アドホックグループを対象とした
協調作業支援フレームワークの提案と実現

小松 純

慶應義塾大学
総合政策学部 卒業論文

概 要

アドホックグループを対象とした協調作業支援フレームワークの提案と実現

本研究の目的はアドホックグループを対象とした協調作業支援環境の構築方法の提案と実現である。本研究では、Peer to Peer(P2P) モデルのサービスにおけるブートストラップ問題、オフライン同期問題を一般的なサービス資源を利用した擬似ノードによって解決することで、これを実現する。

近年、いわゆる「プロジェクトチーム」や「タスクフォース」といった特定の目的の達成のために一時的に構成されるようなグループの事例が多く見られるようになった。このようなグループを本論文では「アドホックグループ」と呼ぶ。

既存の協調作業支援環境の多くは企業等の継続的で大規模な組織を主な対象としており、アドホックグループでの利用においてサーバ導入・運用のコストが大きくなりすぎるという問題がある。P2P モデルによる協調作業支援環境では、サービスの大部分をサーバ等の固定的ノードを用いずに実現できるが、ブートストラップ問題やオフライン同期問題の解決のためには何らかの固定的ノードが必要である。既存の P2P グループウェア等では、この部分をソフトウェアベンダーが運営するサーバを用いて解決、あるいはユーザグループのネットワーク内に、常時ネットワークに接続した固定的ノードを導入するというアプローチによって解決している。しかし、このようなアプローチではソフトウェアベンダーの運営するサーバに依存し続けなければならず、あるいはユーザ自身でのサーバ導入・運営のためのコストを負担する必要があるため好ましくない。

本研究では前述の問題を一般的なサービス資源を利用した擬似ノードを導入することで解決する方法を提案した。そして、一般的なサービス資源としてメールシステムを利用したフレームワークを設計・実装した。

このフレームワークの応用として協調作業支援アプリケーション “collaboware” を実装した。そして、動作試験によって本フレームワークがブートストラップ問題とオフライン同期問題を解決できることを確認した。本研究に関連する研究について紹介し、今後の課題を示した。

キーワード:
協調作業支援, Peer to Peer, オフライン同期問題, ブートストラップ問題

慶應義塾大学総合政策学部
小松純

ABSTRACT

"Proposal and Implementation of CSCW Framework for Ad-hoc Groups"

The purpose of this research is to propose and to implement CSCW framework for ad-hoc groups. In this research, to realize the purpose, solve the bootstrup problem and the offline sycnronization problem on peer to peer(P2P) model service, by pseudo node using general service resources.

In recent years, the cases that groups temporary built for specific objectives what is called as "Project Team" or "Task Force", are appeared offen. In this thesis, such groups are called "ad-hoc group".

The most of existing environments are intended for continuous and large-scale organizations like enterprises, so these environments' initial server construction and operation costs are too high to use in ad-hoc group. In case of P2P model CSCW environments, the most part of services are able to realize, with no fixed role node like a server, but it is necessary to some fixed role node for solving the bootstrup problem and the offline sychronization problem. The existing groupwares solve these problems by the approaches either using servers operated by software vendors or importing fixed role node connecting to network all the time. But these approaches are not good, because they necessary to keep depend on servers operated by software vendors, or they necessary to import own servers and keep operation.

In this research, we proposed a solution of those problems by using pseudo node using general service resources. And we designed and implemented a framework using mail systems as general service resources.

We implemented co-operative work supporting software "collaboware" as application of this framework. And, We confirmed that the framework can solve the bootstrup problem and the offline synchronization problem, by run test. Finally we introduced relational researches, and showed future works.

Keywords:
CSCW, peer to peer, offline synchronization problem, bootstrup problem

Keio University, Fuculty of Policy Management

Jun Komatsu

目 次

図 目 次

表 目 次

第1章　はじめに

本章では、まず本研究の目的を述べる。そして本研究の背景としてコンピュータとネットワークによる協調作業支援の発展及び、アドホックグループについて説明する。また、本論文の残りの部分の構成について示す。

1.1　本研究の目的

本研究は、アドホックグループを対象とした協調作業支援環境の構築方法を提案・実現することを目的とする。

アドホックグループとは何らかの目的達成のために一時的に構成するグループのことである。

1.2　グループ協調作業支援の広がり

コンピュータやネットワークによるグループの協調作業支援は、CSCW(Computer Supported Co-operative Work) の研究として 1980 年代より盛んに研究されるようになった。また大企業等では 1990 年代以降、ホワイトカラーの生産性向上を目的としたグループウェアの導入が進んだ。

1990 年代後半に入ると、個人がコンピュータやインターネットを利用することも珍しいことではなくなった。総務省の平成 16 年版 情報通信白書によると、平成 15 年末におけるインターネットの世帯普及率は 88.1 %であり、9 割近くの世帯がインターネットを利用している [1]。

このような状況の中でコンピュータやネットワークによるグループの協調作業支援は、企業だけでなく、家庭や教育、行政、あるいは地域コミュニティなどの非商業的な場面でも利用されるようになってきている [2]。

1.3　アドホックグループ

アドホックグループとは何らかの目的達成のために一時的に構成されるグループである。

企業等では、従来の職能別組織では対応できない複雑な問題に対応するために、各職能毎の専門家を集めてプロジェクトチームをつくり問題解決にあたるということが頻繁に行われるようになった。この傾向は1つの企業の中に留まらず、企業の壁、または空間や場所の壁を超えて問題解決に取り組んでいくバーチャル・チームの事例は最近多く見られるようになってきた[3]。

身近な例を挙げると、大学の授業等では学生が数人のグループを形成して課題に取り組む、グループワークが盛んに行われている。このようなグループも、授業の課題の達成という目的のもとに一時的に形成されたアドホックグループである。

1.3.1 アドホックグループの性質

アドホックグループの性質について述べる。アドホックグループは一般に以下のような性質を持っている。

一時性 グループの存在期間が一時的であること。

目的志向性 グループが目的志向であること。

組織横断性 グループを構成するメンバーが所属する組織が、複数の組織にまたがっていること。

小規模性 グループを構成するメンバーの数が小さいこと。

流動性 グループを構成するメンバーの変化が流動的であること。

地理分散性 グループを構成するメンバーが地理的に分散していること。

時間分散性 グループを構成するメンバーの活動時間が時間的に分散していること。

1.4 用語の定義

本論文における用語で、特に注意を要するものを表1.1に示す。

1.5 本論文の構成

第2章では、既存のグループ協調作業支援環境としてメーリングリスト、ファイル共有システム、およびグループウェアを取り上げ、アドホックグループでの利用の観点からその特徴と問題点を整理する。

第3章では、アドホックグループを対象とした協調作業支援環境の要件を明らかにし、既存協調作業支援環境の問題点について明らかにした上で、本研究で解

表 1.1: 本論文における用語の定義

順番	用語	意味
1	オーバレイネットワーク (overlay network)	既存のリンクを用いて、その上位層における目的に応じて仮想的なリンクを形成し、構成するネットワーク
2	ロケーション (location)	オーバレイネットワークにおける下位層での識別
3	オンライン (online)	ノードがオーバレイネットワークに接続していること、つまり他の、オーバレイネットワークに接続しているノードに接続しているか、他のオーバレイネットワークに接続しているノードから接続可能な状態にあること。
4	オフライン (offline)	ノードがオーバレイネットワークから切断しているこ、オンラインの逆

決すべき問題として「オフライン同期問題」及び「ブートストラップ問題」を定義する。

第 4 章では、第 3 章で示した問題に対して、「一般的なサービス資源を利用した擬似ノードの導入」による解決を提案する。そしてこの方法による問題解決について全体的なモデルを示す。また本アプローチの既存の方法に対する優位性を考察する。

第 5 章では、4 章で示した問題解決手法のモデルを実際に実現するための P2P フレームワークについて概要を示し、設計と実装について述べる。

第 6 章では、5 章で設計したフレームワークの応用である、協調作業支援環境を実現するアプリケーション “collaboware” について述べる。

第 7 章では、第 3 章で示した問題の解決を確認するためのフレームワークの動作試験について述べる。そして動作試験の結果に基づいて本研究の評価を述べる。

第 8 章では、本研究の関連研究として、Mikan および qwikWeb について紹介し、本研究との比較を行う。

第 9 章では、本研究のまとめを行い、関連研究と今後の課題を示す。

第2章　既存のグループ協調支援環境

本章では既存の協調作業支援環境として、メーリングリスト、ファイル共有システム、およびグループウェアを取り上げ、前章で示したアドホックグループでの利用の観点からその特徴を整理する。

2.1　メーリングリスト

メーリングリストとは、特定の興味分野やテーマについての議論などを目的として、その参加者のメールアドレスのリストを作成し特定のメールアドレス宛てに送られたメールをそのリスト中のメールアドレスに転送することによってグループでのコミュニケーションを実現する仕組みである。

メーリングリストを用いると日常的に利用している電子メールを利用して、グループでのコミュニケーションの場を手軽に実現することができる。電子メールにファイルを添付することによって、テキストだけでなく様々なデータを共有することもできる。

従来はメーリングリストを利用するためには、ISP 等が提供するサービスを利用するか、あるいは、fml[5], Majordomo[6] 等のメーリングリストマネージャのソフトウェアを自身で運用しているメールサーバに導入して利用するといった方法がとられていたが、最近では Yahoo!グループ [7] や QuickML[8] のようなユーザ自身がメーリングリストの作成・管理を手軽に行える無料のサービスが普及してきている。

2.2　ファイル共有システム

ファイル共有システムとは、複数のユーザの間でファイルを共有するためのシステムである。グループの協調作業においてファイルの共有は、協調作業の素材や過程、そして成果物を共有する上で重要である。

特に集団でのソフトウェア開発の分野では、CVS(Concurrent Version System)[9] や subversion[10] のようなバージョン管理機能を持ったシステムが良く利用される。バージョン管理機能とは、ファイルが更新されるたびにバージョン番号を振り、また更新の差分を保存しておくことで、後からファイルの更新過程を追跡すること

ができる機能である。バージョン管理機能により、複数人のユーザによって同じファイルを更新していく場合等に大変便利である。

2.3 グループウェア

グループウェアとは、グループでの作業をコンピュータやネットワークを活用して支援するソフトウェアの総称、あるいは、それらの組み合わせによる統合的なソフトウェアである。本論文ではグループウェアという言葉を後者の意味で捉える。

現在、対象とするグループが取り組む作業の内容、グループの規模、利用形態などに応じて様々なグループウェアが存在する。一般的にグループウェアは、ファイル共有の機能やグループのスケジュール管理機能、掲示板等のコミュニケーション機能を含んでいる。本節では、グループウェアをその実現方法および提供形態で表 2.1 のように分類し、それぞれの特徴について説明する。

表 2.1: グループウェアの分類

		提供形態	
		パッケージソフトウェア	ASP サービス
実現方法	C/S 型	C/S 型グループウェア	ASP 型グループウェア
	P2P 型	P2P 型グループウェア	

2.3.1 クライアント/サーバ型グループウェア

クライアント/サーバ型グループウェアとは、サービスを提供するサーバと、サーバの提供するサービスをただ利用するクライアントとに明確に分かれるネットワークモデルによって実現するグループウェアである。

Lotus Notes/Domino[11] は代表的なグループウェア製品であり、大企業等が全社的に利用するような利用シーンにも対応可能な製品である。

サイボウズ Office 6[12] は、中小企業や、大企業の中の 1 セクション等の中・小規模のグループを対象としたグループウェアである。サーバプログラムを LAN 内に設置されたサーバにインストールし、ユーザは Web ブラウザを用いてサーバにアクセスすることで利用できる。

クライアント/サーバ型グループウェアを利用するためには、サーバの導入と運用が必要である。サーバの導入にはサーバマシンの購入、サーバ構築、担当技術者の確保など、一般的に大きなコストが掛かる。

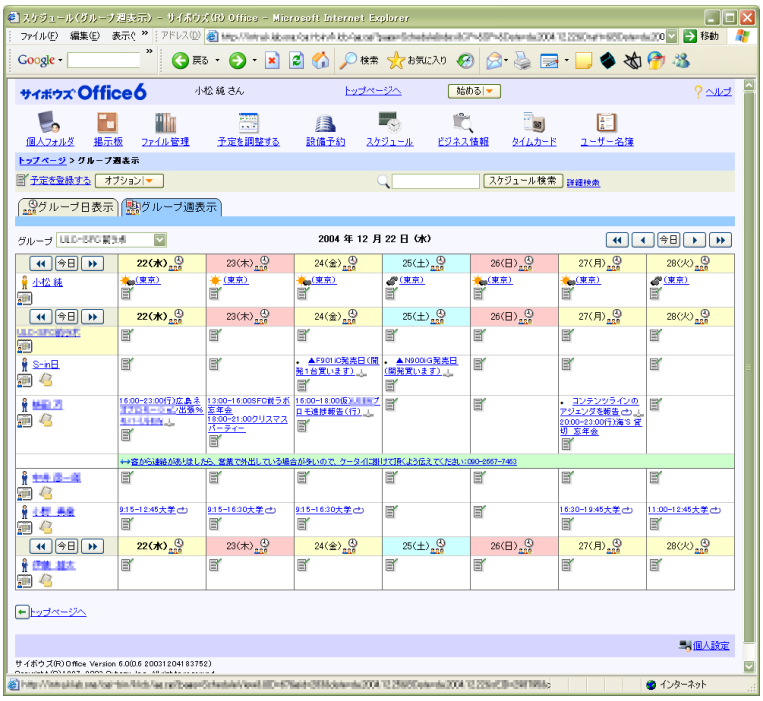

図 2.1: サイボウズ Office 6 画面

図 2.2: Intranets PRO 画面

2.3.2 ASP 型グループウェア

ASP(Application Service Provider) とは、ネットワークを介して、多くの利用者に対しアプリケーションソフトウェアと、それを動作させるサーバ等環境を合わせて提供し、利用者数や、利用期間に応じた利用料を支払うというサービスモデルによってサービスを提供する事業者、およびそのサービス提供形態である [13]。

ASP 型グループウェアのひとつである、イントラネッツ [14] は、伝言板、スケジュール管理、共有アドレス帳、文書共有等のグループウェアの基本的な機能を提供する。イントラネッツでは「Intranets Enterprise」、「Intranets PRO」、「Intranets Basic」の 3 種類のグレードのサービスを用意しており、PRO は月額 5 千円程度の利用料金で利用でき、機能限定版の Intranets Basic については無料で利用することができる。

また、2.1 節で取り上げた Yahoo!グループも、メーリングリストが機能の中心ではあるがファイル共有等の機能も含まれる ASP 型グループウェアの一種である。

ASP 型グループウェアでは 2.3.1 節で述べたようなサーバの導入と運用は必要なく、初期コストの面で大幅に有利である。2.3.1 節で取り上げたクライアント/サーバ型グループウェアと、ASP 型グループウェアの初期・運用コスト比較を表 2.2 に示す。

表 2.2: クライアント/サーバ型・ASP 型 グループウェアコスト比較

	Lotus Notes/Domino	サイボウズ	Intranets PRO
初期コスト	20 万円〜	8 万円〜	5,040 円
サービス利用コスト (月額)	0 円	0 円	5,040 円〜

2.3.3 P2P 型グループウェア

P2P モデルとは、役割の対等なノード (peer) 同士の通信によってサービスを実現するネットワークモデルである。

P2P 型グループウェアとはこの P2P モデルによって実現されているグループウェアである。

本節では、既存の P2P 型グループウェアとして、Groove Virtual Office 及びアリエル・エアワンについて取り上げて説明する。

Groove Virtual Office

Groove Virtual Office は米 Groove Networks 社の製品であり、Lotus Notes の生みの親といわれる Ray Ozzie によって開発された P2P 型グループウェアである [15]。

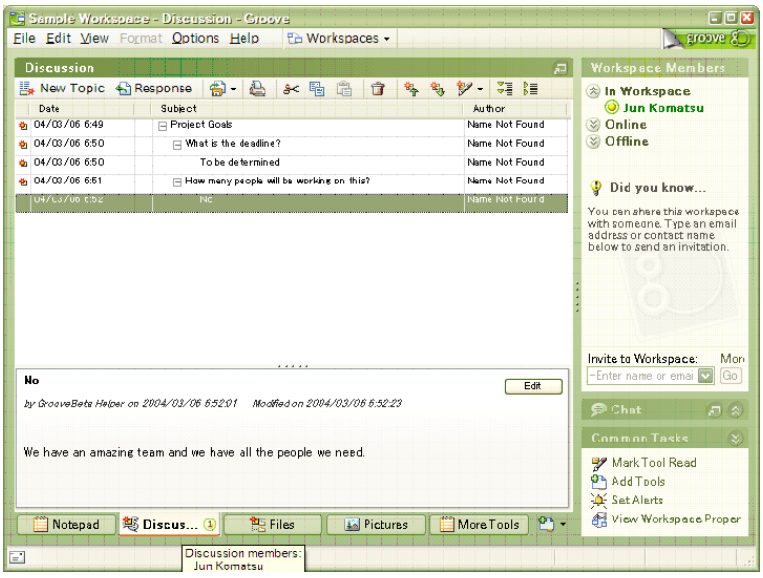

図 2.3: Groove Virtual Office 画面

Groove Virtual Office では、Workspace と呼ばれる共有空間を作成し、その空間を通じて他のメンバとメッセージのやり取りや情報の共有を行う。

Groove のネットワークには、Relay Service 及び、Management Service を提供するサーバが存在する。

Relay Service はノードがネットワークに接続するための最初の接続先ノードとなり、NAT(Network Address Translator) やファイヤウォール等の存在によりノード間で直接通信できない場合に中継接続を行ったり、また、通信相手のノードがネットワークに接続していない場合には、更新差分通知を蓄積し、対象のノードが接続してきたときに蓄積された通信内容を伝える蓄積通信を実現している。

一方 Management Service は、各ユーザの状況を把握し、ドメイン・グループ単位で運用ポリシやアクセス権限を規定し、それを適用することで、ネットワークを、管理可能にする役割を担っている。

Relay Service や Management Service は、通常は Groove Networks 社が運用するサーバを利用する。一方で、Groove Networks 社は企業向けに Groove Enterprise Relay Server および Groove Enterprise Management Server というサーバソフトウェアのパッケージ製品を用意しており、これらのサーバソフトウェアを購入して自前の Relay Service および Management Service を構築することで、Groove Networks 社のサーバを使わずに、自前でこれらのサーバを運用することで、きめ細やかな管理や、要求されるサービスレベルを満たしたサービス運用が可能となる。

アリエル・エアワン

図 2.4: アリエル・エアワン 画面

アリエル・エアワンはアリエルネットワーク社が開発した P2P 型グループウェアである [16]。

アリエル・エアワンでは、ユーザ認証に PKI を利用しており、ユーザはアリエル・エアワンを PC に導入するときに、アリエル社の運営する認証サーバに接続して、証明書を作成し、PC 内に保存する。

アリエル・エアワンでは、ルームという単位で情報の共有を行う。

アリエル・エアワンにも、アリエルネットワーク社が運用する中継サーバは存在するが、その役割は限定されている。

アリエル・エアワンではワークグループ・ノードと呼ばれる仮想ノードをユーザが設置してルームに含めることで、ワークグループ・ノードによってデータの収集や中継接続を行うようになり、ルーム内での情報共有や相互接続が安定するようになる (図 2.5)。

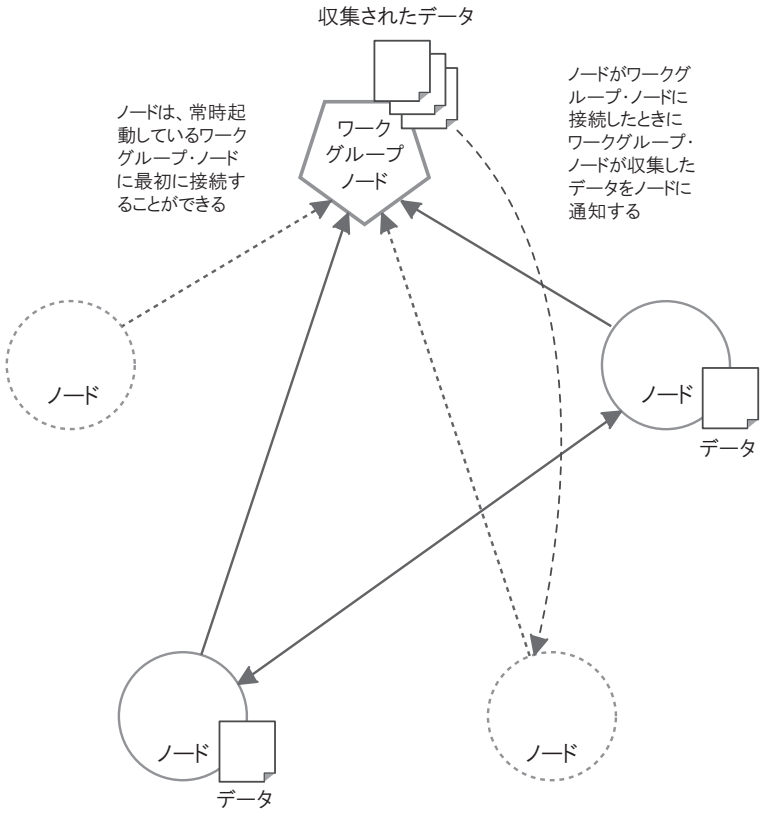

図 2.5: アリエル・エアワンのワークグループ・ノード

第3章　問題の定義

本章では、まずアドホックグループを対象とした協調作業支援環境の要件を明らかにする。そして、既存協調作業支援環境の問題点について述べ、、それを踏まえて、本研究で解決すべき問題である「オフライン同期問題」及び「ブートストラップ問題」を定義し説明する。

3.1　協調作業支援環境の要件

1.3 節で説明したアドホックグループを対象とした協調作業支援環境の要件は、

- 低い初期コスト
- 利用環境の自由
- 高度な専門知識を必要としないこと
- 可用性の維持
- 機密性の維持

である。

以下にそれぞれの要件についての説明を示す。

3.1.1　低い初期コスト

アドホックグループはその一時性から、協調作業支援環境を利用する際に大きな初期コストを負担することが難しい。継続的組織を考えた場合には、初期コストが高くても長期の運用の中で償還していくことが可能だが、アドホックグループでは運用期間が一時的であるために初期コストを低く抑えることが重要となる。

3.1.2　利用環境の自由

アドホックグループではメンバーの所属や所在等が多様であるため、利用環境についても多様であると考えられる。そのためアドホックグループを対象とした協調作業支援環境では、メンバーの利用環境の多様性に対応できるよう、ユーザの利用環境に対して制限が少ないことが求められる。

3.1.3 高度な専門知識を必要としないこと

アドホックグループに、サーバ管理のような高度な技能を持っているメンバーがいるとは限らないため、アドホックグループを対象とした協調作業支援環境においては、サーバ管理のような高度な技能を必要とせずに、環境を利用できる必要がある。

3.1.4 可用性の維持

グループのコミュニケーション基盤として様々な情報の流通を担う協調作業支援環境は、常に利用可能であることが求められる。アドホックグループでは特に、グループのメンバーが同じ空間を共有できないことも多く協調作業支援環境に対する依存度が高まると考えられるため、可用性が維持されることは重要である。

3.1.5 機密性の維持

協調作業支援環境の上でやりとりされる情報には様々なものが含まれる。3.1.4でも述べたように、アドホックグループでは協調作業支援環境への依存度が高まるため、機密性の高い情報についても協調作業支援環境上でやりとりできることが求められる。

3.2 既存協調作業支援環境の問題

本節では3.1節で明らかにした要件を踏まえ、アドホックグループが第3.1章で取り上げた既存の協調作業支援環境を利用する上での問題点について述べる。

3.2.1 クライアント/サーバ型協調作業支援環境の問題

いわゆるクライアント/サーバ型の協調作業支援環境では、サーバ導入にかかるコストや、その運用コストがアドホックグループにとって大きな負担となる。

また、サーバの導入や運用には、多くの場合高度な専門知識が必要となる。しかし、アドホックグループを対象とした場合、3.1.3で述べた理由から、ユーザが自らこのようなサーバを設置し運用することは難しい。

2.3.1節で取り上げたLotus Notes/Dominoやサイボウズでは、LAN内にサーバを設置しての利用が想定されている。LAN内にサーバを設置して利用するような場合においては、LAN外からのアクセスがファイアウォールなどによって妨害され不可能となり、3.1.2節で述べた利用環境の自由が満たされない可能性がある。

3.2.2 ASP 型協調作業支援環境の問題

2.3.2 節で示したような ASP 型の協調作業支援環境を利用することでサーバ導入にかかる初期コストを低く抑えることができる。

また、一般的に ASP 型サービスはインターネットへの接続環境さえあれば、どこからでもアクセスすることができるという利点がある。

システムの導入や運用については ASP 事業者が行うため、ユーザにシステムを運用するための高度な専門知識は求められることはない。

しかし、ASP 型協調作業環境では可用性の維持の面で問題がある可能性がある。多くの ASP 型協調作業環境では、ASP サービス提供者が多くの利用者に対して汎用的なシステムを利用させるモデルをとっており、サーバ資源は共有されている。そのため他の利用者の利用が一時的に集中するなどでサーバやネットワークの資源が枯渇し、通常のサービスを受けられない状態になる可能性が考えられる。

また、機密性の維持の面でも、まず ASP サービス提供者がアドホックグループが協調作業支援環境上でやり取りする様々な情報、利用履歴や利用者の個人情報、その他の情報に触れることになり、この点で幾分機密性が失われている。また、サーバ資源は他の利用者と共用であるため、場合によっては他の利用者にそれらの情報が漏れる危険性も少なからず存在する。

3.3 P2P モデルによる協調作業支援環境の構築

P2P モデルで実現されている協調作業支援環境として、2.3.3 節で Groove Virtual Office とアリエル・エアワンを取り上げた。これらのソフトウェアは、ユーザがソフトウェアをソフトウェアベンダのウェブサイトからダウンロードし、自分の PC にインストールしてグループの設定をすることで利用可能になる。基本的にはユーザ自身によるサーバの導入・運用は不要である。このように P2P モデルによる協調作業支援環境は、3.1.1 節や 3.1.3 節で述べた要件を満たし、アドホックグループでの利用に適していると考えられる。

3.3.1 既存 P2P 型協調作業支援環境の問題点

しかし、2.3.3 節で示したように Groove Virtual Office のネットワークにはソフトウェアベンダが運用している中継サービスが組み込まれているし、アリエル・エアワンではネットワークを安定させるために「ワークスペース・ノード」を設置することを推奨しており、実際にワークスペース・ノードを設置しなかった場合同時にオンラインになっているメンバー間で接続できなかったり、接続が確立するまで時間が掛かったり、また発信した情報が届かなかったりということが発生することがある。

3.4 解決すべき問題

表 3.1: 既存の協調作業支援環境とアドホックグループにおける要件

要件	C/S 型 (Lotus Notes/Domino)	ASP 型 (Intranets PRO)	既存 P2P 型 (アリエル・エアワン)
初期コスト	大きい	小さい	小さい
利用環境の自由	△	○	○
高度な専門知識の要求	大きい	小さい	小さい
可用性の維持	○	△	○
機密性の維持	○	△	○

3.1 節で述べたアドホックグループにおける協調作業支援環境の要件を満たすためには、協調作業支援環境の構築において、ユーザ自身でサーバ的な固定ノードを自身で導入・運用しなくてもよいようにする必要がある。

クライアント・サーバ型の協調作業支援環境では、3.2.1 節で述べたように、サーバの導入・運用に掛かるコストの面で 3.1.1 節や 3.1.3 節で述べた要件を満たさず、ASP 型協調作業支援環境では、3.2.2 節で述べたように、可用性の維持および機密性の維持の点で問題がある。

既存の P2P 型協調作業支援環境においては、3.3 節で述べたように、基本的にはユーザはサーバを必要とせず、要件を満たしているように思える。しかし、実際にはサービスの一部においてソフトウェアベンダの運営するサーバや、「ワークグループ・ノード」のような固定的なノードをネットワーク内に含める必要がある。

このような固定的ノードをネットワークに設置する必要が生じているのは、以下に詳しく説明するブートストラップ問題及び、オフライン同期問題を解決するためであると考えられる。

3.4.1 オフライン同期問題

オフライン同期問題とは、

- 参加しているノードが一時的にオーバレイネットワークから切断することがある

という前提を持ち、

- オーバレイネットワークに接続している全てのノードは、共有されているリソース集合の一貫性・完全性が保たれる

- 何れかのノードが共有されているリソース集合に対して行った変更は、オーバレイネットワーク上の（接続していないノードも含む）全てのノードに共有される

ということが保障されるオーバレイネットワークにおいて、オーバレイネットワークから切断しているノードに情報の伝達ができないことに起因して、このことが保障されなくなるという問題である。

3.4.2 ブートストラップ問題

ブートストラップ問題とは、オーバレイネットワークに接続していないコンピュータがオーバレイネットワークに接続するために、最初に接続するエントリーポイントとなるノードのロケーションを、どのように獲得するかという問題である。

多くのコンピュータは、DHCP 等の機構により動的に IP アドレスの割り当てが行われていることにより、オーバレイネットワークに参加しているノードのロケーション (IP アドレス) も、接続ごとに変化する可能性がある。このような環境下では、ブートストラップ時にオーバレイネットワークに接続している他のノードのロケーションを確定的に把握することができない。

第4章　問題解決のアプローチ

本章では、前章の 3.4.1 節で示したブートストラップ問題、および 3.4.2 節で示したオフライン同期問題に対して、「一般的なサービス資源を利用した擬似ノードの導入」による解決を提案する。そしてこの方法による問題解決について全体的なモデルを示す。また本アプローチの既存の方法に対する優位性を考察する。

4.1　一般的なサービス資源を利用した擬似ノード

前章において、アドホックグループでは利用する協調作業支援環境の実現のために専用のサーバを導入・運用することが難しく、また、特定のサービスプロバイダが提供するサービスに依存することが好ましくないことを示した。

また、P2P モデルを利用することで、サービスの大部分をサーバを用いずに実現できるが、ブートストラップ問題、オフライン同期問題といった問題に対処するために、何らかの固定的ノードの存在が依然必要とされることも示した。

本研究では、前章で示した P2P モデルにおけるオフライン同期問題、ブートストラップ問題をユーザが既に利用可能な一般的なサービス資源を利用した擬似ノードによって解決する方法を提案する。これによって専用のサーバや、それに準じる固定的ノードの設置、もしくは特定のサービスプロバイダが運営するサービスに依存することなく協調作業支援環境を実現でき、結果 3.1 節で述べた要件を満たすことができる。

4.1.1　擬似ノードを実現する一般的なサービス資源の要件

一般的なサービス資源とは、標準化され広く普及しており様々な提供者によって一般的に提供され、一般的なユーザが利用可能なサービス資源である。

擬似ノードとは、ノードと結び付けられ、ノードがオフラインのときに、代わりに他のノードからのメッセージを受信し、蓄積し、ノードがオンラインになったときにその蓄積されたメッセージをノードに通知するノードである。

擬似ノードを実現するための一般的なサービス資源は、以下の要件を満たす必要がある。

- 常時利用可能であること

- 情報の蓄積が可能なこと
- ユーザの識別・認証が可能であること

4.2 システム全体のモデル

前節で述べた一般的なサービス資源を利用した擬似ノードを含んだシステム全体のモデルを図 4.1 に示す。

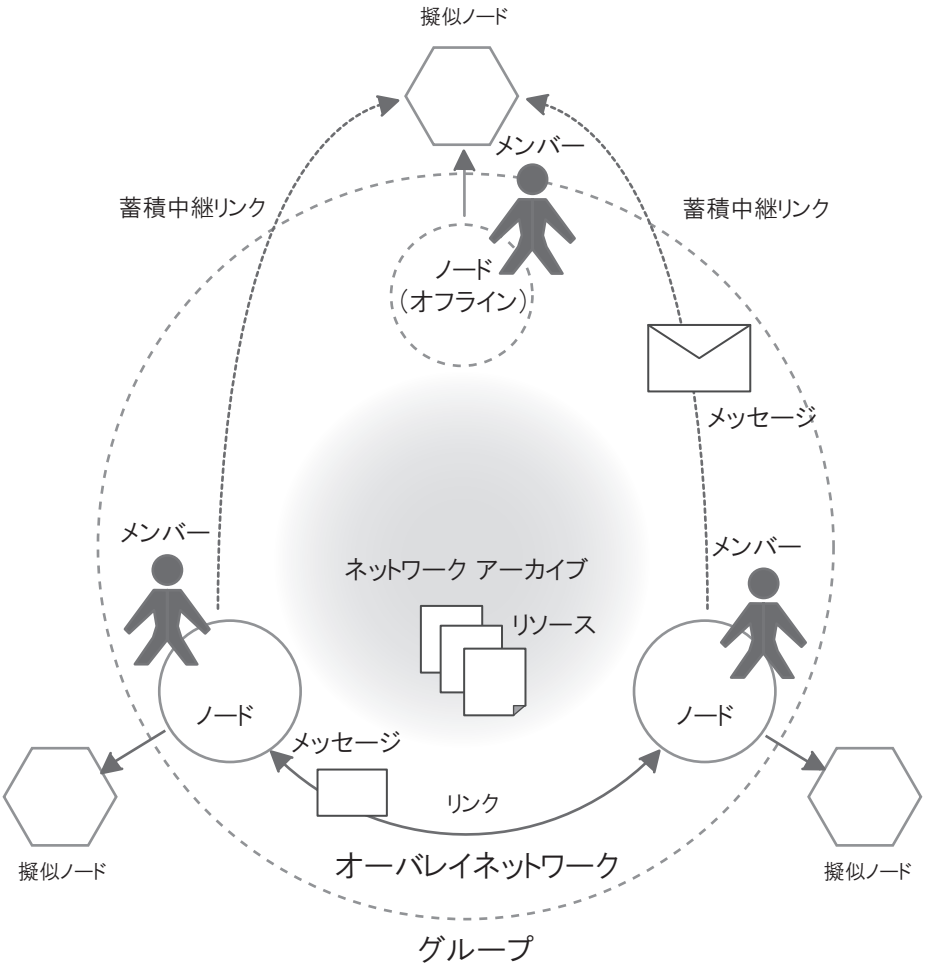

図 4.1: システム全体のモデル

このモデルに含まれる主な抽象とその役割を表 4.1 に示す。

表 4.1: モデルにおける主な抽象とその役割

番号	抽象	役割
1	グループ	同一の目的を持ち協調作業を行うユーザの論理的集合であり、そのための情報を共有する範囲を規定する。
2	メンバー	グループに属するユーザーであり、リソースにアクセスする主体である。
3	ノード	オーバレイネットワークに参加する自律的な存在の単位、オンライン状態とオフライン状態の二つの状態をとる。
4	擬似ノード	蓄積中継リンクを実現するために、対応するノードに対するメッセージを蓄積し、ノードがオーバレイネットワークに接続した時点で蓄積されたメッセージをノードに伝達する擬似的なノード。
5	リンク	ノード間でメッセージをやりとりするための伝送路。
6	蓄積中継リンク	擬似ノードによって実現される、メッセージを送信したい相手がオフラインであったときに擬似ノードを経由して蓄積中継通信を行うための擬似的なリンク。
7	メッセージ	ノード間でやり取りされる様々な情報。
8	リソース	グループによって共有される情報。
9	ネットワークアーカイブ	オーバレイネットワークに参加するノードによって実現される共有リソースの集合。

4.3 本アプローチの優位性

表 4.2: 既存の方式と本研究のアプローチの比較

	本アプローチ	既存 P2P 型	C/S 型	ASP 型
外部サービスプロバイダへの依存	無し	有り	無し	有り
初期導入コスト	小	小&中	大	小
運用コスト	小	小&中	大	小

表 4.2 に既存の協調作業支援環境と本研究のアプローチの比較を示す。

本アプローチは、従来の P2P ネットワークにおけるブートストラップ問題やオフライン同期問題の解決手段である、「中央サーバによる中継接続」や、「常時接続ノードの設置」に比べ、以下の点において優位性があると考えられる。

- **ソフトウェアベンダが提供する中継サーバ等を利用しなくて良い**

  3.3 節で指摘したように、Groove Virtual Office やアリエル・エアワンにおいては、基本的にはソフトウェア提供元が運営するサーバを利用することで、ブートストラップ問題の解決や、オフライン同期問題の解決が図られる。

  しかし、このモデルには 3.3 節で説明したように、ソフトウェアベンダが提供するサービスへの依存が発生する点で問題がある。

  本方式では、ソフトウェアベンダが提供するサーバを使うよりも、ユーザが既に利用可能な一般的なサービス資源を用いることによって特定のソフトウェアベンダやサービスプロバイダが集中的に運用するサーバの利用に依存する必要が無くなる。

- **ユーザが自身でサーバ的固定ノードを用意しなくて良い**

  3.3 節で述べたように、Groove Networks では企業などの大規模な組織において Groove Virtual Office を利用することを想定して、Relay Server というサーバ製品を用意している。前項で述べたような問題は、このように、自前でサーバを設置することで解決されるが、その場合は、やはりサーバ設置のコストの問題がネックとなる。

第5章　フレームワークの設計と実装

本章では、前章で示したモデルを実現する協調作業支援環境のフレームワークの設計と実装について述べる。まず、重要な設計上の決定である擬似ノードを実現する一般的なサービス資源の選定について述べ、その後、フレームワークの仕様について延べ、そしてその詳細と実装について述べる。

5.1　擬似ノードを実現する一般的なサービス資源の選定

4.1.1 節で、擬似ノードを実現する一般的なサービス資源の要件について述べた。

本節では、フレームワークを設計・実装するにあたり、擬似ノードを実現するための一般的なサービス資源について可能性を検討し、本フレームワークで擬似ノードを実現するために利用する一般的なサービス資源について選定する。

WWW

WWW は最も一般的なインターネットサービスのひとつである。WWW は、HTML というハイパーテキスト記述言語によって記述されたウェブページを、ハイパーリンクによって辿っていくことで、分散したホストに存在する関連する情報に次々にアクセスできるシステムである。

WWW のシステムは、ウェブページを公開するウェブサーバと、それを閲覧するウェブブラウザによって構成され、ウェブサーバとウェブブラウザの間の通信は、HTTP というシンプルなプロトコルによって実現されている。

現在、多くの商用 ISP がそのユーザに対して、数 MB から数十 MB 程度のウェブページ公開用のディスクスペースを提供しておりユーザはその領域を使ってウェブページを公開することが可能である。このような ISP のユーザ用ウェブサーバは基本的には常時利用可能であると考えてよいだろう。また、HTTP の基本認証のスキームを利用することでウェブサーバにアクセスするユーザの識別・認証も可能であり、CGI 等のプログラムを利用することで、情報の送信・蓄積も可能になる。しかし、一般的な WWW のサービスは、公に情報を公開することを想定しており、多くの ISP において、特定の領域にユーザの識別・認証を行えるような設定を行うことは、ウェブサーバの設定方法等の知識が必要であり、一般のユーザには難しい。CGI 等のプログラムをウェブサーバ上で利用することも、同様に、一

般のユーザには難しいと考えられる。あるいは、ISP のセキュリティポリシによっては、このような使い方を許容していない可能性も考えられる。

Instant Messaging (Jabber)

Instant Messaging は、同時にネットワークに接続している知人に対して手軽にコミュニケーションをとることができるツールとして、近年急速に普及してきたサービスである。

現在の Instant Messaging のサービスは AOL, MSN, Yahoo! 等、様々な事業者によってそれぞれ独自の内容のサービスとして提供され、それぞれのサービス間での相互運用性は確保されていない。また、これらの事業者によるサービスについては、プロトコルの仕様などもあまり公開されておらず、クライアントプログラムについても、一部クローンは存在するが基本的には事業者が提供するクライアントプログラムの利用を前提としている。

Jabber[17] は、そのような状況を踏まえて開発された Instant Messaging のシステムであり、オープンなプロトコル仕様と、オープンソースによるソフトウェア実装によって、様々なサーバやクライアントが開発されている。

wija[19] は Media Art Online[18] で開発されている、Jabber プロトコルによる Instant Messaging をサポートしたプログラムである。wija の特徴のひとつとして、プラグインによって wija を拡張することが可能な点が挙げられる。

Jabber のシステムでは、クライアント間のメッセージは Jabber サーバを通じて送られ、メッセージの送り先のノードが Jabber サーバに接続していなかったとしても、Jabber サーバ内にそのメッセージが蓄積される。また、Jabber システムでは、「ユーザ名@ドメイン名/リソース名」の形式を持つ Jabber ID によって通信の相手が識別されるようになっており、これによってユーザの識別・認証も可能である。Jabber サーバ自体は jabberd 等のオープンソースの jabber サーバソフトウェアを用いることで誰でも立ち上げることができるが、jabber.org や jabber.jp など既に多くのパブリックな jabber サーバが存在しているのでこれを利用することもできる。

このように、Jabber は擬似ノードを実現する一般的なサービス資源の要件を満たしている。

メール

メールも WWW と同様に最も一般的なインターネットサービスのひとつである。総務省の平成 16 年度 情報通信白書によると、自宅のパソコンからのインターネット利用用途で最も多いのが連絡手段としての「電子メール」(57.6%) であった [1]。このように、電子メールは多くのインターネット利用者にとって身近なサービスであると考えられる。

一般的な商用 ISP は、ほぼ例外なくユーザにメールアカウントを提供している。企業はもとより、最近では大学や高校等の教育機関でも、当たり前のように学生・生徒にメールアカウントが与えられるようになった。

また、hotmail や Yahoo!メール、Gmail[20] 等のサービスを利用すると、ウェブページから情報を入力して登録することで、メールアカウントを無料で発行することができる。このようなサービスでは、ユーザがそのサービスを通じてやりとりするメールの文末部分等に、広告が挿入され、その広告収入によりサービスが成立している。

このように、メールはインターネットを利用しているユーザの大多数が既に利用しており、また利用していないとしても、無料のサービスによりメールアカウントを取得することができ、ユーザが望めば利用可能なものであると考えられる。

メールサービスは最も基本的なインターネットサービスであり、基本的には常に利用可能になっていると考えられる。メールサーバ上には、ユーザが未受信のメールを一時的に蓄積するために、各ユーザごとに数 MB から数十 MB の容量制限を持った領域が確保されているのが一般的である。この領域を利用することで、情報の蓄積も可能である。また、メールアドレスによるユーザの識別ができ、メールアドレスを利用するためには、メールサーバに対してパスワードによる認証を要する。

5.1.1 一般的なサービス資源の比較

表 5.1: 一般的なサービス資源の比較

要件	WWW	Instant Messaging (Jabber)	メール
常に利用可能である	◎	○	◎
情報の蓄積が可能	△	○	○
ユーザの識別・認証が可能	△	◎	◎
広く普及している	◎	△	◎
様々な提供者により一般的に提供	◎	△	◎
一般的なユーザが利用可能	×	○	◎

表 5.1 に、一般的なサービス資源の比較を示す。この表に示した中で、全ての要件及び前提条件を他の一般的なサービス資源に対して高度に達成しているため、メールを擬似ノードを実現する一般的なサービス資源として利用することとする。

5.2　フレームワークの仕様

フレームワークが提供する機能は、

- ネットワークアーカイブへの透過的アクセス
- ネットワーク状態管理
- リンク管理
- メールの送受信
- メッセージの送受信
- ネットワークイベントモデルによるイベントハンドリング

である。

以上の仕様を満たすフレームワークを設計・実装する。

5.3　フレームワークの構成

本節ではフレームワークを構成する要素としてローカルキャッシュの管理、グループメンバーリスト、リンクマネージャについて述べる。

5.3.1　ローカルキャッシュの管理

本フレームワークでは、ネットワークアーカイブと呼ばれる、グループごとに存在する仮想的な情報共有空間を実現する。

ネットワークアーカイブには、「リソース」と「フォルダ」という 2 種類のリソースを保持することができ、フォルダは他のフォルダやリソースを含むことによって階層構造をつくる。

実際には、ネットワークアーカイブはどこかに実体が存在するものではなく、オーバレイネットワークに接続している各ノードのローカルキャッシュ間で、常に内容の同期が取れている前提の元に、1 つの仮想的な情報共有空間を想定しているにすぎない。

フレームワークは、このローカルキャッシュを管理し、オーバレイネットワーク上の他のノードのローカルキャッシュと内容の同期が取れている状態を保つように動作する。

そのために、アプリケーションからのローカルキャッシュに対するリソースの追加・更新・削除等の変更作業が行われる時に、他のノードに後述するメッセージによってその変更作業の内容が伝達される。それぞれのノードは、このようなリソースに対する変更作業のメッセージを受信した場合、その内容を、各ノードのローカルキャッシュに反映する。これによって、アプリケーションに対して、ネットワークアーカイブに対する透過的なアクセスが実現される。

5.3.2 グループメンバーリスト

本フレームワークでは、オーバレイネットワークに接続しているノードは、そのオーバレイネットワークに対応するグループのメンバー全員の情報を把握する。これをグループメンバーリストと呼ぶ。

グループメンバーリストに含まれる情報を表 5.2 に示す。

表 5.2: グループメンバーリストに含まれる情報

番号	情報	説明
1	メンバー識別名	メンバーを一意に識別するための識別名。擬似ノードであるメールサーバで利用するアカウントに対応付けられたメールアドレスを利用する。
2	名前	アプリケーション上で表示する等の目的で利用されるメンバーの名前
3	接続状態	オーバレイネットワークに接続しているか、接続していないかの状態を示す。
4	ロケーション (IP アドレスおよびポート番号)	メンバーに対応付けられたノードに対してリンクを形成するために必要なロケーション情報。
5	情報更新日時	この情報が更新された日時
6	接続リンク数	メンバーに対応付けられたノードがオーバレイネットワーク内のノードと形成しているリンクの数

本フレームワークでは、オーバレイネットワークに接続している各ノードにおいて、このグループメンバーリストの情報を常に最新の正しい状態に保つ。そのために、各メンバーおよびノードの情報が変更された場合には、後述するメッセージによってその変更内容を伝達する。また、それぞれのノードは、このようなグループメンバーリストに対する変更に関するメッセージを受信した場合、その内容を、各ノードのグループメンバーリストに反映する。

5.3.3 リンクマネージャ

リンクマネージャはノードが他のノードとの間に形成するリンクを管理する。

グループメンバーリストが更新され、他のノードの接続状態が変化した場合には、リンクマネージャは、その変化に応じて新しくリンクを接続しようとしたり、既にあるリンクを切断しようとしたりする。

またリンクマネージャは、定期的にリンクの生存確認を行い、下位層での障害等によってリンクが切断された場合等にそのことをフレームワークに通知する。

5.3.4 メール送受信マネージャ

メール送受信マネージャは、蓄積中継リンクによるメッセージ送受信のために利用される。メール送受信マネージャは、メール送信部とメール受信部によって成っており、メール送信部は SMTP によってメールを送信し、メール受信部は、POP3 あるいは IMAP によってメールを受信する。

メールの送信は、オフラインのノードに対してメッセージを送信する必要が生じたときに随時行われ、メールの受信は、ノードがオンラインになるときに行われるのと、ポーリングによりメールの受信が確認される。

5.4 メッセージ

グループ内のノードは、ノードのネットワーク接続状態変化や、ネットワークアーカイブに対する変更をグループ全体で共有するために、相互にメッセージを交換する。メッセージはリンクを通じて交換される。

本フレームワークがノード間でやりとりする、メッセージの種類を表5.3に示す。

表 5.3: メッセージの種類

種類	説明
LOGIN	ネットワークへの接続を通知する
STATE	自ノードの把握しているグループリストを通知する
JOIN	グループへ新たに参加するために既存のグループに対して承認要求を行う
DROP	グループから脱退する
PING	他のノードの応答性を確認する
GROUP	グループメンバーリストの内容を転送する
LOGOUT	ネットワークからの切断を通知する
GET	リソースの取得を要求する
PUT	リソースの登録・更新を要求する
DELETE	リソースの削除を要求する
LIST	フォルダ内のリソースリストを通知する

メッセージの種類には、ノードの状態変化を通知するための、LOGIN, STATE, JOIN, DROP, PING, GROUP, LOGOUT と、ネットワークアーカイブに対する変更を通知・共有するための GET, PUT, DELETE, LIST とがある。

5.4.1 メッセージのフォーマット

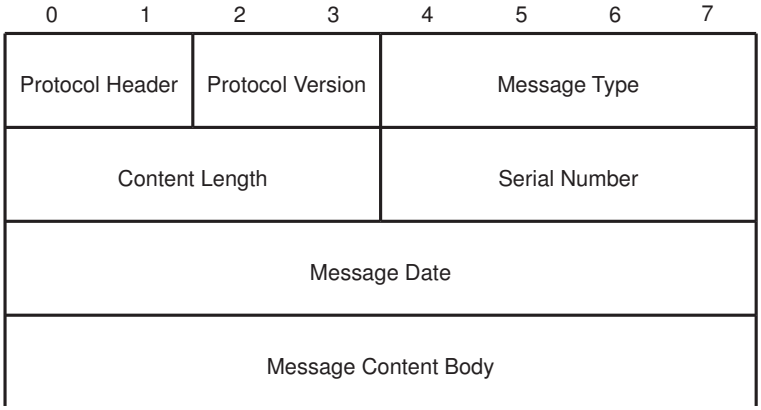

図 5.1: メッセージフォーマット

メッセージのフォーマットを図5.1に示す。メッセージには常に24バイトのヘッダが付いており、先頭から2バイトのプロトコルヘッダ、2バイトのプロトコルバージョン、4バイトのメッセージ種別、4バイトのメッセージの本体部のデータ長、4バイトのメッセージシリアルナンバー、そして、8バイトのメッセージ送信日時(1970年1月1日0時0分[GMT]からの経過ミリ秒)が含まれている。複数バイトの整数についてはネットワークバイトオーダー表現を用いる。

5.4.2 メールによるメッセージの送信

メッセージを送信する対象のノードがオフライン状態の場合、メッセージが蓄積中継リンク、つまりメールによって送信されることになるが、本フレームワークでは、メッセージをメールによって伝送する時に以下の方法に基づいてメッセージを送信する。

まず、メッセージをMIMEエンコードする。そしてエンコードされたメッセージをメールのBody部に設定する。メールのヘッダ部に、本フレームワークのメッセージであることを示す特別なヘッダを追加する。

5.5 ノードの基本動作

ノードの一連の基本動作に沿って、本フレームワークの設計と実装を述べる。

新しいグループの作成

それぞれのノードは、新規にグループを作成することができる。

ノードは新しいグループを識別するための新しいグループ識別名を決定し、その識別名に基づいて新しいグループを作成する。

新しく作られるグループの初期メンバは、グループを作成したメンバのみである。

新しいグループが作成されるとフレームワークによってそのグループメンバリストとネットワークアーカイブのローカルキャッシュが管理されるようになる。

既存グループへの参加

ノードが既存のグループへ新たに参加する場合は、参加対象のグループのに既に参加しているノードに対してグループへの参加許可の要求を JOIN メッセージとして送信する。

参加許可の要求を受けたノードがその要求に対して参加許可する場合は、自ノードのグループメンバーリストに参加許可元のノードの情報を追記し、そのグループメンバーリストの内容をグループ内の他のノード及び、参加要求元のノードに GROUP メッセージとして送る。

グループの他のノードはそのメッセージを受けて、ノード毎のグループメンバーリストの内容を更新する。

参加要求元のノードはグループメンバーリストを受け取ったことで、自身がそのグループへの参加を許可されたことを確認する。

ネットワークへの接続

ノードがネットワークに接続するときには、まず、そのノードがネットワークに接続していない間に他のノードから発信されたメッセージを、各ノードに対応した擬似ノードから取得し、そのメッセージの内容を、グループメンバーリストおよびローカルキャッシュへ反映する。

その後、グループメンバーリストの内容に基づいて、グループ内のオンライン状態のノードに対してリンクを試みる。

リンク接続要求を受けたノードがそれを受容すると、LOGIN メッセージをグループ内の全てのノードに対して通知する。リンクが存在するノードに対してはリンクを経由してメッセージを送信し、リンクが存在しないノードに対しては、擬似ノードに対してメッセージを送信する。LOGIN メッセージを受け取ったグループ内のノードは、グループメンバーリストへその内容を反映し更新する。

ネットワークからの切断

ノードがネットワークから切断する場合は、まず全てのノードに対して、LOGOUT メッセージを送信する。リンクが存在するノードに対してはリンクを経由してメッセージを送信し、リンクが存在しないノードに対しては、擬似ノードに対してメッセージを送信する。その後、全てのリンクを切断する。

ネットワークアーカイブに対する更新

ノードがネットワークアーカイブに対してリソースの新規追加や更新を行うとき、そのノードはグループ内の他の全てのノードに対して PUT メッセージを送る。リンクが存在するノードに対してはリンクを経由してメッセージを送信し、リンクが存在しないノードに対しては、擬似ノードに対してメッセージを送信する。PUT メッセージを受信したノードは、ローカルキャッシュ内の該当リソースと比較し、ローカルキャッシュ内のリソースより新しいと判断された場合、更新された内容をローカルキャッシュに反映する。

第6章　フレームワークの応用

第5章で設計したフレームワークの応用として、協調作業支援環境を実現するアプリケーション collaboware を実装する。

6.1　応用の目的

本アプリケーションを実装する目的は、フレームワークの有効性を検証することにある。実際に協調作業支援を行うためのアプリケーションを実現し、動作を確認することで、フレームワーク自体の有効性について確認することができる。

6.2　実装環境

collaboware は、メンバーの PC 上にで動作するアプリケーションとして実装する。

今回 collaboware の実装には、Java 言語を用い、GUI の実装のために SWT(Standard Widget Toolkit) 及び JFace を用いた。

実装環境として Java 言語を用いることによって、Java VM がインストールされていれば、OS やコンピュータのアーキテクチャが異なっていてもプログラムを実行することができる。アドホックグループにおいては、一般的な企業などと違い、メンバの利用するコンピュータの OS やアーキテクチャについて統一されているという前提を置くことができない。複数の実行環境をサポートできることは、システムの利用可能性を大きくするうえで必要なことである。

従来は、実行速度などのパフォーマンスの面で不利であった Java であるが、近年のコンピュータ自体の性能向上と、VM 技術の進化によって、ネイティブコードのプログラムに比べても遜色のない実行速度を得ることができるようになった。

SWT と JFace はオープンソースの統合開発環境である Eclipse の開発のために産み出された GUI ツールキット及びフレームワークである。JNI(Java Native Interface) を積極的に利用することで、従来の AWT や Swing などの Java 標準の GUI に比べて軽快で OS の GUI コンポーネントとの親和性の高いユーザインタフェイスを実現することができる。

6.3 機能

本アプリケーションは、「プレゼンス共有」、「テキストチャット」および「ファイル共有」の機能を含む。本節では、その詳細を説明する。

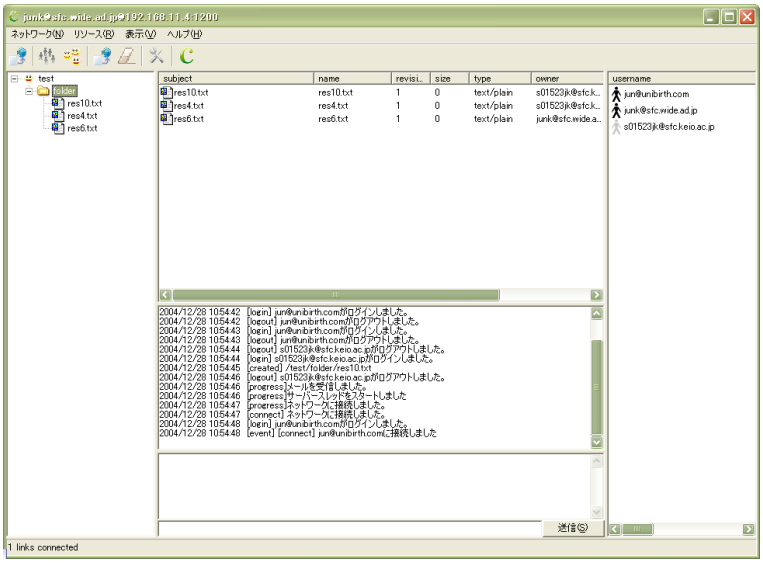

図 6.1: collaboware 画面

6.3.1 プレゼンス共有

同じグループ内の他のメンバーが、ネットワークに接続されているかどうかをリアルタイムに確認する機能を持っている。

また、メンバーはそれぞれ自分の状態を選択、或いは文字によって入力することによって、他のメンバーに自分のステータスを伝えることができる。

6.3.2 テキストチャット

同時にオンライン状態になっているグループ内のメンバー間で、テキストでのリアルタイムのコミュニケーションが可能である。

6.3.3 ファイル共有

グループ内のメンバーでのファイル共有の機能を持つ。この機能はフレームワークの持つネットワークアーカイブを利用したものである。バージョン管理機能を

持っている。

第7章　評価

本章では、本研究で設計・実装した協調作業支援環境が、「ブートストラップ問題」及び「オフライン同期問題」について解決できるかを確認する動作試験について述べる。

7.1　試験環境

今回の実験のための試験環境を図 7.1 に示す。

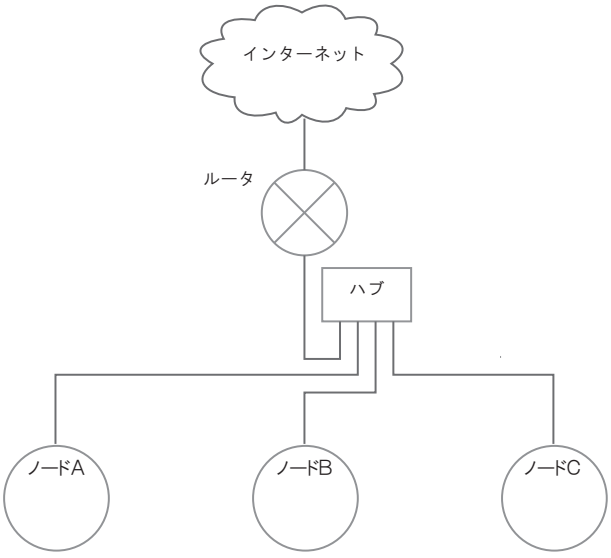

図 7.1: 試験環境

この試験環境は、１つの LAN セグメント内に全てのノードが存在することを想定した試験環境である。

動作試験に参加するノードとしてノードＡ、ノードＢ、およびノードＣの 3 つのノードを用意し、それぞれのノードを、PC は 100BASE-TX 規格のイーサネットでハブと接続し、同じくハブに接続されたルータを経由してインターネットに接続する。

7.2 試験方法

第 6 章で実装を行った collaboware を以下のシナリオに沿って各ノードをそれぞれのオーバレイネットワークに接続したり、オーバレイネットワークから切断したりし、それぞれブートストラップ問題およびオフライン同期問題の解決が図られるかを確認すると共に、その過程でそれぞれのノードがどのような挙動を示すかを観察する。

ノード間、あるいは擬似ノードに対してどのような通信が行われているかを、OGA 氏のフリーソフトウェアである TCP Monitor Plus[23] を利用して観察する。

7.2.1 ブートストラップ問題解決の確認のための動作試験

3.4.2 節で述べたブートストラップ問題の解決を確認するための動作試験シナリオを図 7.2 に示す。

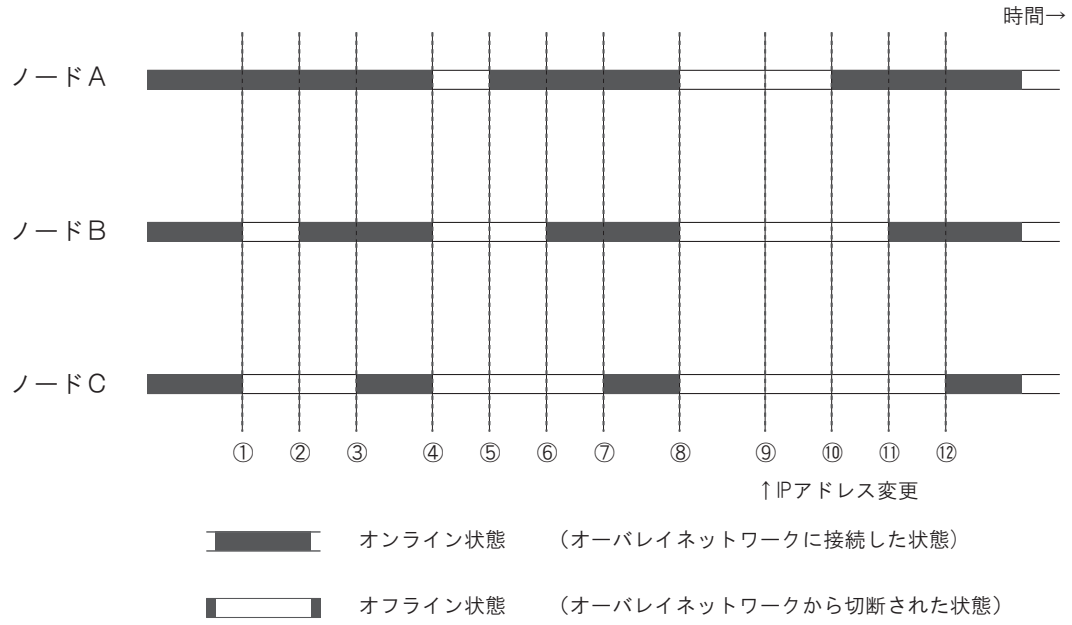

図 7.2: ブートストラップ問題解決の確認のための動作試験シナリオ

図 7.2 は、ノードA、ノードB、ノードCの 3 つのノードのオーバレイネットワークへの接続状態の時間軸に沿った変遷を示したものである。

まず初期状態において全てのノードはオンライン状態であり、3 つのノードは相互に接続されている状態である。①のタイミングでノードBおよびノードCは一旦オフラインとなり、その後②のタイミングでノードBがオンラインになり、③のタイミングでノードCがオンラインになる。次に、④のタイミングで全てのノードがオフラインとなり、その後、⑤のタイミングでノードA、⑥のタイミングで

ノードB、⑦のタイミングでノードCがそれぞれオンラインになる。⑧のタイミングで再度全てのノードがオフラインとなり、⑨のタイミングでは全てのノードに振られているIPアドレスを変更する。その後⑩のタイミングでノードA、⑪のタイミングでノードB、⑫のタイミングでノードCがオンラインとなる。

このシナリオでは、

1. ②および③のタイミングでオーバレイネットワーク内にオンラインのノードが継続的に存在し続ける状態でブートストラップ問題が解決されるか確認し、
2. ⑤・⑥・⑦のタイミングでオーバレイネットワーク内にオンラインのノードが継続的に存在し続けない状態でブートストラップ問題が解決されるか確認し、
3. そして⑨のタイミングでIPアドレスを変更することによって、⑩・⑪・⑫のタイミングでそれぞれのノードにおいて把握している他ノードのロケーションについての情報が正しくないものとなった場合にブートストラップ問題が解決されるかを確認する。

7.2.2 オフライン同期問題解決の確認ための動作試験

3.4.1節で述べたオフライン同期問題の解決を確認するための動作試験シナリオを図7.3に示す。

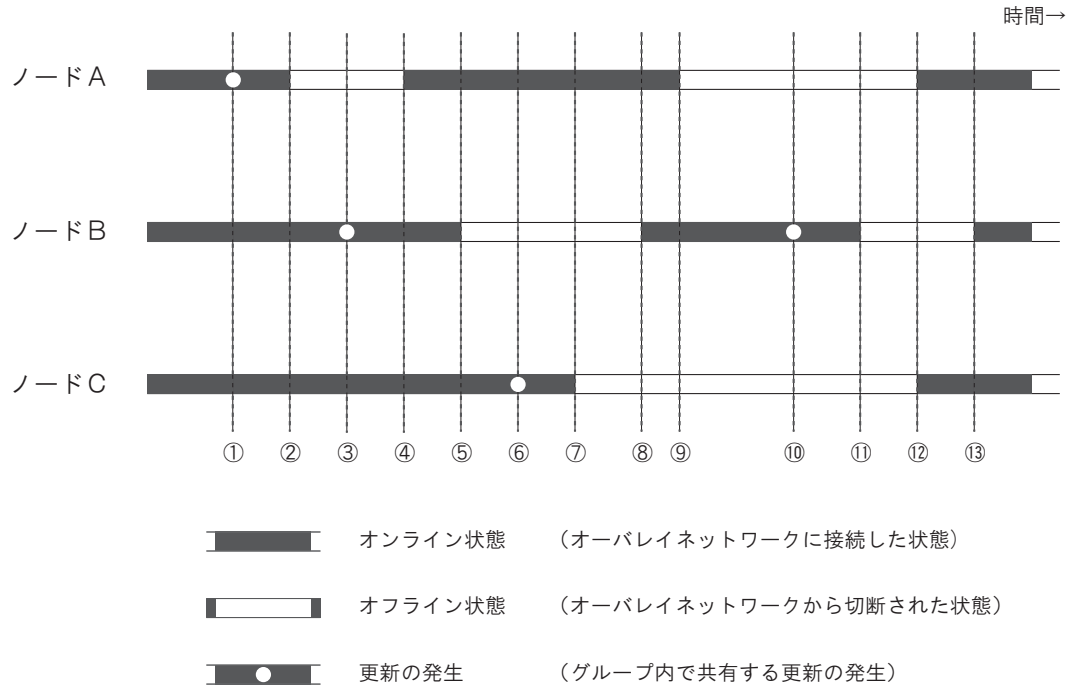

図 7.3: オフライン同期問題解決の確認のための動作試験シナリオ

まず初期状態において全てのノードはオンライン状態であり、3つのノードは相互に接続されている状態である。①のタイミングでノードAで更新が発生する。その後②のタイミングでノードAがオフラインになる。次に③のタイミングでノードBで更新が発生する。その後④のタイミングでノードAがオンラインになり、⑤のタイミングでノードBがオフラインになる。⑥のタイミングでノードCで更新が発生し、その後⑦のタイミングでノードC、⑧のタイミングでノードAがオフラインになり、そして⑨のタイミングでノードBがオンラインになる。⑩のタイミングでノードBで更新が発生し、⑪のタイミングでノードBがオフラインになり、⑫のタイミングでノードAおよびノードCがオンラインに、⑬のタイミングでノードBがオンラインになる。

このシナリオでは、

1. まず全てのノードがオンライン状態である①のタイミングにノードAで発生した更新がノードBおよびノードCに反映されることを確認し、
2. ③のタイミングにノードBで発生した更新が④のタイミングの時点でノードAがオンラインになったときにノードAに反映されることを確認する。
3. 次に⑥のタイミングにノードCで発生した更新が⑧のタイミングでノードBに反映されることを確認する。
4. 最後に、⑩のタイミングにノードBで発生した更新が⑫のタイミングでノードAおよびノードCに反映されることを確認する。

7.3 試験結果

ブートストラップ問題の解決の確認のための動作試験での各ノードの挙動の観察結果を表 7.1 に示す。

ブートストラップ問題の解決の確認を表 7.2 に示す。この動作試験によって、ブートストラップ問題の動作試験の全ての確認事項において、期待通りの挙動を示すことが確認できた。よって、本フレームワークによってブートストラップ問題を解決できることが確認できた。

オフライン同期問題の解決の確認のための動作試験での各ノードの挙動の観察結果を表 7.3 に示す。

オフライン同期問題の解決の確認を表 7.4 に示す。この動作試験によって、オフライン同期問題の動作試験の全ての確認事項において、期待通りの挙動を示すことが確認できた。よって、本フレームワークによってオフライン同期問題を解決できることが確認できた。

表 7.1: ブートストラップ問題の解決の確認のための動作試験結果

	ノードA	ノードB	ノードC
	ノードA、Bと接続 IP addr:192.168.11.2	ノードA、Cと接続 IP addr:192.168.11.6	ノードA、Bと接続 IP addr:192.168.11.3
①	ノードB、Cとのリンクが切断	ノードA、Cから切断	ノードA、Bから切断
②	ノードBから接続	ノードAに接続	
③	ノードCから接続	ノードCから接続	ノードA、Bに接続
④	ノードB，Cから切断	ノードA, Cから切断	ノードA, Bから切断
⑤	オンライン状態		
⑥	ノードBから接続	ノードAへ接続	
⑦	ノードCから接続	ノードCから接続	ノードA、Bへ接続
⑧	ノードB，Cから切断	ノードA, Cから切断	ノードA, Bから切断
⑨	IP アドレスを更新 IP addr:192.168.11.3	IP アドレスを更新 IP addr:192.168.11.2	IP アドレスを更新 IP addr:192.168.11.4
⑩	オンライン状態		
⑪	ノードBから接続	ノードAに接続	
⑫	ノードCから接続	ノードCから接続	ノードA、Bへ接続

表 7.2: ブートストラップ問題の解決の確認

番号	確認事項	確認
1	②および③のタイミングでオーバレイネットワーク内にオンラインのノードが継続的に存在し続ける状態でブートストラップ問題が解決されるか	○
2	⑤・⑥・⑦のタイミングでオーバレイネットワーク内にオンラインのノードが継続的に存在し続けない状態でブートストラップ問題が解決されるか	○
3	そして⑨のタイミングで IP アドレスを変更することによって、⑩・⑪・⑫のタイミングでそれぞれのノードにおいて把握している他ノードのロケーションについての情報が正しくないものとなった場合にブートストラップ問題が解決されるか	○

表 7.3: オフライン同期問題の解決の確認のための動作試験結果

	ノードA	ノードB	ノードC
	ノードA、Bと接続	ノードA、Cと接続	ノードA、Bと接続
①	‘PUT /test/folder/’ を送信	‘PUT /test/folder/’ をノードAより受信、ローカルキャッシュへ反映	‘PUT /test/folder/’ をノードAより受信、ローカルキャッシュへ反映
②	ノードB、Cから切断	ノードAが切断	ノードAが切断
③		‘PUT /test/folder/res4.txt’ を送信（ノードAに対してはメールにカプセル化して送信）	‘PUT /test/folder/res4.txt’ をノードBより受信、ローカルキャッシュへ反映
④	‘PUT /test/folder/res4.txt’ をメールにて受信、ローカルキャッシュへ反映、ノードB、Cに接続、	ノードAから接続	ノードAから接続
⑤	ノードBが切断	ノードA、Cから切断	ノードBが切断
⑥	‘PUT /test/folder/res6.txt’ をノードBより受信、ローカルキャッシュへ反映		‘PUT /test/folder/res6.txt’ を送信（ノードBに対してはメールにカプセル化して送信）
⑦	ノードBが切断		ノードAから切断、オフライン状態へ
⑧	ノードBから接続	‘PUT /test/folder/res6.txt’ をメールにて受信、ローカルキャッシュへ反映、ノードAへ接続、	
⑨	ノードBから切断、オフライン状態へ	ノードAが切断	
⑩		‘PUT /test/folder/res10.txt’ を送信（ノードA、C共にメールにて送信）	
⑪		オフライン状態へ	
⑫	‘PUT /test/folder/res10.txt’ をメールにて受信、ローカルキャッシュへ反映、ノードCへ接続		‘PUT /test/folder/res10.txt’ をメールにて受信、ローカルキャッシュへ反映、ノードAへ接続
⑬	ノードBから接続	ノードA、Cへ接続	ノードBから接続

表 7.4: オフライン同期問題の解決の確認

番号	確認事項	確認
1	全てのノードがオンライン状態である①のタイミングにノードAで発生した更新がノードBおよびノードCに反映される	○
2	③のタイミングにノードBで発生した更新が④のタイミングの時点でノードAがオンラインになったときにノードAに反映される	○
3	⑥のタイミングにノードCで発生した更新が⑧のタイミングでノードBに反映される	○
4	⑩のタイミングにノードBで発生した更新が⑫のタイミングでノードAおよびノードCに反映される	○

第8章　関連研究

本章では、本研究の関連研究を示す。

8.1　Mikan

SMTP/POP3 を利用したファイル共有システム Mikan(みんなで かん たんファイル共有)[24] は、慶應義塾大学大学院で開発されたファイル共有システムである。

Mikan はその名前の由来にも現れているように、短期的な共同作業を想定し、ファイル共有の場を設けるコストが低いことを優先した設計を行っており、SMTP/POP を利用して、数名のグループでのファイル共有の利用の場面で手軽に利用できることが評価されている。

Mikan も本研究と同様にメールをノード間の通信に利用しているが、Mikan が全ての通信をメール経由で実現しようとしているのに対し、本研究では、基本的には直接の通信でメッセージをやりとりし、対象のノードがオフラインの場合のみ、メールを利用するという点で、アプローチの違いが見られる。

8.2　qwikWeb

qwikWeb[25] は、Wiki エンジンと QuickML の機能を組み合わせて実現されたシステムである。qwikWeb では、メールの投稿によって Wiki のページが追加される。このシステムの興味深いところは、ML 上での議論の過程が自動的にウェブページに蓄積され閲覧可能なかたちにまとめられていくということである。また、従来の Wiki が基本的に誰でも編集できる公開型のサービスであったが、この qwikWeb は、QuickML のメーリングリストに登録されたメンバーのみがページを編集可能になるという、アクセス制限の仕組みを持っている。

本研究で実現した協調作業支援環境が、P2P 技術とメールシステムの組み合わせによって実現されているのと似ていることは偶然ではないかもしれない。2 つの別の技術やツールを組み合わせることによって、新しい使い方や機能が実現されている点が興味深い。

第9章　おわりに

本章では、本研究のまとめを行い、今後の課題を示す。

9.1　本研究のまとめ

本研究では、アドホックグループを対象とした協調作業支援環境の要件をあきらかにし、その要件を満たすために必要な問題であるオフライン同期問題およびブートストラップ問題について定義し、その問題を解決するために一般的なサービス資源による擬似ノードを用いた協調作業支援環境のモデルを示し、設計・実装した。

9.2　今後の課題

9.2.1　デプロイメント

本研究で実装した collaboware が、多くのグループで実際に利用され、その利用から得られたフィードバックをアプリケーションおよびフレームワークに反映し、洗練させ、さらに多くのユーザに利用してもらうというスパイラルプロセスを経ることで、本研究の目的であるアドホックグループで手軽に利用できる協調作業支援環境の実現を、より高度に達成できると考える。

9.2.2　リソース保持の高効率化

現状のモデルでは、ネットワークアーカイブ上の全てのリソースをローカルキャッシュに保持する。また、取得した全てのバージョンについても、ローカルキャッシュ内に保持する。しかし、実際の利用の中では、ネットワークアーカイブ内のリソースのうち、ユーザが必要なリソースは一部に限られることが多い。ユーザの利用状況等にあわせて、ノードがキャッシュするリソースを部分化することで、リソース保持の高効率化を行うことが可能であり、今後の課題として取り組んでいきたい。

9.2.3 耐故障性の向上

協調作業支援環境の基盤として、ネットワークアーカイブの耐故障性を向上させることは必要不可欠である。今回の実装では、オーバレイネットワーク上の何れかのノードに予期しない事態が発生して、メッセージの送受信が適切に行われなかった場合や、オーバレイネットワーク上でのリンク、蓄積中継リンクを構成する IP ネットワーク上の問題やメールシステムにおける問題があった場合に、それを正しく認識することができない。これを正しく認識し、適切な対処を行う等の仕組みを組み入れることで耐故障性の向上が図られ、より実用的な協調作業支援フレームワークを実現できると考えられる。

謝辞

本研究を進めるにあたり、ご指導を頂きました慶應義塾大学環境情報学部教授 村井純博士、徳田英幸博士、同学部助教授、同学部助教授 楠本博之博士、中村修博士、同学部専任講師 南政樹氏に感謝致します。

絶えずご指導とご助言を頂きました慶應義塾大学大学院政策・メディア研究科後期博士課程 須子義彦氏に感謝致します。neco KG の仲山昌宏氏、鈴木貴晶氏には日々の研究活動の中で様々な形のご支援を頂きました。感謝致します。

研究を進める中で、常に的確なご助言をして頂いた慶應義塾大学大学院政策・メディア研究科後期博士課程 斉藤賢爾氏に感謝致します。

最後に、ここでこのような素晴らしい人達と出会い、研究を通して自らを成長させる機会を与えて頂いた、両親に対する感謝を持って謝辞を締めさせて頂きます。

参考文献

[1] 平成 16 年版 情報通信白書, 総務省, 2004

[2] 松下 温, 岡田 謙一 : コラボレーションとコミュニケーション, 共立出版, 1995

[3] Jessica Lipnack, Jeffrey Stamps, 榎本英剛訳 : バーチャル・チーム ― ネットワーク時代のチームワークとリーダーシップ, ダイヤモンド社, 1998

[4] ダニエル・ピンク, フリーエージェント社会の到来 - 「雇われない生き方」は何を変えるか, ダイヤモンド社, 2002.

[5] fml project, http://www.fml.org/

[6] Majordomo, http://www.greatcircle.com/majordomo/Welcome.html

[7] Yahoo!グループ, http://groups.yahoo.co.jp

[8] QuickML, http://www.quickml.com/

[9] cvshome.org, https://www.cvshome.org/

[10] subversion.tigris.org, http://subversion.tigris.org/

[11] IBM Lotus, http://www.lotus.com/

[12] サイボウズ株式会社, http://www.cybozu.co.jp/

[13] 藤田一広, ASP におけるコンピュータセキュリティ, pp. 64–78, UNISYS TECHNOLOGY REVIEW 第 72 号, 2002.

[14] Intranets.co.jp, http://www.intranets.co.jp

[15] Groove Virtual Office, http://www.groove.net/

[16] アリエル･エアワン, http://www.ariel-networks.com/product/airone/summary.html

[17] Jabber, http://www.jabber.org/

[18] Media Art Online, http://www.media-art-online.org/

[19] wija, http://www.media-art-online.org/wija/

[20] Gmail, https://gmail.google.com/

[21] 伊藤直樹, P2P コンピューティング 技術解説とアプリケーション , ソフト・リサーチ・センター, 2001.

[22] Eclipse.org Main Page, http://www.eclipse.org

[23] OGA's Web Page, http://hp.vector.co.jp/authors/VA032928/index.html

[24] 西村祐貴, SMTP を利用したファイル共有システムに関する研究, 2003.

[25] qwikWeb - qwik.jp, http://qwik.jp/qwikWeb.html