修 士 論 文

有線ネットワークによる
すれ違い通信の性能改善の研究

北陸先端科学技術大学院大学
情報科学研究科情報科学専攻

八木 辰弥

2015 年 3 月

修 士 論 文

有線ネットワークによる
すれ違い通信の性能改善の研究

指導教員　篠田陽一 教授

審査委員主査　篠田陽一 教授
審査委員　知念 賢一特任准教授
審査委員　丹康雄 教授

北陸先端科学技術大学院大学
情報科学研究科情報科学専攻

1310071 八木 辰弥

提出年月: 2015 年 2 月

概 要

無線通信技術がモバイルノード (以下、ノードと呼ぶ) に実装された事により、特別なインフラを必要とせずノード間が直接情報を交換する「すれ違い通信」が注目されている。すれ違い通信は、ヒューマンモビリティを活用する事で通信が成立する。また、通信の確立時のみノード間の通信経路を形成するため、通信リソース消費は非常に省電力である。このため、災害時のような通信リソースが制限されるような環境においても、活用できる事が期待されている。

すれ違い通信が成立するためには、ヒューマンモビリティ、特定の空間内に存在するノードの密度などが関係する。そのため、通信を行いたいノード間の距離が離れている時には通信自体が成立しない場合がある。また、ノードが密集しやすい地域・場所などにより、ノード密度に差異が存在する。これは、ヒトの居住する地域によってすれ違い通信の発生頻度が極端に変動する事を意味する。すれ違い通信を利用するユーザは、自身が要求する情報と合致する他ユーザとすれ違い通信を行いたいと望んでいる。すれ違い通信を実際に導入しているノードを利用するユーザの要求からこのような事が述べられる。

本研究では、このようなヒトの居住する地域・場所に依存する事なくすれ違い通信がおこなわれるようにするために、すれ違い通信の通信範囲を広域化させる手法として Pocket Warped Network(PWN) を提案する。PWN では、すれ違い通信に対応するノード以外にトンネル・ポイントという機構を設置する。

トンネル・ポイントは、配置された位置の近辺に存在するノードの発見、トンネル・ポイント間でのトンネルの作成、ノードの送信する無線フレームのトンネルを通したリレー機能を持つ。これにより、すれ違い通信が困難であるような場所に存在するノードに対しては、トンネル・ポイントが提供するトンネルを介してすれ違い通信を行えるようにする。

PWN は Kurage,Ikagent,Tako から構成される。Kurage は、トンネル・ポイントがトンネルを作成するにあたり他のトンネル・ポイントの存在を知るための仕組みである。Kurage は既存のネットワーク上に仮想的な通信経路を形成するオーバーレイネットワークを提供する。次に Ikagent はトンネル・ポイントとして動作する。Ikagent はトンネルの作成、トンネルの作成先として他の Ikagent の選択、Tako の情報を収集する機能を持つ。これらの機能を持つ Ikagent の実装を行った。最後に、Tako はノードとして動作する。PWN の実現のために、Tako は特別な機能の追加・変更を行わない。

トンネル・ポイントが存在しない環境と存在する環境をシミュレーションし、すれ違い通信が発生する確率がどの程度向上するか実験を行った。その結果、トンネル・ポイントの台数が増加していくにつれノードが密集する場所が一つの場合においては指数関数的に、ノードの密集が複数に分散されている場合は、クラスタがオーバラップしている時は指数関数的に、そうでない時は対数関数的にすれ違い通信を行える確率が向上する事が分かった。

PWN では、Ikagent がトンネルを作成するにあたり、自身の配下に存在する Tako の情報を利用する。この情報と合致する Tako が存在する他の Ikagent とトンネルを自動的に作成する手法としてトンネル・ポイント選択アルゴリズムを提案する。そのため、Tako は近辺に存在する Ikagent へ自身の情報を送信する。トンネル・ポイント選択アルゴリズムを導入する事により、単純にすれ違い通信の発生頻度が向上するだけでなく、単位時間におけるすれ違い通信による情報交換の品質向上が期待できる。

Ikagent 選択アルゴリズムとして Random_App, Exact_Match, Common_App を考案した。これら三つのアルゴリズムでは Tako が保持するアプリケーション情報を利用する。Random_App は Tako が保持するアプリケーションの中からランダムに一つだけを取得する。そして、他の Ikagent の配下に存在する Tako が同じアプリケーションを保持しているかを調べる。Exact_Match は Tako が保持するアプリケーション全てと一致する Tako が他に存在するかを調べる。Common_App は、Ikagent の配下に存在する全ての Tako のアプリケーション情報を調べ、アプリケーションの集合を作成する。Random_App,Exact_Match は同じアプリケーションを保持する Tako 情報からトンネル作成先としての Ikagent を決定する。Common_App は、作成したアプリケーションの集合と共通する数が多い Ikagent をトンネル作成先として決定する。

各アルゴリズムが正しく動作するかを確認するために、Kurage,Ikagent,Tako を構築した実験環境の下でそれぞれを実行した。その結果、全てのアルゴリズムに対して、決められた取得方法に基づいた結果を返す Tako,Ikagent の情報だけを取得できた事を確認した。これにより、PWN が正しく実現できたと言える。

本研究の成果により、通信自体が困難であった場所に存在するノードともすれ違い通信が行う事が可能となる。また、従来のすれ違い通信では困難であったユーザの要求から通信相手を決定する事が可能となる。これにより、ユーザに対してより有意義なすれ違い通信を提供できる。

目 次

図 目 次

表 目 次

第1章　はじめに

本章では、本研究の背景、目的、本論文の構成を述べる。

1.1　背景

インターネットにおいてユーザへネットワークサービスを提供するためには、通信インフラの構築が一般的である。インターネットにおける通信インフラとは、ユーザが居住する地域まで設置される光ファイバ・海底ケーブル等の固定的な通信回線、また携帯・スマートフォン等の無線端末へ通信を提供するための基地局・無線アンテナ等の固定施設が挙げられる。

このような固定的な場所に設置・構築された通信インフラは地震・洪水等の災害発生時には欠損・断絶する恐れがあり、ユーザへのサービス提供が困難となる場合がある。また災害の規模によっては、通信インフラの復旧が長期化する恐れもある。そのため、通信インフラを提供する企業は、災害に強い通信インフラの構築を目指し、研究・開発が促進されている。2011 年 3 月 11 日に発生した東日本大震災以降、その目的はより強い意向を示している。[1]

そのような災害時にも機能する通信インフラを研究・開発している最中、近年すれ違い通信が注目されている。すれ違い通信は、通信のために特別な通信インフラを必要とせず、無線通信に対応するノード同士で直接情報交換する事が可能な通信である。すれ違い通信はヒューマンモビリティを活用して通信が成立する。また、通信の成立時のみノード間の通信経路が作成され、経路の維持・更新のための計算・処理を行わないため、リソース消費は非常に省電力である。このため、災害時における緊急通信と言った通信リソースが制限されるような環境においても活用できる事が期待されている。

しかし、すれ違い通信による情報の発生にはヒューマンモビリティ、特定の空間に存在するノードの密度等が影響し、ユーザが居住する地域や場所に依存して通信の発生頻度が大きく変動する。このため、すれ違い通信における適切な情報伝搬の手法を提案する事が困難という問題がある。

1.2 目的

本研究では、ヒューマンモビリティ、ノード密度に依存する事なく、すれ違い通信が行えるように、すれ違い通信の通信範囲を広域化する手法を提案する。本研究ではその提案を Pocket Warped Network(PWN) と名付けた。PWN では、すれ違い通信に対応するノード以外に「トンネル・ポイント」と言う新たな機構を設置する。

トンネル・ポイントの実態は、ノードが送信する無線フレームを転送する事が可能なトンネルを作成できる機構である。そのため、トンネルが作成されたトンネル・ポイント間であれば、すれ違い通信を行う事が可能となる。この機構を設置する事で、すれ違い通信が困難となる範囲に存在するノードに対しても、すれ違い通信をおこなえる。

しかし、トンネル・ポイントを導入する事で、すれ違い通信の発生頻度は飛躍的に増大する。このため、単純にトンネル・ポイントを設置しただけでは、個々のノードに対して適切な情報伝搬を行う事は困難となる可能性が高い。

そこで、トンネル・ポイントが自動的にトンネルを作成できる手法が必要となる。これを「トンネル・ポイント選択アルゴリズム」と名付けた。トンネル・ポイント選択アルゴリズムは、トンネル・ポイントが発見したノードの情報と、他のトンネル・ポイントが発見したノードの情報と調べ、他のトンネルの作成先を選択する事である。このため、個々のノードに適した通信相手を選択する事が可能となる。これにより、すれ違い通信の発生頻度が向上するだけでなく、単位時間における情報伝搬の品質向上が期待できる。

1.3 本論文の構成

2 章では、すれ違い通信のより詳細的な説明、および本研究のすれ違い通信のアプローチについて述べる。3 章では、本研究の提案手法 Pocket Warped Network について述べる。4 章では Pocket Warped Network に基づくシステムの設計について述べる。5 章では Pocket Warped Network による動作実験を述べる。6 章では 5 章の実験の考察について述べる。7 章では本研究の今後の課題と展望について述べる。

第2章　すれ違い通信と本研究における性能改善のアプローチ

本章ではすれ違い通信とは何か、その詳細的な説明を固定的なインフラ通信の特徴を述べ、固定的なインフラ通信と比較してすれ違い通信にはどのようなメリットが存在するのかを述べる。また、すれ違い通信の応用例を述べ、実際にすれ違い通信を利用するユーザがすれ違い通信に対して期待する要求を述べる。

そして、本研究の目的であるすれ違い通信の性能改善をどのような観点で着目してアプローチしているのかという事を、すれ違い通信を利用するユーザの要求、要求からのすれ違い通信の問題点、既存研究のすれ違い通信の問題点の解決手法、通信の利用目的による分類をまとめた上で述べる。

2.1　既存の通信インフラ

インターネットを利用するユーザにネットワークサービスを提供するためには、通信インフラの構築・設置が一般的である。インターネット上でデータを送受信するために必要な機器や設備を全て通信インフラと呼ぶ。

このような通信インフラは、UTP や海底ケーブルなどの有線、また携帯やスマートフォンなどの端末に無線電波を提供する基地局といった、物理的に固定された位置に設置・構築されている。そのため、津波・震災などの災害によって通信インフラが損壊した場合は、多くのユーザにネットワークサービスを提供できなくなる場合がある。また、物理的な損壊を回避できた通信インフラであっても同様に、停電によって通信インフラへの電力を供給できず、ネットワークサービスを提供できなくなる場合がある。

2011 年 3 月 11 日に発生した東日本大震災では、固定電話網の交換局、携帯電話網の基地局、中継伝送路などの通信インフラが損壊し、また長期停電によって通信インフラへの電源供給が停止した。この結果、多くのユーザがネットワークを利用できなくなったという事態に陥った [2]。

2.2　すれ違い通信

節 2.1 では、一般的なネットワークでサービスを提供するために、主流となっている固定的なインフラについて、その特徴を述べた。本節では、すれ違い通信について、その特徴を固定的なインフラ通信と比較して述べる。

すれ違い通信とは、ヒューマンモビリティを活用する事で、無線通信に対応するノード同士が直接情報交換を行う通信機能の事である。図 2.2 はすれ違い通信の動作を表した図である。

図 2.2 の一番目の図より、A と B は共にすれ違い通信の機能を持つノードを示す。また、ノードを囲む丸枠の線は、各ノードが放出する無線電波の到達範囲を示す。前述の通り、すれ違い通信ではヒューマンモビリティを活用する。この図では、二つのノードが黒い矢印線の方向に向かって直線に移動する事を仮定する。

図 2.2 の二番目の図では、ノード A と B が移動をおこないお互いの無線電波の到達範囲に存在している事を示している。すれ違い通信では、互いのノードが無線電波の到達範囲に存在した場合に、通信が確立しノード同士が直接情報交換を行う。

情報交換を完了したノード A と B はヒューマンモビリティに基づいて再び移動を行う。この動作はノードが無線通信を中断するまで繰り返される。このようにすれ違い通信は、通信の確立のために特別な通信インフラを必要としない。そのため、災害時の緊急通信といった、通信インフラの故障・ダウン等の通信リソースが制限されるような環境においても活用できる事が期待されている。

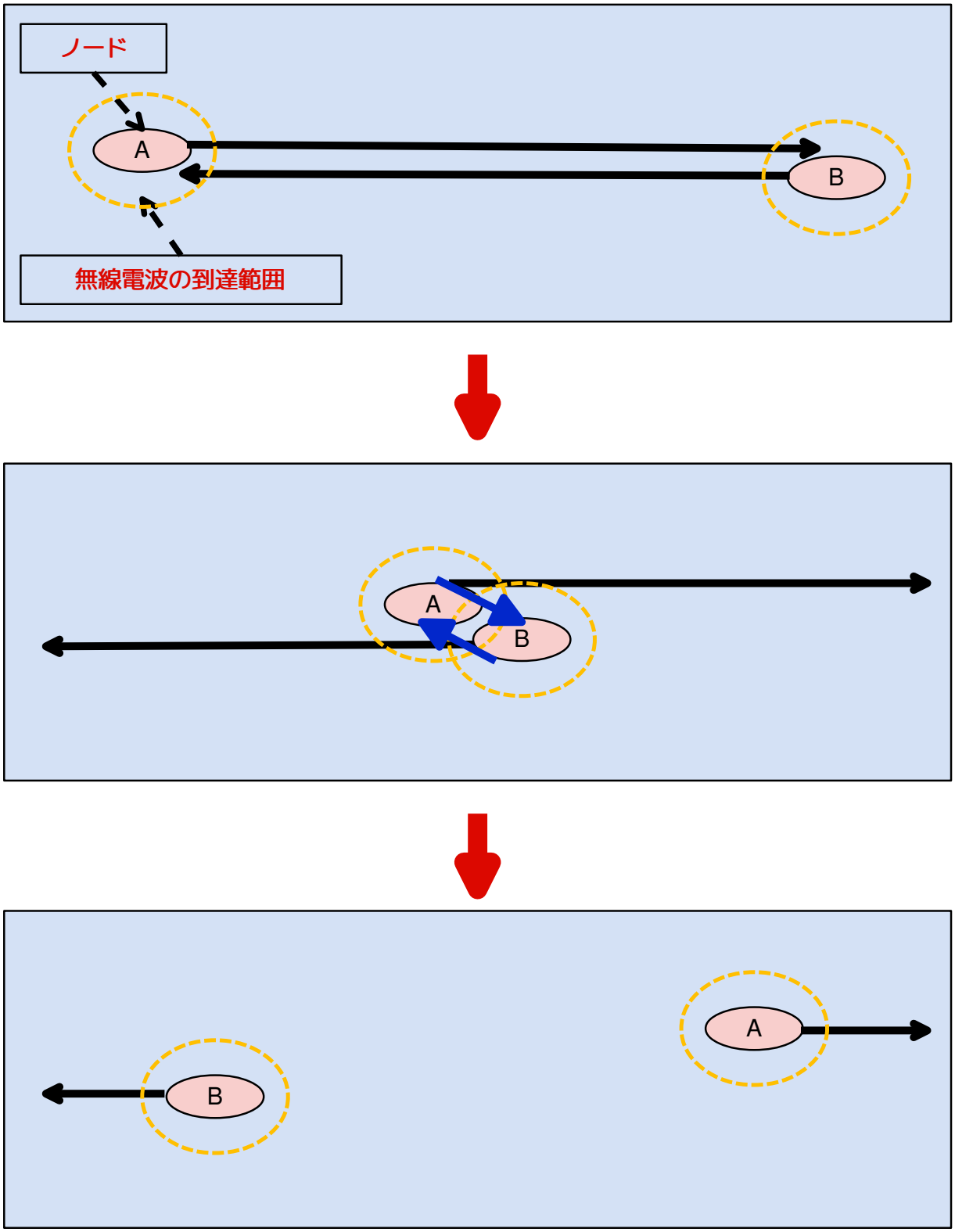

図 2.1: すれ違い通信の動作

2.2.1 すれ違い通信の応用例

節 2.2 では、すれ違い通信の動作について詳細的に述べた。この項では、実際にすれ違い通信を利用するにあたっての応用例を述べる。すれ違い通信の応用例として主に、以下の二つが挙げられる。

1. ユニキャストによるノードへの情報の伝搬

   すれ違い通信ではユニキャストによって、送信元ノードから宛先ノードまで情報を伝搬する事が可能である。その具体例として Pocket Switched Network(PSN) と呼ばれるものがある。

   PSN は固定的なインフラを必要とせずに、無線通信に対応するノードで構成されるネットワークである。また PSN で対象とされるノードの特徴として、人間の手によって持ち運びが容易であり、無線電波によって他の PSN に対応しているノードと通信が可能な端末とされている。

   PSN では情報の転送のためにヒューマンモビリティを採用している。そのため、PSN におけるルーティングの目的は有線ネットワークやモバイルアドホックネットワーク (以下、MANET と呼ぶ) のルーティングの目的とは異なる。有線ネットワークでは通信経路が常に存在し、ルーティングの目的はネットワークサービスの品質を満たす最善経路を見つける事である。また、MANET は無線で接続可能なノード (PC、スマートフォン等) をアクセスポイントが仲介する事なく、相互に接続する形態をとっており限定された領域内で構築可能なネットワークである。MANET ではノードが頻繁に移動をおこなっており、通信経路の作成・削除が繰り返される。しかし、通信経路が絶えず変化する中でも各ノードは経路表の更新・維持を行う。そのため、指定された宛先までの情報の送信の際には、現在利用可能な通信経路の中から最善な経路を見つけ送信を行う。したがって、MANET におけるルーティングの目的は有線ネットワークと同様である [3]。

   PSN では、長時間通信経路が存在しないと仮定されている。そのため、PSN におけるルーティングの目的は、宛先までの最善な経路をみつける事ではなく情報の伝搬率を最大にする事である。その目的達成のために PSN では、送信された情報を受信したノードは情報に変更を加えずそのまま自身のノードに保存する事が可能である。これにより、多数のノードに同じ情報を転送する事が可能である。したがって、PSN では情報の配信率を最大にするためにはどうすればいいのかという前提の元で、最適な手法を提案するためのルーティングアルゴリズムの研究・開発が促進されている [4]。

   PSN のようなノードの情報を別ノードへ転送できる機能を持つアプリケーションの開発をおこなえば、すれ違い通信でも PSN と同様の動作を行う事が可能となる。

2. 複数のノードへの情報の伝搬

すれ違い通信に対応するノードが複数存在する場合、個々のノードに対し同じ内容の情報を伝搬する事が可能である。上記の応用例と同じく PSN もこの動作に対応可能であるが、ここでは、この利用目的で通信を行う具体例を述べる。

その具体例としてゲーム機器間の通信に利用されている。ゲーム機器本体にすれ違い通信に対応するアプリケーションを登録する事で、同一のアプリケーションを登録している他のゲーム機器とすれ違い通信を行う事が可能となり、アプリケーション内の情報が交換される。また、アプリケーションの中にはすれ違い通信の発生頻度による特典が付加されるものも存在する。

このように、すれ違い通信はユーザのゲームエンタテイメントを向上させるための楽しみ方の一つとしても利用されている [5]。

2.3　本研究の性能改善のアプローチ

節 2.2 ではすれ違い通信についてその特徴を固定的なインフラ通信と比較して述べた。本節では、すれ違い通信をどのような観点から分析・着目して性能改善を行うのか、そのアプローチを述べる。

2.3.1　すれ違い通信を利用するユーザの要求分析

本節ではすれ違い通信を利用するユーザの視点から、すれ違い通信に対する要求を整理する。以下はすれ違い通信を利用している環境で要求されるものである。

1. 通信の発生頻度の向上
   単純に移動を行う間、通信の発生頻度が多くなると、その分情報交換が行われユーザの保持する情報の品質が向上する。

2. 意図しない移動中における情報交換
   すれ違い通信を利用するユーザは、情報交換だけを目的として移動を行う可能性は低いと考えられる。そのため、通勤・通学等のユーザの都合による移動の最中でも、情報交換がおこなわれるとすれ違い通信によるサービスの品質を保つ事ができる。

3. ユーザの嗜好に近いノードと情報交換
   すれ違い通信における情報には、ノードに登録されているアプリケーションからユーザの嗜好を時折、表している。個々のユーザの嗜好に近い、または合致するような情報を保持するノードと通信を行う事は、すれ違い通信のサービスの品質をより向上させる。

2.3.2 ユーザの要求からのすれ違い通信の問題点

項 2.3.1 ではすれ違い通信におけるユーザの要求を述べた。本項では、ユーザの要求を満たすために、考慮しなければならないすれ違い通信の問題点を述べる。図 2.3.2 はユーザの要求からのすれ違い通信の問題点を表したものである。赤い丸枠で囲まれているものが問題点を示す。また、赤い丸枠に割り振られている番号は以下の問題に対応する。

1. ノードの密度による通信頻度の格差

   すれ違い通信によって、[1] ユーザクラスタ (以下、クラスタと呼ぶ。) が展開される。クラスタ内に存在するノードの密度が大きければクラスタ内におけるすれ違い通信の発生頻度も高くなる。その反面、ノードの密度が小さければすれ違い通信の発生頻度が低くなる。これは、人間が居住する地域・環境等によってすれ違い通信の発生頻度に格差が発生する問題を引き起こす。

   例えば、東京や大阪等といった都心では人間が密集しやすい環境・施設が比較的多数存在し、すれ違い通信の発生頻度は大きくなる。その反面、人口が少ない地方や人間が密集しやすい環境・施設が存在しない場所で展開されるクラスタではすれ違い通信の発生頻度が小さくなる可能性がある。

2. クラスタ間の伝搬

   上記 1 番目の問題点に対して、クラスタを超えた通信を行う事で解決する事が可能である。しかし、すれ違い通信に対応するノードの移動はヒューマンモビリティに依存している。そのため、クラスタ間の物理的な距離が大きく離れている場合は通信に時間がかかる場合がある。あるいは、通信の確立自体おこなわれない可能性も存在する。

[1] ここで述べるユーザクラスタとは、一つの物理的な空間内である程度の移動をおこなえば、容易にすれ違い通信をおこなえるユーザの集合体の事を指す。図 2.3.2 では、図中の空間内にクラスタ U とクラスタ S が存在していると仮定している。

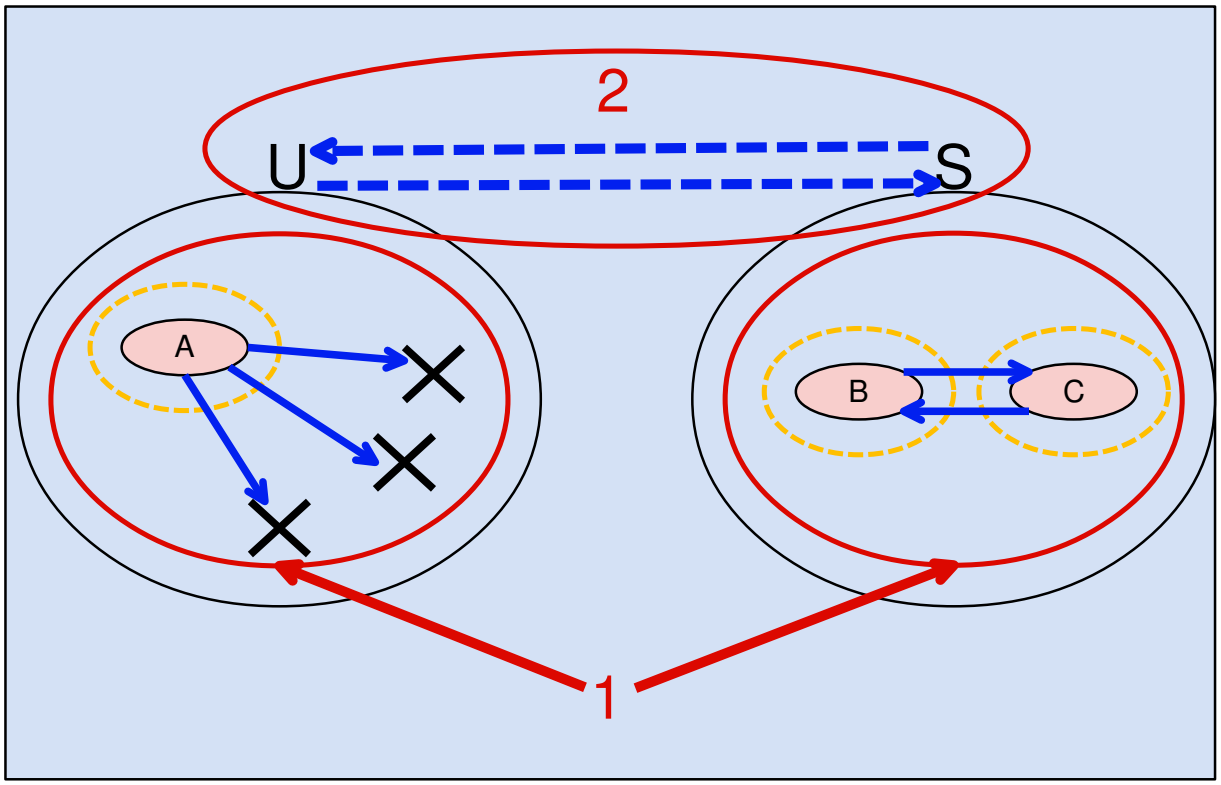

図 2.2: すれ違い通信の問題点

2.3.3 関連研究による解決手法

節 2.3.2 では、ユーザの要求を実現するために考慮しなければならない問題点を述べた。本節では、関連研究である PSN における問題点を解決する手法として PSN のルーティングアルゴリズムがどのようなアプローチから発案されているかを述べる。

PSN もすれ違い通信と同様にノードの移動手段としてヒューマンモビリティを採用している。そのため、PSN にルーティングアルゴリズムを考慮する際にはすれ違い通信と同様の問題点を考慮しなければならない。したがって、伝搬率を最大にする事を目的とする PSN のルーティングのアプローチを纏める事はすれ違い通信の性能を改善するという目的に関連する。以下は、PSN のルーティングアルゴリズムを分類した三つのアプローチである。

1. Encounter-Based Approach

   Per-Contact-Based Approach とも呼ばれる。適切な転送先を決定するために個々の PSN ノードに、遭遇したノード数、遭遇時間といった他ノードと遭遇した際の履歴情報を利用する。Encounter-Based Approach のルーティングアルゴリズムの例として、transit-contact approach[6] や Jones らによって提案された代表的な Encounter-Based Approach がある [7]。

2. Social-Based Approach

   クラスタをつくって生活しようとする人間の持つ基本的な傾向である社会性を PSN ルーティングの性能を向上させるために採用したものである。例えば、クラスタ内で人気の高い人間は他の者よりも多くの人間と対話を行うと考えられ、その人間が保持するノードを中心に、情報を伝搬すれば一つのクラスタ内での情報の伝搬やクラスタを超えた通信に転送ができると考えられている。このような sSocial-Based Approach によるルーティングアルゴリズムの例として、BUBBLE[8] や SimBet[9] 等がある。

3. Location-Based Approach

   情報の転送先を決定するために、ヒューマンモビリティの空間的な特性を利用する。例えば、同じような移動パターンを持つ二人の人間は出逢う可能性が高い事を前提とし、同じような移動パターンを持つ人間が保持するノードに対し情報伝搬を行う。また、人間が頻繁に訪れる場所 (家や会社など) を保存する事で、宛先の場所を訪れる可能性が高いノードに対して情報の伝搬を行う事で、伝搬率を高めるといった手法などが Location-Based Approach に含まれる。Location-Based Approach のルーティングアルゴリズムの例として HOME[10] や MobySpace[11] などがあたる。

表 2.1: 通信の利用目的による分類

	一般的な通信		すれ違い通信	
	HTTP	FTP	本研究の提案システム	PSN
pull	○ http get	○ ftp get	○ 要求する情報を 保持するノードと すれ違い通信	×
push	○ http post	○ ftp put	×	○ unicast, multicast

2.3.4 通信の利用目的による分類

ここで、すれ違い通信の性能を改善するという目的にあたって、本研究で提案するシステムがどこに着目したかを明確にするために、インターネットの通信における利用目的の用途を述べる。

表 2.1 は、汎用的な PC で利用されている「一般的な通信」と「すれ違い通信」の場合において、利用目的においての通信の対応をまとめた表である。通信の利用目的は以下の二つに分類する。

1. pull

   通信相手が保持する情報をある程度理解しており、相手の情報を取得するための通信

2. push

   情報を伝搬・拡散させるための通信

ファイル転送のプロトコルである FTP などでの一般的な通信では、表 2.1 より、pull,push どちらの利用目的においても対応する機能が存在している。そのため、目的によっての使い分けが可能であると言える。

その反面、すれ違い通信の場合関連研究である PSN のルーティングアルゴリズムは、項 2.3.3 より、情報の伝搬・拡散させる点から発案されているため、push に対応していると言える。しかし、ノードが要求する情報を保持する相手とのすれ違い通信は考慮されておらず pull には対応していないと言える。

しかし、項 2.3.2 より、すれ違い通信を利用するユーザの要求から、ユーザが要求する情報を取得する事が可能な pull の利用目的も必要であると考えられる。

そこで、本研究においては pull の通信に着目し、すれ違い通信においても pull の利用目的に対応するシステムの実現を可能とした。

第3章　Pocket Warped Network(PWN)の提案

本章では、項2.3.2で述べたクラスタによるすれ違い通信の差異、またクラスタを超える伝搬が困難という問題点を解決するための手法としてPocket Warped Network(PWN)を提案する。

3.1　PWNの概要

PWNは、ノードの通信範囲を仮想的に広域化させる手法である。そのために、PWNではすれ違い通信に対応するノードの他に、トンネル・ポイントと呼ぶ機構を追加する。

トンネル・ポイントはクラスタ間を結ぶ仮想回線(以下、トンネルと呼ぶ)を作成する機能を持つ。トンネル・ポイントは、ノードが伝搬する情報をトンネル・ポイントに記憶する機能は持たない。その代わりに、トンネル・ポイント間を接続し作成されたトンネル上で、ノードの情報を転送する。これにより、本来クラスタを超える通信に関してモビリティ依存であったものに対して、トンネルを通る事による転送を可能とする。

図3.1は、PWNの概要を表した図である。クラスタUにはノードA、クラスタSにはノードB,Cが存在する。またUにはトンネル・ポイントW1、Sにはトンネル・ポイントW2が配置されており、W1とW2間で一つのトンネルが既に作成されているものとする。

この図では、クラスタUに存在するノードAが、クラスタSに存在するノードB,Cとすれ違い通信を行う場合の動作を述べる。まずAは自身の通信範囲へ情報を伝搬する。この時、Aの通信範囲内にはトンネル・ポイントW1が存在する。W1はノードAが伝搬した情報を検知し、Aの情報をそのままトンネルを通りW2へと転送を行う。これによりクラスタUに存在するAの情報がクラスタSへと到達する。そして、転送先に存在するトンネル・ポイントW2はAの情報をノードB,Cに転送する。これにより、クラスタの間で物理的な距離が離れている場合でもトンネル・ポイント間にトンネルが作成されている時にはクラスタを超えたすれ違い通信が可能となる。

ノードAからはトンネル間のすれ違い通信においても、ノードB,CがAの通信範囲内に存在するかのような挙動を見せる。この挙動から提案手法をPocket Warped Networkと名付けた。PWNは、ルーティングの観点から見た場合はトンネルが作成されたトンネル・ポイント間のみ通信経路が維持されるため、網羅性はない。また、トンネル・ポイントにはノードの情報をそのまま記憶(ストア)する機能は存在しないため、ノードの観点

からはリアルタイム性がない。その反面、トンネル・ポイントが記憶する要素はルーティング、ストアの機能を持つ事と比べて少なくなるため、トンネル・ポイントに要求される性能が厳しくならないというメリットが考えられる。

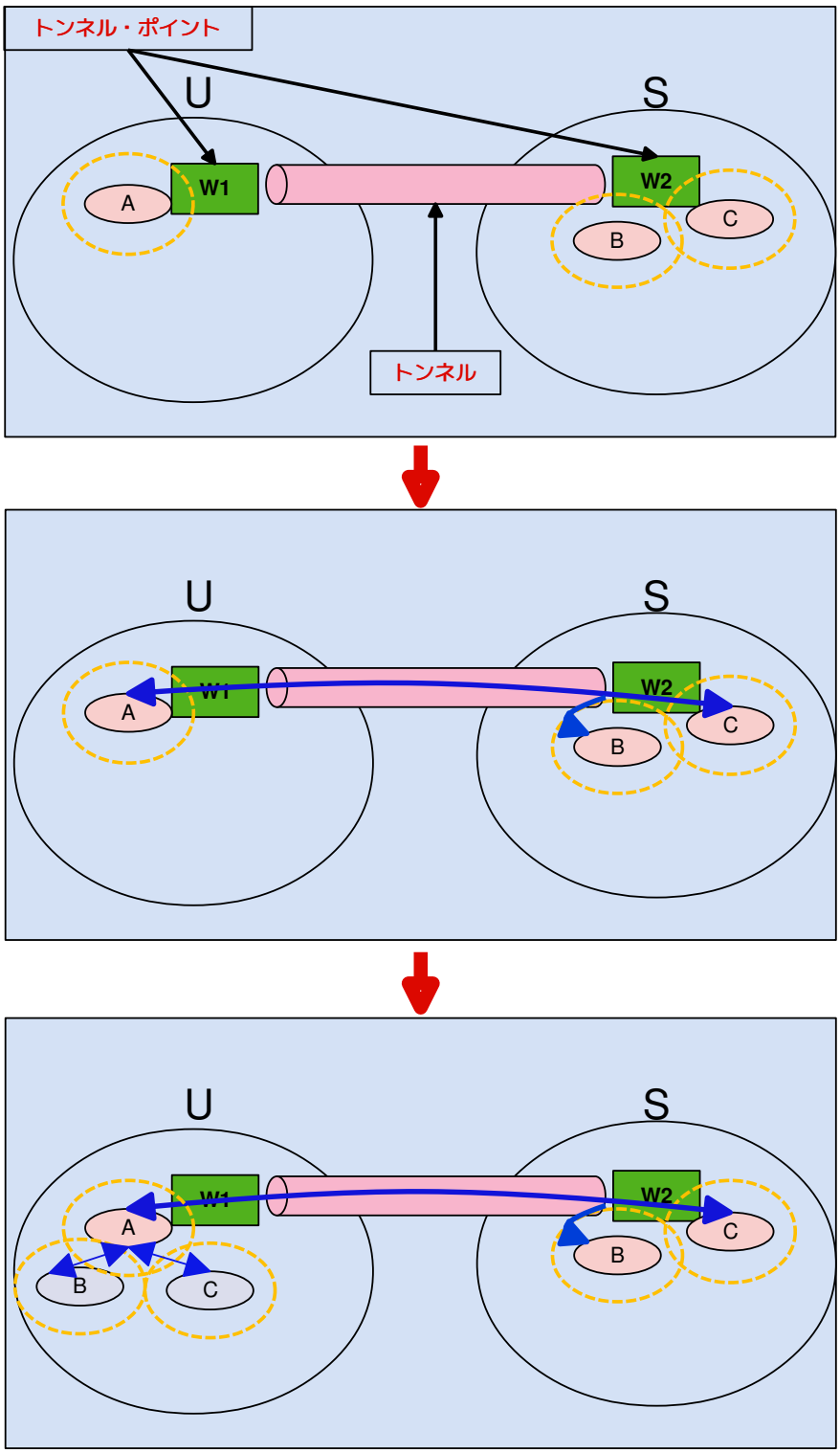

図 3.1: PWN の概要

3.2 従来のすれ違い通信とPWNによるすれ違い通信の発生頻度の比較実験

この節では、PWN の提案を導入する事ですれ違い通信の発生頻度が従来のすれ違い通信と比較してどの程度まで向上するのかを述べる。そのためのシミュレーションを実装し、トンネル・ポイントがない場合、ある場合によってすれ違い通信の発生頻度を調べる比較実験をおこなった。

3.2.1 比較実験におけるシミュレーション環境

本実験では、以下四つの環境を想定する。

1. ノードが一つのクラスタに配置されている場合 (トンネル・ポイントなし)

   多数のすれ違い通信に対応するノードに対して、指定した領域内からランダムの位置を決定し、配置を行う。各ノードには、固定されたすれ違い通信の範囲が決定されている。したがって、すれ違い通信は二つのノードがお互いの通信範囲に存在した場合に発生する。

   ノードの通信範囲はノードの配置された位置を中心とする半径 30m と想定する。この 30m という通信範囲は既存のすれ違い通信に対応するノードとして任天堂の製品である Nintendo 3DS®における推奨距離である。[12]

   また、ノードが配置される領域は以下のようになる。

$$\text{ノードが配置される領域}\,(k^2m) = 20000m * 20000m = 400k^2m \tag{3.1}$$

   これは日本の面積と、Nintendo 3DS の国内累計販売台数から Nintendo 3DS の密度を計算し、それぞれの数値を千分比し計算した面積である。

   日本の面積は 377,960km^2,Nintendo 3DS の販売台数は約 15,037,116 台販売された (2014 年 1 月までの累計)[13]。以上より、面積は

$$\frac{15,037.116}{377.900} = 39.7912... \approx 40 \tag{3.2}$$

   となり、1km^2 あたり、約 40 のノードが存在する事になる。配置される最大ノード数を 16,000 台とし、この台数が 400km^2 内に配置される。よって、以下の三つの環境もノードの配置領域に関しては同様の範囲で行う。尚、図 3.2.1 にガウス分布によって配置されたノードの配置図を掲載する。また、ガウス分布の平均値、標準偏差は表 3.1 に従う。

表 3.1: 1,2 番の環境におけるクラスタの概要

		ガウス分布の平均値		ガウス分布の標準偏差	
クラスタ番号	ノード数	x	y	x	y
1	16000	0.0	0.0	2500.00	2500.00

2. ノードが一つのクラスタに配置されている場合 (トンネル・ポイントあり) 1 番の条件の他に、特定の領域にトンネル・ポイントを配置する。トンネル・ポイントの配置領域はノードと同様に、400km^2 以内となる。また、トンネル・ポイントがノードを発見するために利用する無線の通信範囲は、ノードと同様に半径 30m と仮定する。また、トンネル・ポイントを配置するにあたって、非階層型クラスタリングアルゴリズムである k-means++法によって配置される。k-means++法に関しては次節のトンネル・ポイントの配置方法で述べる。

3. ノードが複数のクラスタに配置されている場合 (トンネル・ポイントなし)

   最大配置領域である 400km^2 の範囲内で分布配置されるクラスタを複数作成するように変更を加える。これにより、擬似的にノードが居住する地域・密集する場所内での限定された中ですれ違い通信が行われるような環境の構成が可能となる。本実験では、表 3.2 のクラスタに従って、ノード 16,000 台を配置している。図 3.2.1 が各クラスタからガウス分布によって配置された状態となる。

4. ノードが特定の場所に密集して配置されている場合 (トンネル・ポイントあり)

   3 番の条件の他に、トンネル・ポイントを配置する。トンネル・ポイントの配置アルゴリズムは 2 番と同様に k-means++法を採用している。また、ノードの配置に関しては表 3.2、図 3.2.1 と同様である。

3.2.2 トンネル・ポイントの配置方法

比較実験では、トンネル・ポイントを自動的に配置するように行う。これにより、ノードが密集するような場所・拠点を作成する事が可能となる。特に表 3.2 より、明示的にクラスタを構成しノードの配置を決定した場合において、付加価値を付ける事が可能となる。

比較実験において、トンネル・ポイントの配置方法として、非階層型クラスタリングのアルゴリズムである k-means++法を採用した。k-means++法は、同じく非階層型クラスタリングのアルゴリズムである k-means 法の改良方法である。従って、k-means++法を述べる前に k-means 法を述べる。

- k-means 法

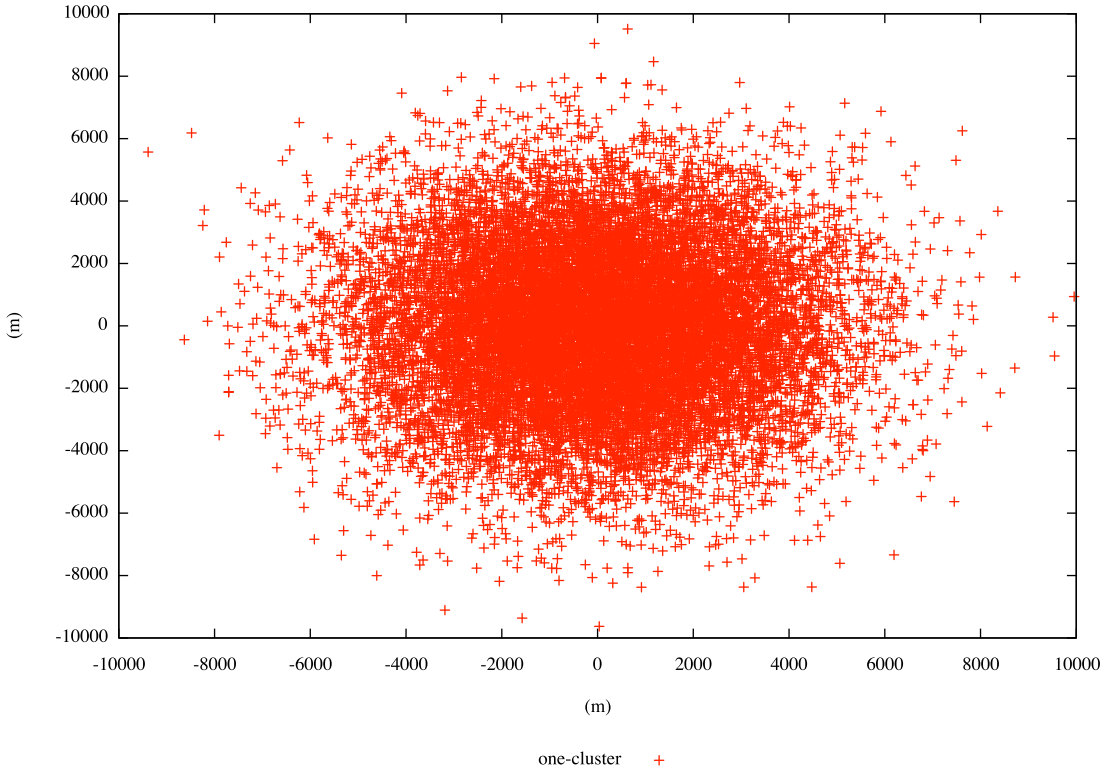

図 3.2: ノードの配置図 (クラスター一つ)

表 3.2: 3,4 番の環境における各クラスタの概要

		ガウス分布の平均値		ガウス分布の標準偏差	
クラスタ番号	ノード数	x	y	x	y
1	637	5430.0	5430.0	1142.50	1142.50
2	1791	0.0	-1800.0	181.25	181.25
3	490	-600.0	-4000.0	308.75	308.75
4	1152	-800.0	-2500.0	193.75	193.75
5	3000	-1800.0	-2800.0	250.00	5000.00
6	149	-3000.0	-1052.0	256.25	256.25
7	966	-3000.0	-3225.0	282.50	282.50
8	6000	-5000.0	-3500.0	500.00	500.00
9	1052	-4745.0	-5095.0	173.00	173.00
10	602	-6200.0	-5000.0	275.00	275.00
11	161	-8890.0	-9190.0	193.75	193.75

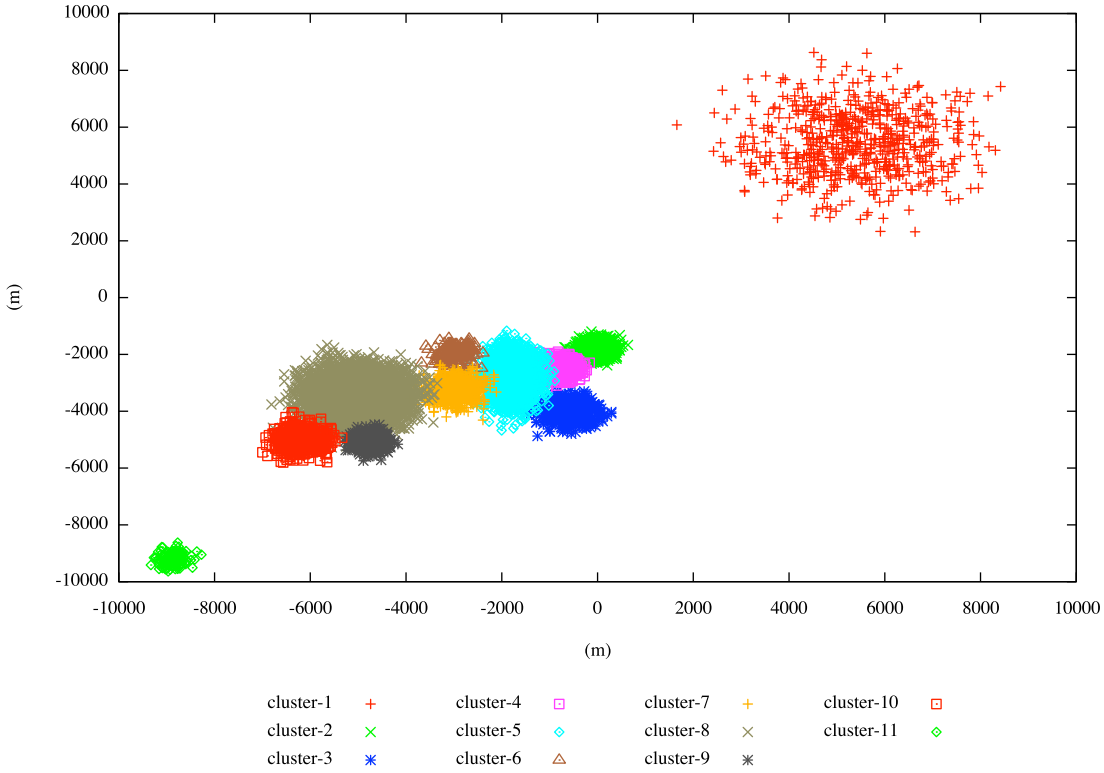

図 3.3: ノードの配置図 (複数クラスタ)

k-means 法は、式 3.3 の評価関数 ϕ を最小化するようにクラスタ中心点を見つける事により、データ X を k 個のクラスタに分割する。

$$\phi = \sum_{x_i \in X} \min_{c_j \in C} ||x_i - c_j||^2 \tag{3.3}$$

$x_i \in \{1, ..., n\}$ は各データとなり、$X = \{1, ..., n\}$,n はデータの総数となる。$c_j \in \{1, ..., k\}$ は各クラスタの中心点となり、$C = \{1, ...k\}$,k はクラスタの総数となる。このように、k-means 法は各データ点から最短距離となるクラスタの中心点との距離の総和が最小となるようなクラスタ中心点を計算する事でクラスタリングを行う。k-means 法のアルゴリズムを以下に示す。

1. 任意の k 個のクラスタ中心点 $c_j \in 1, ..., k$ をランダムに選択し初期値を決定する。
2. 各データ点 $x_i \in 1, ...n$ を最も近いクラスタ c_j に割り当てる。
3. 各クラスタ中心点 $c_j \in 1, ..., k$ を式 3.4 に従ってクラスタ中心点を求める。

$$c_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i \tag{3.4}$$

   ここで、C_j は、各クラスタ j に含まれるデータの集合を示し、$|C_j|$ は、クラスタ C_j に含まれるデータの総数を示す。

4. クラスタに変化がなくなるまで、step 2,3 を繰り返す。

k-means 法は、クラスタの平均を利用して、k 個に分割するため他のクラスタリングアルゴリズムと比較して、計算量が少なく済むという利点がある。その反面、ランダムに決定される初期値によって、最終的な結果が依存するため明示的には特定のクラスタに含まれないはずのデータまでもクラスタに含まれてしまう問題が発生する。[14] [15]

- k-means++法

k-menas++法は k-means 法における初期値設定方法を改良したものである。k-means++では、全データに対して $D(x_j)$,$x_j \in \{1, ..., n\}$ を求める。Dx_j は、データ x_j と既に決定されたクラスタ中心点との最短距離を示す。k-means++法のアルゴリズムを以下に示す。

a. 一つ目のクラスタ中心点 c_1 をデータ X からランダムに選択する。
b. 新たなクラスタ中心点 c_j を $x_i \in Dx_i$ から確率 $\frac{D(x_i)^2}{\sum_{x \in X} D(x)^2}$ に基づいて選ぶ。
c. クラスタ中心点を k 個選ぶまで b. を繰り返す。
d. k 個選択後、k-means 法アルゴリズムの 2.-4. と同様の処理を行う。

k-means++法はクラスタ中心点との最短距離が大きいデータが次のクラスタ中心点に選ばれやすくする。また、確率的に選ぶ事で常に最大のものが選ばれるわけではない。このため、外れ値の問題にも対応可能である。[14] [15]

3.2.3 比較実験の評価軸

本節では比較実験において、何を指標にして評価するのかを述べる。比較実験に関してはすれ違い通信を行う際、モビリティの考慮をしない。そのため、ノードは配置された初期位置から移動を行わない。従って、すれ違い通信は現在の位置から半径 30m の範囲内に存在する他のノード、トンネル・ポイントが存在する場合はすれ違い通信が発生する。

通信範囲にノードが存在する場合は、そのノードとすれ違い通信が行える。また、トンネル・ポイントの場合は、全てのトンネル・ポイントが一つの仮想空間に所属していると仮定している。そのため、トンネル・ポイントの通信範囲内にノードが存在する場合、全てのトンネル・ポイント上で検出した全ノードとすれ違い通信が行える。この時、重複するノードまた自身のノードに関してはすれ違い通信の発生回数には含まない。

ノード、トンネル・ポイントのどちらにせよ、評価軸に関しては、全ノードに対してすれ違い通信が発生する確率を表す。その確率は、

$$\text{確率}\,(P) = \frac{N}{{}_n\mathrm{C}_2} \tag{3.5}$$

で表され、N = 通信可能な 2 ノードの数、${}_n\mathrm{C}_2$ = 全体の 2 ノードの組み合わせとなる。また、トンネル・ポイントのみによって発生するすれ違い通信と従来のすれ違い通信との発生回数を比較するために、トンネル・ポイントが存在する場合の確率 (P) には、ノード間の直接的なすれ違い通信の発生は加算しない。

3.2.4 比較実験の結果

最初に、一つのクラスタ配置、複数のクラスタ配置において従来のすれ違い通信による確率結果を以下の表で述べる。表 3.3 より、配置位置におけるすれ違い通信の確率は双方

表 3.3: ノード配置依存によるすれ違い通信の発生確率

配置の種類	配置ノード数	確率 (%)
クラスタ (一つ) 配置	16,000	0.0186 %
クラスタ (複数) 配置	16,000	0.1791 %

共通して極端に低い事が分かる。一つのクラスタ配置は $20\mathrm{km}^2$ * $20\mathrm{km}^2$ 内に一様に分布されるため、拡散した多数のノードとすれ違い通信を行うのは非常に困難である。

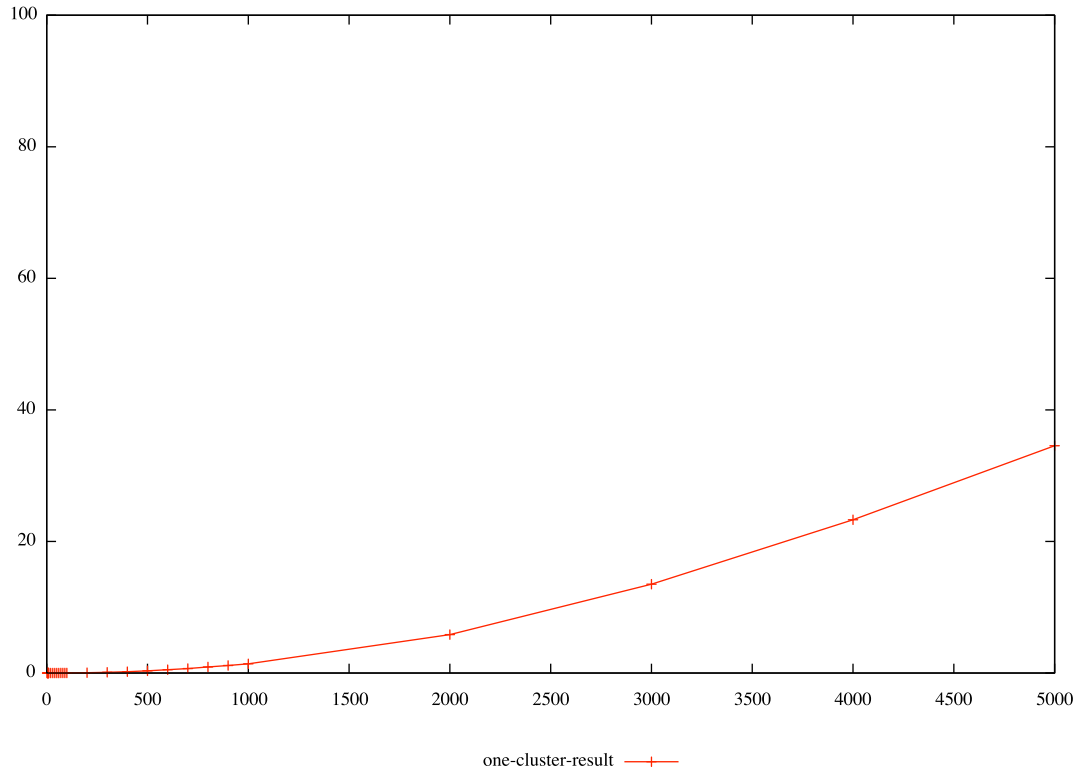

図 3.4: トンネル・ポイント設置後のすれ違い通信の発生確率 (一つのクラスタ)

対して、複数のクラスタ配置は各クラスタ毎にノードが密集されているため、一つのクラスタ配置と比較するとすれ違い通信の発生する確率は約 10 倍までに向上するが、それでも別クラスタに存在するノードとはすれ違い通信が行えないため全体の確率としては低い。

各配置手法に対して、トンネル・ポイントを k-means++によって配置位置を決定した際のすれ違い通信の確率は以下のようになった。表 3.2.4 は一つのクラスタにおいてノードを配置した時の、表 3.2.4 は複数のクラスタにおいてノードを配置した時において、全体的なすれ違い通信の発生確率の変化をグラフで表したものである。結果からクラスタが一つの場合、1000 台までは単調に増加していく傾向があり、トンネル・ポイントが 1000 台間隔で増加していくと全体のすれ違い通信の発生確率が指数関数的に増加している事が分かった。また、クラスタが複数の場合では、トンネル・ポイントが 1000 台以降増加するにあたって、指数関数的に増加している事がわかる。しかし、4000 台と 5000 台では発生確率の増加が指数関数的にならず、対数関数的に増加していく傾向が見られる。

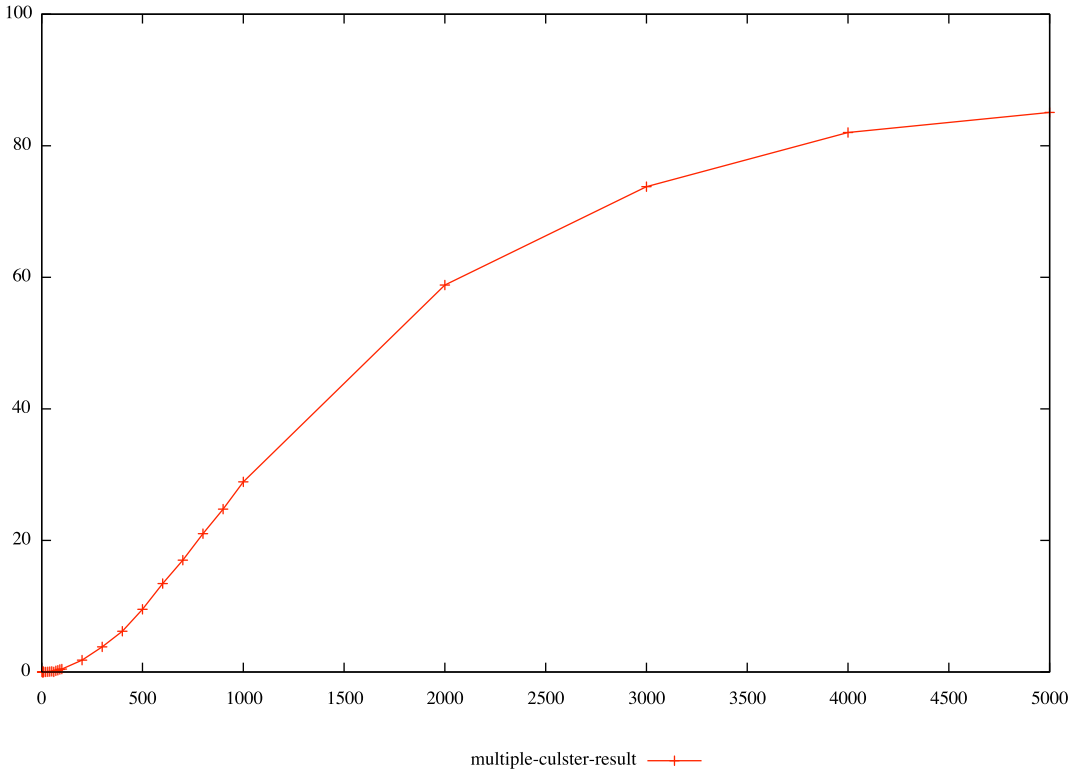

図 3.5: トンネル・ポイント設置後のすれ違い通信の発生確率 (複数クラスタ)

3.2.5 比較実験からの考察

比較実験から、ノードが密集するクラスタの数によってトンネル・ポイントを配置する事によって、次の事が言えると考えられる。まず 1,2 番の実験環境のようなクラスタ一つの場合によるノードの配置では、トンネル・ポイントの数が増加していくにつれ、すれ違い通信が行える確率は急激に増加する事が期待される。今回の実験においては、ノードが 1000 単位で増加していくにつれ指数関数的にすれ違い通信の発生確率も向上したため、クラスタ数が一つにおいてのトンネル・ポイントの配置によるすれ違い通信の恩恵を飛躍的にノードが受ける事ができると考えられる。1,2 番のような環境の例は大学内のキャンパス、テーマパーク、娯楽施設と言った今回の実験よりも比較的狭い領域、少ないノード数においても一つの場所・拠点に密集するような環境が想定される。

また、3,4 番の実験環境のようなクラスタ数が複数の場合によるノードの配置では、各クラスタの範囲内においてノードの密度が比較的高い場所にトンネル・ポイントを置くだけで、すれ違い通信の発生確率が飛躍的向上する事が見られた。そのため、ノードの密度が高い場所に配置する事で、1,2 番の実験環境とは比較的少ないトンネル・ポイントの数によって、すれ違い通信の発生確率はより向上できる事が言える。

その反面、4000 台以降では飛躍的に増加する傾向が見られなかった。3,4 番の環境では 9 個のクラスタをオーバーラップさせ、2 個のクラスタは独立している。このため、オーバーラップするクラスタであれば、トンネル・ポイントの配置によって、通信の発生確率が飛躍的に向上するが、独立したクラスタ間ではトンネル・ポイントの数を大量に配置しない限り、トンネル・ポイントによるすれ違い通信の恩恵を受けられない事が言える。

このような独立したクラスタ間に存在するノードと通信を行うためには、ノードの密度が高い場所に配置する以外に、高密度な場所から離れた位置に存在する複数、あるいはたった一つのノードを対象として、その近傍にトンネル・ポイントを配置すると言った事を行わなければならない。しかし、トンネル・ポイントの利点としてトンネル・ポイントの機能を持たせるためのマシンには節 3 で述べた通り、それほど厳しい性能を要求されない。そのため、PWN を利用したいユーザがトンネル・ポイントを自身で用意する事で、このような問題は解決できると考えられる。

3.3 トンネル・ポイント選択アルゴリズム

比較実験の結果からトンネル・ポイントを利用する事で、各ノードすれ違い通信を行える確率は飛躍的に向上する事が分かった。しかし、単純にすれ違い通信を行える確率が増加するだけでは、節 2.3.1 で述べた中で、特にユーザの嗜好に合ったすれ違い通信と言うユーザの要求を満たすのは困難である。本来すれ違い通信は、範囲内に存在するノードから通信相手が選択される。そのため、トンネル・ポイント上で発見されたノードの中には、ユーザの要求を満たす事が可能な情報を保持するノードが多数存在する可能性が高い。従って、ユーザ視点からはこのようなノードと優先的にすれ違い通信を行いたいが、

単純に情報をトンネル・ポイント間で転送するだけでは、優先してすれ違い通信を行うのは困難である。

そのようなノードと優先してすれ違い通信を行えるようにするために、本研究ではトンネル・ポイント間でトンネルを作成するにあたって、ノードが保持する情報を利用して、多数存在するトンネル・ポイントの中から自動的にトンネルを作成するための選択候補を決定するための機構を提案する。これをトンネル・ポイント選択アルゴリズムと名付けた。トンネル・ポイント選択アルゴリズムの詳細を次節に述べる。

3.4 トンネル・ポイント選択アルゴリズムの一例

本節ではトンネル・ポイント選択アルゴリズムを実装するにあたって、どのような観点から考慮すればいいのか、その一例を述べる。図 3.4 はクラスタ U に存在しているノード A がトンネルを介してすれ違い通信を行う事を仮定している図である。トンネル・ポイント W1 のトンネル作成の組み合わせとしてクラスタ U の W2、クラスタ T の W3 の二通りある。そこで、ノード A に対して W1 は W2 と W3 のどちらとトンネルを作成するのが適切かを考える。この W1 がトンネル先を決定するための考慮する要素を「トンネル・ポイント選択アルゴリズム」と呼ぶ。

この例でによるトンネル・ポイント選択の方法として、ノード A がすれ違い通信に登録しているアプリケーションの集合を調べ、他のクラスタに存在する各ノードのアプリケーション集合と比較し、共通しているものがあるか調べる。共通しているアプリケーションがものが一つでもあればトンネルの作成先として有効だとみなし、評価値をつける。これをクラスタ毎に調べ評価値が最大となるクラスタ先に存在するトンネル・ポイントをトンネルの作成先と判断する。

図 3.4 に存在する各ノードのアプリケーションの集合をまとめたものが表 3.4 となる。ノード A は p1,p2,p4 という三つのアプリケーションをすれ違い通信における情報配信に利用するために登録している。ここで、クラスタ S に存在するノードに関して、B は p3 を登録しておりノード A とすれ違い通信を行う必要性はないと考えられる。同じくノード C も p1 だけしか登録しておらず、ノード A からは一つのアプリケーションにした対応していない。対して、クラスタ T に存在するノード D は p1,p2,p3,p4 と A のアプリケーションを全て網羅している。同じく E に関しても p2,p4 を登録しているので二つのアプリケーションに対応している。以上より、クラスタ U の W1 はクラスタ S とトンネルを作成するよりも、クラスタ T との間でトンネルを作成する方が、ノード A に対して有効性が高いと考えられる。これにより、W1 は W2 よりも W3 を優先してトンネルを作成する。

このようにノードの情報を利用してトンネルの作成先を選択する事によって、ユーザが要求する情報を取得できる pull の通信に対応できると考えられる。本節で述べた例はあくまでトンネル・ポイント選択アルゴリズムの本の一部の例として述べたが、このような手法を導入する事により、単純にすれ違い通信の頻度を増加させるだけでなく、単位時間におけるすれ違い通信の品質向上が期待できる。

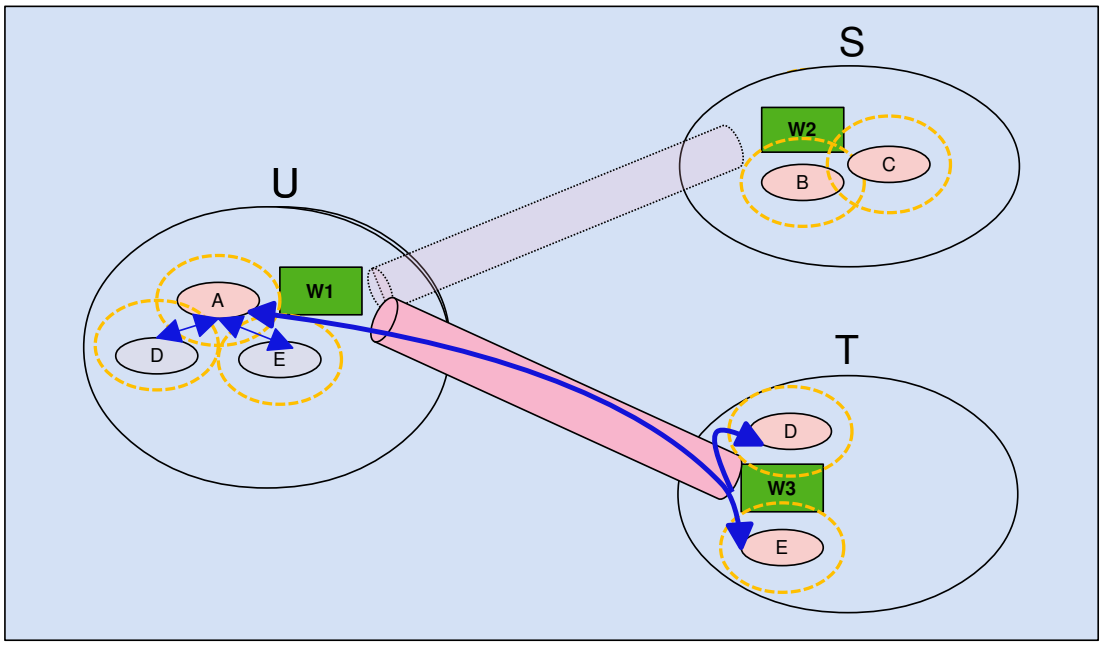

図 3.6: トンネル・ポイント選択アルゴリズムの一例

表 3.4: 各ノードが登録しているアプリケーション

ノード名	A	B	C	D	E
アプリケーション	{p1, p2, p4}	{p3}	{p1}	{p1, p2, p3 p4}	{p2, p4}

3.5　PWN で想定されるノード

通信範囲を広域化する手法、ユーザの要求に一致するような情報を取得する手法の双方トンネル・ポイントが行う。そのため、PWN ではすれ違い通信に対応するノードそのものに機能を追加、または変更を加えない。したがって PWN においてすれ違い通信の性能改善の対象となるノードは、ノード以外にすれ違い通信で情報を配信可能なアプリケーションが存在する事を想定する。

これにより、ノード上で起動するすれ違い通信に対応するアプリケーション間の情報交換の品質を向上させる事が期待できる。

第4章　提案手法に基づくシステムの設計

本章では、提案手法であるPWNに基づくシステムの設計を述べる。

4.1　全体構成

図4.1に、システムの概要を示す。また、以下にシステムにおける各機構の仕組みを述べる。

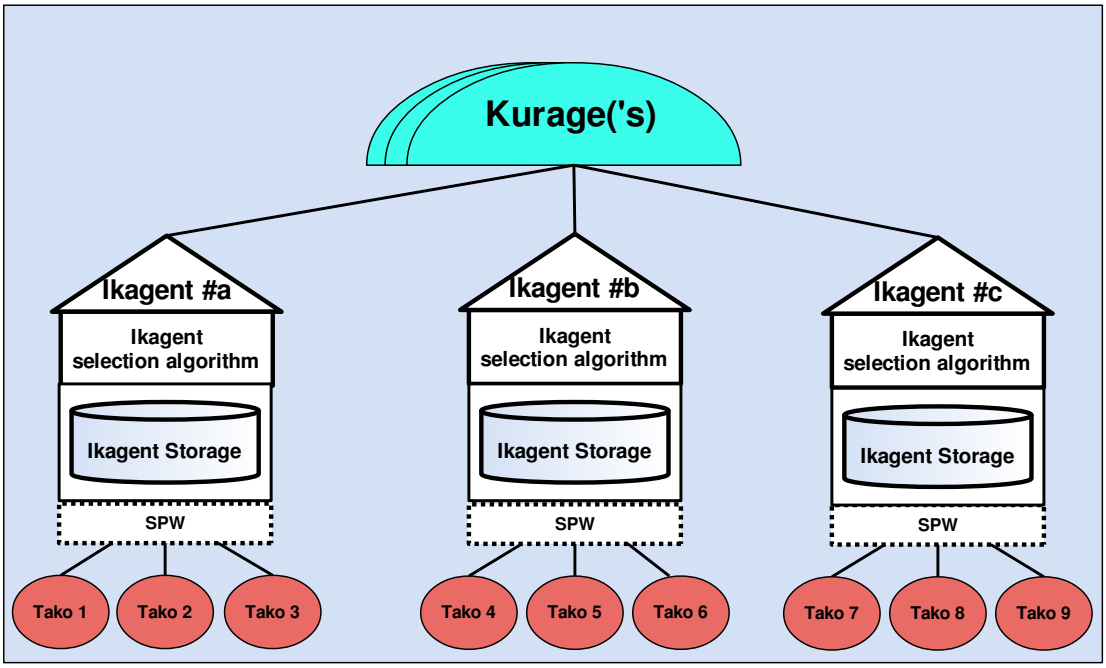

図 4.1: 全体構成

1. Kurage

   トンネル・ポイント間でトンネルを作成するためには、トンネルの作成要求を行うトンネル・ポイントと作成先のトンネル・ポイントのお互いを発見できる仕組みが必要となる。Kurage はその仕組みである。

   Kurage は、オーバーレイネットワークを提供する機能を持つ。オーバーレイネットワークは既存の通信経路上に利用目的に応じて仮想経路を作成し、その上でサービスを提供するネットワークとされる [16]。そのため、Kurage とトンネル・ポイント間で利用するソフトウェア、プロトコルを決定する事で、オーバーレイネットワーク上に存在するトンネル・ポイント間は同一の仮想空間内であれば、お互いを発見する事が可能となる。

   また、Kurage は他の Kurage と接続する事によって、Kurage 間でトンネル・ポイントの情報を共有する事が可能となる。つまり、Kurage はオーバーレイネットワークを提供するための機構群である。

   このオーバーレイネットワークを提供する機構として、本研究では Internet Relay Chat(IRC) を採用した [17] [18]。 そのため、Kurage は IRC サーバとなる。

2. Ikagent

   Ikagent はトンネル・ポイントとして動作する。そのため、Ikagent は Kurage に接続され、ノードの情報の転送、及び Ikagent 選択アルゴリズムの機能を持つ。また、

外部プログラムとしてSPWの機能を呼び出す。SPWはノードの情報を転送する機能と、Ikagentの通信範囲に存在するノードの情報を発見する機能を持つ。そのため、SPWによって発見されたノードの情報を記憶領域に格納しIkagent選択アルゴリズムを動作させる事によって、ノードの要求と一致する情報を持つIkagentとトンネルを作成する事が可能となる。本研究では、IkagentはIRCクライアントとなる[17] [18]。

3. Tako

   Takoはすれ違い通信に対応するノードである。Takoは自身の通信範囲内へ情報を伝搬し続けるというすれ違い通信の動作を繰り返す。そのため、Takoにはカーネルモジュールの追加・変更といったPWNの構成のために、特別な機能を追加しない。

4.2 Ikagent 内の記憶領域部分の設計

Ikagent の記憶領域は SPW によって発見された Tako 情報、及び同一空間内に存在する Ikagent の Tako 情報を格納する。Ikagent 内に存在する Tako 情報を格納するタイミングは、SPW が送信するメッセージに依存する。SPW のメッセージは以下の三種類である。

- NEW

  Ikagent の近辺で新しく発見された Tako が存在する場合、対象となった Tako の情報を通知する。Ikagent がこの情報を受け取った時、対象となる Tako の情報を追加する。

- UPD

  NEW によって発見された Tako の情報に変更があった場合に送信され、変更された情報を通知する。Ikagent がこの情報を受け取った時、対象となる Tako の情報を更新する。

- DEL

  Tako が Ikagent の近辺に存在しなくなった場合に、対象となった Tako の情報を通知する。Ikagent がこの情報を受け取った時、記憶領域から対象となる Tako の情報を削除する。

本研究では、Ikagent が SPW の通知によって Tako の情報の追加・更新・削除を行うためのソフトウェアとして、SQLite を採用した [19]。また、Ikagent 選択アルゴリズムでトンネルを作成するにあたり、他の Ikagent 内に存在する Tako の情報を指標とする。その手法を次節で述べる。

4.3 接続された Ikagent 間における Tako 情報取得部分の設計

Kurage に接続された Ikagent は他の Ikagent の配下に存在する Tako の情報を取得する事が可能となる。そのため、Ikagent 選択アルゴリズムを実行する Ikagent は、Tako の情報を取得した後に要求と一致する Tako が存在する Ikagent をトンネルの作成先として選択する。その Tako の情報を取得するための手法として二つの方法を提案した。

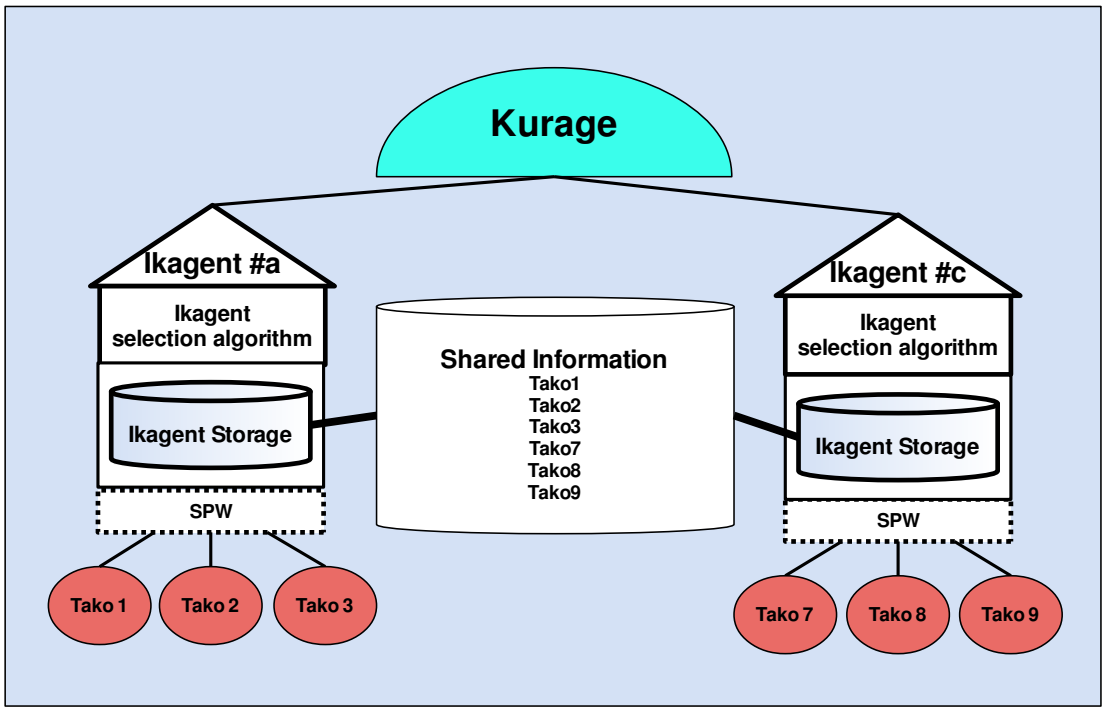

図 4.2: Status Share(SS) 型の概要

4.3.1 Status Share(SS) 型

Status Share(SS) 型は、同一の仮想空間内に存在する Ikagent 間で Tako の情報を共有する手法である。図 4.3.1 は SS 型の概要である。

図 4.3.1 では、Ikagent#a と Ikagent#c は同じ Kurage に接続し、同一の仮想空間に存在していると仮定する。その際、Ikagent#a は Tako1,2,3 の情報を、Ikagent#c は Tako7,8,9 の情報をお互いの SPW の NEW メッセージによって検出した場合にそれぞれの記憶領域にその情報が格納される。その時、NEW メッセージを Kurage を介する事により、Ikagent#a, または Ikagent#c に向けて Tako 情報が転送される。これにより、お互いの Ikagent は自身が実行する SPW で検出した Tako 情報以外にも他の Ikagent で検出された Tako 情報を格納する事が可能となる。

図 4.3.1 の場合、お互いの記憶領域に格納される Tako 情報は最大で Tako1,2,3,7,8,9 の 6 台が共有される。このように、自身の配下に存在する Tako 情報を他の Ikagent にも通知する事によって、Tako 情報を受信した Ikagent はトンネルの作成先を選択するための指標を保持する事が可能となる。

4.3.2 Distribution Query(DQ) 型

Distribution Query(DQ) 型は同一の仮想空間内に存在する Ikagent に向けて分散検索をかける手法である。

同一の仮想空間内に存在する Ikagent 内の Tako 情報を自身の Ikagent の記憶領域に格納した後に、Ikagent 選択アルゴリズムによってトンネルを作成先を選択する SS 型に対して、DQ 型は仮想空間内の Tako 情報を各 Ikagent は共有しない。その代わりに、トンネルの作成要求を行う Ikagent は Ikagent 選択アルゴリズムを実行し、トンネルの作成先を決めるための指標として自身の Tako 情報を利用する。Ikagent 選択アルゴリズムによって、選択された一つ (または、複数) の Tako 情報を纏めて、仮想空間内に Tako 情報を転送し、分散検索をかける。

Tako 情報を受け取った Ikagent は、その情報を処理し検索元の Tako 情報と一致するような Tako 情報を自身が持っているかどうかを調べる。検索元の Tako 情報と一致するような Tako 情報を持っている Ikagent のみが応答として一つ (または、複数) の Tako 情報を纏めて結果として返答メッセージを転送する。

図 4.3.2 は Distribution Query の概要を表したものである。

この図では、Ikagent 三台が同一の仮想空間に存在し、Ikagent#a が分散検索をかけた際の各 Ikagent の挙動を述べる。また、Ikagent 選択アルゴリズムの例としてこの図では、Ikagent に格納されている Tako 情報からトンネルの作成先を決めるための Tako を選択し、更に、選ばれた Tako 内のアプリケーションの情報から一つ、ランダムに選択する。これにより、検索メッセージとしては選択された Tako 情報と選択されたアプリケーションの情報一つを纏めたものとなる。

Ikagent#a の中に存在する Tako から Tako3 を選択し、更にその中からアプリケーション情報として p3 を選択する。従って、検索メッセージの中身には、少なくとも Tako3, の情報、p3 の情報が付加される。検索メッセージは Kurage に到達し、更に同一の仮想空間に存在する Ikagent へ向けて転送される。

検索メッセージを受け取った Ikagent#c,Ikagent#b は自身の配下に存在する Tako の情報を調べる処理を行う。この処理によって、自身の配下に検索メッセージに一致するような Tako 情報を持つ Ikagent のみが結果を返す。

図 4.3.2 では、Ikagent#b の配下に存在する Tako6 のアプリケーションに p3 が登録されている。そのため、Ikagent#b は結果として Tako6 の情報を応答メッセージに纏めて返答する。返答メッセージは Kurage に到達し、検索を送信した Ikagent#a に転送される。Ikagent#c も同様に検索メッセージの情報と一致する Tako がいるか調べる。しかし、Ikagent#c の配下には、p3 のアプリケーションを持っている Tako が存在しない。そのため、Ikagent#c は処理を終了し、結果を返す事はない。

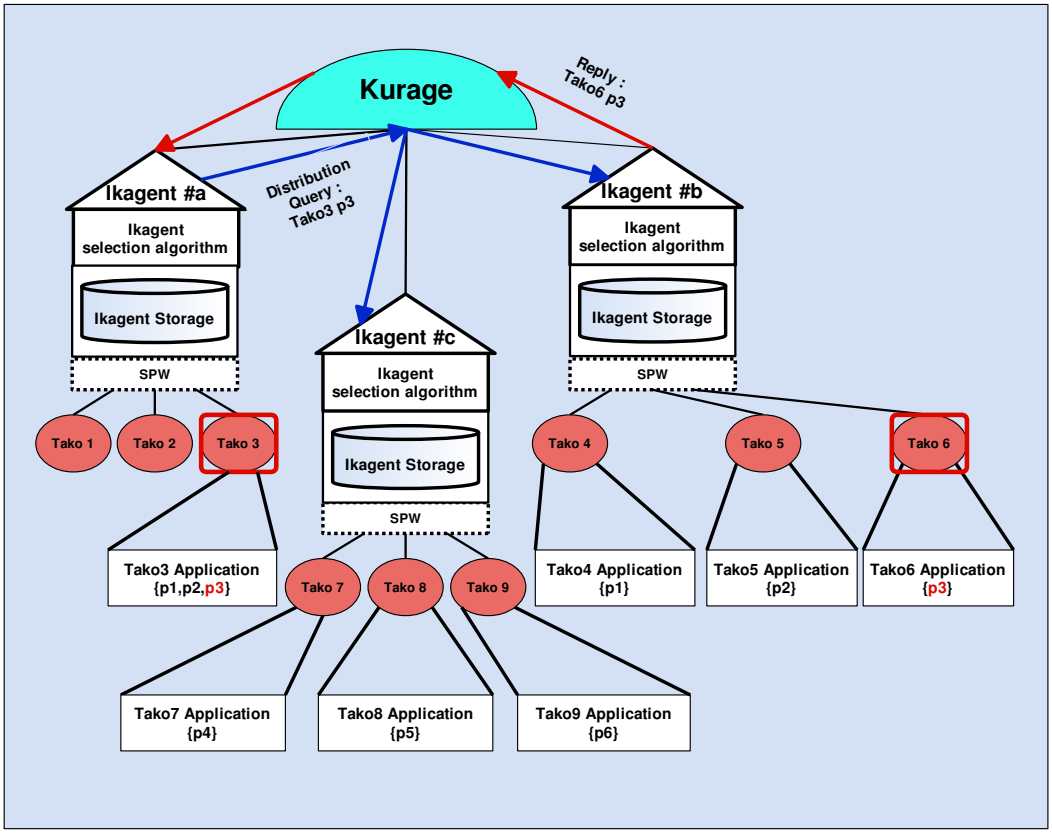

図 4.3: Distribution Query(DQ) 型の概要

4.3.3 SS 型, DQ 型の利用用途

4.3.1,4.3.2 より、同一の仮想空間に存在する Ikagent の Tako 情報を取得する 2 つの手法を述べた。この節では SS 型、DQ 型の利用用途を述べる。

SS 型は同一の仮想空間内に存在する Ikagent 間で全ての Tako 情報を取得し、共有する。そのため、トンネルの作成先としての Ikagent 候補は同一の仮想空間内に存在する全 Ikagent となる。全ての Tako 情報が一つの Ikagent の記憶領域に格納されるため、Tako の状態を Ikagent 間で知る事が可能となる。そのメリットとして、Ikagent 観点からはトンネルを作成する場合は、Ikagent 選択アルゴリズムを実行する Ikagent のみで処理を行うため、他の Ikagent にはトンネル作成に関する負荷がかからなくなる。

しかし、仮想空間内の Ikagent 数が多い場合、その分共有する Tako 情報の数も増える可能性がある。そのため、共有に関するネットワークトラフィックが増大する可能性がある。尚、仮想空間内の Ikagent 数が少なく、共有する Tako 情報の数が多い場合は、共有の際のメッセージ送信に対して共有元で輻輳制御を行う事により帯域幅の溢れをある程度まで抑える処理は可能である。しかし、Ikagent の数が多い場合には個々の Ikagent に対して輻輳制御を行ってしまうと全 Tako 情報が Ikagent 間で共有されるまでに莫大な時間がかかる恐れがある。そのため、実際に共有される Tako 情報が少なくとも、Ikagent 数が多い場合には SS 型は不向きである。

DQ 型は同一の仮想空間内に存在する Ikagent に対して分散検索をかける。そして、結果を返す Ikagent は検索メッセージからの情報と一致するような Tako 情報を持つ Ikagent のみとなる。そのため、結果として返す Ikagent のみがトンネルの作成先の候補として挙がるため、Tako の視点からは要求と一致するような Ikagent 先の Tako とすれ違い通信を行える可能性が高くなる。

しかし、仮想空間内の Ikagent 数が少ない場合、結果を返す Ikagent の数も少なくなる。あるいは全く返す事がない場合も考えられる。そのため、ある程度仮想空間内の Ikagent 数が多くないと DQ 型による恩恵を受けられない場合がある。

図 4.1 より SS 型、DQ 型それぞれの特徴を纏めた。これにより SS 型、DQ 型の利用するにあたっての、用途の判断となる。

4.4 Ikagent 選択アルゴリズムの設計

本節では、実際に設計を行った Ikagent 選択アルゴリズムとして三つの手法を述べる。

4.4.1 Random_App

Random_App は、SPW の NEW メッセージによって検出された Tako に登録されているアプリケーションの内、一つをランダムに選択しそのアプリケーションを持つ Tako が存在する Ikagent をトンネルの作成候補として選ぶアルゴリズムである。

表 4.1: SS 型、DQ 型のまとめ

	SS 型	DQ 型
概要	Ikagent 間で全ての Tako 情報を共有	分散検索をかけ結果だけを格納
メリット	Tako の状態が即座に適応される、 Ikagent 選択アルゴリズムを実行する Ikagent のみでトンネルを作成可能	要求に一致する Ikagent のみが トンネル作成先の候補となる
デメリット	Ikagent 数が多いとネットワークの 帯域幅を消費する可能性有	検索をかける Ikagent の数が少ないと DQ 型の恩恵が受けられない可能性有
利用用途	仮想空間の Ikagent 数が少ない時に有効	仮想空間 Ikagent の数が多い時に有効

図 4.4.1 は SS 型による Random_App の概要である。この図では、Ikagent#a に存在する Tako1 が NEW メッセージによって検出された後、Random_App の挙動を述べる。Tako1 のアプリケーションから p3 のアプリケーションが選択された。SS 型の場合、同一の仮想空間上に存在する Ikagent 内の Tako 全ての情報が共有されるため、Ikagent#a は自身の記憶領域から p3 のアプリケーションを保持している Tako を検索する。Ikagent#b には Tako2 が p3 のアプリケーションを保持している。そのため、Tako2 の情報と、Ikagent#b の情報が結果として返される。これにより、Ikagent#b の先に存在する Tako3 との直接的なすれ違い通信を行う事が可能となる。

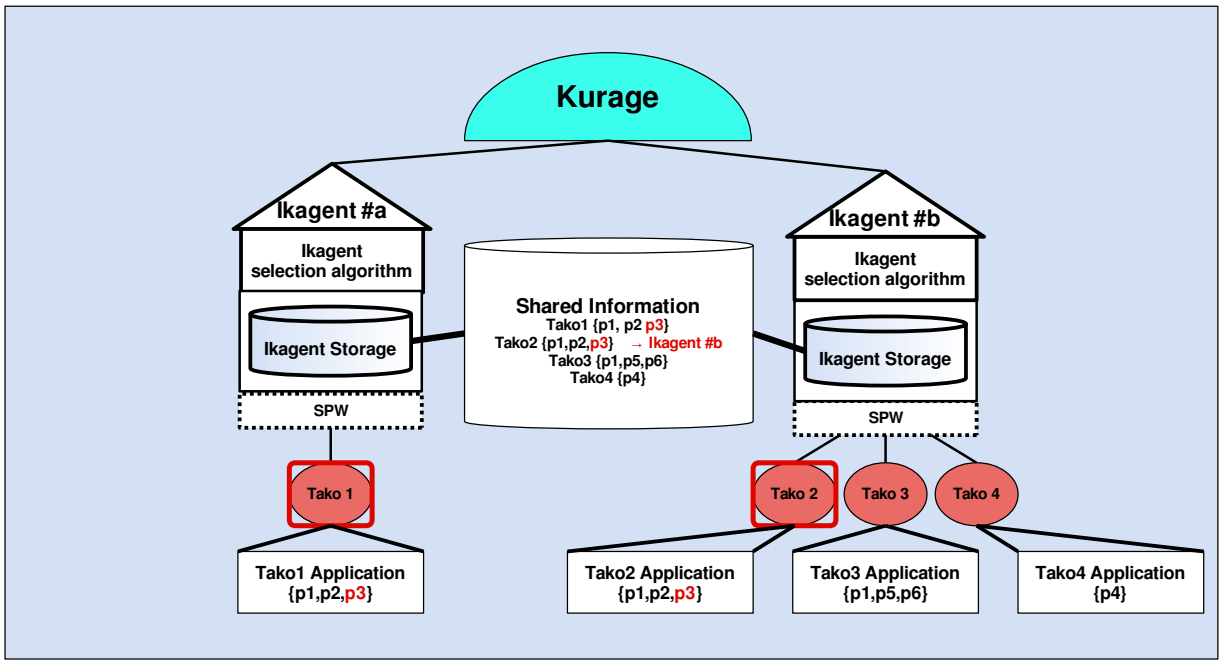

図 4.4: Random_App の概要 (SS 型)

図 4.4.1 は DQ 型による Random_App の概要である。図 4.4.1 と同様に Tako1 からランダムに選択された p3 のアプリケーションからトンネルの作成候補を選択する。検索メッセージに Tako1 の情報と選択されたアプリケーションの情報を付加する。この検索メッセージは Kurage に送信され、Ikagent#b に転送される。検索メッセージを受信した Ikagent#b は Random_App によって作成された検索メッセージと認識して処理を行う。そのため、DQ 型による検索メッセージの中身にはどのようなアルゴリズムを利用して作成されたかが分かるようにアルゴリズム名も付加されている。Ikagent#b は受信したメッセージの中のアプリケーションを調べ、自身の配下に p3 を保持する Tako が存在するか検索する。現在、Tako2 が p3 のアプリケーションを持っているため結果として応答メッセージに Tako2 の情報と Ikagent#b の情報を付加し返送する。仮に、Ikagent#b に p3 のアプリケーションを保持する Tako が存在しなかった場合は、メッセージの返送を行わない。

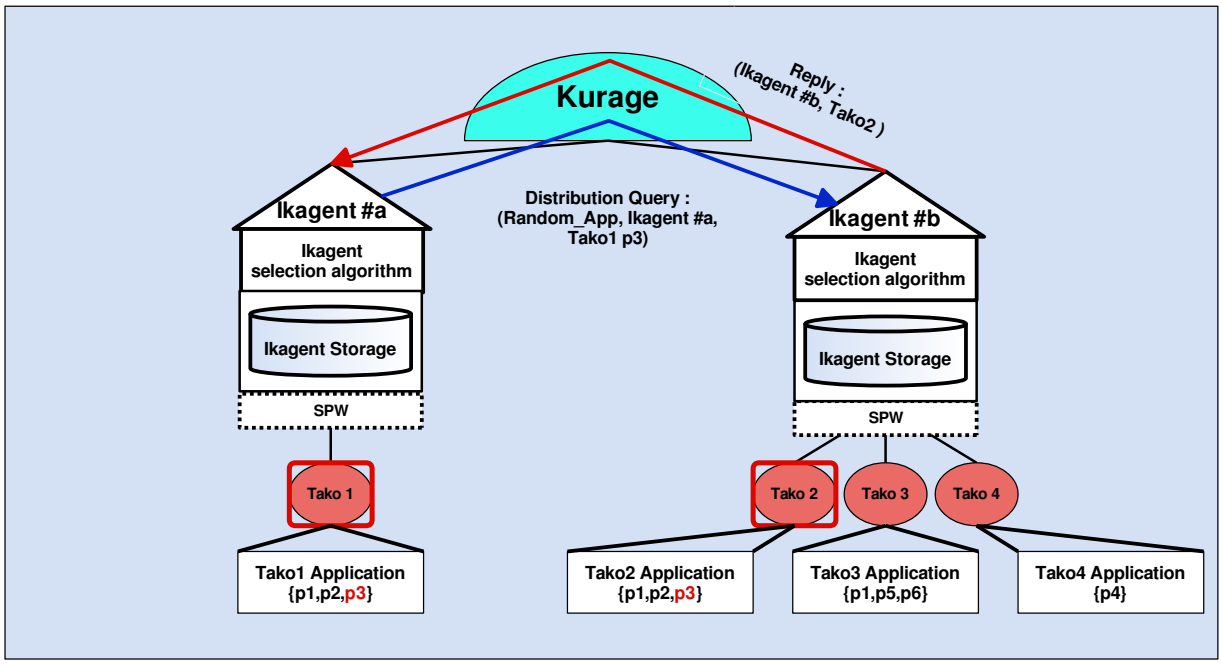

図 4.5: Random_App の概要 (DQ 型)

4.4.2 Exact_Match

Exact_Match は、SPW の NEW メッセージによって検出された Tako に登録されているアプリケーション全てと一致する Tako が存在する Ikagent をトンネルの作成候補として選ぶアルゴリズムである。

図 4.4.2 は SS 型による Exact_Match の概要である。この図では、Ikagent#a に存在する Tako1 が NEW メッセージによって検出された後、Exact_Match の概要を表す。Ikagent#a の Tako1 は p1,p2,p3 の三つのアプリケーションを保持している。そのため、記憶領域から Tako1 と完全に一致するアプリケーションを保持する Tako は Ikagent#b の Tako2 のみである。これにより、結果として Tako2 と Ikagent#b の結果が返される。

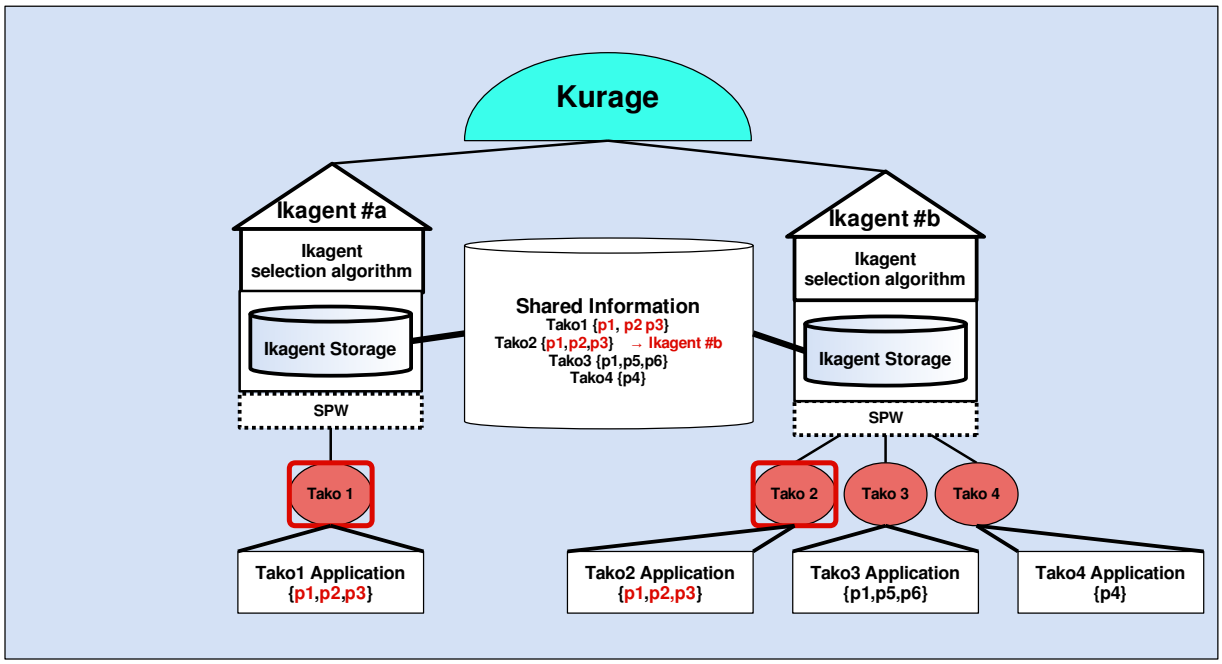

図 4.6: Exact_Match の動作 (SS 型)

図 4.4.2 は DQ 型による Exact_Match の動作である。図 4.4.2 と同様に Tako1 の全アプリケーションからトンネルの作成候補を選択する。検索メッセージに Tako1 の情報と全アプリケーションの情報を付加する。この検索メッセージは Kurage に送信され、Ikagent#b に転送される。

検索メッセージを受信した Ikagent#b は Exact_Match によって作成された検索メッセージと認識して処理を行う。

Ikagent#b は受信したメッセージの中のアプリケーションを調べ、自身の配下に Tako1 と同等のアプリケーションを保持する Tako が存在するか検索する。現在、Tako2 が p1,p2,p3 のアプリケーションを持っているため結果として応答メッセージに Tako2 の情報と Ikagent#b の情報を付加し返送する。Random_App 同様、Ikagent#b に Tako1 と同等のアプリケーションを保持する Tako が存在しなかった場合は、メッセージの返送を行わない。

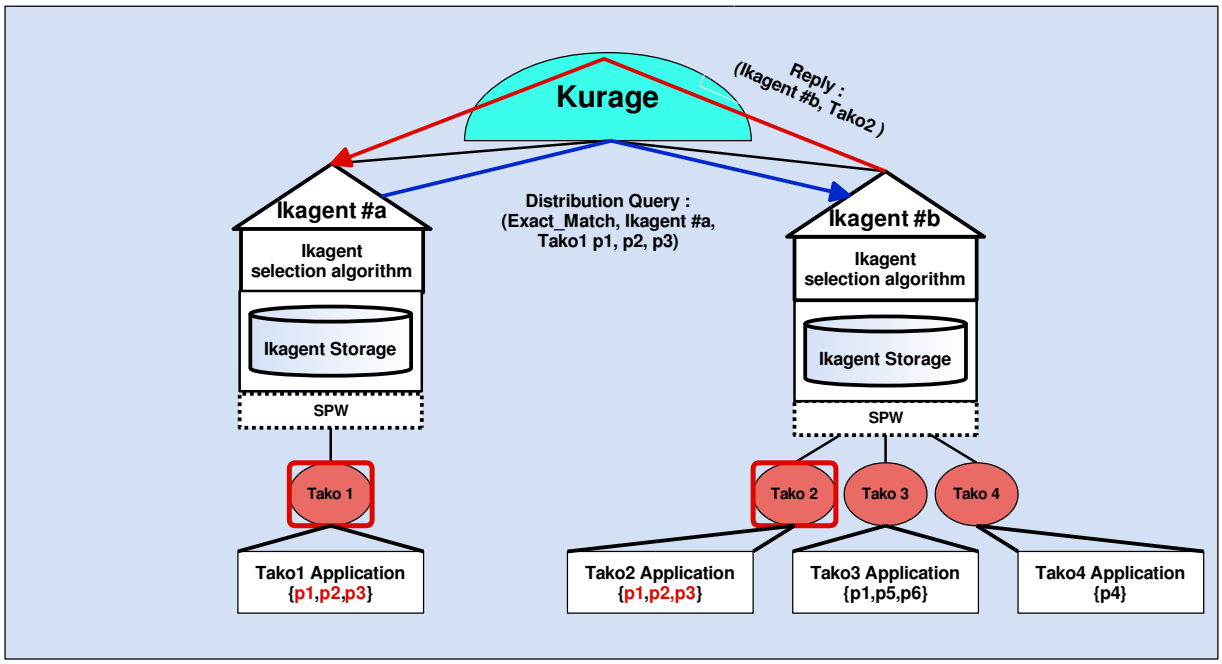

図 4.7: Exact_Match の概要 (DQ 型)

4.4.3 Common_App

Common_App は、Ikagent 内に存在する Tako 情報全てを利用するアルゴリズムである。まず、アプリケーションの情報が格納される空のリストを作成する。そして、Ikagent は自身の配下に存在する Tako のアプリケーションを全て調べる。その時、リストにまだ格納されていないアプリケーションが存在する場合は、そのアプリケーションの情報をリストに追加する。つまり、同一の Ikagent 内におけるアプリケーションの集合を作成する。この時、同じアプリケーションを登録している Tako が複数存在する場合は、その数に関係なく一つだけを集合に追加する。そのため、集合内のアプリケーションの情報は重複せずそれぞれ独立した情報が格納されたリストとなる。

作成されたアプリケーションの集合と、他の Ikagent の Tako のアプリケーションとを比較する。この時、比較対象となった Ikagent に存在する Tako の内、自身で作成した集合要素と一致するアプリケーションがある場合、加算を行い評価値をつける。この時、この評価値が大きい Ikagent ほど、トンネルの作成候補として優先的に選択される。

図 4.4.3 は SS 型による Common_App の概要である。Ikagent#a が Common_App を実行した時の挙動を表す。

Common_App を実行した際、Ikagent#a に存在する Tako のアプリケーションを調べる。現在、Ikagent#a には Tako1 のみが存在するため、Common_App で作成されるアプリケーションのリストには p1,p2,p3 が格納される。

このリストと、Ikagent#b,Ikagent#c に存在する Tako のアプリケーションを調べる。

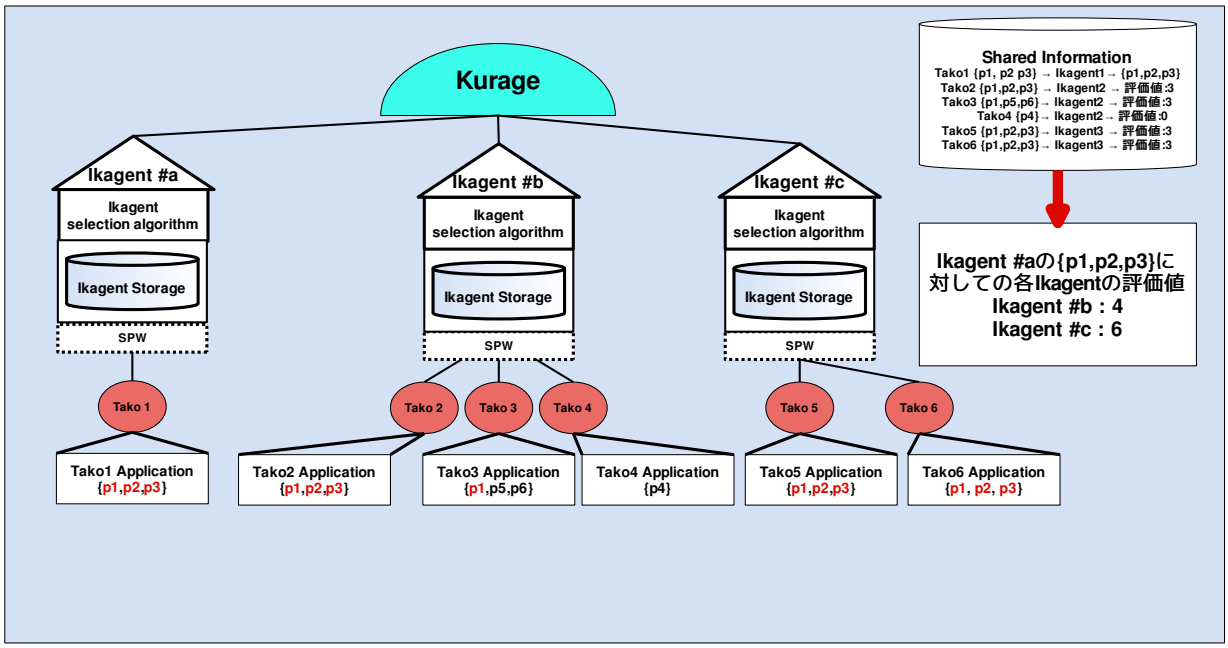

図 4.8: Common_App の概要 (SS 型)

Ikagent#b の場合 Tako2 が p1,p2,p3 を持っているため、一致する各アプリケーションにつき、1 加算した評価値が作成される。更に、Tako3 が p1 を持っているため、最終的に Ikagent#a に対しての Ikagent#b の評価は 3 + 1 の 4 となる。

また、Ikagent#c の場合は、Tako5,Tako6 がそれぞれ p1,p2,p3 を保持しているため、評価値は 3 + 3 の 6 となる。この評価値を降順にソートを行うと、Ikagent#c,Ikagent#b という順序となる。このように、存在する Tako によって作成された集合から評価値を付ける事によって、複数の Tako に対して有用性が高い Ikagent がトンネル作成先の候補となる。

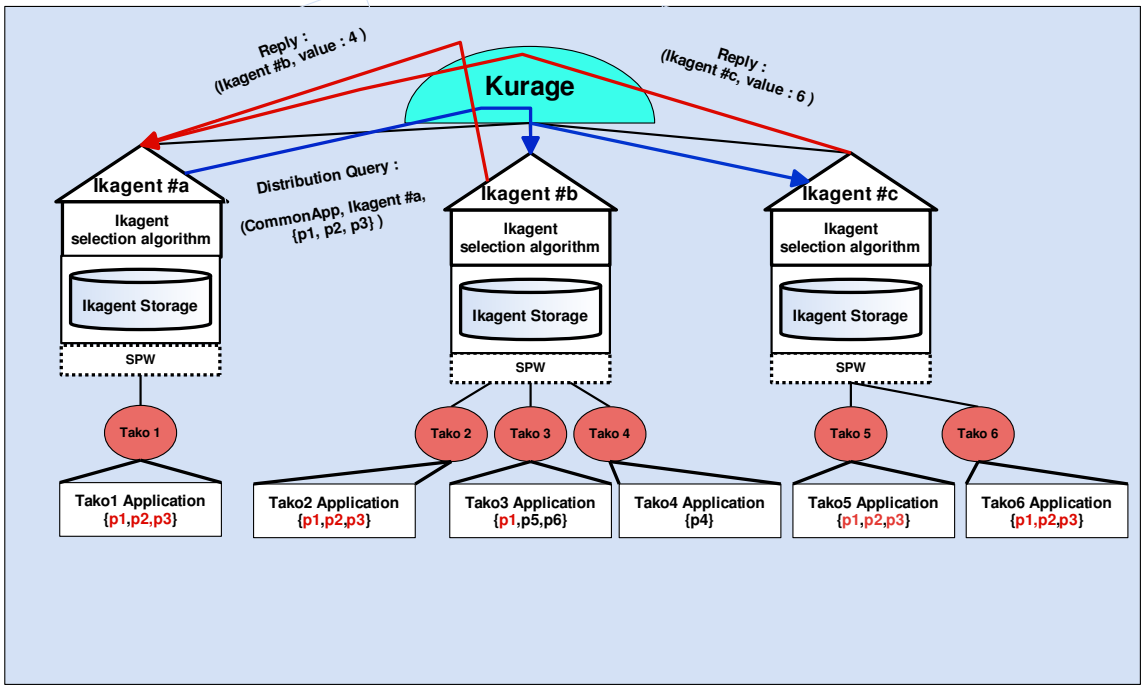

図 4.9: Common_App の概要 (DQ 型)

図 4.4.3 は DQ 型による Common_App の概要である。図 4.4.2 と同様に Tako1 の全アプリケーションからトンネルの作成候補を選択する。

検索メッセージに Common_App と作成したアプリケーションのリストを付加する。この検索メッセージは Kurage に送信され、Ikagent#b,Ikagent#c に転送される。

検索メッセージを受信した Ikagent#b,Ikagent#c は Common_App によって作成された検索メッセージと認識して処理を行う。

Ikagent#b,Ikagent#c 共に、受信したメッセージの中のアプリケーションのリストを調べ、自身の配下にリストと同様のアプリケーションを保持する Tako が存在するか検索する。Ikagent#b の場合、Tako2 が p1,p2,p3 を保持しているため、評価値として 3、Tako3 が p1 を保持しているため 1、合計 4 の評価値が付けられる。Ikagent#c の場合、Tako5 が p1,p2,p3 のアプリケーションを持っているため 3、Tako6 も同様に p1,p2,p3 を持っているため 3 の合計 6 の評価値が付けられる。

返送メッセージに Ikagent の情報と評価値を付加し返送する。Ikagent#a には、Ikagent#b と Ikagent#c の返送メッセージが返ってくる。そのメッセージか r なお評価値をソートすると Ikagent#c の方が評価値が大きいため、Ikagent#a は Ikagent#c とトンネル作成を優先する。

第5章　動作実験

本章は、PWN を構成するために設計した本システムが設計通りに動作するか動作実験を行いその結果を述べる。

5.1　動作実験におけるエミュレーション環境

動作実験を行うために、Kurage として IRC サーバ 1 台、Ikagent として IRC クライアントを 5 台仮想マシン上で動作させる。また、各 Ikagent 内では 10 台の Tako が接続されていると想定し、擬似端末を実行する。各 Ikagent 内に存在する Tako 情報は表**??**に掲載する。Ikagent 選択アルゴリズムの動作実験では Ikagent1 に焦点をあてて各アルゴリズムにおいての結果を取得する。そのため、Ikagent1 を実行する仮想マシンは DQ 型で実行する場合、ネットワークのパケットを tcpdump によって、また SS 型で実行する場合、cat コマンドによって、ファイルに出力する。これは、各 Ikagent 選択アルゴリズムによって出力された結果を取得するためである。

擬似端末の動作は NEW,DEL の二つの動作のみで行う。UPD は、実際にすれ違い通信が行われた後に Tako のアプリケーションのリストに更新を加えるためのメッセージであるため、擬似端末による UPD を考慮した実装が困難であるためである。また、同一の Ikagent 内に存在する各擬似端末の動作間にはランダムに与えられた待機時間を持つ。図 5.1 に、エミュレーション環境のトポロジを表す。

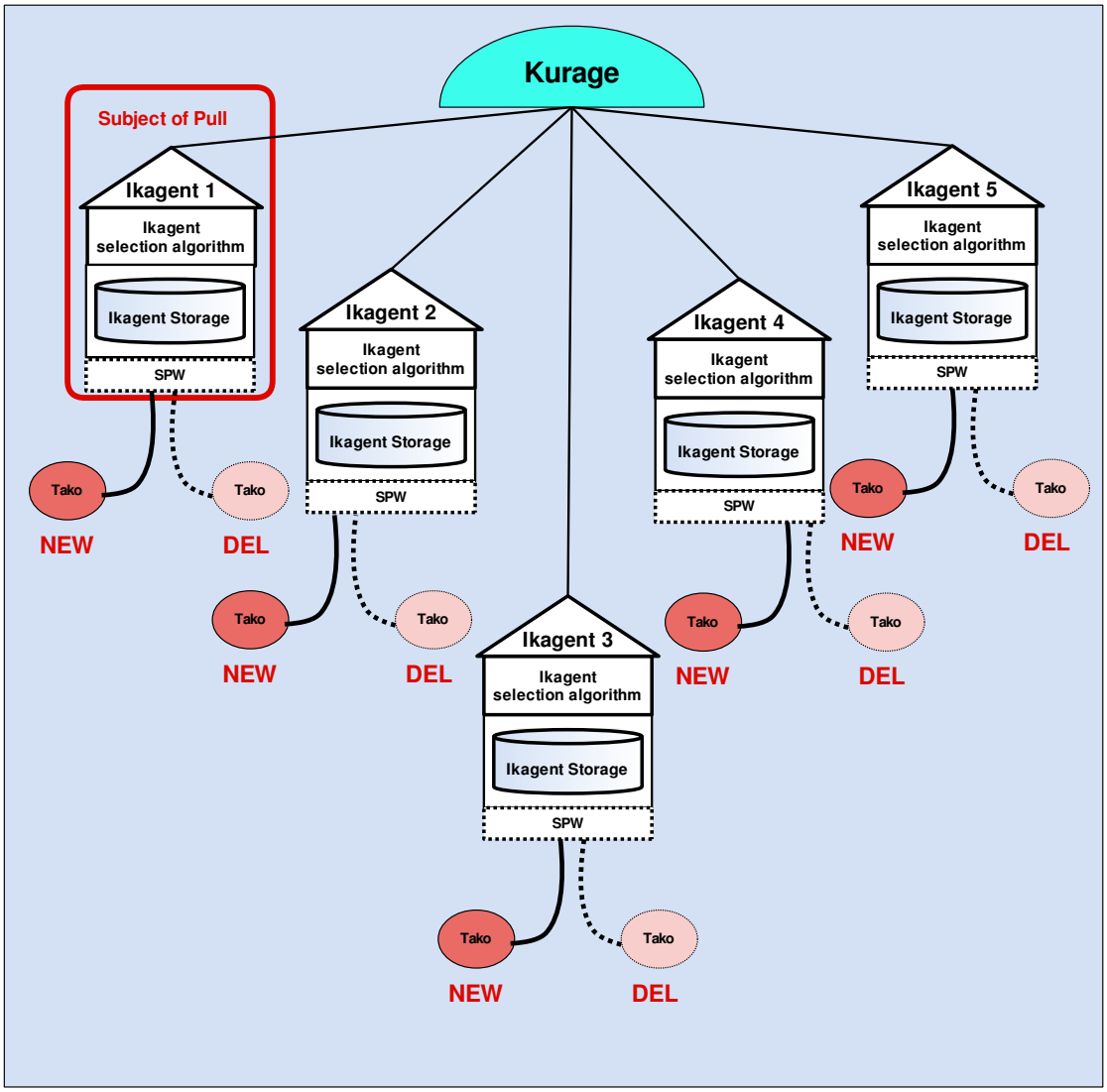

図 5.1: エミュレーション環境のトポロジ

5.2 動作実験の結果

5.2.1 SS 型による実験結果

SS 型は全 Ikagent 内で全 Tako の情報を共有するため、Tako の挙動、また Ikagent の挙動が他の Ikagent にどう影響するかを述べる。

- 同一仮想空間に新たな Ikagent が接続された場合

  既存の仮想空間に新たな Ikagent が所属した場合、以前から所属していた Ikagent には所属した事を通知するメッセージが Kurage から転送される。そして、同一の仮想空間に所属している Ikagent 間であれば、Tako 情報の通知を行う事が可能になった事を確認した。

- 仮想空間に所属する Ikagent が新たな Tako を発見した場合

  他の Ikagent が SPW によって Tako を検出した場合、Ikagent は所属するグループへと検出を意味するメッセージである NEW-TAKO を転送する。これにより、全ての Ikagent で NEW-TAKO を検出し、データベースに格納している事を確認した。

- 発見された Tako が Ikagent から存在しなくなった場合 Ikagent から Tako が存在しなくなった時、つまり SPW が DEL メッセージを送信した時、Ikagent から DEL の対象となった Tako の情報が削除される。DEL の対象となった Tako 情報を送信するメッセージが DEL-TAKO となり仮想空間上に転送される事を確認し、全ての Ikagent から DEL の対象となった Tako の情報がデータベースからも削除されている事を確認した。

- Ikagent が実行を終了した場合

  Ikagent の実行が終了した時には、終了の対象となった Ikagent 内の Tako の情報を全て削除するための通知として DEL-IKAGENT が転送される。このメッセージは削除の対象となった Ikagent の情報のみが付加される。各 Tako の情報には、どの Ikagent が発見されたも、削除の対象となった Ikagent の情報に関連した Tako の情報が削除されている事も確認する事が出来た。

5.2.2 DQ 型による実験結果

DQ 型は、分散検索をかけて結果をデータベースに格納するため、Ikagent が新しく所属する以外に、検索メッセージの送信や応答メッセージの到着において、どのような動作結果をもたらすかを述べる。

- 同一仮想空間に新たな Ikagent が接続された場合

  SS 型と同様に、同一の仮想空間に所属している Ikagent 間であれば、Tako 情報の通知を行う事が可能になった事を確認した。

- 仮想空間に所属する Ikagent が新たな Tako を発見した場合

  他の Ikagent が SPW によって Tako を検出した場合、各アルゴリズムによって分散検索のための QUERY メッセージを転送する。これにより、全ての Ikagent で QUERY を検出し、各 Ikagent は検索に対しての処理を行う事を確認した。

- 応答メッセージの送信

  検索メッセージから各 Ikagent 選択アルゴリズムを実行し、結果を返すための応答メッセージを作成する。検索の対象となった Tako 情報の要求と合致した Ikagent だけが応答を返す事を確認し、要求と合致しない Ikagent は応答を返さない事を確認した。

5.2.3 各 Ikagent 選択アルゴリズムの実験結果

各 Ikagent 選択アルゴリズムを SS 型、DQ 型で実行させアルゴリズムのプログラムが出力する結果から、決定された Tako,Ikagent の数を調べる。Random_App,Exact_Match は個々の Tako と検索対象となる要素が一致した時に Tako の情報が返されるため、これら二つのアルゴリズムに返しては結果を返した Tako 数を調べる。Common_App は、Tako の情報を集合として格納してから Ikagent の情報を結果として返すため Common_App に関しては、結果を返した Ikagent 数を調べる。結果の抽出に関しては Ikagent1 を実行しているマシンに tcpdump を実行し、取得した Tako,Ikagent 情報を調べる。

SS 型,DQ 型における Random_App,Exact_Match を Ikagent1 で 1000 秒実行した際に結果を返した Tako 数 (重複した Tako の結果は含まない) は表 5.1、表 5.3 のようになった。

また、Common_App に関しては表 5.2、表 5.4 のような結果となった。

表 5.1: SS 型による Random_App,Exact_Match の実行結果

実行したアルゴリズム	結果を返した Tako 数	Ikagent2	Ikagent3	Ikagent4	Ikagent5
Random_App	32	7	9	7	9
Exact_Match	5	3	1	0	1

表 5.2: SS 型による Common_App の実行結果

実行したアルゴリズム	結果を返した Ikagent 数
Common_App	4

表 5.3: DQ 型による Random_App,Exact_Match の実行結果

実行したアルゴリズム	結果を返した Tako 数	Ikagent2	Ikagent3	Ikagent4	Ikagent5
Random_App	32	5	9	9	9
Exact_Match	4	2	1	0	1

表 5.4: DQ 型による Common_App の実行結果

実行したアルゴリズム	結果を返した Ikagent 数
Common_App	4

第6章　動作実験からの考察

5章より、本システムの動作実験を行い、その結果を纏めた。本章では、この結果から各設計案についての考察を述べる。

6.1　SS型についての考察

SS型は、同一の仮想空間上に存在するIkagent全てにTako情報を通知する。そのため、一度Tako情報が通知されれば全てのTakoとすれ違い通信を行えるようになる。今回の実験では、各Takoにランダムの待機時間をもたせているものの、各Ikagentに全てのTako情報が反映されるためにある程度の時間をかければ全てのTakoの存在を知る事が可能となる。

その反面通知されるTako情報が多くなると、その分転送されるメッセージ数も増加する。従って、IkagentがTako情報を記憶するためのディスクよりも先にIkagentがメッセージを転送する際のネットワークがボトルネックとなる。

今回の実験ではTakoの数が少ない事、Ikagentが新しく仮想空間に所属してきた時に、新たに所属してきたIkagentがそれ以前までに発見したTakoを通知するような仕組みを取っていない。そのため、一つのIkagentから発見されたTako数が多くなったとしてもネットワークの帯域を急激に消費してしまうような事態は起こらなかった。しかしIkagentが多数存在し、またTako情報が通知されるタイミングによってはネットワークトラフィックが増加してしまい、Ikagentを実行するマシンのパフォーマンスによってはダウンしてしまう可能性もある。そのため、Tako情報の通知に対して制限をかけると言った通知に関して直接的な対策を行う必要があると考えられる。

6.2　DQ型についての考察

DQ型は検索をかけて応答を返したIkagentのみがトンネル作成先の候補となる。そのため、今回の動作実験において同じIkagent数、同じTako数で実行した場合SS型よりもネットワーク全体に転送されるメッセージは比較的少なくなる。そのため、DQ型では仮想空間内のIkagent内に存在する全てのTako情報を知る事は出来ないが、代わりにIkagent選択アルゴリズムに基づいてノードの要求する情報を取得する事が可能となり、格納するTako情報を犠牲にする代わりに結果はより良いものとなると言える。

その反面、検索に対してIkagentが調べるTako数が少ないと結果を返す事が困難となる。特に、その問題を顕著に表したのがDQ型で実行するExact_Matchである。Exact_Matchは完全一致なため、検索対象となったTako情報と一致するものが存在しない限りIkagentが応答を返す事がない。今回の実験では各Ikagent選択アルゴリズムの純粋な性能を測るために要求条件を厳しくしすぎた結果、応答を返す事がなかった場合に要求条件をある程度簡単にし再検索をかけるという事をしていない。そのため、DQ型においてIkagent選択アルゴリズムを作成するにあたっては、再送の機能も考慮する事が必要となる。

6.3 各Ikagent選択アルゴリズムについての考察

本研究では、Ikagent選択アルゴリズムとしてRandom_App,Exact_Match,Common_Appの設計を行った。実験結果として得られた結果はDQ型、SS型の何れにせよある程度の時間をかければ全てのアルゴリズムに対して要求を満たすTako,Ikagentの情報を得る事が可能となった。そのため、新規にIkagent選択アルゴリズムを設計するにあたってはTako数、Ikagent数の要素を考慮する必要はなくどのようなIkagent、Takoの情報を取得したいかだけを考慮する事によって、ユーザに対してより有意義なすれ違い通信を行う事が可能と考えられる。

第7章　おわりに

本研究では、すれ違い通信の性能改善を目的とし、ノードの情報を転送するトンネル・ポイントを配置する事によって構築されるネットワークとして Pocket Warped Network を提案した。トンネル・ポイントを利用する事によって、すれ違い通信が発生する確率は飛躍的に向上する事となる。

また、すれ違い通信においてそれまで実現するのが困難であった pull の利用目的に応じた通信を Ikagent 選択アルゴリズムを提案・設計する事で可能となる。

PWN を構築するにあたってノードに対して直接変更を加える事なく、ノード上で起動するアプリケーションの情報交換の品質を改良するという事を目的とした。今回の実験では Nintendo 3DS を対象としたが、それ以外にもすれ違い通信でアプリケーションの情報交換が可能なノードであれば、PWN を活用できると期待される。

また、PWN の規模によって SS 型、DQ 型という Ikagent,Tako の情報を取得するための方法を設計した。そのため、用途によっての使い分けが可能となる。

7.1　今後の課題

7.1.1　物理ノードの Tako による PWN の動作検証

節 5 より、本研究での実験は Tako を擬似端末として行い結果を出した。そのため今後は、Tako を物理ノードに変更し実際にユーザが利用する環境においての動作検証が必要と考えられる。物理ノードの候補としては、Nintendo 3DS が挙げられる。今回、全ての実験に関して Nintendo 3DS に関係した環境、ノードの挙動等を考慮して擬似端末や実験環境の構築を行った。そのため Nintendo 3DS 本体における PWN の動作検証に対しても、類似性が高い環境上でヒューマンモビリティを考慮した実験が可能と考えられる。

7.1.2　アプリケーション以外のパラメータを利用した Ikagent 選択アルゴリズムの設計

Ikagent 選択アルゴリズムの設計として、Random_App,Exact_Match,Common_App の設計を行った。これらはすれ違い通信においてアプリケーションの種類を利用して いる。そのため、トンネルの張り直しを行ったとしても同じ Ikagent をトンネルの作成先の候補

として選択する可能性が高い。そのため、アプリケーション以外のパラメータを利用する事でこの問題は解決できると考えられる。例として、Tako 同士のすれ違い通信の履歴を Ikagent で記録し、その記録を利用して Ikagent の選択を行うアルゴリズムが第一に考えられる。これにより、Ikagent 内ですれ違い通信を行う機会が少ない Tako や、すれ違い通信が集中して発生する Tako の存在を知る事が可能となり、Tako の個々の情報から Ikagent を選択する事が出来ると考えられる。

7.1.3 NAT の内側に存在する Ikagent との通信

本研究では、Ikagent にグローバル IP アドレスが付加されている事を想定してシステムの設計を行った。そのため、同一の仮想空間に存在する限り Ikagent 間のトンネル作成は容易に行えると考えられる。

しかし、一般的に家庭内のネットワークはプライベート IP アドレス空間であり、NAT によってグローバル IP アドレスに変換されてから通信を行う。そのため、NAT の内側に存在する Ikagent とトンネルを作成するための方法を提案する事で、PWN によるすれ違い通信の発生頻度がより向上すると考えられる。

7.1.4 Ikagent 間の通信の並列処理化

Ikagent 間で送受信される Tako 情報の処理は単一のスレッドで待ち受けを行っている。そのため、複数の Ikagent が同時に Tako 情報を送信した場合、前スレッドの実行が完了するまでは次の Tako 情報の処理を行う事が出来ない。従って、Tako 情報の送受信における処理を並列化する事で、スケーラビリティの向上が期待できると考えられる。

謝辞

本研究を行うにあたり、多くの方々からの多大なる御助言や御協力をいただきました。その方々一つ一つの御協力がなければ本研究は成立しませんでした。心より厚く御礼申し上げます。

本研究を進めていくにあたって、主指導教官で篠田陽一教授には、様々な御助言、適切な御指導、外部プログラムの寄与を賜りました。心から深く感謝致します。また、副指導教官である丹康雄教授、知念賢一特任准教授、また副テーマ指導教官である田中宏和准教授には本研究に関しての助言をいただきました。心から感謝いたします。

本研究室の宇多仁助教授、井上朋哉助教授には本研究に関して活発な議論や多大なる御指導を賜りました。心から感謝致します。

IIJ 技術研究所の長 健二朗室長には研究に関する御助言をいただきました。心から感謝致します。

株式会社創夢の井上 潔第三開発部部長には研究に関する御助言及び今後の生活の面での相談と多様な面で支えていただきました。ここに深く感謝致します。

本研究室の博士後期課程の明石邦夫氏には研究の方針について多くの御意見を頂きました。心から感謝致します。

本研究室修了生の大野夏希氏には研究の方針について多くのご意見を頂きました。心から感謝致します。

本研究室博士後期課程の Muhammad Imaran Tariq 氏、博士前期課程の岩橋鉱司氏、加藤邦章氏、成田圭介氏、岩本祐真氏、園田真人氏、可児友邦氏には研究以外でも普段の生活の面で支えて頂きました。ここに心から感謝致します。

最後に研究や生活を支えてくれた家族に感謝します。

参考文献

[1] 災害に強いインフラを支える業務用移動無線におけるアドホックネットワーク技術の活用

http://www.soumu.go.jp/main_content/000287475.pdf

[2] 崩壊した通信インフラ、災害に強いネットワークとは — 日経エレクトロニクス Online

http://techon.nikkeibp.co.jp/article/HONSHI/20110526/192147/

[3] 第 4 回 MANET におけるさまざまなプロトコル

http://internet.watch.impress.co.jp/www/column/wp2p/wp2p04.htm

[4] Shengling Wang, Min Liu, Xiuzhen Cheng, Min Song

Routing in pocket switched networks

Wireless Communications, IEEE , Vol. 19, No. 7, pp67–73, 2012.

[5] すれ違い通信といつの間に通信 — ニンテンドー 3DS — Nitendo - 任天堂

http://www.nintendo.co.jp/3ds/hardware/features/network.html

[6] Wei Gao, Guohoug Cao

On exploiting transient contact patterns for data forwarding in Delay Tolerant Networks

Network Protocols(ICNP), 2010 18th IEEE International Conference on, pp193–202, 2010

[7] Evan P.C Jones, Lily Li, Jakub K. Schmidtke, Paul A.S. Ward

Practical Routing in Delay-Tolerant Networks

IEEE Transactions on Mobile Computing, Issue No. 8, pp943-959, 2007.

[8] Pan Hui, Crowcroft, J, Yoneki, E

BUBBLE Rap: Social-Based Forwarding in Delay-Tolerant Networks

Mobile Computing, IEEE Transactions on, Vol. 10, Issue 11, pp1576–1589, 2010.

[9] E. Daly, M. Haahr
Social Network Analysis for Routing in Disconnected Delay-Tolerant Manets
Mobile Computing, IEEE Transactions on, Vol. 8, Issue 5, pp606-621, 2008.

[10] S. Wang et al.
A Home Based Wireless Relay Selection Strategy in Pocket Switched Networks
submitted to IEEE JSAC Theories and Methods for Advanced Wireless Relays, 2011.

[11] Laguay, J, Friedman, T, Conan, V
Evaluating Mobility Pattern Space Routing for DTNs
INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings, pp1–10, 2006.

[12] 本体の主な機能 — Nintendo 3DS — 任天堂
http://www.nintendo.co.jp/3ds/hardware/specs/index.html

[13] ニンテンドー 3DS の国内累計販売台数が 1500 万台を突破! — ファミ通.com —
http://www.famitsu.com/news/201401/28047239.html

[14] 小野田崇, 坂井美帆, 山田誠二
k-means 法の様々な初期値設定によるクラスタリング結果の実験的比較
http://www.ymd.nii.ac.jp/lab/publication/conference/2011/JSAI-2011-Onoda.pdf

[15] David Arthur, Sergei Vassilvitskii
k-means++: The Advantages of Careful Seeding
http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf

[16] 第 XVII 部 オーバーレイネットワークによる統合分散環境
http://www.wide.ad.jp/project/document/reports/pdf2002/part17.pdf

[17] 第 XVI 部 IRC の運用状況とデータ解析
http://www.wide.ad.jp/project/document/reports/pdf2002/part16.pdf

[18] RFC 1459 - Internet Relay Chat Protocol - IETF Tools
https://tools.ietf.org/html/rfc1459

[19] SQLite
http://www.sqlite.org