 16 個の異なる MNSInfo-ID を発信でき，1 つの空間に独立した複数の MNS を設定できる．携帯電話では背面に搭載された RFID タグリーダを用いてこのタグを読み取る．

図 2.22: 携帯電話に取り付けられたアクティブ RFID タグリーダ

具体的な動作手順を以下に示す．携帯電話内蔵のアプリケーションは読み取ったタグ ID を元に (1)，ユーザの端末を MNS にログインさせるために必要となる情報をプロキシサーバに対し問い合わせる (2)．プロキシサーバには MNSInfo-ID と MNSInfo の対応情報が XML 形式で保管されており，検索要求を受けたプロキシサーバはこのデータから該当する MNSInfo を探して携帯電話に通知する (3)．携帯電話は User-ID をプロキシサーバ経由で MNS サーバに登録し (4)，同時に MNSInfo と User-ID をユーザ端末に通知 (5) してログインを要請する (6)．ログイン要請を受けた端末は MNS サーバにアクセスしてログイン手続きを行い (7)，完了すれば MNS に参加してサービスを利用できる (8).

2.6 おわりに

本章では，ユーザが柔軟に端末のグループを作り出し，それに基づいてサービス間における通信制御を行う枠組みとして MNS システムを示した．MNS はネットワークレベルでユーザが構築する仮想的な閉域グループである．端末を複数の MNS に同時に参加させ，端末上で動くサービスを各 MNS と結びつけることによってサービス指向グルーピングを実現する．また，本稿では Windows 上での実装について述べたが，現在では多様化する端末環境を考慮して Linux 上での実装も行っている．今後は，端末管理機構の詳細化や新たな適用シナリオなどの検討を進める．

第3章

/ 適応的な多地点配信サービスのための機能ユニット指向ミドルウェア

3.1 はじめに

本章では，グループ内におけるブロードキャストチャネルの利用法の 1 つであるリアルタイムデータ配信機構をアプリケーション／サービスに提供する手法について論じる．第 2 章で述べたサービス指向グルーピング機構では，いわゆる第 3.5 層で実現するために既存のブロードキャストチャネルが利用できないことが問題となる．一般に，大規模ファイル配信やビデオストリームの配信，ビデオ会議といった多地点通信型のアプリケーション／サービスにおいては，効率的，かつスケーラブルな多地点配信技術，すなわちマルチキャスト技術が必須となる．ここで，グループ内におけるブロードキャストというのはマルチキャストと同義であるといえる．そこで，本章ではマルチキャストを如何に効果的に提供できるかということについて述べる．

マルチキャストは，スイッチやルータなどのネットワーク層において達成されるものと，エンド側のホストのアプリケーション層において達成されるものに大きく分類される．前者は “IP マルチキャスト”，後者は “アプリケーション層マルチキャスト（ALM: Application-level Multicast）” とそれぞれ呼ばれる．IP マルチキャストのコンセプトは広く受け入れられ，現在ではその機能がほとんどのルータに実装されている．しかしながら IP マルチキャストは，ドメイン間ルーティング，グループ管理，マルチキャストアドレス割当，リライアビリティ，輻輳制御，フロー制御，セキュリティ，そしてビジネスモデルの欠如といった，アーキテクチャに根本的に起因する多くの問題を抱えている [33]．これらの問題のために，IP マルチキャストのインターネットにおける実用化は進んでいない．これらの問題の一部に対する部分的解決策は数多く提案されているが，それらは他の往々にして問題をより複雑なものとする．

一方，エンドユーザが扱うホストの CPU の高速化やメモリの大容量化，アクセスリンクを含めたネットワークの広帯域化などによって，ピアツーピア（P2P: Peer-to-Peer）技術を用いた通信やコンピューティングが可能となった [34]．P2P 技術は洗練された既存のユニキャスト技術をそのまま流用できること，およびインターネットへの展開が容易であることから，多地点通信への応用が盛んに研究され，様々な ALM 方式が提案されてきた [15, 35–39]．ALM ではパケットの複製やマルチキャストルーティング，グループ管理などのマルチキャストに関する機能を，アプリケーション層において提供される．ホストは自律的に論理的なオーバレイネットワークを形成し，それに沿ってホスト間を P2P でデータを中継，すなわちバケツリレー式に転送する．

これまで，ALM は積極的に研究対象とされ，ALM のための様々な仕組みやアルゴリズムが提案されてきた [40–50]．これらの仕組みやアルゴリズムによって，多様なアプリケー

ションからの基本的要求のほとんどを満たすことができると言えよう．そのため，実用的観点から今後どのようにして ALM システムを開発，展開していくかを検討することが重要であると考える．ここで問題となるのは，どのようすれば効率的に ALM アプリケーションを実装できるかということである．アプリケーションの設計において共通する機構や機能を含むにも関わらず，従来のようにマルチキャストに関するすべての機能を個々のアプリケーションごとに独立に開発して組み込むのは冗長であり，時間の浪費を生み出すことになる．

そこで本章では，開発上の冗長性を軽減し，最小限の労力で様々な ALM 機能をアプリケーションに組み込めるようにすることで開発効率を改善するためのミドルウェアを提供することを目的とする．これに向けて，*RelayCast* と呼ぶ機能ユニット指向 ALM ミドルウェアを開発する．RelayCast は，最小かつ基本的な ALM 機能の組をユニットとして提供し，また新たなアルゴリズムを柔軟に追加できる枠組みをもつ．ビデオや音声のキャプチャ，コーデック，マルチウィンドウなどといったそれぞれのアプリケーションに固有の機能は，RelayCast が築く通信基盤上に実装されていくことになる．

まず RelayCast を設計するにあたり，既存の ALM システムを鳥瞰した際に主として “オーバレイネットワーク構築機能” と “マルチキャストルーティング機能” という 2 つの機能を抽象化できるということに着目した．これらの機能のことを，ここでは “機能ユニット” と呼ぶ．既存の ALM システムでは，これらの機能ユニットの振る舞いがそれぞれ異なっていると言うことができる．たとえば，End System Multicast（ESM）[35] では，ホスト間の遅延をメトリックとしてリンクコストを計算し NARADA プロトコルによってメッシュ型のオーバレイネットワークを構築する．その上で shortest widest path アルゴリズムを用いてマルチキャストツリーを構築する．また Scattercast [36] では，同時接続数（リンクストレス）と遅延に応じてメッシュ型オーバレイネットワークを構築し，その上で DVMRP（distance vector multicast routing protocol）[51] と同様の振る舞いをするマルチキャストルーティングプロトコルを走らせる．

各機能ユニットは複数の “コンポーネント” を含む．コンポーネントとは，それぞれの機能ユニットのために個別に提案されたアルゴリズムのことである．上の例で言えば，ESM ではオーバレイネットワーク構築機能において，ホスト間の遅延をもとにリンクコストを計算するコンポーネントと，オーバレイネットワークのトポロジを決める NARADA プロトコルのコンポーネントを採用していると見なせる．また，マルチキャストルーティング機能においては shortest widest path アルゴリズムのコンポーネントを採用する．このようなアプローチは，IETF RMT WG（Reliable Multicast Transport Working Group）[52] において議論されたリライアブルマルチキャストのためのビルディングブロック [53] でも採

用されている．すなわち，コンポーネントはこのビルディングブロックに対応する概念である．

機能ユニットとコンポーネントという基本設計によって，開発者は RelayCast 上においてアプリケーション固有の機能や ALM システムのためのアルゴリズムを柔軟に追加することができる．しかしながら，上述した 2 つの主要機能ユニットにはいくつかの共有モジュール機能が存在するため，これらをさらに分割することによってミドルウェアをよりスマートにし，冗長性を排除することができる．その共有モジュール機能とは，他のホストとパケットを送受信する通信機能，他のホストに関する情報を蓄えるデータベース機能，メトリックとして用いられるネットワーク状態を測定する評価機能のことである．これら共有モジュール機能を 2 つの主要機能ユニットから切り離し，それぞれ機能ユニットとして新たに定義する．切り離されたユニットの機能は，ユニット間に定義されるインタフェースを通して利用されることになる．

本章では，上記の設計指針に基づいた RelayCast のプロトタイプ実装と，そのプロトタイプを用いてローカルエリアネットワーク上で行った実験についても述べる．この実験の目的の 1 つは，機能分離によって非実用的な性能低下が生じていないことを調べることである．実用的観点から，このような実装と実験を通して本ミドルウェアアーキテクチャの実用性を示すことは妥当であると考える．

本章での主な成果内容は以下の通りである．まず，ALM に関して 2 つの基本機能を抽象化し，共有モジュール機能を分割し，RelayCast と呼ぶ ALM ミドルウェアの設計を行った．RelayCast は，オーバレイネットワーク構築機能，マルチキャストルーティング機能，通信機能，データベース機能，評価機能の 5 つの機能ユニットから構成される．次に，この設計指針に基づき，RelayCast のプロトタイプを実装した．最後に，実験を通して RelayCast の実用性について検討した．

本章の残りの構成は次の通りである．3.2 節では，ALM 機能の抽象化と各機能に要求される基本コンポーネントについて述べる．3.3 節では，RelayCast のアーキテクチャを示す．3.4 節では，プロトタイプの実装について述べる．3.5 節では，初期実験の結果を示し，RelayCast の実用性について議論する．最後に 3.6 節にて本章のまとめを述べる．

3.2 アプリケーション層マルチキャスト機能の抽象化

3.2.1 アプリケーション層マルチキャスト

既存の ALM システムは，メッシュ優先手法 [15,35,36] とツリー優先手法 [37,39,41,42] の 2 つに分類される．メッシュ優先手法では，ホストのみを構成要素とする論理的なネットワーク，すなわちメッシュ型のオーバレイネットワークを構築し，その上でマルチキャストルーティングを実行することによってマルチキャストツリーを構築する．一方ツリー優先手法では，オーバレイネットワークを構築せず，直接にホストをノードとするマルチキャストツリーを構築する．本章では以下の理由からメッシュ優先手法を前提とする．

- 同一グループ内で複数のツリーを構築する際に，グループ管理を何度も繰り返し行う必要がない．
- 標準的なマルチキャストルーティングプロトコルを採用できる．
- ツリーの最適化が比較的容易であるため，グループ全体での品質保証を行い易い．
- メッシュ型のオーバレイトポロジを組織するので，ホストの故障に対してより耐性のある構造を提供できる．

ALM の動作は，1）ホスト発見，2）オーバレイネットワーク構築，3）マルチキャストルーティングという 3 つの基本機能から構成される．まずホスト発見では，あるマルチキャストセッションに参加しようとするホストが，そのセッションに属する他のホストを発見する．ホスト発見方法としては 3 つの方法が考えられる．すなわち，Web などの他の媒体を用いた広告からブートストラップホストを把握する方法，ディレクトリサーバを用いてセッションに参加する他のエンドホストのアドレスを把握する方法，ブロードキャストチャネルを用いてフラッディング的に検索を行う方法である．

次に，オーバレイネットワーク構築では，それぞれのホストはセッションに参加する他ホストのリストを獲得し，リストに含まれる複数のホストとの間に双方向論理リンクを確立する．このような論理リンクの集まりによって，オーバレイネットワークが構築される．その後，より良い接続環境を求めてそれぞれのホストは自律的にオーバレイネットワークの最適化を行う．その様子を図 3.1 に示す．ホストは定期的に他のエンドホストとの間の論理リンクのコストを計算し，コスト計算の結果に基づいて新たに有用な論理リンクを張り，あるいは無用と判断された論理リンクを切断する．既存の ALM システムでは，隣接ホストの選び方やリンクコストの計算方法のアルゴリズムが個々で異なる．

最後に，マルチキャストルーティングでは，オーバレイネットワーク上でホストをルータに見立ててマルチキャストルーティングプロトコルを実行することによって，データ配

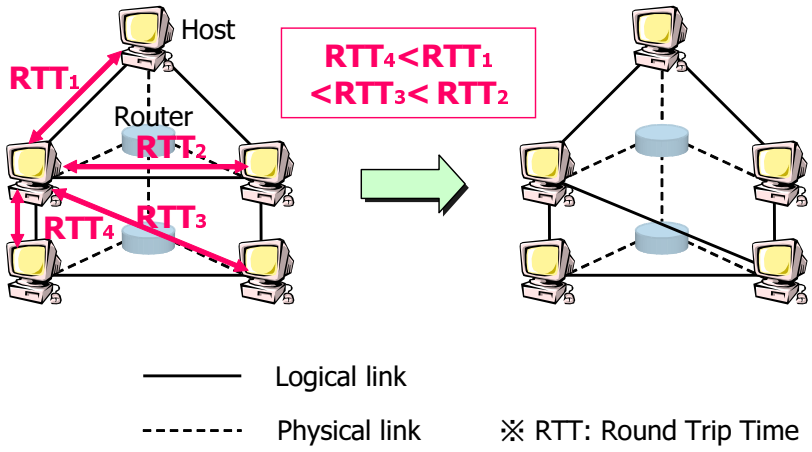


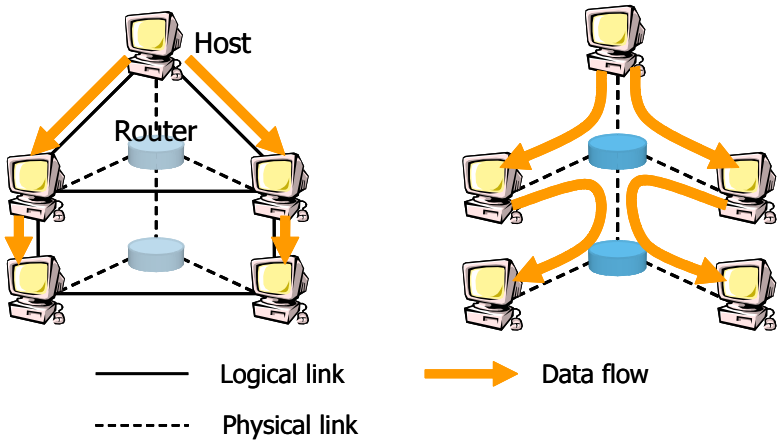
図 3.1: オーバレイネットワークの最適化

信のための論理的なマルチキャストツリーを形成する．その配信木の様子を図 3.2(a) に示す．このとき，物理ネットワーク上での実際のデータフローは図 3.2(b) のようになる．既存の ALM システムでは，それぞれ異なったマルチキャストルーティングプロトコルを利用する．

基本的な動作は以上の通りであるが，現在では多くの ALM 方式が提案されている [15, 35–50]．これらは ALM を実現するためのアルゴリズムについて提案しており，マルチキャストに関する機能が個々のサービスごとに独立に開発され，組み込まれているというのが現状である．

3.2.2 ALM ミドルウェア

ALM のアプリケーションを開発するにあたり，そのプロセスは若干複雑になり得る．その 1 つの理由は，開発者がルータとしてのホストの動作まで考慮しなければならないことである．つまり，アクセスリンクの輻輳やアプリケーションエラー，マシンクラッシュなどのホスト故障によってオーバレイネットワークとマルチキャストツリーの分断が生じた際に，それらを修復するコードを記述しなければならないということである．分断が生じると，ホストがデータを受け取ることができないのである．もう 1 つの理由としては，開発者が実装のために ALM システムに適用されるアルゴリズムの詳細を完全に理解しなければならないことが挙げられる．しかしこれは逆に，ALM がアプリケーションの要求に柔軟に対応できることも意味する．



(a) Logical data flow. (b) Physical data flow.

図 3.2: アプリケーション層マルチキャストにおける配信木

このような状況は，ALM のインターネットへの展開を妨げかねない．IP マルチキャストの場合，3.1 節で述べたように多くの問題を含んではいるが，開発者はソケットオプションを設定するだけでその機能を利用することができ，ルータの故障にまで考慮する必要はない．ALM もまた，アプリケーション開発の段階においては IP マルチキャストと同様の利便性を持つべきである．

上記の考察から，ALM ミドルウェアの導入によって ALM アプリケーションの開発，展開を助長することが期待される．ミドルウェアは 3.2.1 節で述べた 3 つの機能を提供するのが望まれる．一般に，ホスト発見機能は外部の発見機構を必要とするため，ミドルウェア単体だけで提供できる機能ではない．しかしながら，ALM を MyNetSpace 内におけるブロードキャストチャネルとして利用する場合，ホスト発見は MyNetspace がもつ端末管理機構によって達成されることとなる．そこで本章では，セッションに参加するホストはあらかじめ他のメンバを知っていると仮定する．したがって，オーバレイネットワーク構築機能とマルチキャストルーティング機能に関して抽象化を行い，ミドルウェアに組み込むこととする．

このようにアプリケーションの外側に ALM 機能を実装するシステムとしては，Yoid [37], PeerCast [54]，Overlay Socket [55] が提案されている．Yoid は，ツリー優先手法を採用した ALM システムの 1 つであり，YTMP（Yoid Tree Management Protocol）を用いてマルチキャストツリーを構築する．Yoid の実装では，*wrappers* というローカルプロキシの形でソフトウェアを構成することで，既存アプリケーションをサポートする．3.4 節で述べる

表 3.1: オーバレイネットワーク構築機能のコンポーネント

Component		Applied situation
Optimization Metric	Delay	To guarantee delay
	Bandwidth	To guarantee bandwidth
	Delay and bandwidth	To guarantee delay and bandwidth
Topology	Full flat	Resources of end-hosts are nearly equal
	Hierarchical	Rresources of end-hosts are unequal or Group is relatively large

RelayCast プロトタイプの実装では，Yoid の実装形態と同様の方式を採っている．PeerCast は，アプリケーションとトランスポート層の間にピアリング層を導入する．ピアリング層によって，マルチキャストツリーの最適化やホストの脱退によって生じるツリー再構築を，アプリケーションから隠す．

Yoid や PeerCast では，それぞれが提案するアルゴリズムをミドルウェアとして提供することを目的としている．そのため，アプリケーションからの多様な基本的要求に対応するためのアーキテクチャについては考慮されていない．これに対し本章では，3.2.3 節，および 3.2.4 節で述べるような ALM 機能の抽象化を行い，アプリケーションからの多様な基本的要求を満たすための機能ユニット指向なアーキテクチャを提案する．

Overlay Socket では，ALM 機能の抽象化を RelayCast と同様，すなわち 3.2.3 節，および 3.2.4 節や文献 [56] で述べるものと同レベルで行っている．しかしながら，Overlay Socket は提案する単一の ALM 手法 [40] を Socket API 上に組み込むだけに留まる．本章ではさらに，基本機能を機能ユニットとして定義し，各機能ユニットに対して個別に提案されたアルゴリズムをコンポーネントとして扱うことを検討した．RelayCast では，適切なコンポーネントを選択することで多様なアプリケーションからの基本的要求を満たすことができる．

3.2.3 オーバレイネットワーク構築機能の抽象化

オーバレイネットワーク構築機能の基本コンポーネントを表 3.1 に示す．この機能において，最適化メトリックを決定するアルゴリズムとオーバレイネットワークのトポロジを

表 3.2: マルチキャストルーティング機能のコンポーネント

Component	Applied situation
DVMRP	A few source end-hosts
PIM-SM/CBT	Many source end-hosts
Multi-path routing protocol	Continuous communication
Shortest widest path algorithm	Bandwidth guarantee and delay guarantee

形成するアルゴリズムがコンポーネントとなり得る．最適化メトリックは論理リンクのコスト計算のための基準となり，オーバレイネットワークトポロジは隣接ホストの選択方法に影響を与える．

最適化メトリックとして通常は，帯域幅か遅延，もしくはその両方が候補となる．帯域幅の測定には，pchar やパケットペアなどを用いる能動的測定法と，受信レートを測る受動的測定法がある．遅延の測定には，ホスト間のエンドツーエンドの RTT（Round Trip Time）を測定する方法 [15] と，ネットワーク上のランドマークまでの RTT を測定する方法 [57] などがある．

オーバレイネットワークトポロジとしては，完全フラット型トポロジと階層型トポロジが挙げられる．すべてのホストが対等な関係にある完全フラット型トポロジは，ホストがもつリソース（処理能力やネットワークの状況）がほぼ均一な場合に適している．また，階層型トポロジは，各ホストがもつリソースが不均一で，参加しているホストの数が比較的多い場合に適している．

3.2.4 マルチキャストルーティング機能の抽象化

マルチキャストルーティング機能の基本コンポーネントを表 3.2 に示す．この機能では，マルチキャストツリーを構築するためのアルゴリズム（すなわちルーティングプロトコル）がコンポーネントとなり得る．マルチキャストツリーの種類としては，最短木，共有木，複数経路木，SWPT（Shortest Widest Path Tree）が基本的なものとして挙げられる．

最短木は，DVMRP [51] と同様のプロトコルによって，オーバレイネットワーク上においてソースホストからの論理ホップ数を最小にするように構築される．最短木は送信者が特定の数人の場合に適している．共有木は，スーパーノード（参加ホストの代表）をラン

デブポイントとして利用し，PIM-SM [58]（Protocol Independent Multicast-Sparse Mode）や CBT [59]（Core Based Trees）と同様のプロトコルを実行することで構築される．この共有木はグループの全員が送信者となり得る場合には有効である．しかし，ALM ではルータと比較して信頼性の低いホストをランデブポイントとして利用するため制御は複雑になる．複数経路木は，マルチパスルーティングプロトコル [56, 60] を適用することにより実現される．このプロトコルでは，ホスト故障に伴うマルチキャストツリー切断時の再構築遅延を最小限にするための冗長パスを各ホストにあらかじめ与えておく．そのため，アプリケーションからの通信継続性要求が厳しい場合などに適する．SWPT は，エンドツーエンドの帯域幅が一定値以上となる経路の中から，ホップ数が最小となる経路を選択することにより構築される [35]．帯域幅も必要で，かつ遅延保証もある程度必要な場合に有効である．

3.3 RelayCast

3.3.1 アーキテクチャ

RelayCast の設計は基本的に次に述べるアプローチに基づく．まず ALM の基本機能を機能ユニットとして定義し，次にそれらのユニットから共有モジュール機能を分割する．3.1 節でも述べたが，基本機能のユニットとはオーバレイネットワーク構築機能とマルチキャストルーティング機能である．これらのユニットはいくつかのモジュール機能を共有する．すなわち，他のホストとパケットを送受信する通信機能，他のホストに関する情報を蓄えるデータベース機能，メトリックとして用いられるネットワーク状態を測定する評価機能である．これら共有モジュール機能をオーバレイネットワーク構築機能とマルチキャストルーティング機能から切り離し，それぞれ *P2PCom*，*HostList*，*MetricEstimator* という機能ユニットとして定義する．

また，共有機能を分離したオーバレイネットワーク構築機能とマルチキャストルーティング機能を，それぞれ *LogicNet*，*McastTree* という機能ユニットとして再定義する．LogicNet と McastTree はアルゴリズムが異なる複数のコンポーネントをもち，適切なコンポーネントを選択することによって多様なアプリケーションからの要求を満たすことを可能にする．

さらに，それぞれの機能ユニット間にインタフェースを定義する．機能ユニット間インタフェースとは，分割された他の機能ユニットやソケット API を利用するためのプログラミングインタフェースである．このインタフェースによって，アプリケーション開発者が RelayCast に新たにコンポーネントの追加するのを補助する．

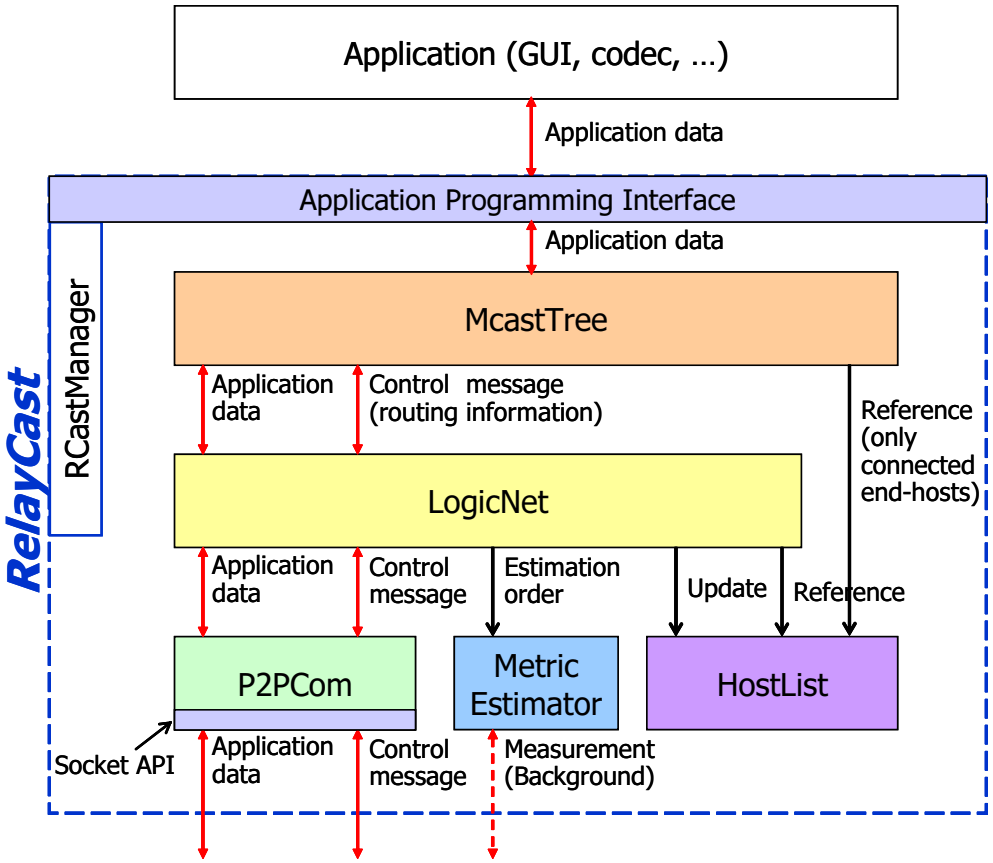


図 3.3: RelayCast アーキテクチャ

図 3.3 に RelayCast のアーキテクチャを示す．図で破線で囲まれた部分が RelayCast である．上述したように，RelayCast は 5 つの機能ユニットとそれらの間を結ぶインタフェースから構成される．片方向の矢印は API，または機能ユニット間インタフェースによる操作を表し，両方向の矢印はそれらの操作によってデータパケットのやり取りが生じることを表す．RCastManager はコンポーネントの選択などを記した設定ファイルを読み込み，それに従って各機能ユニットの管理を行う．また，RCastManager はマルチキャストセッションごとに 1 プロセスとして生成される．

3.3.2 機能ユニットとインタフェース

本節では，それぞれの機能ユニットの役割を述べ，ユニットがもつべきインタフェースについて示す.

McastTree

McastTree はマルチキャストツリーの構築と，それに沿ったパケットルーティングを行う．アプリケーション自身の転送データ，およびマルチキャストルーティングの制御メッセージ（ルーティング情報）を，下流のホストを指定して LogicNet 経由で P2PCom に渡す．ホストが送信者である場合には，シーケンス番号を記した（アプリケーションレベルの）ヘッダを付加する．各ホストはパケットに付加されているシーケンス番号をチェックすることにより，ループを防止する.

McastTree の機能ユニット間インタフェースとしては，論理リンク接続先のホストと制御メッセージ（ルーティング情報）を送受信する *SendMsg / RecvMsg*，接続されたホストとアプリケーションのデータを送受信する *SendDat / RecvDat*，接続されたホストの情報データベースを参照する *GetCnnHostInfo* が定義される.

LogicNet

LogicNet はオーバレイネットワークを維持し，最適化を行う．主な操作は，MetricEstimator から得られる測定結果に基づいて論理リンクの張替えを行うことである．LoginNet は，オーバレイネットワーク構築の制御メッセージ（ホストリスト，セッション参加/脱退，論理リンクの接続/切断）を，HostList から選択した対向のエンドホストを指定して P2PCom に渡す．LogicNet は，P2PCom から制御メッセージを受け取って処理し，さらに McastTree にその制御メッセージを渡す．アプリケーションのデータを処理する場合は，LogicNet は P2PCom または McastTree に対してデータをそのまま転送する.

LogicNet の機能ユニット間インタフェースとしては，制御メッセージ（オーバレイネットワーク維持，最適化，など）を送受信する *SendMsg / RecvMsg*，アプリケーションのデータを送受信する *SendDat / RecvDat*，他のホストの情報データベースを参照，更新する *GetHostInfo / PutHostInfo*，リンクコスト計算に用いるメトリックであるネットワーク状態の測定・評価を開始する *EstimateStart* が定義される．EstimateStart は，呼び出されると測定・評価をバックグラウンドで開始し，結果を *PutHostInfo* を用いて HostList に格納する.

MetricEstimator

MetricEstimator はネットワーク状態（遅延，帯域幅）を測定，評価する．通常では，測定プロセスはバックグラウンドで実行される．MetricEstimator には，RTT や帯域の様々な測定手法や，ping や pchar などの外部プログラムを他の機能ユニットとは独立して組み込むことができる．測定結果は LogicNet に渡されて，HostList に保存される．

MetricEstimator の機能ユニット間インタフェースとしては，ネットワーク状態を実際に測定・評価する *Estimate* が定義される．

P2PCom

P2PCom は P2P ネットワーク上で，セッションに参加している他ホストと直接に通信を行う．P2PCom は他ホストに対するソケットを持ち，LogicNet によって指定されたホストに対して制御メッセージ，およびアプリケーションデータを転送する．また，P2PCom には他の機能ユニットとは独立に，STUN [61] や Teredo [62] といった NAT 越え技術を組み込むことが可能である．

P2PCom の機能ユニット間インタフェースとしては，制御メッセージを送受信する *SendMsg / RecvMsg*，アプリケーションのデータを送受信する *SendDat / RecvDat* が定義される．

HostList

HostList はホストの情報のリストを管理するデータベースである．ホストの情報には，IP アドレスや GUID，ネットワーク状態の測定結果，状態（参加中/脱退済，論理リンクの有無）が含まれる．リストは，LogicNet によって更新され，LogicNet と McastTree の双方から参照される．

HostList はデータベースであるため，機能ユニット間インタフェースは定義されない．

3.3.3 組み合わせポリシ

RelayCast では適切なコンポーネントを選択することによって，アプリケーションの要求を満たす．マルチキャストセッションの開始時にそのオーナーがコンポーネントの組み合わせを決め，他のユーザはその決定に従う形をとる．

コンポーネントの選択の組み合わせは，ALM システムの性能を左右する要因の 1 つである．したがって，アプリケーションに対して要求に基づいた適切な選択の組み合わせを

表 3.3: 組み合わせポリシの例

Application	Overlay Network Construction	Multicast Routing
Streaming media	Bandwidth and delay (metric)	DVMRP, SWPA
Interactive application	Delay (metric)	Multiple DVMRP, Multiple SWPA, PIM-SM/CBT
Bulk-data transfer	Bandwidth (metric)	DVMRP, SWPA

(SWPA = Shortest widest path algorithm)

提供する必要がある．また，選択されたコンポーネントの衝突も回避しなければならない．たとえば，McastTree ユニットに遅延を保証するコンポーネントの適用が求められるにも関わらず，LoginNet ユニットで遅延を犠牲にするコンポーネントが適用されてしまった場合には，RelayCast 内でコンポーネントの衝突が生じてしまうだろう．これらを回避する最も単純な方法は，1 つ 1 つのコンポーネントをユーザが指定することである．しかしながら，それぞれのコンポーネントの役割について理解していないユーザでは，組み合わせとしては可能であるが，あまり効果的ではない組み合わせを選ぶ可能性がある．

もう 1 つの方法は，定性的なコンポーネント選択のポリシをあらかじめ決めておくことである．つまり，ユーザが通信モデルやグループ人数，利用アプリケーションなどのプロパティを選択すると，適切な組み合わせが RelayCast に適用される．特定のアプリケーションに対して，定性的に有効であると考えられるコンポーネントの組み合わせの 1 例を表 3.3 に示す．たとえば，帯域幅保証と遅延保証を要求するストリーム配信型アプリケーションの場合には，LoginNet ユニットの最適化メトリックに帯域幅と遅延の両方が選択され，McastTree ユニットのマルチキャストルーティングプロトコルに最短パスを構築する DVMRP や Shortest widest path アルゴリズムが選択される．また，ユーザが通信継続性を求める，すなわちホスト故障による配信木分断の影響を最小限に抑えたい場合には，マルチパスルーティングを選択することも考えられる．今後，コンポーネント選択のポリシを十分検討し，またこのポリシを提供する方法を検討する必要がある．

```
/** Control message header */
struct rcast_msg_hdr {
    unsigned int     sessionID;   // Multicast session identifier
    unsigned int     srcID;       // Host identifier of message sender
    unsigned short   msg;         // Control message number
    unsigned short   seqNum;      // Sequence number
};

/** Application data header */
struct rcast_dat_hdr {
    unsigned int     sessionID;   // Multicast session identifier
    unsigned int     srcID;       // Source host identifier
    unsigned short   seqNum;      // Sequence number
    unsigned short   chnl;        // Channel number
};
```

図 3.4: ヘッダ構造体の定義

3.4 実装

3.4.1 実装の概要

本章のこれまでに述べた内容に基づいて，Linux（kernel-2.4）上にて C++言語を用いて RelayCast プロトタイプの実装を行った．5 つの機能ユニットと RCastManager をそれぞれオブジェクトとしてクラスで定義し，機能ユニット間インタフェースをそのクラス内でのメソッドの形で実現した．現在のところ，MetricEstimator はホスト間の遅延（平滑化 RTT）を測定できる．LogicNet は完全フラット型のオーバレイネットワークトポロジを形成し，MetricEstimator で測定した遅延を最適化メトリックとして利用する．すなわち，LogicNet は遅延に基づいたメッシュ型オーバレイネットワークを構築することができる．また，ゴシップ型の情報流通を行う NameDropper アルゴリズム [36] を用いて，ホストの IP アドレスリストを各ホスト間で交換しあうこともできる．McastTree には，DVMRP 及びマルチパスルーティングプロトコルのコンポーネントを実装した．送信ホストごとにマルチキャストツリーを構築することにより，送信ホストが複数存在する場合，すなわち多対多通信にも対応する．

図 3.4 に，他のホストとやり取りする制御メッセージとアプリケーションデータに乗せるヘッダ構造体を示す．現在のところ，ヘッダは処理に必要な情報のみを含む．制御メッ

```
/** Session information **/
struct session_info {
    int          version;     // Version number of RelayCast
    unsigned int sessionID;   // Multicast session identifier
    unsigned int ownerID;     // Host identifier of session owner
    int          numSender;   // The number of senders
                              // (if 0xffffffff, all hosts are senders)
    id_list*     sendersID;   // Host identifiers of permitted senders
    short        numChnl;     // The number of channels
    short        cmpMetric;   // Optimization metric components (RTT|BW)
    short        cmpTopology; // Overlay topology components (FLAT|HIER)
    short        cmpRouting;  // Multicast Routing components
                              // (DVMRP|PIMSM|MP|SWP)
};
```

図 3.5: セッション構造体の定義

セージヘッダには，マルチキャストセッション識別子，送信元ホスト識別子，メッセージ番号，シーケンス番号が含まれる．アプリケーションデータヘッダには，セッション識別子，送信元ホスト識別子，シーケンス番号，チャネル番号を含む．チャネル番号は，同一のマルチキャストツリーに複数のデータフローを流す際のフロー判別に用いられる．

また，所望するマルチキャストセッションを開始する場合，そのセッションの開始者（オーナー）は図 3.5 に示す情報を設定ファイルに記述する．このファイルはプログラム起動時に読み込まれ，構造体に格納される．途中参加者はブートストラップとなるホストにアクセスしたときにこの情報を受け取り，それに従って設定を自動的に行う．現在の仕様ではホストの IP アドレスと識別子は全単射の関係にある．sessionID はセッションを区別するためのものであり，ランダムに割り当てられる．特定の送信者だけが遅れるようにするには，あらかじめ許可する送信者の識別子 senderID を指定しておく．1 つのマルチキャストツリーにいくつのデータフローを流すかを決めるには numChnl を指定する．最後に，どのコンポーネントを設定するのかを cmpMetric, cmpTopology, cmpRouting に指定する．

3.4.2 実装モデル

RelayCast の動作確認を行うために，アプリケーションには VIC（video conferencing tool）[63] を用いた．VIC は UDP ユニキャストを用いて 2 ホスト間で双方向通信を行うこ

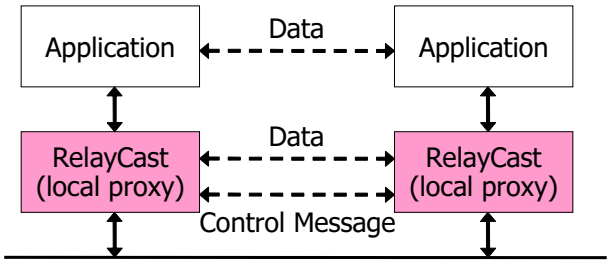


図 3.6: ローカルプロキシモデルによる実装

とができる．VIC のユニキャスト通信機能と RelayCast を用いて，ALM による多拠点ビデオ会議システムを構築する．実装では，VIC に RelayCast を組み込むのではなく，RelayCast をローカルプロキシとして実現する．

図 3.6 に，ローカルプロキシ型の実装モデルにおけるアプリケーションとミドルウェア（RelayCast）の関係の概念図を示す．このモデルでは，アプリケーションは直接的にはローカルホスト内でミドルウェアとだけ通信を行う．ミドルウェアは配信木を構築して，マルチキャストセッションに属する全ホストにアプリケーションからのデータを転送する．このモデルの利点は，既存のアプリケーションに対してほとんど変更を要しないこと，また個々のアプリケーションからは完全に分離した形でデバッグできるのでミドルウェアの開発に注力することができることである．逆に欠点は，2 度もソケット API を通してパケットを処理しなければならないために性能が制限されることである．

プログラム起動時には以下のように，マルチキャストセッションに参加しているホストの IP アドレスリストと設定ファイルをオプションで指定し，コマンドを入力する．今回の実装では，マルチキャストセッションごとに 1 つローカルプロキシを起動するように設定している．

```
% > rcast -l [IP address list] -f [configuration file]
% > vic localhost/[middleware port]
```

次に，ミドルウェア内部の実装フレームワークについて議論する．フレームワークの設計には，キュー制御型のルータモデルとイベント駆動型のスイッチモデルがある．ルータモデルでは，パケットは一旦キューにバッファリングされ，送信スレッドと受信スレッドが独立にキューの読み書きを行う．スイッチモデルでは，パケットが到着すると即座に，そのパケットの受け取り，ルーティングテーブルの参照，それに従ったパケットの転送と

いう一連の操作が実行される．スイッチモデルでは一般的に高いスループットを出すことが可能であり，高速な転送能力をもつオーバレイネットワークインフラストラクチャ構築を目指す RON [13] ではこのスイッチモデルを採用している．したがって，RelayCast もまた高速な処理性能を重視し，内部の実装フレームワークにはスイッチモデルを採用する．

3.4.3 機能ユニット間インタフェースと API の定義

図 3.7 に実装した機能ユニット間インタフェースの定義一覧を示す．それぞれのインタフェースの役割については，3.3.2 節において既に述べているため，ここでの説明は割愛する．なお，McastTree クラスと LogicNet クラスにおいて引数とされている char *data は，すでに図 3.4 に示したヘッダを含むものとする．McastTree.GetCnnHostInfo() 関数の引数は，ホストの識別子を指定するのではなく，論理リンクを確立しているホストを $0 \leq$cnn_host_num$< CONNECT_MAX$（最大接続数）の番号で指定することで，接続しているホストの情報を知る．また，MetricEstimator.Estimate() 関数の引数である char opt には，実際にネットワーク状態を測定する項目を指定する．現在のところ，‘0’ ならば遅延と帯域の両方，‘1’ ならば遅延，‘2’ ならば帯域をそれぞれ測定する．

また，アプリケーションから RelayCast を利用するためのインタフェースとして，図 3.8 に示す API を定義した．ただし，本実装ではローカルプロキシモデルを採用しているため，アプリケーションから直接 API を叩くのではなく，ローカルプロキシから API を呼び出す形となっている．実際には API が受け取ったパケットをプロセス間通信を用いて各 RCastManager プロセスに渡す形を取るのが望ましいが，本実装ではローカルプロキシに組み込む形を取り，同一プロセス内でのポインタ渡しによるメモリコピーをするにとどまる．これはローカルプロキシモデルによる性能劣化を相殺するための措置であることを明記しておく．

マルチキャストセッションを指定して RCastManager を作成する場合には，CreateRCastManager() という API を呼ぶ．引数には参加ホストの IP アドレスリストファイルと設定ファイルを指定する．オーナーとして新たにセッションを立ち上げる場合には，設定ファイルには 3.4.1 節で述べたセッション情報を記述する必要がある．返値は 32 ビットの RCastManager 識別子であり，これ以外の API の呼び出しにはこの識別子を指定する．

セッションのグループに参加，脱退する場合には，JoinGroup() / LeaveGroup() という API を呼ぶ．また，アプリケーションが RelayCast を通してデータを送受信する場合には，SendDat() / RecvDat() という API を呼ぶ．

```
/** class McastTree **/
unsigned short SendMsg(unsigned int dst_id, char *data, size_t dlen);
unsigned short RecvMsg(unsigned int *snd_id, char *data, size_t *dlen);
ssize_t SendDat(unsigned int dst_id, char *data, size_t dlen);
ssize_t RecvDat(unsigned int *fwd_id, char *data);
HostInfo* GetCnnHostInfo(int cnn_host_num);

/** class LogicNet **/
unsigned short SendMsg(unsigned int dst_id, char *data, size_t dlen);
unsigned short RecvMsg(unsigned int *snd_id, char *data, size_t *dlen);
ssize_t SendDat(unsigned int dst_id, char *data, size_t dlen);
ssize_t RecvDat(unsigned int *fwd_id, char *data);
HostInfo* GetHostInfo(unsigned int id);
int PutHostInfo(unsigned int id, HostInfo *hinfo);
int EstimateStart(unsigned int id, MetricInfo *minfo);

/** class MetricEstimator **/
int Estimate(in_addr ip, MetricInfo *minfo, char opt);

/** class P2PCom **/
ssize_t SendMsg(sockaddr_in *addr, char *buf, size_t len);
ssize_t RecvMsg(sockaddr_in *addr, char *buf);
ssize_t SendDat(sockaddr_in *addr, char *buf, size_t len);
ssize_t RecvDat(sockaddr_in *addr, char *buf);

/** classHostList **/
// not defined.
```

図 3.7: 実装した機能ユニット間インタフェースの定義一覧

```
/** API **/
rcast_manager_id CreateRCastManager(FILE *iplist, FILE *config);
bool JoinGroup(rcast_manager_id rcmngr_id);
bool LeaveGroup(rcast_manager_id rcmngr_id);
ssize_t SendDat(rcast_manager_id rcmngr_id, char *data, int dlen);
ssize_t RecvDat(rcast_manager_id rcmngr_id, char *data);
```

図 3.8: 実装した API の定義一覧

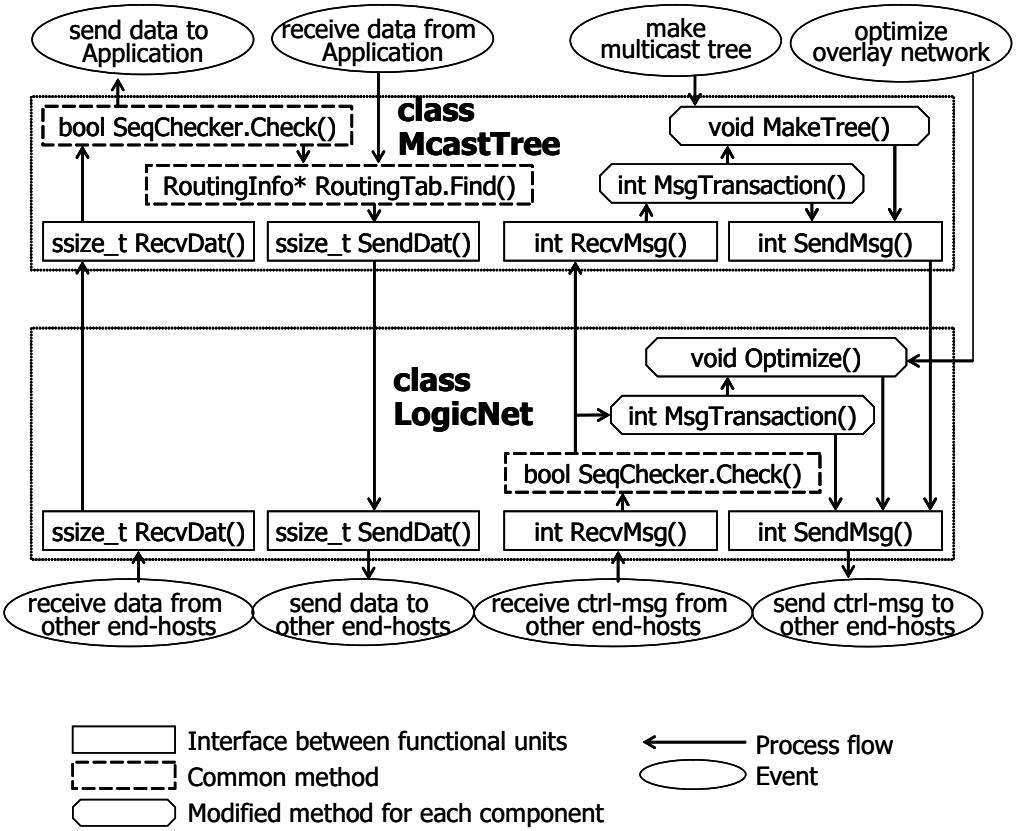


図 3.9: LogicNet クラスと McastTree クラスにおけるイベント処理フロー

3.4.4 イベント処理

図 3.9 に，イベント処理の基礎となるフレームワーク，すなわちイベントが発生してから LoginNet クラスとと McastTree クラスの一連のメソッドによって処理されるプロセスフローを示す．発生するイベントとしては，アプリケーション及び他のエンドホストからのアプリケーションデータの受信，他のエンドホストからの制御メッセージの受信，オーバレイネットワークの最適化，マルチキャストツリーの構築がある．

他のエンドホストからアプリケーションデータのパケットを受信した場合，まずパケットを LogicNet から McastTree まで上げ（LogicNet.RecvDat() → McastTree.RecvDat()），そのパケットのソース ID とシーケンス番号を調べ（McastTree.SeqChecker.Check()），番号が重複しているなら破棄する．次に，ローカルホスト内のアプリケーションに対してそのパケットを送る．そして，ソース（送信元エンドホスト）の ID をインデックスとしてルーティングテー

ブルを引き（McsatTree.RoutingTab.Find()），ルーティング情報（RoutingInfo）を得る．その情報に基づいて転送先エンドホストを決定してパケットを転送する（McastTree.SendDat() → LogicNet.SendDat()）．

アプリケーションからアプリケーションデータのパケットを受信した場合，ソース ID をインデックスとしてルーティングテーブルを引き（McsatTree.RoutingTab.Find()），ルーティング情報に基づいて転送先エンドホストを決定してパケットを転送する（McastTree.SendDat() → LogicNet.SendDat()）．

他のエンドホストからコントロールメッセージを受信した場合，まず LogicNet で受信し（LogicNet.RecvMsg()），シーケンスチェックを行い（LogicNet.SeqChecker.Check()），そのメッセージに対し適切な処理を行う（LogicNet.MsgTransaction()）．このとき，メッセージが LogicNet 用であれば処理される．必要であれば，他のエンドホストに対しメッセージを送信または転送を行う（LogicNet.SendMsg()）．その後，メッセージは McastTree に上げられ（McastTree.RecvMsg()），McastTree 内で処理される（McastTree.MsgTransaction()）．

オーバレイネットワークの最適化を行う場合，まず最適化のための計算を行う（LogicNet.Optimize()）．具体的には，メトリックを用いてリンクコストを計算し，オーバレイトポロジ形成アルゴリズムによってどの論理リンクを切断し，どのエンドホストと論理リンクを張るかということを計算する．この計算に基づき，他のエンドホストに対して論理リンクの接続，切断のメッセージを送信する（LogicNet.SendMsg()）．

マルチキャストツリーの構築を行う場合，ルーティング情報を含んだメッセージを作り（McastTree.MakeTree()），他のエンドホストに対してメッセージを送信する（McastTree.SendMsg() → LogicNet.SendMsg()）．また同時に，ルーティングテーブル（RoutingTab）の情報を更新する（McastTree.MakeTree()）．マルチキャストツリーの構築イベントは，ルーティングテーブルの更新時間間隔が経過したとき，及び他のエンドホストからルーティング情報を含んだメッセージを受信したときに発生する．

3.4.5 コンポーネント選択

実装では，適切なコンポーネントの選択には特定のメソッドに変更を加えることで対応する．つまり，メソッド内の対応する部分の手続きを切り替えることに相当する．図 3.9 において八角形のブロックで示したメソッドが，選択されたコンポーネントごとに変更を加える必要のあるメソッドである．

最適化メトリックとオーバレイネットワークトポロジに関するコンポーネントを選択する場合，これらが直接関係する void LogicNet.Optimize() での処理を主に変更する．また，

表 3.4: 各コンポーネント実装に要したソースコードの行数

Total	**4523** lines
Basic Framework	3656 lines
Full-flat topology	97 lines
Delay Metric	*347 lines
DVMRP	131 lines
Multi-path Routing Protocol	292 lines

(*including the RTT measurement method in the MetricEstimator unit.)

オーバレイネットワークトポロジを決定するアルゴリズムに特別な制御メッセージを用いる際には，それに応じたメッセージ処理の記述を int LogicNet.MsgTransaction() に追加する必要がある．

マルチキャストルーティングプロトコルのコンポーネントを選択する場合，void McastTree.MakeTree() での処理を主に変更する．このメソッドは，他ホストとルーティングメッセージを交換したり，ルーティングテーブル（class RoutingTab）がもつルーティング情報（class RoutingInfo）を更新したりする．ルーティングプロトコルにおいて特別な制御メッセージを用いる際には，やはり，それに対応するメッセージ処理の記述を int LogicNet.MsgTransaction() に追加する必要がある．

表 3.4 に基本フレームワークとそれぞれの主なコンポーネントの実装に要したプログラムソースコードの行数を示す．単純なソースコード行数の比較がミドルウェアの定量的評価とはならない．しかしながらこれを示すことは，ソフトウェア開発者に対する本ミドルウェアのアーキテクチャ的優位性を表す大まかな基準となり得るであろう．表 3.4 によれば，実装におけるそれぞれのコンポーネントのソースコード行数は，最大でも全体の 10%程度である．ただし，ここで示したコード量は最小限のエラー処理しか含んでいないものであり，単に指標として示したものであることに留意されたい．

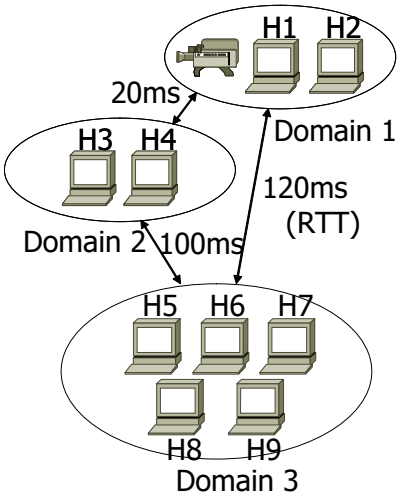


図 3.10: 実験ネットワークトポロジ

3.5 動作検証

3.5.1 実験の構成

本節では，ローカルエリアネットワークにおいて行った実験の結果について示し，提案したミドルウェアの実用性について検証を行う．実験では，サービスモデルとして 1 対多のライブストリーム配信を想定する．本来は多対多通信をサポートしているが，マルチキャストツリー構築の解析を容易にするためにソースホスト（送信者）は 1 台に限定する．

実験トポロジは，図 3.10 に示されるように 9 台のホスト Hi（$i = 1,2,\cdots,9$）によって構成される．H1∼H4 のスペックは，CPU が Pentium III 700MHz，メモリが SDRAM 512MB である．また H5∼H9 のスペックは，CPU が Pentium IV 2.0GHz，メモリが SDRAM 512MB である．実際の環境ではホストが属するドメインが異なっていることが十分想定される．そのため，実験ではドメインを 3 つに分けて，実験トポロジに dummynet ルータ [64] を導入してドメイン間に遅延を発生させる．

実験で選択したコンポーネントを表 3.5 に示す．オーバレイネットワークの最適化メトリックには遅延を採用し，トポロジには完全フラット型を採用する．ただし，各ホストが確立できる論理リンクの上限を 5 本に制限し，ソースホストである H1 のみ上限を 3 本にした．マルチキャストルーティングプロトコルとしては，DVMRP とマルチパスルーティングプロトコルを選択する．DVMRP のルーティングテーブル更新時間間隔を 60 秒とし

表 3.5: 実験において適用したコンポーネント一覧

Functional unit	Component
LogicNet	Optimization metric: Delay Topology: Full flat
McastTree	DVMRP & Multi-path routing protocol

た．また，マルチパスルーティングプロトコルの傾き [56] の最小値を 0.1 とした．ルーティングメトリックはどちらも，オーバレイネットワーク上における H1 からの論理ホップ数とした．

実験では H1 から順にプログラムを起動させる．H1 からは固定ビットレートで 1.5 Mbps のビデオストリームを送信する．検討事項として，構築されるマルチキャストツリーと各ホストで測定される受信スループットを示す．

3.5.2 実験結果

図 3.11 に受信スループットの変化が安定した状態（実験開始後約 350 秒）におけるマルチキャストツリーを示す．また，各ホスト（H4, H7∼H9）で測定された受信スループットを図 3.12 に示す．プログラムを起動させると，最初の論理リンクの確立後 2∼3 秒でデータの受信が可能となる．ソースの次ホップに常に位置した H2∼H4 では安定して受信が可能であったが，それらの下流に位置する H5∼H9 ではオーバレイの最適化を行う際に不安定になることがあった（実験開始後 230∼267 秒の間）．なお，オーバレイネットワークの最適化に関しては，集中的に計算を行っているのではなく，各エンドホストが自律分散的に計算しているので，必ずしも最適解とはならないことに注意したい．図 3.11 の状態において，H3 と H4 における転送のオーバヘッドは 10ms 程度であった．

次に，図 3.11 の状態でエンドホスト H2 と H4 を強制終了させた（実験開始後 363 秒）．マルチキャストツリーの下流側に位置する H7∼H9 は瞬間的に十分なスループットを得られなくなるが，3∼5 秒で復旧できたことが分かる．このとき，マルチパスルーティングによって再構築された直後のマルチキャストツリーを図 3.13 に示す．H8 と H9 は，上流となるエンドホストをそれぞれ H6 と H5 に切り替えている．その際，ホストにおける受信スループットがゼロに陥ることはない．しかしながら，各エンドホストの受信スループッ

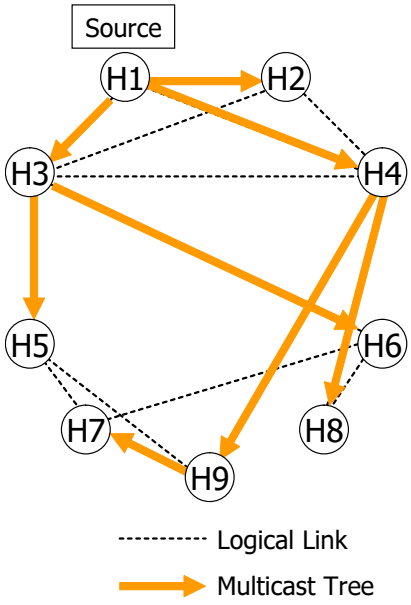


図 3.11: 安定状態におけるオーバレイネットワークとマルチキャストツリー

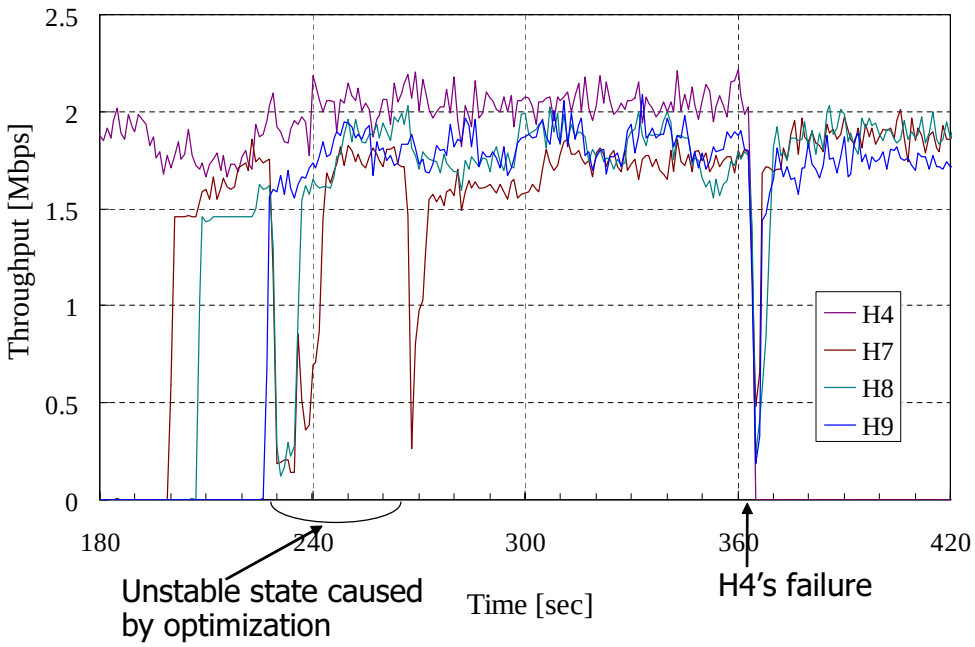


図 3.12: 各ホストで測定された受信スループット

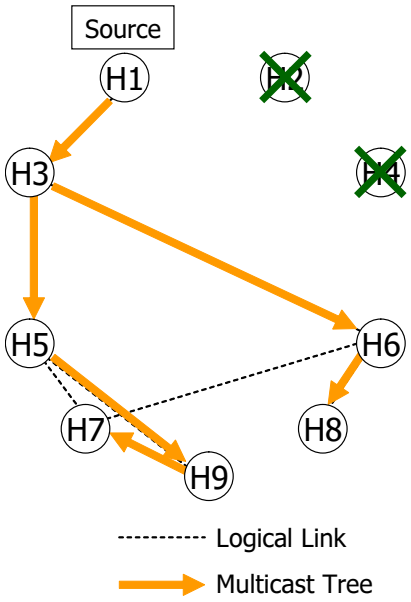


図 3.13: ホスト故障発生直後において再構築されたマルチキャストツリー

トは 1.5~2.0 Mbps となり，ストリームのレートよりも若干多くなっている．

3.5.3 考察

RelayCast の実現化の性について議論する．実装では，まず図 3.9 に示した基本フレームワークを構成した．それからそのフレームワークに，3.4.5 節で述べたように特定のメソッドを変更することによって DVMRP やマルチパスルーティングプロトコルなどのコンポーネントを組み込んでいった．また実験では，McastTree ユニットのマルチキャストルーティングコンポーネントとしてマルチパスルーティングプロトコルを採用した場合のアプリケーションの動作について検証した．そして，性能上のボトルネックが RelayCast によって生じていないことを確認した．ここでボトルネックとは，パケットがホストを経由するときにその内部で発生する処理オーバヘッドが非実用的に大きいこと，コンポーネントとして組み込んだアルゴリズムの動作が阻害されないこと，ミドルウェアの転送レートがアプリケーションが発生するデータの転送レートよりも低くなることを意味している．

本章では，ALM ミドルウェアを用いることで ALM の性能を改善するのではなく，ミドルウェアを設計し，開発することを目的としている．したがって，上で述べてきたような

アプリケーション開発と 1 つの実験シナリオをもって，RelayCast 上におけるアプリケーション開発プロセスと実際の振る舞いを検証するのは十分な妥当性をもつ．

以上の考察から，提案したミドルウェアアーキテクチャがアプリケーション開発において汎用性と実用性をもつと言える．それゆえ，RelayCast は開発における冗長性を削減し，開発者が最小限の労力で様々な ALM アプリケーションを効率よく実装することを助長することができるであろう．

3.6 おわりに

本章では，ALM を用いて特定グループ内における適応的なリアルタイムブロードキャスト配信を行うためのミドルウェアとして RelayCast を提案し，そのアーキテクチャと実装，および動作検証ついて述べた．RelayCast は，オーバレイネットワーク構築機能とマルチキャストルーティング機能という ALM に関する基本機能を提供する機能ユニット指向ミドルウェアアーキテクチャをもつ．また，提案した RelayCast のプロトタイプをローカルプロキシの形で実装し，動作検証を通してその振る舞いと有効性を確認した．現在のところ検証を行っているアプリケーションはライブストリーム配信であるが，再送制御やキャッシングといったコンポーネントを組み込むことによって，RelayCast はより幅広いアプリケーションに対して適用できると考える．今後の課題として，マルチキャストセッションの記述，オーバレイネットワークのモニタリングなどのより高度な API の導入，アプリケーションに適したコンポーネントの選択ポリシの決定方法について検討を進めている．

第4章

/インフォーマルコミュニケーション支援プレゼンス機構

4.1 はじめに

第 2 章，および第 3 章では，ユーザ主導型ネットワークサービスを実現するにあたり，ネットワークとしてどのようなフレームワークを具備するべきかということについて論じてきた．これまで述べたようにユーザ主導でネットワークを管理する環境では，ユーザ同士でアプリケーション／サービスやデバイス，端末を共有し，協調作業を行うようになるであろう．その際，ユーザ間においてどのようにコミュニケーションを成立させて，協調作業を円滑に開始させるかということが重要となる．ネットワークを介した遠隔環境においては，通常のように顔を合わせてコミュニケーションをとることが非常に難しくなってしまうため，これを技術的にどのように支援していくかが課題となる．

協調作業を円滑に行うために，従来から様々な方法が研究・提案されてきている [65,66]．それらの多くは「いかにしてネットワークを用いることで物理的な距離をのりこえるか」「いかにして多くの情報を伝えることでスムーズなコミュニケーションを行うか」ということに主眼が置かれている．そのような研究も重要ではあるものの，情報が多く伝わりすぎることのデメリットや，ネットワークにより距離が短縮されすぎることのデメリットについても考慮する必要があろう．

地理的に離れた場所にいる同僚と何らかのコミュニケーションシステムを用いて打ち合わせを行う際には，音声に加えて映像も用いた方が，また映像の解像度やフレームレートはより高い方が質の高いコミュニケーションを行うことができるというのは自明である．このような,「必要なときだけつなぐ」システムにおいては，伝える情報量を増やすことはその目的と良く合致する.「必要なときだけつなぐ」という利用形態は，離れた場所にいる相手との会議や打ち合わせなどといったフォーマルなコミュニケーションを行うのには適している．

一方，インフォーマルなコミュニケーションは偶発的であり，相手も話題も実際にコミュニケーションが始まるまで不定であるという特徴をもつ [67]．また，日常のインフォーマルなコミュニケーションを重ねることで人間関係はより親密なものとなり，またそのようなコミュニケーションの中から有益なアイディアなどが生まれることも考えられる．さらに，チャットなどのコミュニケーションをとるに至らないまでも，離れた場所にいる仲間がどんなことをしているのか，あるいは別の部屋はどんな雰囲気なのかということが “なんとなくわかる” ことで日常生活を円滑に行うことが期待できる．ほかにも，大学の研究室や会社のオフィスにおいては，そこに所属するメンバーは固定ではなく，年に何回か新人が加わるというのが一般的である．そのような新人と既存のメンバーが協調作業を行うためには信頼関係の醸成が不可欠であり，そのためには普段のインフォーマルなコミュニ

ケーションを積み重ねると共に，日常生活の様子といったものを伝えることで「つながっている感」をもたせることが有意義であると考えられる．またこのような「つながっている感」を重視する傾向は，インターネットの発展と共に成長してきた若い世代で顕著であることが報告されており [68, 69]，このような傾向は今後も続くものと考えられる．

上記のような特徴をもつインフォーマルコミュニケーションの支援においては,「必要なときだけつなぐ」のではなく,「常につないでおく」というスタイルが必要である．そして常につないでおくことで，ユーザ自身やユーザの周囲の状況をプレゼンス情報として他のユーザに広告することができると考える．このような観点から，本研究ではプレゼンス情報を適切な形で常時ユーザに呈示することで，前述の「つながっている感」をユーザが感じ取ることができると共に，適切な時機に適切な話題でコミュニケーションを行うことが期待できるプレゼンスの活用に注目する．

プレゼンス情報として自身の状況を常時他のユーザに広告することを考えた場合，情報量の多い通信手段が優れた通信手段であるとは必ずしも言うことができない．確かに，情報を受け取る側にとっては情報量が増えることは歓迎すべき事である．音声だけよりも映像が付加されていた方が，発信者側がどのような状況下に置かれていてどのようなことをしているのかと言うことが詳しくわかる．しかし，一般的な双方向のコミュニケーションにおいては受信者は同時に発信者でもある．発信者の立場から見れば，映像のような情報量の多い媒体で常に自分の状況を他のユーザに広告されるのはプライバシや心理的な観点から非常に抵抗が大きい．また，たとえば自分の位置情報を他のユーザに公開することを考えた場合，相手に応じてどの程度細かく自分の位置情報を伝えるかを変えたいという要求も考えられる．

このように，常につないでおくことでいわばコミュニケーションを行うことへのハードルを下げることを目的とする場合には，要求される情報量や情報の粒度が，フォーマルなコミュニケーションを目的として必要に応じてつながれる場合と異なってくる．そこで，本研究では適切に調節された情報量をもつプレゼンスの伝達を通して，ユーザのインフォーマルなコミュニケーションの支援を行うことを目指す．プレゼンスの種類は極めて多岐に渡り，それらを全て利用することは現実問題として不可能である．そのため，どの種のプレゼンスを利用するかという点が問題になる．究極的にはある 1 種のプレゼンスを伝えることで本研究で目標としているようなコミュニケーションの支援ができることが理想である．本研究の一環としてそのプレゼンスがどのようなものであるかについて検討を行っているが，現時点ではその解を見いだすに至っていない．さらなる検討を有意義なものとするためには，どのような情報をプレゼンスとしてどのような形で他のユーザへと伝達することがインフォーマルなコミュニケーションの支援に効果的であるかを，実際に試用して

いく中で見極める必要がある．以上に鑑み，本章では様々な種類のプレゼンス情報を扱うことのできるシステムの形態について検討を加えると共に，実際に ACDC（Atmosphere Carrier for Daily Communication）システムというプレゼンス情報伝達プラットフォームの構築について示す．また，いくつかのプレゼンス情報を利用できるよう実装したアプリケーションを示し，プレゼンスがコミュニケーションに影響を与えうるかという初動的な実験について報告する．

本章の残りの構成は次の通りである．4.2 節では，プレゼンス情報を用いてインフォーマルコミュニケーションを支援するシステムを検討する際の留意点について説明する．4.3 節では，プレゼンスシステムの実現手法について述べる．4.4 節では，プレゼンスシステムとして実装したテストベッド，およびアプリケーションについて示す．4.5 節では，実装したプレゼンスシステムを用いて初動的な評価実験について述べる．最後に 4.6 節でまとめと今後の課題を述べる．

4.2 インフォーマルコミュニケーションとプレゼンス

4.2.1 プレゼンスの実現

一般的に言えば，プレゼンスとは “ユーザの状況を伝える情報” と定義することができる．ユーザが置かれている環境についての情報を伝達することで，ネットワークコミュニケーションを円滑に行おうというのがその目的である．そのようなユーザを取り巻く環境についての情報というものは，通常のフェイストゥーフェイスのコミュニケーションにおいては各自が意識的に，あるいは無意識のうちに，様々な五感によって把握しているものである．具体的には，以下のようなタイプに分類することができる．

- 相手が今何をやっているか，どのような場所にいるかと言うようなそれぞれの人間に関する客観的な事実としてとらえられる事柄.
- 暇そうかあるいは忙しそうか，何を考えているのか，機嫌はいいのか悪いのかといったような人間の内面に関するもの.（これらの情報は，一般には他人は五感により観察された事柄をもとに主観に基づいた推測を行う）
- 周囲の明るさや温度など，各々の人間の周囲の状況を表し，場合によっては複数の人間で共通の値となるもの.

フェイストゥーフェイスのコミュニケーションにおいては各々の人間がごく自然に取得し，理解しているこれらの情報により，コミュニケーションの開始のタイミングが調節さ

れたり，場合によっては話題の提起も行われる．具体的な例としては,「相手が忙しそうな場合には急ぎでない用件であったら後で話すことにする」ということや,「機嫌が悪そうだからくだらない軽口を叩くのはやめておこう」といったことは我々がしばしば経験することである．

しかし，ネットワークコミュニケーションにおいては，これらの情報はコミュニケーションを開始する前には得ることができない．勿論，コミュニケーションを開始した後ではコミュニケーションに用いるメディアに応じて様々な情報を得ることができるが，そもそものコミュニケーションを行うかどうかという状況判断にはそれらの情報は用いることができない．その結果として,「忙しいときにどうでもいい話題で話しかけられる」「席を離れている間にビデオチャットに誘われる」というような，話し手（caller）と受け手（callee）との間での状況認識の不一致により双方に不利益をもたらすような事態が生じる可能性が考えられる．このような事態は，ネットワークコミュニケーションの進歩により，コミュニケーションメディアとして同期性をもつものが大きく台頭してきたことによりもたらされ，旧来の E-mail によるコミュニケーションの時代にはあまり考慮されることのなかった新しい問題である．

このような caller・callee 双方への不利益を解決するための手段として考慮されているのが，いわゆるプレゼンス情報である．フェイストゥーフェイスのコミュニケーションの場合にはコミュニケーションを始める前に様々な形で取得されうる情報を，ネットワークを通じて「プレゼンス情報」としてあらかじめコミュニケーションが開始される前にネットワーク上の周囲のユーザに広告しておくことで，ネットワークコミュニケーションにおいてもフェイストゥーフェイスのコミュニケーションと同様に，ユーザがコミュニケーションを行う機会や話題についての判断材料として用いようとするものである．ユーザの周囲に偏在するあらゆる情報をネットワークを用いて伝送し，離れた場所にいるユーザに適切な形で呈示することにより，離れた場所にいるユーザがどのようなことを行っているのかということはもとより，ユーザがそこに存在していると言うこと自体を「存在感」として「何となく」伝えられることが期待できる．

しかし，一言で「ユーザに関する情報」と言ってもその種類は極めて多岐にわたる．また，ネットワークによって伝達を行うことが前提であるので扱う情報は計算機が扱える電子情報である必要があるが，我々の身の回りに存在する様々な情報のうちそのまま計算機で扱える形で存在しているのはごく一部である．ほとんどの情報はセンサなど何らかの方法によって電子化することではじめて計算機で扱い，ネットワークを用いて伝送することができるようになる．また，情報の種類によってはセンサなどを用いても直接得ることができず，得られたデータと過去のデータから推定によってしか求められない情報というも

のも考えられる．また，そのように多岐にわたる膨大な情報をどのような形でユーザに呈示するのかと言うことも大きな問題であると言える．

このように，プレゼンス情報のネットワークを介したやりとりが実現されることによって，ネットワークコミュニケーションはより豊かなものになり，フェイストゥーフェイスのコミュニケーションと同等なものになることが究極的には期待される．しかし，そのためには解決しなくてはならない問題が数多く存在し，それらを解決することではじめて前述のような目標が実現され得る．

4.2.2 従来におけるプレゼンス情報の活用

プレゼンス情報については，現在様々な場面での利用が検討されているが，その中でもコンピュータ支援型協調作業（CSCW: Computer-Supported Cooperative Work）と呼ばれる分野において特に注目を集めている．CSCW とはその名の通り，複数の人間による協調作業を計算機を用いることで支援しようという取り組みである．CSCW に分類される取り組みとしては数多くのものがあるが，それらは大きく二つに分類することができる．すなわち，協調作業を行うユーザ同士が物理的にすぐそばにいる場合と遠く離れた場所にいる場合である．これらのうち，プレゼンスが大きく関係してくるのはユーザ同士がフェイストゥーフェイスのコミュニケーションを行うことが不可能な後者である．

協調作業というのは，作業形態がどういうものであれ作業者同士がコミュニケーションを行うことが必要不可欠である．そこで，ユーザ同士のコミュニケーションをより豊かにするための方法としてコミュニケーションを行う際により多くの情報を伝えることで目的を果たそうという試みは割合古くから行われてきた [70–72]．これらの試みでは，コミュニケーションのメディアとしてテキストだけではなく音声や映像をあわせて用いるとともに，ユーザを映した画像を周囲のユーザに対して常に公開しておくことで離れた場所にいるユーザ同士を結びつけることも行っている．しかし，これらの試みが行われてから久しいが，今日このような方法が広く一般に広まっているとは言い難い．

このような方法の普及を妨げている要因として，Ellen Isaacs らは以下のような点を挙げている [73].

- たとえ相手ユーザの姿が映像で映し出されても，相手が何に集中しているのかそれだけではわからないという問題
- 各ユーザごとにカメラを設置することなどコスト面の問題
- 常に自分の姿を他人に見られてしまうというプライバシの問題
- 純粋に技術上の実装の難しさの問題

これらの観点から，CSCW の分野においては実際にコミュニケーションを行うメディアとして，音声や映像ではなくインスタントメッセージの利点が見直されつつある [74].

一般的に ICQ [75] に代表されるインスタントメッセンジャでは，ユーザ間であらかじめ通知しておいた ID を使用して，ユーザ同士がテキストベースのリアルタイムメッセージ交換によるコミュニケーションを行う．最近では MSN Messenger [76] 等の様に音声通信やビデオチャットなどをサポートしてよりリッチなコミュニケーションを行うことを目指すものもある．これらのインスタントメッセンジャでは，ユーザが各自の端末上でクライアントソフトウェアを起動してサーバに接続することによってユーザ間の同期的なコミュニケーションが可能になる．各ユーザのクライアントソフトウェア上には，ユーザ ID の一覧（BuddyList）に基づいてそれぞれオンラインでコミュニケーション可能かあるいはオフラインでコミュニケーションがとれない状況にあるかが一覧で示される．自分がオンラインの間に他のユーザがオンラインになった場合には，その情報は瞬時にクライアントソフトウェアに反映される．また，多くのインスタントメッセンジャでは，端末の前で様々に変化するユーザの状態を「退席中」「電話中」「取り込み中」といったユーザの状況を知らせるプレゼンス機能を備えている．また，本来は自分の名前を “スクリーンネーム” としてフリーテキストで記述する機能を用いて，名前と共に自分の置かれている状況や気分などをで書き込むことも一部のユーザの間で行われている．このような傾向は十代の若いユーザの中に多く見られ，また日常的にインスタントメッセージサービスを多く使うヘビーユーザほどその傾向が顕著であることが報告されている [69].

他のコミュニケーションメディアには見られない特性として，インスタントメッセージは同期的なコミュニケーションと非同期コミュニケーションの双方に対応可能であるという点が挙げられる [77]．インスタントメッセージは本来同期的なコミュニケーションを行うために開発されたが，コミュニケーションに用いられるメディアが音声や映像ではなく文字情報であるため，長時間に渡ってメッセージが保持されるという特徴がある．そのため，メッセージ受信時に何らかの理由で応答できない状況にあっても，ユーザは自分の都合がついたときにメッセージを送ってきたユーザに返信し，その時点から改めて同期的なコミュニケーションを始めるということができる．

このようなインスタントメッセージのもつ特性を利用し，これを拡張して地理的に離れた場所に存在しているユーザ同士のコミュニケーション，中でも特にフェイストゥーフェイスの環境においてはごく自然に行われるインフォーマルなコミュニケーションを支援する試みがいくつかなされている．Chuah は他のユーザからはメッセンジャーの 1 ユーザとして見えるプログラム（bot）を用意し，BuddyList 内で同一のストリーミング放送を視聴しているユーザの情報などを提供することを試みている [78]．ストリーミング放送を見る

という行為自体の日常生活における占める割合がそもそも非常に少ないという点が難点であるが，これによって「リビングで一緒に同じテレビを見ているような感覚」が得られるとしている．

3[68] では，再生中の曲をユーザ同士で共有して一緒に同じ音楽を聴くということを試みている．この「同じものを聴いている」ということにより同じ部屋で一緒に音楽を聴いているような感覚をユーザに与え，いわゆるユーザ同士の「繋がっている感」を得る事を目指している．ユーザが視聴するものが音楽か映像かという違いはあるが，コンセプトとしては Chuah のアプローチ [78] に近く，どちらも「共有感」のコミュニケーションにおける役割について示す 1 例といえる．

ConChat [79] ではユーザの様々なコンテキスト要素を定義し，システムがユーザのコンテキストをルールベースで解釈しユーザのコミュニケーションを手助けすることを試みている．しかし，コミュニケーションがなされること自体を前提条件とし，行われているコミュニケーションをいかに豊かなものにするかということを目的としており，コミュニケーションを生起させることにもプレゼンスを活用する本研究とは着眼点が異なっている．また，全てのコンテキストに対応するルールを記述することは難しいと考えられる．

また，プレゼンスという概念は各々のユーザ（人間）についての様々な情報であることが一般的であるが，Cooltown プロジェクト [80,81] においては対象を人間についてのみから全てのモノ（オブジェクト）に拡張し，それらのオブジェクトが全てそれぞれのプレゼンスを表記する Web ページをもつ．各ユーザが適切に管理・設定されたアクセス権に基づいてそれらのプレゼンス情報にアクセスすることにより，ユーザは様々なオブジェクトについての情報を得る．全てのオブジェクトが Web 表記をもつことの実現性・妥当性についてと共に，そのような環境での膨大な情報量を処理することの実現可能性についても検討を行う必要がある．

森川らはユーザ周辺の情報完了に関する情報などをユーザ状況に応じて集約し，体系化するサービスプラットフォームについて検討を行っている [82]．情報をユーザ自身の関わるものとユーザ周辺環境を表すものに分類するという点は本研究でのプレゼンス情報の扱いと同様の手法であるといえるが，その目的はコンテキストアウェアサービスの提供のためである．プロファイル情報の体系化に重点を置いている点が本研究とは異なっている．

発信者・受信者双方に最適なコミュニケーション手段を提供する試み [83] もなされている．しかし，プレゼンス情報は主にシステムがユーザに最適なコミュニケーション手段を判断することに用いられており，プレゼンスによってコミュニケーションを活性化させるという本研究とは立場が異なる．また，選択肢としてシステムが用意したコミュニケーション手段しか提供されず拡張性という観点から問題がある．

4.2.3 インフォーマルコミュニケーション支援への適用に関する検討事項

個人のプレゼンスと環境のプレゼンス

ある一人のユーザに注目した場合，そのユーザに関するプレゼンス情報は，一般にユーザ自身に由来するものとユーザを取り巻く環境に由来するものの 2 つに大別される [82]. ユーザ自身に由来するものとしてはユーザの位置情報，忙しさ，何をしているかといったものが挙げられ，一方，環境に由来するものとしては明るさ，温度，騒音レベルといったものが挙げられる．前者と後者を比較した場合，後者の特徴として実世界において比較的近距離に存在するユーザ同士は，同一かもしくはそれに準じた情報をプレゼンスとして保持するという点が挙げられる．

ここで扱うシステムは屋内で用いることを想定しており，前段で述べた「環境」は概ね「部屋」に置き換えて扱うことができる．部屋に由来するプレゼンスとしては，前述の明るさなどの他に，張りつめている／落ち着いている等といった部屋の雰囲気といったものも考えられ，この場合部屋の雰囲気を生じさせているのはその部屋に在室している人間である．このようにプレゼンスの種類によってはユーザ自身に由来するものと周囲の環境に由来するものが密接に関連する場合もあるため，本研究で検討するシステムにおいても両者とも扱える枠組みをもつことを目指す．

プレゼンス情報の伝達

プレゼンス情報を利用する場合には，4.1 で述べたように情報量を適度に制限することが重要となる．ユーザの状況を詳しく伝えるという観点からは，伝えるプレゼンス情報は粒度が細かい方が望ましいが，あまりに細かいプレゼンス情報は発信者の行動や置かれた状況というものを丸裸にしてしまう.「オフィスにおいて勤務時間中の社員の様子を上司が効率的に管理する」という目的の場合などにおいては，このようなユーザのプライバシが大きく制限されるシステムが許容されるというケースも考えられるが，本章で目的としているような同僚同士のコミュニケーションを支援するようなシステムの場合には，ユーザのプライバシは最大限尊重されるべきである．また，そのようにユーザのプライバシを充分に考慮したシステムでなくてはユーザは気軽にシステムを利用することができず,「日常のインフォーマルなコミュニケーションを支援する」というシステムの目的を果たすことも困難になる．

そこで，発信者の周囲の実空間に存在する膨大な情報を，適度な情報量に制限して離れた場所の受信者にネットワークを介して伝達することを考える．その際，元の情報を削減

して情報の粒度を落とす場所については以下のような 3 つの方法が考えられる．

1 つ目は実空間情報を取得する時点で削減する場合であり，情報の取得にカメラなどのような高機能なデバイスを用いず，低機能なセンサなどを用いる場合がこれに該当する．そもそも発信者についての高度に詳細な情報を取得しないので，ユーザへの心理的な安心感は高いと言える．しかしその一方で，情報を加工したり複数の種類の情報を組み合わせてユーザに関する情報を作成するといったようなことを考えた場合には，実際に受信者に提示するより多くの情報を取得したいと言うことも考えられるが，そのような場合にはこの手法を用いる利点は薄れる．

2 つ目は発信者から受信者へと情報を伝えるネットワーク上で情報を削減する場合であり，ネットワーク上にサーバをおいて各ユーザに関する情報を全てサーバに蓄え，そのサーバ上で情報の粒度などを調整するということなどが考えられる．この手法ではサーバを管理する管理者が必要になるが，一般に管理者はサーバ上の全ての情報にアクセスできるため，自分のプレゼンス情報をそのようなサーバに置くことに対してユーザが抵抗を覚える可能性も否定できない．

3 つ目の手法は，ユーザに関する情報は全てユーザ自身の管理下に置いておき，相手ユーザに応じて粒度を調整して情報を送信すると共に，受信者側では各ユーザから送られてきた情報を必要に応じて合成し，ユーザに呈示する手法である．この手法は，上記の 2 つの手法と比較してユーザのシステムに対する安心感と，プレゼンス情報の処理に対する柔軟性のバランスがとれていると言える．

以上の観点から，プレゼンス情報の流通に関しては各ユーザが自身の情報を管理し，必要に応じて粒度を調整し他のユーザに公開するという手法を本研究では用いる．

ユーザとシステムとの関係

本章での目的を実現するようなシステムにおいては，なされるコミュニケーションはユーザの利益となると共に，ユーザにとって有用であるようなものでなくてはならない．集中して考え事をしているのにくだらないことで話しかけられたり，相手に「今どこにいるのか」ということのみを聞くために行われるコミュニケーションは，いくらコミュニケーションの回数が増えたところでユーザにとって有用ではない．その一方で，自分の作業が一息ついてちょっと雑談したいと思っているユーザが同士がコミュニケーションを行うことができれば，そのようなコミュニケーションは双方のユーザにとって有用であると推測できる．上記の例のように，同様の状況に置かれていたりあるいは同じ事柄に興味をもつユーザ同士に対し，互いの存在を知らせることでその事柄についてユーザにとって有

益なコミュニケーションを行えることが期待できる．ユーザの興味を軸にユーザ同士を結びつけること有用性は，Ping や Trackback を備えた blog サービスの昨今の隆盛からも見て取れる．

このように，ユーザにとっての有用性を向上させることが最終的な目標となるが，そのこととコミュニケーションの回数や頻度を増やすと言うことは必ずしも一致しない．ユーザのコミュニケーションを支援する際にはコミュニケーションの回数は多いほどよいということは必ずしも言えず，またコミュニケーションの内容によってはそのコミュニケーションを行ったことでユーザが不利益を被る場合もあるということに留意する必要がある．しかし，適切なプレゼンス情報を活用することで無用なコミュニケーションを減らすと共に，ユーザにとって有益なコミュニケーションを活性化させることが期待できる．ゆえに，本章で検討するシステムにおいてもプレゼンスを活用し，インフォーマルコミュニケーションをユーザにとって有益なものとするための支援を行うものとする．

また，ユーザ同士のインフォーマルなコミュニケーションを促進させることを目標とするシステムにおいては，システムとユーザとの心理的な距離を適度に保つことが重要であると考えられる．ユーザが自分の作業に集中しているときでも,「口やかましく」他のユーザの様子を報告してくるようなシステムは非常に鬱陶しいものであるし，逆にふとした瞬間に他のユーザの様子をうかがおうと思ったときに，ユーザが多くの労力を割かなくてはいけないようなものでは，そもそもシステム自体をあまり使おうという気分になれない．

これは主に，ユーザに対して情報を提示する出力側のユーザインターフェース（UI: User Interface）についてであるが，入力側の UI についても同様のことが言える．システムが提供するプレゼンスの種類を増やす事にのみ注力し，その結果としてユーザが手動で入力しなくてはならない項目が膨大になってしまっては意味が無く，自動で取得できる情報は自動で取得してユーザへの負担を少なくしなくてはならない．

ユーザが自発的に使いたいと考え，かつ実際に継続的に使用するようなシステムであるためには,「ユーザの使用コスト < システムより得られるメリット」である必要がある．そのためには，システムが提供する機能を豊富にし，ユーザが享受するメリットを増やすことも勿論重要であるが，それと同時にユーザがシステムを使用するために必要とするコストを下げることも重要である．そのためには，システムの使い勝手，中でも特に，ユーザとシステムを直接結びつける UI の使用感を良好なものにする必要がある．一般的に，プロトタイプとして実験的に実装されたシステムでは，UI の実装は本質でないとされることが多いが，本章で述べるようなシステムにおいては UI もまたシステムを構成する大きな要素であり，そのでき不できがシステムの使用感を大きく左右するという点で重要な要素であるといえる．そこで，UI は適度にユーザとの距離を保ちつつ，適度に主張すると

いう特性が求められると考える．

このような観点から，筆者らは本章のようなシステムにおいては，一般に本質でないとされることが多い UI についてもシステムを構成する重要な要素であり，充分な検討を加える必要があると考える．

4.3 ACDC システム

4.3.1 システム構成

4.2 節での検討をふまえた上で，インフォーマルなコミュニケーションの支援を目的とした ACDC システムの設計を行う．

ユビキタス情報社会が実現した暁には，様々な情報が電子化されると共に，ユーザはセンサ類をはじめとして数多くのデバイスを利用することができるようになる．実空間に存在するユーザの状態をプレゼンスとして他のユーザに広告するためには，それらのデバイス類を用いてユーザのコンテキストをコンピュータが扱える形に電子化し，ネットワーク上に投影する機構が必要となる．それらのデバイス類は種類も管理者も様々であるが，機能から分別すると，図 4.1 に示すように，プレゼンス情報の対象である所有者，情報を要求し受信する受信者の他に，情報源（キーボード，センサなど），情報提示装置（ディスプレイ，アクチュエータなど）が存在する．そして，システムには以下の条件が求められる．

- プレゼンス情報は所有者ごとに多様な情報源から集めて保存管理されること
- 所有者と受信者の関係を反映して公開する情報を制御できること
- 複数人から受信した情報を合成解析して付加情報やサービストリガーを生成する処理ができること

本システムでは上記の条件を満たすため，計算処理・データベース（DB: Data Base）・通信の各機能を備えるとともに周囲のデバイスなどと連携するユーザエージェント（UA: User Agent）を用い，これにより実空間に存在しているユーザをネットワーク空間に投影させるものとする．このような環境における情報の取得，情報の伝達および合成，そして UA のそれぞれについて以下で検討を行う．

4.3.2 情報の取得

前述したような環境の中でユーザ（所有者）のプレゼンス情報を作成・通知する際には，ユーザの周囲に存在するセンサを利用して大量かつ高度なコンテキストを利用すること

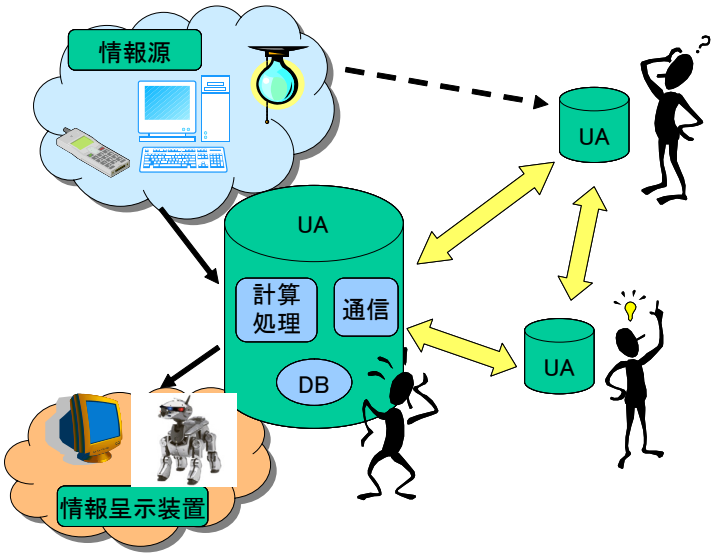


図 4.1: ACDC システム概要

や [80, 81]，各受信者に対して個別に情報公開ポリシを設定し，各受信者との関係に応じた粒度のプレゼンス情報を公開できるよう，所有者を中心に情報源と保存者が連携して所有者に関する高度な情報生成が行われることが必要となる．また，情報源となるデバイスも各ユーザが占有して用いるものや複数ユーザに共有されるものなど様々な形態があり，それらに対応できる枠組みである必要がある．

4.3.3 情報の伝達と合成

各ユーザごとに生成されたプレゼンス情報は，他のユーザへと広告されてはじめて意味をもつ．その際には，所有者は各ユーザごとに異なる公開ポリシをもつので，所有者と受信者が一対一で結びつけられて情報の伝達を行えることが必要となる．そのためには，各ユーザが一意の識別子を保持し，その識別子を用いて情報の伝達を行うことが必要となる．

また，一般に各ユーザの交遊関係は異なり，各所有者はそれぞれ異なったユーザリスト（BuddyList）をもつ上に，それぞれの相手に対して様々に異なる公開ポリシを設定することが考えられるため，各ユーザが受信者として受け取る情報はユーザごとに異なっている．さらに，同一の部屋などにいることがわかっている複数のユーザのプレゼンス情報を組み合わせることで，それらのユーザがいる環境に関するプレゼンス情報を取得することもで

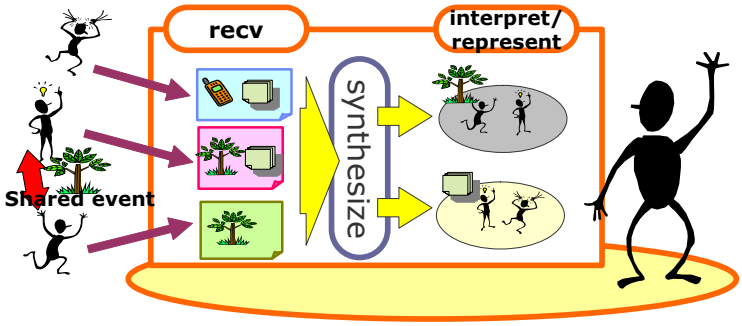


図 4.2: 受信者側での情報合成

きる．そこで，図 4.2 に示すように，他のユーザから受け取ったプレゼンスをそれぞれの受信者が各々解釈・合成を行うことで，各受信者の立場でプレゼンスを新たに創成しユーザに呈示できることが期待できる．

4.3.4 ユーザエージェント

実空間のユーザの存在をネットワーク空間に投影し，ユーザが利用しやすい形で提供するために，情報の収集保存機能，相手ごとにフィルタされた情報を伝える通信機能，合成等の受信情報の処理機能，および UI や API を併せもつことが UA には求められる．

図 4.3 に UA の構成図を示す．前述した主要な 3 つの機能は相互に密に連携処理を行う．また，相手にプレゼンスを通達する際は必ず通信部を通す必要があるが，これらの要素は UI を含め同一端末にある必要はなく，たとえば UI と通信機能は携帯電話を利用し，保存機能・処理機能は遠隔サーバで実行させるというシナリオも考えられる．ただし 1 ユーザ 1 エージェントに対応し，たとえば複数のユーザに共同で利用される DB サーバを置く場合であっても，あるユーザの保存部は他のユーザの保存部に直接アクセスできず，各ユーザごとに独自に管理される．

通信部（Network Transaction）

図 4.3 下の通信部は他の UA とのプレゼンスや制御メッセージ通信を担う．そのためには，互いの UA の通信識別子を使った接続処理から始まり，制御メッセージやプレゼンス情報そのもの，およびインスタントメッセージのテキストデータなどの伝送処理，切断処理までを行う必要がある．

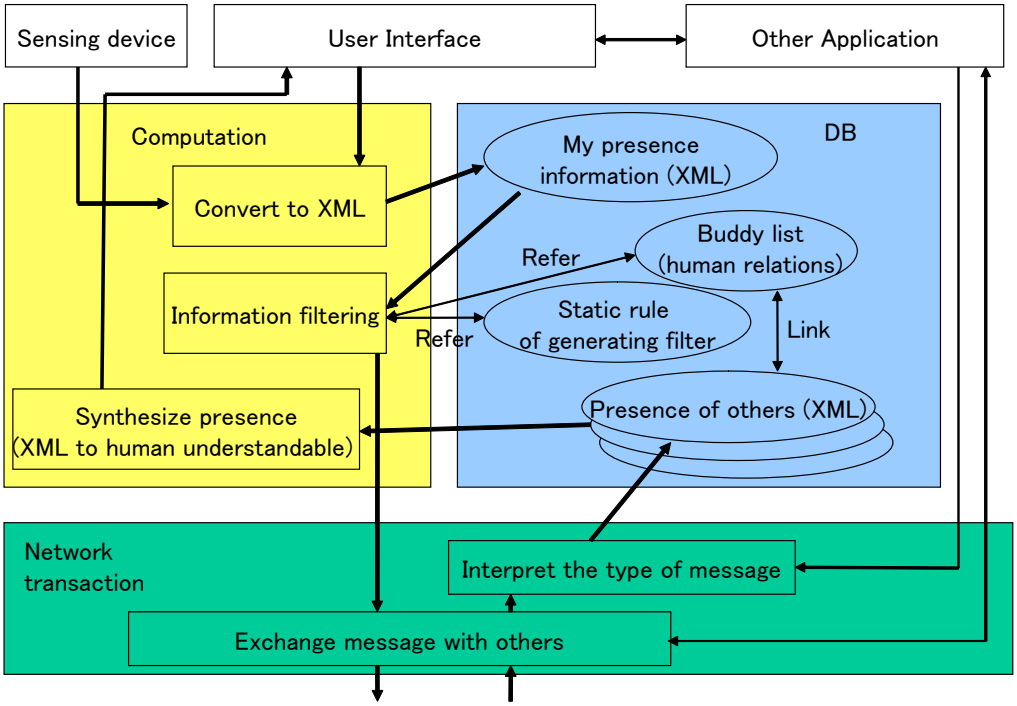


図 4.3: ユーザエージェントの構成

これらのプロトコルはアプリ独自でも良いが，SIP [84] がセッションの制御プロトコルとして標準となりつつあり，SIP アドレスを識別子にして接続・切断処理を行い，メッセージのペイロードに情報を載せて伝達することができる．SIP は IETF により標準化がなされこれに準拠したライブラリが多数公開されており，これを利用することで実装上の労力を低減することができ，またペイロードに多様なデータを格納できるため拡張性に優れているという利点がある．具体的には，SIP をシグナリングプロトコルとして用い，ペイロードにプレゼンス情報や制御メッセージを載せ，受信時に送信元の確認，メッセージの種類の判定を行って保存部やアプリにデータを渡すという手法が考えられる．

保存部（DB）

図 4.3 右上の保存部はユーザ自身および受信した他ユーザのプレゼンス情報を保存する．自身に関する情報は，情報源から集めた後にデータ処理を施したプレゼンス情報，情報送信先リストである BuddyList，および公開ポリシを情報に適用するためのフィルタ規則を含む．また，プレゼンス情報は同一属性でも多様な表記方法があるため（例：位置などの粒度，相対・絶対など），それらを基本的に全て保存し，送信先との公開ポリシに応じて情報を制御することとする．データ保存形式は，抽出や合成，データ変換などの管理・計算の容易性といった観点から XML を用いる．

保存されているデータは，基本的にそれぞれの情報がアップデートされた時点で新しいデータに置き換えられ，古いデータは破棄される．ただし，複数の時刻のプレゼンス情報を合成することによって新たなプレゼンスを創成するという場合も考えられ，このような要求がある場合には，図 4.4 に示すようにイベントの生起によりプレゼンス情報が更新されてから次にイベントが生起してプレゼンスが更新されるまでを 1 世代として，指定された世代の分の過去のプレゼンス情報についても時系列別に保存しておくものとする．

処理部（Computation）

図 4.3 左上の処理部は各種のデータ変換・合成や情報のフィルタ処理を行う．情報所有者側では，センサや UI などの情報源のデータからコンテキスト処理をして保存部に渡すこと，および保存部のプレゼンス情報とフィルタ規則から他人に公開する自身のプレゼンス情報の生成処理を行う．また，情報受信側では，BuddyList 中の相手から受信した情報群を解析し，共通項目やユーザとの関係から合成プレゼンスを生成し，ユーザやアプリケーションに提示可能な状態に変換する．

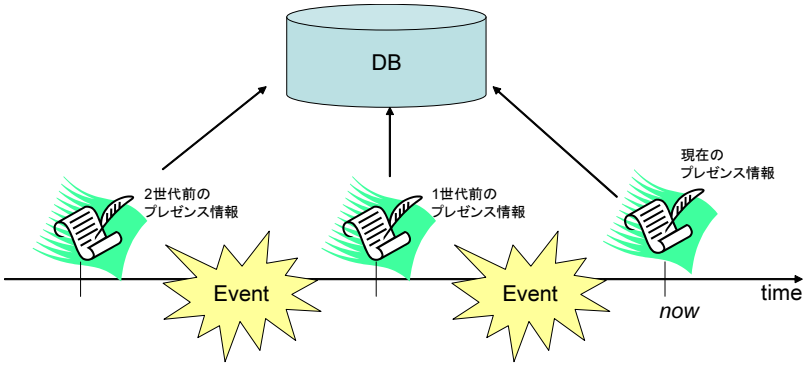


図 4.4: 過去のプレゼンスのデータベースへの記録

センサやユーザインタフェースのような情報源のデータが変化したことを検出するには，情報源自体がデータの変化を検出してイベントとして通知する方法と，ユーザエージェント側が定期的に情報を取得して前回のデータとの比較を行い，データに変更が加えられたかどうかをチェックする方法（ポーリング）の 2 通りが考えられるが，そのどちらにも対応できるものとする．

実際にプレゼンス情報を他のユーザに向けて発信する際には，保存部に保存されているプレゼンス情報とフィルタ規則から他人に公開する自分自身のプレゼンス情報の生成処理を行う．発信のタイミングは，センサやユーザインタフェースといった情報源のデータが変化して自身のプレゼンス情報が変化した場合に自分から情報を発信する場合と，他のユーザエージェントから情報を通知することを要求され，それに応える形で自身のプレゼンス情報を送信する場合が考えられる．前者は，後者に比べて BuddyList 内の多くのユーザに一斉に自身のプレゼンス情報を送信することになるため，そのような負荷に耐えられるだけの計算能力資源およびネットワーク資源をユーザエージェントは備える必要がある．

また，情報受信者側では BuddyList 中の相手から受信したプレゼンス情報群を解析する．これにより，それらのプレゼンス情報軍のなかに含まれる共通項目や，自身と相手ユーザとの関係から合成プレゼンスを生成し，ユーザやアプリケーションに提示可能な状態に変換する．変換されたデータは API を通じて他のアプリケーションに通知されたり，ユーザインタフェースを通じてユーザに呈示される．

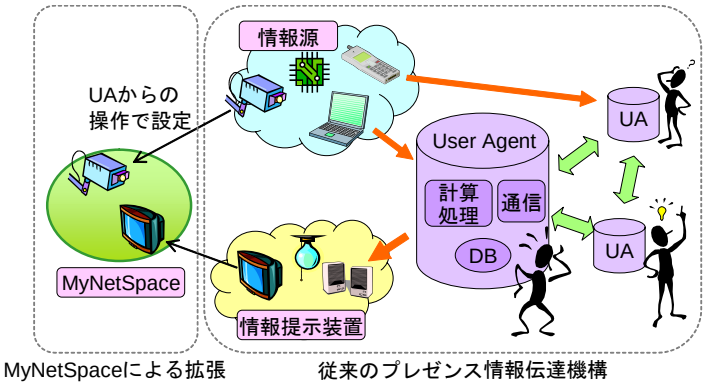


図 4.5: プレゼンスシステムと MyNetSpace による拡張部分

4.3.5 情報源と情報提示装置との直接通信

本章で述べたシステムではすべてのプレゼンスが UA を介してやり取りされるが，このモデルに当てはまらない場合も存在する．たとえば，環境に設置されたビデオカメラの映像を自分のプレゼンスとして公開したいことが考えられる．このように大容量となる情報源を利用する場合，UA 自体が携帯端末上などに搭載される可能性を考慮すると，UA を通して全員に配信するのは現実的ではない．つまり，情報源から情報提示装置へと直接に，しかし UA 間の関係に基づいて安全に情報を伝達する制御機構が要求される．

加えて，ユビキタス環境が整うに伴って情報源や情報提示装置となるアプリケーションは多様化する一方，単機能化・部品化していき，プレゼンスシステムだけでなく様々な形で再利用可能になっていくと予想される．このことから，上述の制御機構は共通プラットフォームとして提供されるのが効率的である．

現在，第 2 章で述べた MyNetSpace（MNS）システムをこのような共通プラットフォームとして利用することをシナリオの 1 つとして考えている．そこで以下では，MNS システムを適用することで既存プレゼンスシステムの改善を図る．

図 4.5 の左側の枠内に示すように，直接に通信を行わせたい情報源と情報提示装置を MNS に取り込むことで実現する．前提として，MNS システムが動作する環境，すなわち情報源，および情報提示装置というアプリケーションが載る端末において MNS アナライザと MNS マネージャが動作する環境でなければならない．また，UA とユーザが使用する情報源，情報提示装置が載る端末との間には，あらかじめ信頼関係があるとする．

ユーザ A から B（複数人でも構わない）にプレゼンスを公開するときの具体的な手順を

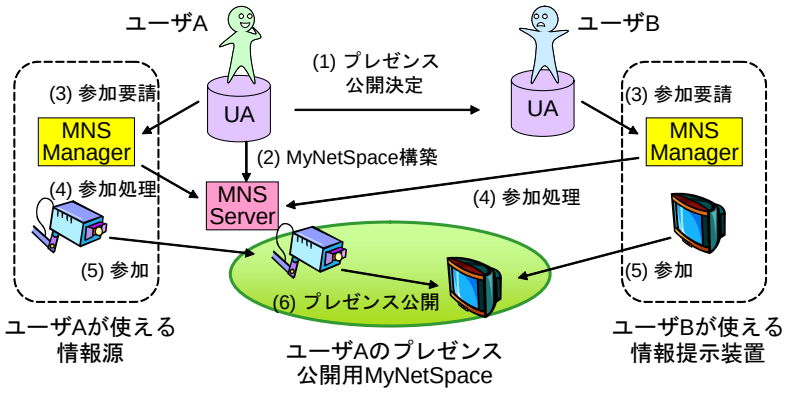


図 4.6: MyNetSpace を用いたプレゼンス公開手順

図 4.6 に示す．まず，ユーザ A が使用できる情報源から直接にプレゼンスを公開することを決定する．ユーザ A は，MNS サーバのプロセスを起動し，プレゼンス公開用 MNS を構築する．その後，直接に公開したい情報源が載る端末の MNS マネージャに対して MNS への参加要請メッセージを送り，情報源を MNS へと取り込む．参加要請メッセージには，MNS サーバの情報と MNS への参加認証のための情報が含まれる．それらの情報に従って MNS サーバと MNS マネージャの間で参加処理が行われ，端末に適切な設定が行われる．一方，ユーザ B に対しては UA を通じて，プレゼンスを情報源から直接取得するようにメッセージを送る．このメッセージには，MNS サーバの情報と MNS への参加認証情報も含まれる．ユーザ B はプレゼンス受信の許否を決め，許可する場合にはどの情報提示装置で受信するかを決める．その後，受信させたい情報提示装置が載る端末の MNS マネージャに対して MNS への参加要請メッセージを送り，MNS へと参加させる．最後に，指定された情報源からプレゼンスを受信するように情報提示装置を操作する．

4.4 実装

4.4.1 実装の概要

本節では，主目的であるプレゼンスを用いてコミュニケーションの活性化を図ることに対する妥当性の確認と，さらなる課題の抽出を行うために実装したアプリケーションについて説明を行う．

各ユーザの PC 上でクライアントを動作させることを前提とし，Windows 上で実装を行っ

た．また，コミュニケーション機能とプレゼンス機能を兼ね備えたツールとして MSN [76] や ICQ [75] に代表されるメッセンジャーが存在するが，本実装ではこのようなメッセンジャー機能をベースとしつつ，独自のプレゼンス機能を搭載したアプリケーションを実装した．

最初にも述べたように，全てのプレゼンスを実装することは実際問題として不可能であるので，本実装では現時点で有用であると期待できるプレゼンス情報を数多くユーザに呈示することでコミュニケーションの支援を目指すものとした．

4.4.2 実装フレームワーク

様々なプレゼンス情報をやりとりする際に用いる，アプリケーションの土台となるフレームワークについて述べる．シグナリングプロトコルとしては 4.3.4 で述べたように SIP を用い，各ユーザの識別子として SIP アドレスを用いた．また，SIP ライブラリとして GNU oSIP library [85] を用い，これを用いて作成した Win32 DLL を.NET Framework 上で動作する C#で記述したアプリケーションから呼び出す構造とした．そして，各ユーザのプレゼンス情報は SIP の NOTIFY メッセージのペイロードに格納され BuddyList 内のユーザに広告される．ネットワーク上には SIP サーバが設けてあるが，これはネットワークレベルでの各ユーザの SIP アドレスと IP アドレスの対応付けにのみ用いられており，プレゼンス情報レベルではサーバレスの構成としている．

また，プレゼンスの最終的なユーザへの呈示は基本的に PC のモニターによっているが，一部情報を AIBO [86] を通して実空間へ反映させる事も試みている．図 4.7 にスクリーンショットを示す．左がツールチップによる情報表示の，右がデスクトップ上へのポップアップによる注意喚起の例である．

4.4.3 シグナリングプロトコル

本節では，実装で利用した SIP のシグナリングプロトコルを示す．

プレゼンス情報を受け取る側（subscriber）と発信する側（notifier）との間のパケットの流れを図 4.8 に示す．まず，各ユーザがアプリケーションを操作し REGISTER メッセージを SIP サーバに送信することでサインインすると同時に，SIP の SUBSCRIBE メッセージが BuddyList 内の各ユーザの SIP アドレスに対して送付される．その時点でオフラインのユーザについては SIP サーバから 404（Not Found）応答がかえされる．一方，既にサインインしているユーザに対しては SIP サーバによりメッセージが転送される．SUBSCRIBE

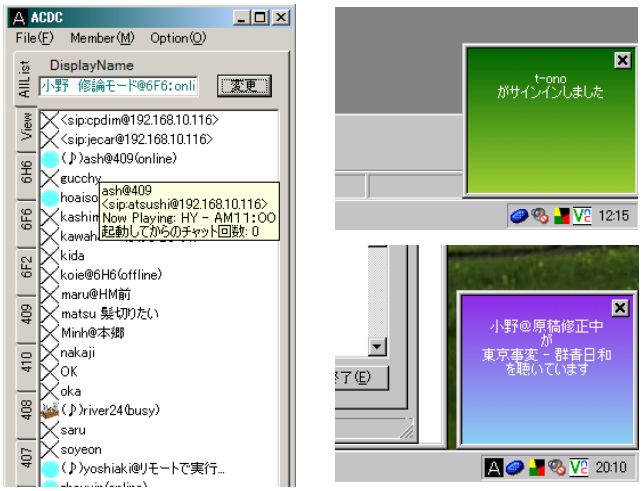


図 4.7: ACDC スクリーンショット

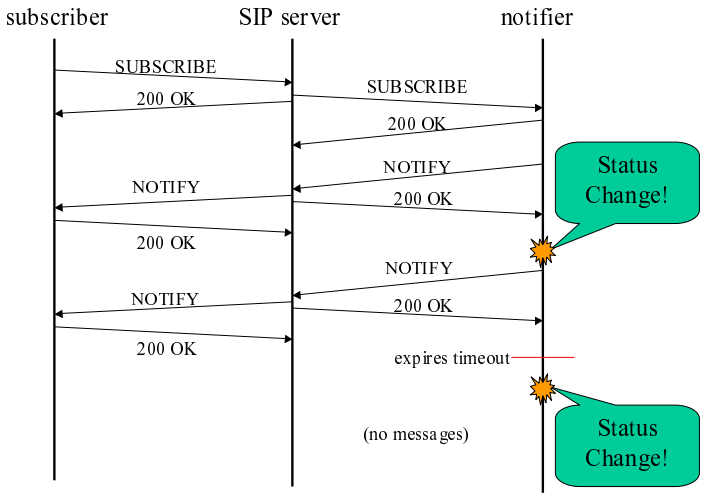


図 4.8: プレゼンス情報伝達の際のメッセージング

を受け取ったユーザエージェントは，自身のその時点のプレゼンス情報を格納した NOTIFY メッセージを送り返し，これによってサインインしたユーザは他のオンラインのユーザのプレゼンス情報を知ることができる．また，SUBSCRIBE メッセージにはヘッダに expires（有効期間）フィールドが定められており，これが有効である間に自身のプレゼンス情報が変化した場合には新たなプレゼンス情報を NOTIFY メッセージに格納して再び送信する．各々の UA は，自身が送信した SUBSCRIBE メッセージの有効期限が切れる前に新たな SUBSCRIBE メッセージを送信し，有効期限が失効することを防ぐ．

以上は SIP の標準に基づいた動作であるが，このような動作を行うことによってあるユーザのプレゼンスが変化した場合にその情報を即座に他のユーザに伝達することができる．また，アプリケーションエラーやネットワークの不調などで正常終了せずにユーザがシステムを利用出来なくなった場合でも，実際にはオフラインのユーザが他のユーザからオンラインに見えている期間の上限を SUBSCRIBE メッセージのヘッダの expires とすることができる．expires の値を小さくすれば異常終了したユーザをすぐにオフラインとして認識出来るが，この値の大小と SUBSCRIBE メッセージのトラフィック量はトレードオフの関係にあるため適度な大きさにする必要がある．本実装では expires の値はユーザが指定出来るようにしたが，デフォルトでは 300 秒とした．

また，前述の通り本アプリケーションはインスタントメッセンジャーをベースとしており，テキストチャットによるコミュニケーションをサポートしている．SIP は当初セッションの制御目的で定められたプロトコルであるが，今日ではインスタントメッセージの伝送についても RFC で正式に標準化されており [87]，本アプリケーションもこれに準拠したプロトコルを用いている．すなわち，話しかける側（caller）と話しかけられる側（callee）との間で図 4.9 に示した手順によりセッションを確立し，このセッション内で MESSAGE メッセージをやりとりしてインスタントメッセージングを実現している．具体的にセッションを確立し，会話が行われ，セッションが終了するまでの手順は以下の通りである．

1. caller が callee 宛に INVITE メッセージを送る．メッセージはまず SIP サーバに届けられる．
2. メッセージを受け取った SIP サーバは caller に 100（Trying）を応答すると共に，INVITE メッセージを callee に転送する．
3. INVITE メッセージを受け取った callee は 180（Ringing）応答を SIP サーバを通じて caller へ送る．
4. callee がセッションへの招待を受諾した場合には同様に 200（OK）を caller へ応答する．VoIP（Voice over IP）などのアプリケーションの場合にはユーザによる確認があってはじめて 200 が返信されるが，本アプリケーションではインスタントメッ

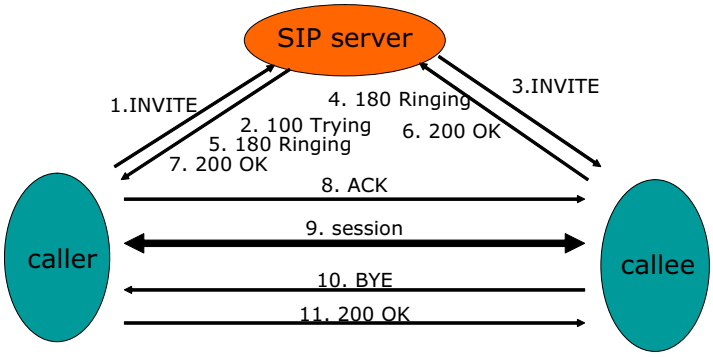


図 4.9: SIP によるチャットセッションの確立

セージングであるため，ユーザの操作を特に必要とすることなく自動的に 200 応答を返信する仕様としている．

5. 200 応答を受け取った caller は ACK メッセージを callee に送り，これによって caller と callee の間でセッションが確立する．
6. ユーザがインスタントメッセージの会話ウィンドウを閉じるなどの操作を行いセッションを終了する場合には，BYE メッセージを送信する．
7. BYE メッセージを受信した場合には 200 応答を返し，これによってセッションは終了する．

それぞれの SIP パケットがどのセッションに含まれるかは，SIP パケットのヘッダーに含まれる Call-ID フィールドの値による．Call-ID はセッションの識別子であり，これが同一のパケット同士がひとつのセッションに関係する SIP パケットとして認識される．なお，2 人以上のユーザが同時に参加するチャット（マルチチャット）の場合には，すべての参加ユーザ間にフルメッシュで一対一のセッションを設けている．その際のそれぞれのセッションの Call-ID は全て同一である．

4.4.4 プレゼンス情報の種類

ステータス

自身の状況について,「PC から離れている」「多忙につき話しかけるな」等と言ったことを意味する数種類のステータスから選びプレゼンスとして他のユーザに広告する．多くのインスタントメッセンジャーソフトにおいて実装されている極めて基本的なプレゼンス情

報である．本実装においては,「オンライン」「取り込み中」「休憩中」「電話中」「退席中」の中から，ユーザが手動でマウス操作により選択するものとした．

スクリーンネーム

自身の名前として相手ユーザの画面に表示されるものであり，前項と同様に多くのインスタントメッセンジャーソフトが備えている機能である．本来ユーザの同定は SIP アドレスなどのアカウントにより可能であるが，ユーザにとってアカウント名と実際の名前を結びつけることは難しいことがあるため，自身の名前をフリーテキストで入力することでそれが相手に表示されるというものである．このように名前を入れることを意図して設けられたスクリーンネームであるが，一部のユーザのなかにはここに名前以外に近況報告のような短信を併記するユーザも見られる．

部屋の雰囲気

部屋の「雰囲気」を測る尺度には様々なものが考えられるが，本実装ではそのうち部屋の空気がどの程度緊張しているかという軸に注目してこれをプレゼンスとして広告するシステムを実装した．ただし,「どの程度部屋が緊張しているか」というコンテキストを自動で取得することは極めて困難であるため，これをユーザが手動で入力するシステムとした．具体的には，各ユーザは自分の在室している部屋名と，0 から 100 までの整数値をもつ “緊張度” をプレゼンスとして他のユーザに広告する．一方で，他のユーザから受け取ったプレゼンス情報を解釈し，各部屋ごとに在室しているユーザの申告する緊張度を加算平均して部屋の緊張度を求め，ユーザに提示する．なお，緊張度は他のユーザのプレゼンスが更新され新たなプレゼンスが届くたびに再計算される．そして，新たに求められた緊張度が以前より増加して閾値を超えた場合は，デスクトップの隅に図 4.7 右のようなポップアップを表示させユーザに注意を喚起する．

BGM 情報

ユーザ同士が共通の趣味を発見して話が弾む事を期待し，話題の「種」を提供することを目的として，ユーザが BGM として再生している音楽の曲情報を取得しプレゼンスとして他のユーザに広告するようにした．具体的には Winamp を COM サーバにするプラグインを用い，本アプリケーションとの間で COM による通信を行って再生中の曲のメタ情報を取得している．

他のユーザから受信した BGM 情報は，図 4.7 左のように BuddyList をマウスポインタでポイントしたときに表示されるツールチップを用いてユーザに表示するものとした．この方法ではユーザがマウス操作を行わない限り情報が表示されず一覧性がよくないという欠点があるが，BuddyList の表示が煩雑になるのを防ぐことを優先しこの方法を採用した．また同時に，一定時間ごとに BuddyList 内のユーザのうち音楽を聴いているユーザを一人ランダムで選択し，図 4.7 右のようなポップアップを表示させてそのユーザが聴いている曲情報を表示させることも行った．

会話回数

ユーザの忙しさとコミュニケーションの回数について考察すると，忙しいときはコミュニケーションの量が減り，暇なときはその逆と言うような何らかの相関があると期待される．そこで，アプリケーションを起動してからの IM のセッションの累積数をプレゼンスとして広告することとした．なおユーザへの表示は，前述の BGM 情報と同様に図 4.7 左のようなツールチップを用いた表示としている．

コミュニケーションプレゼンス

実空間においては誰と誰が会話をしているかという情報は，人間関係を窺い知る上で重要な要素となっている．そこでこれをネットワークコミュニケーションにも発展させ，ネットワークコミュニケーションで誰と誰が会話をしているかという情報をプレゼンスとして他のユーザに伝達させることを目的とするのが “コミュニケーションプレゼンス” [88, 89] である．本実装ではコミュニケーションとして文字チャットを扱い，誰と誰がチャットを行っているかを BuddyList 内の他のユーザに広告する．この際実際にプレゼンス情報として伝達されるのは，会話のセッション ID と会話への参加人数だけであり，会話に参加している全てのユーザのプレゼンスを受け取れるユーザのみが，誰と誰が会話をしているかを実際に把握できる仕様としている．ユーザに呈示する UI 画面は図 4.10 の様になっており，左がどのユーザも会話をしていない状態，右が一部のユーザが会話をしている状態である．図中の中心の点が自分を，周囲の点が他のユーザを表しており，会話をしているユーザは線でつながって表示される．

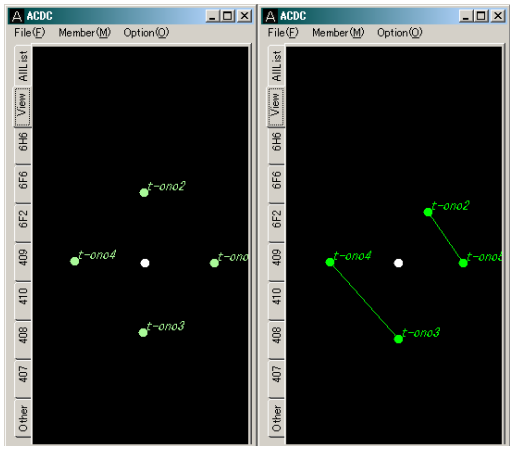


図 4.10: コミュニケーションプレゼンス

4.4.5 実空間との入出力連携

入力連携

ユーザの在室している部屋がうるさいか静かかというプレゼンスを伝達することで，部屋の「活気」をある程度伝達できるのではないかという意図のもとに部屋の騒音レベルを 3 段階で広告することにした．騒音レベルの取得は AIBO に搭載されているマイクを用い，騒音レベルは「AIBO のプレゼンス」としてユーザに広告される．ユーザは BuddyList の AIBO をマウスでポイントすることにより表示されるツールチップにより，その AIBO が置かれた部屋の騒音レベルを知ることができる．

出力連携

PC のモニタ以外を通じてプレゼンス空間と実空間をつなぐ試みとして，AIBO を用いて部屋の雰囲気を実空間に反映させることを行った．具体的には，それぞれサーバにより制御される AIBO を各部屋に一匹ずつ配置する．各々の AIBO は他のユーザからは一般ユーザと同様の 1 クライアントとして見える．各ユーザは AIBO を BuddyList に登録し，他のユーザへと同様に自分のプレゼンスを AIBO に通知する．そして AIBO は受信したプレゼンスを合成し，自分がいる場所以外の部屋の “緊張度” が上昇した場合に図 4.11 のような「おびえた」仕種をする．これにより，ユーザの意識が PC 以外を向いている場合でもユーザに遠隔地の状況が変化したことを伝えると共に，AIBO の仕種による一種の「癒

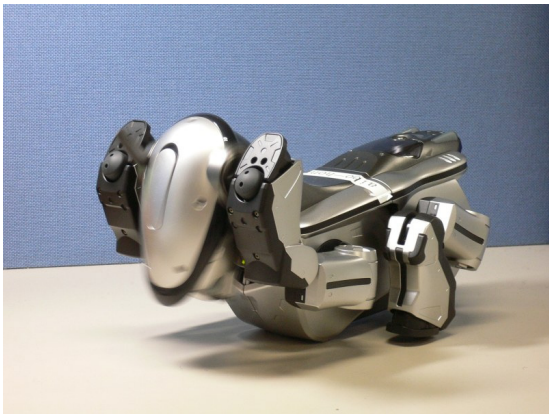


図 4.11: AIBO による情報呈示例

し」効果が期待できる．

4.5 コミュニケーションに与える影響の評価

4.5.1 実験の目的と構成

実装したテストベッド，およびアプリケーションを用いて初動評価実験を行った．本実験の主たる目的は，インフォーマルなコミュニケーションにおけるプレゼンス情報の与える影響を明らかにし，今後の検討を行うにあたっての判断材料とすることである．今回の実験においては上記の目的を果たすために，ユーザが利用できるプレゼンス情報の種類を変化させた場合に，コミュニケーションがどのように変化するかということを実験により明らかにする．実際に測定した項目は表 4.1 に示すとおりである．

これらの項目を測定項目として選択した意図は以下の通りである．C はコミュニケーションの量を最も直接的に反映すると考えられる．一方で，N_i や L_i（$i = m, o$）は一回あたりのコミュニケーションがどの程度「盛り上がったか」ということの指標になると考えられる．また，t が小さいことはユーザの行っているタスクの中でチャットの優先度が高いことを意味していると考えられ，そのような場合にはユーザの欲求に沿ったコミュニケーションが行われているということが期待できる．

実験は表 4.2 に示す条件で 3 回，実験 1，実験 2，実験 3 の順に行った．それぞれの実験期間は平日の 3 日間であり，実験の被験者は全て本研究室に所属している学生 26 名である．被験者は全員互いに面識があり，約半数が柏の，残り半数が本郷の研究室に自分の

表 4.1: 測定項目

C	チャット回数
t	相手が最後に発言してから自分が発言するまでの時間差
N_m	1 回のチャットにおける自分の発言回数
L_m	1 回のチャットにおける自分の発言文字数
N_o	1 回のチャットにおける他人の発言回数
L_o	1 回のチャットにおける他人の発言文字数

表 4.2: 実験条件

	利用できるプレゼンス情報
実験 1	スクリーンネーム及びステータスのみ
実験 2	なし（SIP アドレスのみ）
実験 3	本システムで実装した全ての情報

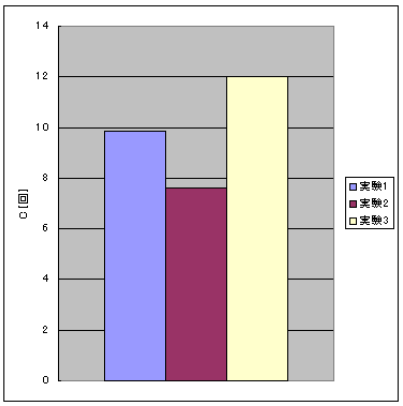


図 4.12: ユーザあたりのチャット回数 C の平均値

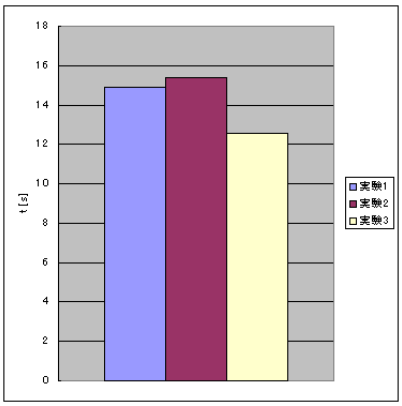


図 4.13: ユーザあたりの発言時間差 t の平均値

席をもち基本的にそこで研究を行っている．

それぞれの実験条件の設定は，以下のような意図による．すなわち，実験 1 では現行のインスタントメッセンジャーソフトと同様のプレゼンス情報を提供することで，これでなされるコミュニケーションは一般のインスタントメッセンジャーソフトによるコミュニケーションとほぼ同等のものと考えられる．これに対し，実験 2 においては利用できるプレゼンス情報を極限まで減らし，ユーザが相手ユーザについて知ることができる情報は，相手がオンラインかオフラインかと言うことだけである．これにより，プレゼンス情報が全く利用できない場合にコミュニケーションはどのような形になるのかということを知ることが目的である．一方，実験 3 では今回実装を行った全てのプレゼンス情報を利用し，結果を実験 1 や実験 2 と比較することで今回実装を行ったプレゼンス情報がコミュニケーションの支援にどの程度効果的であるかを観察する．

4.5.2 結果と考察

実験の結果，得られたデータ集計したものを図 4.12〜図 4.17 に示す．図 4.12 および図 4.13 は，それぞれ C および t のユーザ一人あたりの平均値を示す．また，図 4.14，図 4.15，図 4.16 および図 4.17 はそれぞれインスタントメッセージによる会話セッション 1 回あたりの N_m，L_m，N_o 及び L_o の平均値を示す．なおいずれのデータも，それぞれの実験期間中に 1 回以下しかチャットによる会話をしなかったユーザは除外してある．

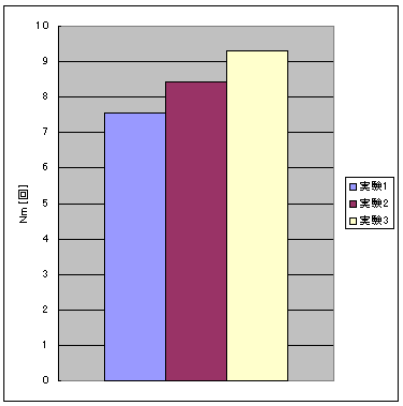


図 4.14: 1 回のチャットにおける自分の発言回数 N_m の平均値

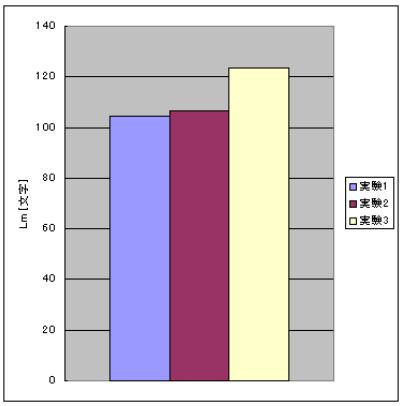


図 4.15: 1 回のチャットにおける自分の発言文字数 L_m の平均値

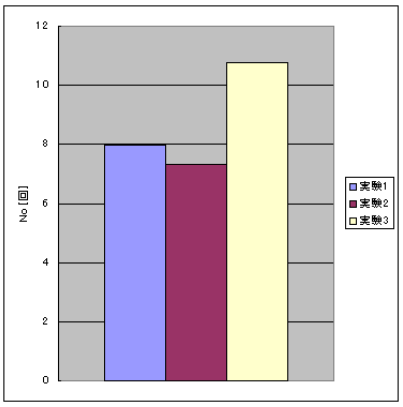


図 4.16: 1 回のチャットにおける他人の発言回数 N_o の平均値

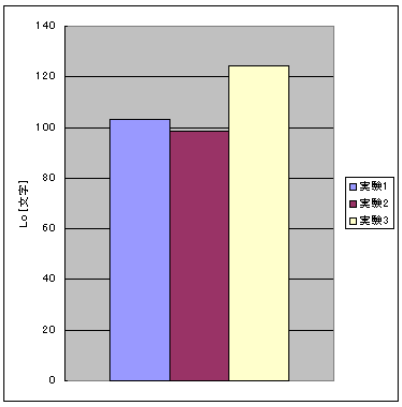


図 4.17: 1 回のチャットにおける他人の発言文字数 L_o の平均値

まず，それぞれの図より実験 2 において全体的にユーザ間のコミュニケーションが低調であったことがわかる．また，図 4.12 と図 4.14〜4.17 を比較した場合，実験 1 は実験 2 や実験 3 と比べてチャットの回数自体は割合多いものの，チャット 1 回あたりの発言回数や発言文字数といったものはそれほど多くない．これは，実験 1 の場合は特に実験開始間もない時間帯において，被験者がシステムに慣熟していないためテスト目的でチャットを行ったり，あるいは被験者同士でアプリケーションの使い方について相談したりしていたことが原因として考えられる．このようなテスト目的や事務的連絡が目的で行われたチャットでは，そのチャットが行われている間に伝送された情報量（バイト量）は比較的少量であることが想像され，用事が済んだ時点で速やかにコミュニケーションが終了しているものと思われる．

その一方で，2 番目に行われてユーザがある程度アプリケーションの操作に慣熟していると思われる実験 2 では，C は 3 つの実験中で最低となっているのに対して N_m と L_m は実験 1 よりも大きくなっている．これはコミュニケーションの回数自体は少ないもののそのコミュニケーションにおいて発言回数は実験 1 より多かったかったということを意味している．実験 2 においては利用できるプレゼンス情報は，相手ユーザがオンラインかオフラインかと言うことだけであり，相手がどこにいるかという情報などは全く利用できない．このため，会話を行う際に通常のコミュニケーション内容に加えて，実験 1 では行われることの少なかった「今どこにいるのか」ということを始めとする相手の状況を問う発言が一定量付け加わったためではないかと考えられる．またこれは，実験を行った本研究室の居室が柏と本郷にそれぞれ 2 部屋ずつあり，特に柏の居室は 2 部屋が廊下で隔てられており相互に「向こうの部屋に今誰がいるか」ということを確認するのが困難であるため，たとえば柏にいるユーザにとって，自分の部屋に姿が見えないがオンライン状態になっているユーザがいた場合,「自分がいない方の柏の部屋にいる」のか「本郷のどちらかの部屋にいる」のかが区別がつかないことも一因として考えられる．しかし以上の検討は推測に基づいており，さらなる正確な検討を行うためには会話内容を全て記録し，どのような会話が行われたかを分析する必要がある．

一方で，実験 3 においては実験 1 や実験 2 の場合と比較して，図 4.12 よりチャット回数が多いことが，図 4.13 より t の値が小さいことが見て取れる．これはコミュニケーションの回数が多いと共に，相手の発言を受け取ってから自分が発言するまで，あるいは逆に自分が発言を送ってから相手の返信が届くまでの時間差が少ないことを意味しており，コミュニケーションがより同期的に行われたということを示している．また図 4.14〜図 4.17 からも，実験 3 の値は実験 1 や実験 2 よりも全体的に大きく，コミュニケーションが全体として活発だったことが推測できる．このような結果となった要因としては，特に t の値

の大小に関してはユーザが PC の前にいるかどうかというユーザのステータスが重要なことも勿論であるが，チャットで話すというコミュニケーションがユーザにとってどれほど重要性を持っているか，ユーザがコミュニケーションを何か他の作業を行いながら片手間に行うのではなくコミュニケーションを自身のメインの作業として行っているかと言うことに関連しているとも考えられる．そのような観点からは，実験 3 においてなされたコミュニケーションはユーザの興味に即したものであったということができ，BGM 情報や部屋の雰囲気といったプレゼンス情報によって，ユーザの興味に即した話題を提供することができたものと推測できる．

4.6 おわりに

本章では物理的に離れた拠点間での，日常生活の大部分を占めるインフォーマルなコミュニケーションの活性化を狙い，プレゼンスがコミュニケーションを効果的に開始させる能力をもつことに注目し，これがどのように構成されるかについて分析・検討を述べた．また，実際にインフォーマルなコミュニケーションへのプレゼンスの影響を検証すべく，ACDC システムと呼ぶプレゼンス情報プラットフォームの設計と，テストベッド・アプリケーションの実装を示した．

現在，実装したアプリケーションを実際に使用することで課題の抽出を行い解決策の検討を行っている．また，利用できるプレゼンス項目を変化させた場合におけるコミュニケーションの質・量への影響について評価を行い，これをもとにインフォーマルなコミュニケーションの支援能力のより高いプレゼンスについての検討を行っている．

第5章

/結論

5.1 本論文における成果

本論文では，ユーザ主導なサービス指向ネットワーキングを実現するという立場から，ユーザの要求に応じたネットワークの個人化と，それに基づいたユーザ間，サービス間の動的な通信制御を支援するためのネットワークミドルウェア，およびソフトウェア構成を論じた．現在，環境や端末，ユーザ，コンテキスト，サービスなどの多様化によって，ユビキタス・モバイル・IP ネットワークを収容し支えるための新しいネットワークアーキテクチャの必要性が求められている．このような状況において，ユーザ要求・サービスの多様さと，遍在リソースのカスタマイズ・管理の困難さ・複雑さというトレードオフにある関係を，それぞれ一定のレベルで両立させなければならない．そのため，個人によるネットワーク管理の複雑さを排除しつつユーザの多様な要求に応えられる枠組み，いわゆるユーザ主導ネットワーク構築技術が重要となる．

これに対する具体的な取り組みとして，ユーザの要求に応じた適応的なアクセス制御を目的とし，物理ネットワークトポロジに依存しないサービス指向な端末グルーピング機構について検討した．また，構築されたグループ内における効率的かつ適応的なブロードキャスト配信機能の提供を目的とし，アプリケーション層マルチキャスト（ALM）を用いて多地点配信サービスを支援するミドルウェアのアーキテクチャについて検討した．さらには，複数ユーザがネットワークを共有する場合にはユーザ同士の信頼関係を常時から醸成しておくことも重要な課題の 1 つであると考え，インフォーマルなコミュニケーションの活性化を支援するためのプレゼンス機構について検討した．本論文では以上の検討を通じて，ユーザ主導型ネットワークサービスとその構成のあり方についての議論を行った．

以下に本論文における主たる成果を示す．

- ユーザがサービス単位で柔軟なアクセス制御を行うための共通基盤をネットワークレベルで提供するサービス指向端末グルーピング機構，MyNetSpace（MNS）システムについて示した．仮想的にグルーピングされた端末上で動作するサービスに対して安全な内部通信機能と端末管理機能を提供し，また端末が複数のグループに同時に参加できる機構を実現した．

- 特定グループ内においてリアルタイムに多地点配信を行うアプリケーションの開発効率を改善するための ALM ミドルウェア，RelayCast のアーキテクチャについて示した．最小限かつ基本的な機能をユニット化することで，最小限の労力で多様なアプリケーションからの要求に応えることができるフレームワークを実現した．

- インフォーマルコミュニケーションを支援するための様々なプレゼンス情報の伝達

に対応しうるプラットフォームについて示した．ユーザごとに自律分散的動作できる UA を配置し，また受信側におけるプレゼンス情報合成できる機構を組むことで，実際に試用してプレゼンスが与える影響を見極めるための基盤を実現した．

また，上記 3 つの提案した機構，および実証アプリケーションについて実装を行い，実用性の検証，外部機構との連携による応用，ユーザに与える影響の調査について報告した．それぞれが実現可能性，適用可能性をもつことを検証し，またユーザに対して一定の影響を与えうることを確認した．

5.2 今後の展望

今後の展望としては以下のものが挙げられる．

- サービス検出機構の構築
  第 2 章で述べたように，IP 層から分離した形，いわゆる第 3.5 層にてサービス指向グルーピング機構である MNS システムを実現する際，サービス検出機構をどのように提供するかが重要な課題となる．UPnP（Universal Plug and Play）[90] に代表されるような既存のサービス・デバイス検出機構はサブネットを単位とした運用を前提としている．この問題は既存のブロードキャストチャネルを利用できないことに起因するため，MNS システム上では用いることはできない．したがって，MNS システムに適したサービス検出機構を用意する必要がある．これに対する解決案の 1 つは，MNS サーバをディレクトリサーバに見立ててサービス登録をする方法である．この際，クライアントとなる MNS マネージャが自端末で動くサービスを検出し，自動で MNS サーバに登録する仕組みが求められる．また，サービスの記述形式の規定や，サービス検出機構をユーザに提供するするための API をどのような形にしていくかなどを検討する必要がある．

- より高度な API の導入
  ユーザの要求に応じてネットワークを個人化し利用するにあたり，その状況をモニタリングするための API を規定することが課題となる．たとえば MNS システムであれば，仮想ネットワークインタフェースにどのような MNS がマッピングされているのかを調べるための API が要求される．また RelayCast であれば，どのようなオーバレイネットワークやマルチキャストツリーが形成されているのかを通知する API が要求される．後者に関しては，目的は異なるが，Nakao ら [91] がより汎用的

なオーバレイネットワーク構築機能の提供に向けて測定やモニタリングのための API を提供する仕組みを提案しており，それらを参考にすることも必要であろう．

- 安全性に対する検討
  インターネット上に新たな機構やサービスを導入する際，セキュリティに対する検討を行うことが必ず要求される．本論文でも通信路の暗号化やそれに用いる暗号鍵の配布の仕方などについては検討を加えている．しかし，実際には端末まで含めたシステム全体に関して，セキュリティ上問題となる箇所を指摘していく必要があると考える．

参考文献

[1] M. Weiser. The Computer for the Twenty-First Century. *Scientific American*, pp. 94–104, Sept. 1991.

[2] D. D. Clark. The design philosophy of the DARPA internet protocols. *Proc. ACM SIGCOMM 1988*, pp. 106–114, Aug. 1988.

[3] P. Srisuresh. Traditional IP Network Address Translator (Traditional NAT). RFC 3022, IETF, Jan. 2001.

[4] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460, IETF, Dec. 1998.

[5] C. Tschudin and R. Gold. Network Pointers. *Proc. First ACM Workshop on Hot Topics in Networks (HotNets-I)*, pp. 23–28, Oct. 2002.

[6] B. Ford. Unmanaged Internet Protocol. *Proc. Second ACM Workshop on Hot Topics in Networks (HotNets-II)*, pp. 93–98, Nov. 2003.

[7] R. Droms. Dynamic Host Configuration Protocol. RFC 1531, IETF, Oct. 1993.

[8] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and M. Walfish. A Layered Naming Architecture for the Internet. *Proc. ACM SIGCOMM 2004*, pp. 343–352, Sept. 2004.

[9] K. R. Sollins. Designing for Scale and Differentiation. *Proc. ACM SIGCOMM 2003 Workshop on Future Directions in Network Architecture (FDNA-03)*, pp. 267–276, Aug. 2003.

[10] J. D. Touch, Y.-S. Wang, L. Eggert, and G. G. Finn. A Virtual Internet Architecture. *Proc. ACM SIGCOMM 2003 Workshop on Future Directions in Network Architecture (FDNA-03)*, Aug. 2003.

[11] K. Psounis. Active Networks: Applications, Security, Safety and Architectures. *IEEE Communications Surveys and Tutorials*, Vol. 2, No. 1, pp. 2–16, First Quarter 1999.

[12] D. Wetherall, J. Guttag, and D. Tennenhouse. ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols. *Proc. First IEEE International Conference on Open Architectures and Network Programming (OPENARCH '98)*, pp. 117–129, April 1998.

[13] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient Overlay Networks. *Proc. 18th ACM Symposium on Operating Systems Principles (SOSP 2001)*, pp. 131–145, Oct. 2001.

[14] L. Subramanian, I. Stoica, H. Balakrishnan, and R. Katz. OverQoS: Offering Internet QoS Using Overlays. *Proc. First ACM Workshop on Hot Topics in Networks (HotNets-I)*, pp. 1–16, Oct. 2002.

[15] Y. h. Chu, S. G. Rao, and H. Zhang. A Case for End System Multicast. *Proc. ACM SIGMETRICS 2000*, pp. 1–12, June 2000.

[16] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet Indirection Infrastructure. *Proc. ACM SIGCOMM 2002*, pp. 73–88, Aug. 2002.

[17] A. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure Overlay Services. *Poc. ACM SIGCOMM 2002*, pp. 61–72, Aug. 2002.

[18] K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. Little, and G. Zorn. Point-to-Point Tunneling Protocol (PPTP). RFC 2637, IETF, July 1999.

[19] W. Townsley, A. Valencia, A. Rubens, G. Pall, G. Zorn, and B. Palter. Layer Two Tunneling Protocol (L2TP). RFC 2661, IETF, Aug. 1999.

[20] OpenVPN. http://openvpn.sourceforge.net/.

[21] PacketiX VPN 2.0 (SoftEther VPN 2.0). http://www.softether.com/jp/vpn/.

[22] Zone Labs: ZoneAlarm. http://www.zonealarm.com/.

[23] 藤田範人, 小出俊夫, 石川雄一, 塚本明. ピアツーピア型レイヤ 2 仮想ネットワークの提案. 電子情報通信学会ソサイエティ大会, No. B-6-59, Sept. 2004.

[24] S. Aoyagi, M. Takizawa, M. Saito, H. Aida, and H. Tokuda. ELA: A Fully Distributed VPN System over Peer-to-Peer Network. *Proc. 2005 Symposium on Applications and the Internet (SAINT 2005)*, pp. 89–92, Jan. 2005.

[25] JXTA. http://www.jxta.org/.

[26] R. Braden. Requirements for Internet Hosts – Communication Layers. RFC 1122, IETF, Oct. 1989.

[27] NDIS Developer's Reference. http://www.ndis.com/.

[28] netfiler/iptables project. http://www.netfilter.org/.

[29] R. Rivest. The MD5 Message-Digest Algorithm. RFC 1321, IETF, April 1992.

[30] D. Eastlake and P. Jones. US Secure Hash Algorithm 1 (SHA1). RFC 3174, IETF, Sept. 2001.

[31] R. Pereira and R. Adams. The ESP CBC-Mode Cipher Algorithms. RFC 2451, IETF, Nov. 1998.

[32] 電子タグリーダ搭載携帯電話試作機. http://www.kddi.com/corporate/news_release/2005/0302/.

[33] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen. Deployment Issues for the IP Multicast Service and Architecture. *IEEE Network*, Vol. 1, No. 14, pp. 78–88, Jan. 2000.

[34] A. Oram, editor. *PEER-TO-PEER Harnessing the Power of Disruptive Technologies*. O'Reilly, March 2001.

[35] Y.-H. Chu, S. G. Rao, S. Seshan, and H. Zhang. Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture. *Proc. ACM SIGCOMM 2001*, pp. 55–67, Aug. 2001.

[36] Y. Chawathe. Scattercast: An Adaptable Broadcast Distribution Framework. *ACM Multimedia Systems Journal*, Vol. 9, No. 1, pp. 104–118, July 2003.

[37] P. Francis. Yoid: Extending the Internet Multicast Architecture. http://www.icir.org/yoid/, April 2000.

[38] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel. ALMI: An Application Level Multicast Infrastructure. *Proc. Third USENIX Symposium on Internet Technologies and Systems (USITS '01)*, pp. 49–60, March 2001.

[39] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole. Overcast: Reliable Multicasting with an Overlay Network. *Proc. Fourth USENIX Symposium on Operating System Design and Implementation (OSDI 2000)*, pp. 197–212, Oct. 2000.

[40] J. Liebeherr and T. K. Beam. HyperCast: A Protocol for Maintaining Multicast Group Members in a Logical Hypercube Topology. *Proc. First International Workshop on Networked Group Communication (NGC '99)*, pp. 72–89, July 1999.

[41] B. Zhang, S. Jamin, and L. Zhang. Host Multicast: A Framework for Delivering Multicast To End Users. *Proc. IEEE Infocom 2002*, Vol. 3, pp. 1366–1375, June 2002.

[42] W. Wang, D. Helder, S. Jamin, and L. Zhang. Overlay Optimizations for End-host Multicast. *Proc. Fourth International Workshop on Networked Group Communication (NGC 2002)*, pp. 154–161, Oct. 2002.

[43] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable Application Layer Multicast. *Proc. ACM SIGCOMM 2002*, pp. 205–217, Aug. 2002.

[44] S. Y. Shi and J. S. Turner. Routing in Overlay Multicast Networks. *Proc. IEEE Infocom 2002*, Vol. 3, pp. 1200–1208, June 2002.

[45] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai. Distributing Streaming Media Content Using Cooperative Networking. *Proc. 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2002)*, pp. 177–186, May 2002.

[46] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz, and J. D. Kubiatowicz. Bayeux: An Architecture for Scalble and Fault-tolerant Wide-area Data Dissemination. *Proc. 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2001)*, pp. 124–133, June 2001.

[47] A. Rowstron, A.-M. Kermarrec, M. Castro, and P. Druschel. SCRIBE: The design of a large-scale event notification infrastructure. *Proc. Third International Workshop on Networked Group Communication (NGC 2001)*, pp. 30–43, Nov. 2001.

[48] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Application-level Multicast using Content-Addressable Networks. *Proc. Third International Workshop on Networked Group Communication (NGC 2001)*, pp. 14–29, Nov. 2001.

[49] S. Banerjee, C. Kommareddy, K. Kar, S. Bhattacharjee, and S. Khuller. Construction of an Efficient Overlay Multicast Infrastructure for Real-time Applications. *Proc. IEEE Infocom 2003*, Vol. 2, pp. 1521–1531, April 2003.

[50] Z. Xu, M. Mahalingam, and M. Karlsson. Turning Heterogeneity into an Advantage in Overlay Routing. *Proc. IEEE Infocom 2003*, Vol. 2, pp. 1499–1509, April 2003.

[51] T. Pusateri. Distance Vector Multicast Routing Protocol. *Internet-draft, draft-ietf-idmr-dvmrp-v3-10.txt*, Aug. 2000.

[52] IETF Reliable Multicast Transport Working Group. http://www.ietf.org/html.charters/rmt-charter.html.

[53] B. Whetten, L. Vicisano, R. Kermode, M. Handley, S. Floyd, and M. Luby. Reliable Multicast Transport Building Blocks for One-to-Many Bulk-Data Transfer. *IETF RFC3048*, Jan. 2001.

[54] M. Bawa, H. Deshpande, and H. Garcia-Molina. Transience of Peers and Streaming Media. *Proc. First ACM Workshop on Hot Topics in Networks (HotNets-I)*, pp. 107–112, Oct. 2002.

[55] J. Liebeherr, J. Wang, and G. Zhang. Overlay Socket: Programming Overlay Networks. *Proc. Fifth International Workshop on Networked Group Communication (NGC 2003)*, pp. 242–253, Sept. 2003.

[56] N. Mimura, K. Nakauchi, H. Morikawa, and T. Aoyama. RelayCast: A Middleware for Application-level Multicast Services. *Proc. Third International Workshop on Global and Peer-to-Peer Computing on Large Scale Distributed Systems (GP2PC 2003)*, pp. 434–441, May 2003.

[57] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-Aware Overlay Construction and Server Selection. *Proc. IEEE Infocom 2002*, pp. 1190–1199, June 2002.

[58] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei. Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification. *RFC 2362*, June 1998.

[59] A. Ballardie. Core Based Trees (CBT) Multicast Routing Architecture. *RFC 2201*, Sept. 1997.

[60] 中内清秀, チェンヤンユン, 森川博之, 青山友紀. ピア・ツー・ピアマルチキャストのためのマルチパスルーティングアルゴリズム. 情報処理学会マルチメディア，分散，協調とモバイルシンポジウム（DICOMO）, pp. 319–324, June 2001.

[61] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs). *IETF RFC3489*, March 2003.

[62] C. Huitema. Teredo: Tunneling IPv6 over UDP through NATs. *Internet Draft, draft-huitema-v6ops-teredo-05.txt*, April 2005.

[63] VIC - Video Conferencing Tool. http://www-mice.cs.ucl.ac.uk/multimedia/software/vic/.

[64] L. Rizzo. Dummynet: a simple approach to the evaluation of network protocols. *ACM Computer Communication Review*, Vol. 27, No. 1, pp. 31–41, 1997.

[65] C. Neustaedter and S. Greenberg. Balancing Privacy and Awareness in Home Media Spaces. *Proc. Fifth International Conference on Ubiquitous Computing (UbiComp 2003)*, pp. 297–314, Oct. 2003.

[66] P. Dourish, A. Adler, V. Bellotti, and A. Henderson. Your Place or Mine? Learning from Long-Term Use of Audio-Video Communication. *Computer Supported Cooperative Work*, Vol. 5, No. 1, pp. 33–62, 1996.

[67] 北陸先端科学技術大学院大学. ナレッジ・サイエンス. http://www.kousakusha.com/ks/, 2002.

[68] M. Zaner, E. Chung, and T. Savage. 3 degree and Net Generation : Designing for Inner Circles of Friends. *Proc. UbiComp 2003 Workshop on Intimate Ubiquitous Computing*, pp. 50–52, Oct. 2003.

[69] R. E. Grinter and L. Palen. Instant Messaging in Teen Life. *Proc. 2002 ACM Conference on Computer Supported Cooperative Work (CSCW 2002)*, pp. 21–30, Nov. 2002.

[70] S. A. Bly, S. R. Harrison, and S. Irwin. Media Spaces: Bringing People Together in a Video, Audio, and Computing Environment. *ACM Communications*, Vol. 36, No. 1, pp. 28–46, 1993.

[71] R. S. Fish, R. E. Kraut, R. W. Root, and R. E. Rice. Video as a Technology for Informal Communication. *ACM Communications*, Vol. 36, No. 1, pp. 48–61, 1993.

[72] D. Hindus, M. S. Ackerman, S. Mainwaring, and B. Starr. Thunderwire: a Field Study of an Audio-only Media Space. *Proc. 1996 ACM Conference on Computer Supported Cooperative Work (CSCW 1996)*, pp. 238–247, Nov. 1996.

[73] E. Isaacs, A. Walendowski, S. Whittaker, D. J. Schiano, and C. Kamm. The Character, Functions, and Styles of Instant Messaging in the Workplace. *Proc. 2002 ACM Conference on Computer Supported Cooperative Work (CSCW 2002)*, pp. 11–20, Nov. 2002.

[74] S. W. B. A. Nardi and E. Bradner. Interaction and Outeraction: Instant Messaging in Action. *Proc. 2000 ACM Conference on Computer Supported Cooperative Work (CSCW 2000)*, pp. 79–88, Nov. 2000.

[75] ICQ Instant Messenger. http://www.icq.com/.

[76] MSN Messenger. http://messenger.msn.com/.

[77] M. Handel and J. D. Herbsleb. What Is Chat Doing in the Workplace? *Proc. 2002 ACM Conference on Computer Supported Cooperative Work (CSCW 2002)*, pp. 1–10, Nov. 2002.

[78] M. Chuah. Reality Instant Messaging: Injecting A Dose of Reality into Online Chat. *Proc. 2003 Conference on Human Factors in Computing Systems (CHI 2003)*, pp. 926–927, April 2003.

[79] A. Ranganathan, R. H. Campbell, A. Ravi, and A. Mahajan. ConChat: A Context-Aware Chat Program. *IEEE Pervasive Computing*, Vol. 1, No. 3, pp. 52–58, July–Sept. 2002.

[80] T. Kindberg, J. Barton, J. Morgan, G. Becker, D. Caswell, P. Debaty, G. Gopal, M. Frid, V. Krishnan, H. Morris, J. Schettino, and B. Serra. People, Places, Things: Web Presence for the Real World. *HPLabs Technical Report HPL-2000-16*, Feb. 2000.

[81] P. Debaty and D. Caswell. Uniform Web Presence Architecture for People, Places, and Things. *IEEE Personal Communications*, Vol. 8, No. 4, pp. 6–11, Aug. 2001.

[82] D. Morikawa, M. Honjo, A. Yamaguchi, and M. Ohashi. A Proposal of User Profile Management Framework for Context-Aware Service. *Proc. IEEE/IPSJ Symposium on Applications and the Internet Workshops (SAINT2005 Ubiquitous Networking and Enablers to Context Aware Services Workshop)*, pp. 184–187, Jan. 2005.

[83] 内田達人, 敷田幹文. 状況情報の利用による分散型コミュニケーション支援法の提案. 情報処理学会研究報告, No. 2003-GN-49, Oct. 2003.

[84] . Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261, IETF, June 2002.

[85] The GNU oSIP library. http://www.gnu.org/software/osip/.

[86] AIBO. http://www.jp.aibo.com/.

[87] E. B. Campbell, J. Rosenberg, H. Schulzrinne, C. Huitema, and D. Gurle. Session Initiation Protocol (SIP) Extension for Instant Messaging. RFC 3428, IETF, Dec. 2002.

[88] パンサェーンジュター, 川田雅人, 小野孝之, 三村和, 森川博之, 青山友紀. 複数ユーザ同士の関係を考慮したプレゼンス交換と合成手法. 電子情報通信学会技術研究報告, No. IN2003-275, March 2004.

[89] M. Kawada, N. Mimura, H. Morikawa, and T. Aoyama. A User-Oriented Presence Synthesizing System for Facilitating On-line Communication. *Proc. IEEE/IPSJ Symposium on Applications and the Internet Workshops (SAINT2005 Teamware Workshop)*, pp. 242–245, Jan. 2005.

[90] UPnP (Universal Plug and Play) Forum. http://www.upnp.org/.

[91] A. Nakao, L. Peterson, and A. Bavier. A Routing Underlay for Overlay Networks. *Proc. ACM SIGCOMM 2003*, pp. 11–18, Aug. 2003.

発表文献

論文誌

[1] N. Mimura, K. Nakauchi, H. Morikawa, and T. Aoyama. Functional Unit Oriented Middleware for Application-Level Multicast Services, *IEICE Transactions on Communications, IEICE/IEEE Joint Special Section on Autonomous Decentralized Systems*, vol.E88–B, no.12, pp.4442–4450, Dec. 2005.

[2] 三村 和, 飛岡 良明, 平井 肇, 森川 博之, 青山 友紀. 柔軟なアクセス制御を実現するためのサービス指向グルーピング機構. 電子情報通信学会論文誌. (投稿予定)

[3] 三村 和, 小野 孝之, 川田 雅人, 森川 博之, 青山 友紀. インフォーマルコミュニケーション支援プレゼンス機構. 情報通信学会論文誌. (投稿予定)

国際会議

[4] N. Mimura, K. Nakauchi, H. Morikawa, and T. Aoyama. RelayCast: A Middleware for Application-level Multicast Services. *Proc. Third International Workshop on Global and Peer-to-Peer Computing on Large Scale Distributed Systems (GP2PC 2003)*, pp.434–441, May 2003.

[5] M. Kawada, N. Mimura, H. Morikawa, and T. Aoyama. A User-Oriented Presence Synthesizing System for Facilitating On-line Communication. *Proc. IEEE/IPSJ Symposium on Applications and the Internet Workshops (SAINT2005 Teamware Workshop)*, pp.242–245, Jan. 2005.

[6] N. Mimura, K. Nakauchi, H. Morikawa, and T. Aoyama. A Middleware Approach for Supporting Application-level Multicast Services. *Proc. First International Workshop on Ad Hoc, Sensor and P2P Networks (AHSP 2005)*, pp.689–694, April 2005.

国内会議，研究会

[7] 三村 和, 伊庭 斉志. 進化型探索手法による３次元箱詰め問題の解法. 情報処理学会全国大会, 7k-5, March 2001.

[8] 三村 和, 中内 清秀, 森川 博之, 青山 友紀. ピアツーピアマルチキャストミドルウェアの実装. 電子情報通信学会総合大会, B-6-85, March 2002.

[9] 三村 和, 中内 清秀, 森川 博之, 青山 友紀. RelayCast: ピアツーピア型ストリーム配信のためのミドルウェア. 電子情報通信学会技術研究報告, IN2002-42, July 2002.

[10] 三村 和, 中内 清秀, 森川 博之, 青山 友紀. アプリケーションレベルマルチキャストサービスのためのミドルウェアの動作検証. 電子情報通信学会総合大会, B-6-244, March 2003.

[11] 川田 雅人, 中村 岳, 三村 和, 森川 博之, 青山 友紀. ピアツーピア型多対多ビデオコミュニケーションのためのホスト構成. 情報処理学会マルチメディア, 分散, 協調とモバイル（DICOMO2003）シンポジウム, pp.341–344, June 2003.

[12] 川田 雅人, 正岡 直也, 三村 和, 森川 博之, 青山 友紀. ホスト特性の異種性を考慮した多対多オーバレイルーティングの実装評価. 電子情報通信学会ソサイエティ大会, B-7-57, Sept. 2003.

[13] パンサェーン ジュター, 川田 雅人, 小野 孝之, 三村 和, 森川 博之, 青山, 友紀. 複数ユーザ同士の関係を考慮したプレゼンス交換と合成手法. 電子情報通信学会技術研究報告, IN2003-275, NS2003-320, March 2004.

[14] 小野 孝之, 川田 雅人, パンサェーン ジュター, 三村 和, 森川 博之, 青山 友紀. ユーザ主導型プレゼンスシステムの設計と実装. 電子情報通信学会総合大会, SB-3-4, March 2004.

[15] 三村 和, 飛岡 良明, 森川 博之, 青山 友紀. MyNetSpace を実現するためのネットワーク空間の管理・制御機構 電子情報通信学会ソサイエティ大会, B-6-43, Sept. 2004.

[16] 飛岡 良明, 三村 和, 森川 博之, 青山 友紀. ユーザポリシを反映した仮想端末によるネットワーク空間の構築 電子情報通信学会ソサイエティ大会, B-6-44, Sept. 2004

[17] 飛岡 良明, 三村 和, 森川 博之, 青山 友紀. MyNetSpace：柔軟なアクセス制御を実現するためのユーザ主導仮想閉域ネットワーク. 電子情報通信学会技術研究報告, NS2004-224, March 2005.

[18] 小野 孝之, 三村 和, 川原 圭博, 森川 博之, 青山 友紀. インフォーマルコミュニケーションを支援するプレゼンス技術. 電子情報通信学会技術研究報告, NS2004-296, March 2005.

[19] 飛岡 良明, 三村 和, 森川 博之, 青山 友紀. MyNetSpace における仮想端末間通信機構の設計と実装. 電子情報通信学会総合大会, B-6-1, March 2005.

[20] 三村 和, 飛岡 良明, 森川 博之, 青山 友紀. 柔軟なサービスアクセス制御を実現するユーザ主導仮想閉域ネットワーク. 電子情報通信学会総合大会, B-6-2, March 2005.

[21] 林 辰也, 飛岡 良明, 三村 和, 森川 博之, 青山 友紀. MyNetSpace を構築するための端末管理機構. 電子情報通信学会総合大会, B-6-3, March 2005.

[22] 小野 孝之, 三村 和, 川原 圭博, 森川 博之, 青山 友紀. プレゼンスを活用したインフォーマルコミュニケーション支援システムの試作. 電子情報通信学会総合大会, B-20-10, March 2005.

[23] 三村 和, 平井 肇, 森川 博之, 青山 友紀. MyNetSpace における内部通信機構の実装と評価. 電子情報通信学会ソサイエティ大会, B-6-30, Sept. 2005.

[24] 平井 肇, 三村 和, 森川 博之, 青山 友紀. タグリーダ付携帯電話を用いた近傍仮想ネットワークの検出・参加機構. 電子情報通信学会ソサイエティ大会, B-6-31, Sept. 2005.

[25] 三村 和, 飛岡 良明, 森川 博之, 青山 友紀. サービス指向グルーピング機構を用いたユーザ主導ネットワークの構築. 第 13 回マルチメディア通信と分散処理（DPS）ワークショップ, pp.290–294, Nov. 2005.

[26] 平井 肇, 三村 和, 天野 翔, 森川 博之, 青山 友紀. MyNetSpace：柔軟なアクセス制御を実現するためのユーザ主導仮想閉域ネットワーク. 電子情報通信学会技術研究報告, NS2004-224, March 2005.

[27] 三村 和, 平井 肇, 森川 博之, 青山 友紀. 仮想端末グルーピング機構によるユーザ主導型プレゼンスシステムの改善. 電子情報通信学会総合大会, March 2006.

[28] 平井 肇, 三村 和, 森川 博之, 青山 友紀. 仮想端末グルーピング機構によるサービス指向ネットワークの実現. 電子情報通信学会総合大会, March 2006.

謝辞

終わりに鑑み，博士論文研究を通じて終始御指導を賜った指導教官である森川博之助教授に深謝の意を表します．本研究を進める過程で，知識はもちろんのこと，研究に対する心構えなども御教授くださり，また打ち合わせのみならず常日頃から有益な御指導，御批評，御鞭撻いただきました．今後も機会があれば，是非とも御一緒できればと存じます．

青山友紀教授には御指導を賜ったばかりでなく，研究室内外において研究環境を維持するのに御尽力いただき，ここに深謝の意を表します．本研究を進める上で常に的確な御批判，御鞭撻をいただき，またその一言一言に叱咤激励されました．そのような過程で青山教授に認めていただけたことは，研究を進める上で非常に励みになりました．

研究室の環境の整備に御尽力してくださった渡邉廣次助手，川原圭博助手，今泉英明助手，川北敦子秘書，宮島史子秘書に感謝の意を表します．また，かつて研究室にて助手をされ，現在は多方面にて御活躍中の鄭武龍さん，國頭吾郎さん，南正輝さん，王渓さんにも感謝の意を表します．上記の方々には研究だけでなく，生活の面においても非常にお世話になり，多大なる厚恩を賜りました．

そして研究生活の上では，博士課程の同学年である滕睿さん，グェンホアイソンさん，金子晋丈さん，岡敏生さん，パベルプピレフさん，孫咏梅さんをはじめとし，諸先輩・後輩の方々と数多くの議論をさせていただきました．彼らにもまた，感謝せずにはいられません．特に既に大学院を卒業された中内清秀さん，川田雅人さん，飛岡良明さんには，本研究を遂行するにあたって信念や研究の方向性などの面で多大なる影響と助力を賜りました．

もとより，本論文は筆者個人の力でできたものではなく，研究室の方々のみならず，多数の方々からの御教示や御協力をいただきました．最後にこれらの方々と，これまで筆者に勉励の機会を与え，そして支えてくれた両親に心から感謝の意を表します．

2006 年 12 月 16 日