平成20年度　　修士論文

# クライアント群の近傍証明と<br>ファイルアクセス制御への応用

指導教官<br>
松尾　啓志　教授<br>
齋藤　彰一　准教授

名古屋工業大学大学院　工学研究科情報工学専攻<br>
平成19年度入学　19417590番

中山　英威

# 目 次

# 第1章

# はじめに

近年、携帯型計算機は一人が一台以上所持することが当り前となりつつあり、端末自体のサイズも小さくなってきている。人はどこにいても計算機からネットワークに接続することが可能となり、ネットワークを利用したサービスは益々有用となってきている。

一例として、会議やイベントにおいて参加者全員に資料を配布することを考える。参加者には資料を閲覧可能としたいが、一方で、それらの資料を参加者でない外部の人間には情報を漏らしたくない、というケースは少なくない。最も一般的な資料の配布方法は、紙の資料を作成して配布する方法である。しかし、外部への情報漏洩を防ぐことを考えた場合、紙資料は例えば会議終了後に回収して処分せねばならず、非常にコストがかかる。

そこで、ネットワークにより資料を閲覧させることで、資料配布のコストを大幅に低下させることができる。ただ代わりに、外部への情報漏洩が容易となることは明らかであり、インターネットを通じて全世界に発信される等、被害範囲が非常に大きい。

参加者のみを特定する方法としては、参加者にパスワードを教えてアクセスを制限する、GPS 等の位置情報を元に参加者かどうかを判断する [1][2][3]、といった方法もあるが、これらはそのような端末の環境やパスワードの配布といった準備が必要である [4][5]。このため、誰もがどのような状況でも使えるというようものではない。そこで本研究では、端末の機能やその環境とは関係なく、ネットワークによって参加者のみに資料を閲覧させる手法について提案する。

以下、第 2 章で端末の位置を把握する既存の手法ついて、第 3 章では、第 2 章で述べたものを用いずにアクセス制御を行う提案手法を述べる。また、実装については第 4 章で示す。そして、第 5 章で、提案手法を実際の端末を用いて実験及び考察を行い、第 6 章で結論をまとめる。

# 第2章

# 既存手法

特定の無線端末にのみアクセス許可を与える既存の手法を述べる。最も簡単なアクセス許可を与える方法としては、クライアントがサーバやファイルへのアクセスする際に認証すればよい。しかし、許可されていない端末でも認証によってファイルへのアクセスが可能となるため、適切ではない。本章では、特定端末外はアクセスが不可能となる、サーバクライアントモデルについてアクセス許可を与えるものについて述べる。

## 2.1 サブネットを用いたアクセス許可

無線のアクセスポイントのサブネットを用いることで、特定範囲内の無線端末にアクセス許可を与えることができる [6]。サブネットマスクは、ルータであるアクセスポイントの IP アドレスのうち、サブネット ID の部分である。無線クライアントがアクセスポイントを用いてサーバにアクセスする場合、アクセスポイントに接続する全てのクライアントは同じサブネット ID を使用する。特定のサブネット ID にのみアクセス許可を与えることで、そのサブネット ID を持つアクセスポイントの電波範囲内にあるクライアントのみにアクセス許可を与えることができる。

ただしこの手法では、前述のようにサーバにアクセスするクライアントが全て同じサブネット ID を持っている必要がある。また、サブネットごとに許可を与える場合、各アクセスポイントごとに別々のサブネット ID を与える必要がある。

## 2.2 アクセス制御リストによる許可

Windows OS では、**アクセス制御リスト (Access Control List:ACL)** というものを利用して、ファイルやフォルダ等へのアクセス許可を管理している。ユーザやグループ等に対して利用可能な権限や拒否を定義したものを**アクセス制御エントリ (Access Control Entry:ACE)** と言い、これを集めたものが ACL である [7]。クライアントに許可を与える ACE を追加もしくはこれを拒否することで、サーバ上のファイルへ特定のクライアントのみがアクセスできる。

しかし、新規クライアントがアクセスを要求した場合、管理者にはこのクライアントが許可を出せるクライアントなのかを知る術がなく、あらかじめアクセスを予定していたものにしか対応できない。

# 第3章

# 提案手法

本章では提案手法として、他のクライアントとの相互証明により「クライアントが集団内に存在するか否か」の判断を行う手法、及びこの判断によるファイルへのアクセス制御について説明する。

本研究では、多数の端末 (クライアント) が集まっている場所を参加者の集団であると見做す。集まったクライアントに対してアクセス許可を出すことで、参加者に対するアクセス許可を実現する。また、アクセス許可を受けた参加者が集団から離れた時、その参加者が情報を外部に持ち出す可能性があり、これに対するアクセス制御についても行う。ただし、本研究においては、アクセスするクライアントの殆どは虚偽報告などの不正操作は行わないものとして考える。

## 3.1 クライアント群による相対位置証明

本節ではクライアント同士の位置証明手法について述べる。無線端末が自らの位置をサーバに教える場合、自己申告による位置情報では虚偽報告の可能性があるため、GPS 機能やアクセスポイントのサブネット機能による第三者による証明が必要となる。しかし、GPS 機能がどの端末にもついているわけではない。また、アクセスポイントによる証明を行う場合、全ての端末アドレスのサブネットが同じでなければならなかったり、アクセスポイントで場所を特定できるようにサブネットを割り振らなければならないなど、制約面で問題が多い。

そこで、会議やイベントなどにおいては、幾つもの端末が同じようにサーバ上の同

一の資料にアクセスすることに着目する。クライアント自身の証明を近傍にいる他のクライアントに任せ、これを相互に行うことにより参加者の全員の位置を確認する。他のクライアントに証明されないクライアントはどこか遠くにあるクライアント、即ち、集団から離れた場所からアクセスしたクライアントであると判断する。

### 3.1.1 クライアントリストの作成

他のクライアントに自身を証明してもらうために、クライアントは自身の MAC アドレスや IP アドレスといった情報を他に渡す必要があり、逆に他のクライアントを証明するにはそれらの情報を自身が得る必要がある。各クライアントは近傍に存在するクライアントを相互に証明し、それらの証明をサーバに集めることで、サーバは所持しているファイルへのアクセス許可を与えるクライアントを判定する。

クライアントがサーバにアクセス要求を行った際、サーバはそのクライアントのアドレスを入手する。同様にアクセス要求があった全クライアントから入手した全アドレスを**クライアントリスト**として作成し、要求を行った各クライアントに返信する。ここで入手するアドレスはクライアントの位置は考慮せず、アクセス要求のあった全てのクライアントに対して行う。

図 3.1 では、クライアント A は集団から離れた場所に存在するが、クライアントリスト作成時にはこれが集団内にあるのか外部からアクセスを試みているのか分からないため、クライアントリストでは区別せずにリストへ追加する。本節では、以降図 3.1 のクライアント群を用いて近傍クライアントの証明について説明する。

### 3.1.2 近傍リストの作成

クライアントリスト作成後、クライアントが近傍に存在することの確認と、そのクライアントについてのサーバへの証明を行う。クライアントの証明には、無線 LAN のアドホックモードを利用する。アドホックモードでは、クライアント同士が直接通信を行うため、クライアント間の距離をある程度近付けなければ、通信を行うことができない。これを利用して、クライアントから目的のアドレスに向けて 1 ホップ PING

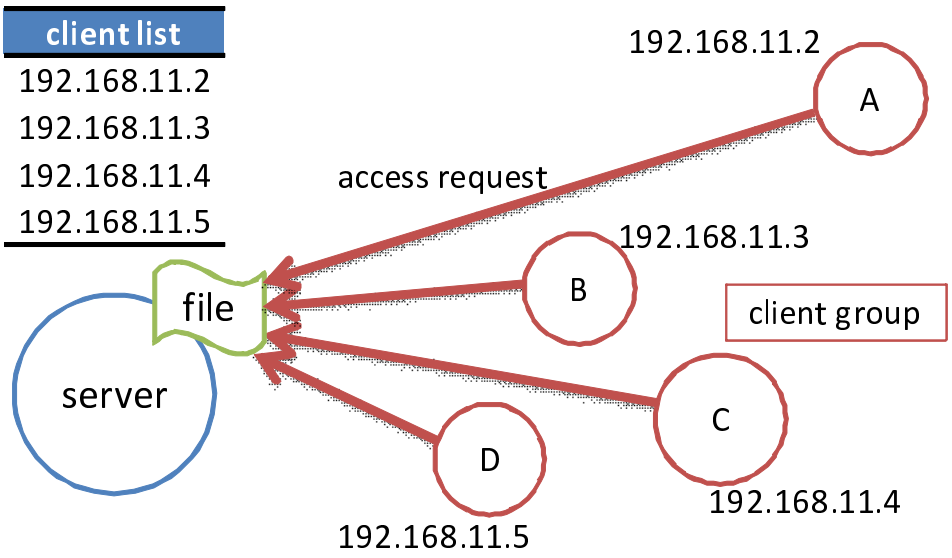



図 3.1: クライアントリストの作成

を発信する。クライアント間が近ければ PING は発信元のクライアントに返り、発信先のクライアントが自身の近くにいることが分かる。クライアントリストの全アドレスに対して同様の処理を行い、PING が返ったアドレスのリストを**近傍リスト**として作成する。

図 3.2 はクライアント C の近傍リストの作成を示している。C はクライアントリストにある全てのアドレス（自身を含む）に PING を発信する。近傍に存在するクライアント B,D、及び自身に対しては PING が返るため、そのアドレスが近傍リストに追加される。クライアント A は集団から外れた場所にあり PING が届かないため、アドレスが近傍リストには加えられない。各クライアントについても同様に、サーバからのクライアントリストの受信に対して近傍リストをサーバに返信する。

### 3.1.3　サーバによる近傍リストの集計

サーバが近傍リストを基にクライアントへのアクセス許可の判定を行う。サーバはクライアントリストにある各クライアントに、集団内に存在する可能性として**スコア**を設ける。近傍リストを受信した際、クライアントが他の近傍リストによって証明され

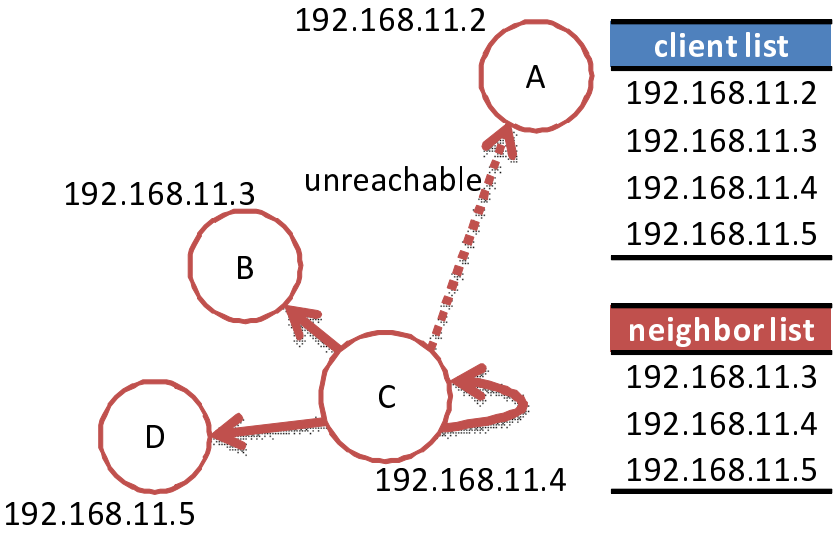



図 3.2: 近傍リストの作成

るごとにそのクライアントの持つスコアを増加する。アクセス要求を受けた全クライアントに対して同様の処理を行い、このスコアの結果により判定を行う。

判定は多くのスコアを得たクライアント、つまり、多くの他クライアントに証明されるクライアントについてアクセス許可を出す方式である。逆に、他のクライアントから証明されていないクライアント、もしくは証明するクライアントが少ないものについては、外部からアクセスを試みたクライアント、または、集団から少し離れた所からアクセスしようとしたクライアントだと判定し、許可しない。

図 3.3 の下部にある近傍リスト (neighbor list) は各クライアントからサーバが受信した近傍リストである。サーバはクライアントリスト上の全クライアントがそれぞれ近傍リストで何回証明されているかを数え、証明された回数をスコア (図 3.3 の左上部) とする。図中の 192.168.11.2 のクライアント A はその証明回数が自分自身を証明したのみであり、他クライアントによる証明が得られていないことが分かる。詳しい判定方法は、後述の 3.2 節で示すが、ここではスコアの低いクライアント A をアクセス不許可とし、残りの B,C,D についてはアクセスを許可する。

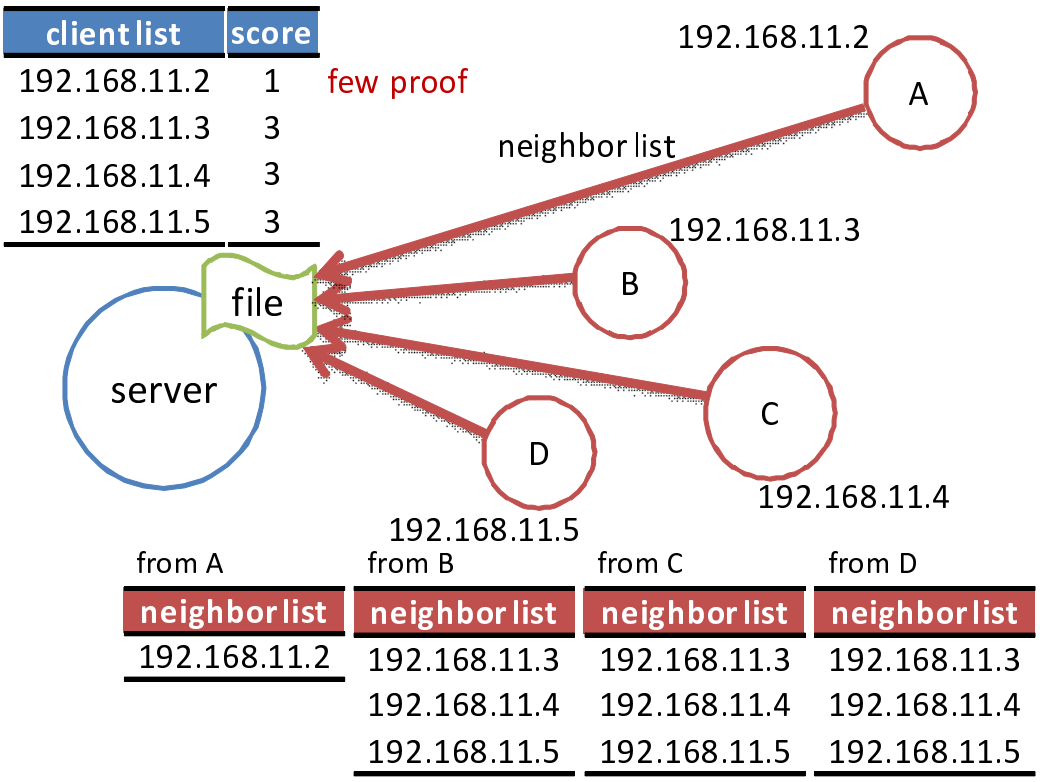



図 3.3: 近傍リストによるアクセス許可判定

## 3.2　アクセス許可の判定

サーバが各クライアントに対して、アクセス許可と不許可の判定を行う手法について述べる。本節では、図 3.4 のクライアント群のモデルと、このときの各クライアントからサーバに送信される近傍リストを用いて述べる。

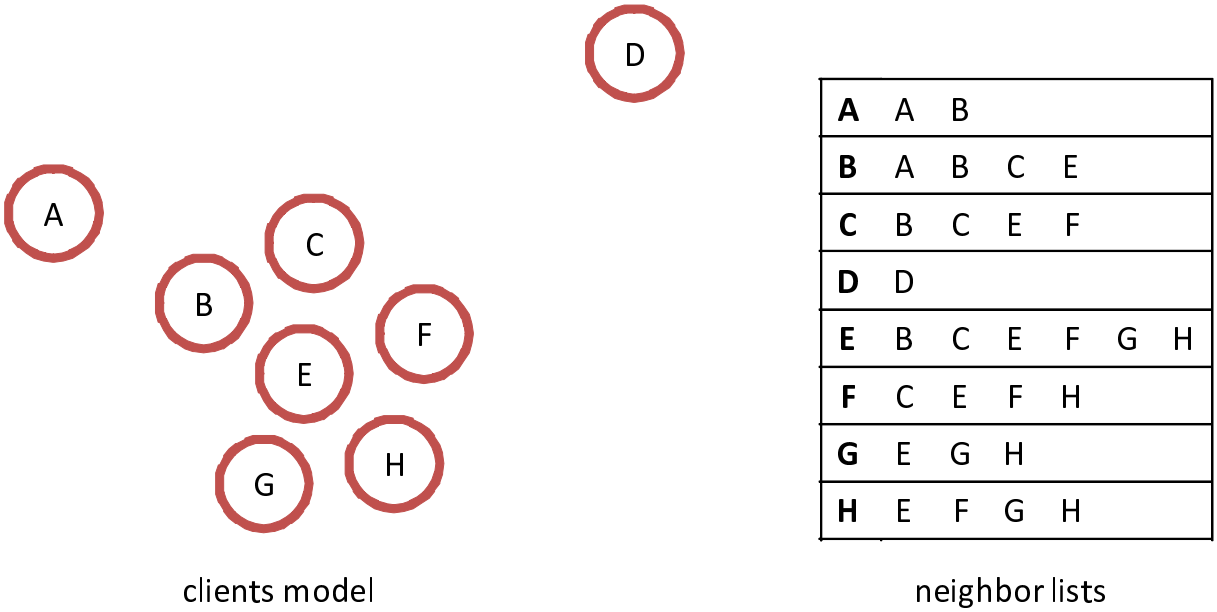



図 3.4: クライアント群モデル

クライアント B,C,E,F,G,H を参加者の集団とし、クライアント A は集団の近くに存在するクライアント、クライアント D は十分離れた場所にいる外部からアクセスを試みるクライアントとする。

### 3.2.1　単純加算による判定

アクセス許可の判定方法として、近傍リストによる証明を単純に加算した場合を考える。図 3.4 のモデルにおいて、3.1 節で述べたように、近傍リストによる証明回数をスコアとした場合、近傍リストより、他クライアントからの証明が多い E のクライアントは集団内のクライアントとして許可を、A や D のように証明の少ないものは集合の外部にいるクライアントである可能性が高く不許可を出す（図 3.5 参照）。証明回数

には、自分自身への証明も含むものとしている。

| | スコア | 判定 |
|---|---|---|
| A | 2 | 不許可 |
| B | 4 | 許可 |
| C | 4 | 許可 |
| D | 1 | 不許可 |
| E | 6 | 許可 |
| F | 4 | 許可 |
| G | 3 | 不許可 |
| H | 4 | 許可 |

図 3.5: 近傍リストから得られた証明数と判定 (閾値 3 の場合)

ただし、クライアント B,C,F,G,H のように、主観によって証明が多いとも少ないとも取れるクライアントへの判定には、何らかの閾値を用意して判定を出す必要がある。図 3.5 では、閾値をスコア 3 として、これより大きいスコアについてアクセス許可を出すものとしている。このため、参加者として集団内に存在するはずのクライアント G にはアクセス許可が与えられない。つまり、情報漏洩の危険度や参加者の分散程度を考慮に入れて閾値を決める必要がある。また、図 3.5 の各スコアは、図 3.4 で各クライアントが近傍リストとして証明を行った数と等しいが、常に等しくなるとは限らない。これについては次小節で説明する。

### 3.2.2 誤情報を考慮に入れた判定

クライアントがサーバに対して実際とは異った証明を行った場合について説明する。3.1 節で述べた通り、クライアントはそれぞれ、自身の近傍にあるクライアントの情報をサーバに送信することで、相互に自らが集団内に存在することを証明する。しかし、図 3.6 のように、各クライアントがサーバに送信する情報が必ずしも正しい情報であるとは限らない。ここではクライアント E が本来は近傍に存在するはずのクライアン

トCを、何らかの理由により近傍リストに加えなかった時を示している。

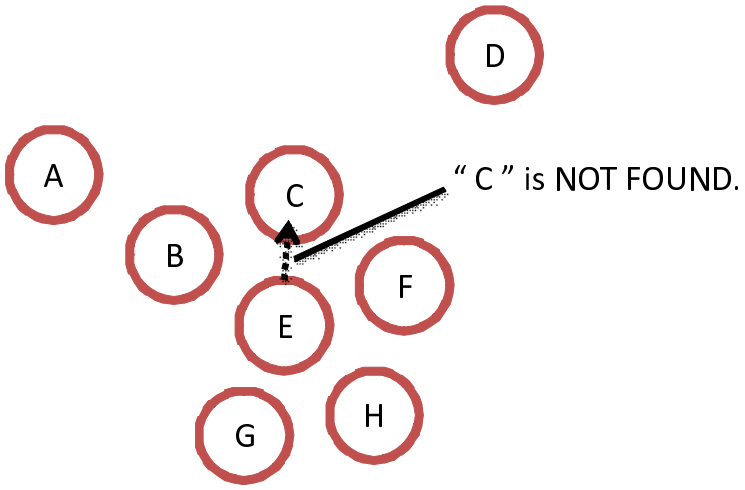



図 3.6: 誤った近傍リストの作成

このような本来とは違う近傍リストを作成する理由として、以下のような場合が考えられる。

- クライアントの移動などにより、近傍クライアントが変化した場合
- 悪意により、実際に観測した近傍クライアントとは別のリストを送信した場合

前者については、常に端末の位置が変化し得る無線通信では非常に起りやすい。この場合、再度 3.1 節の処理を行うことで問題を解決できる可能性が高いが、クライアント数が増えたるに従い、通信トラフィックが大幅に増加する。後者に至っては、クライアント自身に悪意があるため、再度同様の処理を行っても問題が解決できないため、対策が必要となる。

サーバに送信する近傍リストが実際と正しくない場合は、次の 2 通りがある。

- 近傍には存在しないクライアントが、近傍リストに追加される場合
- 近傍に存在するクライアントが、近傍リストから外される場合

前者の場合、外部のクライアントが集団内にいることを装うことがを考えられる。例として図 3.4 のクライアント D が集団内に存在することを装う場合を考える。クライアント D は集団から離れた位置にあるため、本来ならば D により証明される他クライアントは存在しないが、ここで D があたかも集団内に存在するかのように近傍リストにクライアント C,F を追加することで、サーバに対して実際とは異なる証明を行うことも可能である。しかし、虚偽報告を行ったクライアント D は実際には近傍に存在しないクライアントについての証明をするが、D 自身についての証明はクライアント C,F に行われない。このため、証明の相互一致ではなく、証明された数によりアクセス許可の判断を行うことで、虚偽報告を行うことによるクライアント D のメリットは無くなり、情報漏洩の面でも問題が無いと言える。

次に、近傍に存在するクライアントを近傍リストが証明しない場合を考える。ここで問題となるのは、集団内のクライアントが近傍リストによる他クライアントの証明を行わないことによる、アクセスの妨害である。

| | |
|---|---|
| **A** | A B |
| **B** | A B C E |
| **C** | B C E F |
| **D** | D |
| **E** | B E F G H |
| **F** | C E F H |
| **G** | E G H |
| **H** | E F G H |

| | スコア | 判定 |
|---|---|---|
| **A** | 2 | 不許可 |
| **B** | 4 | 許可 |
| **C** | 3 | 不許可 |
| **D** | 1 | 不許可 |
| **E** | 6 | 許可 |
| **F** | 4 | 許可 |
| **G** | 3 | 不許可 |
| **H** | 4 | 許可 |

図 3.7: 近傍リストの虚偽報告によるアクセス妨害

図 3.6 の場合、クライアント C に対する証明は、図 3.7 のように自身を除いてクライアント B,F からしか得られない。図 3.5 と同様にアクセス許可には閾値 3 より多い証明が必要だとすると、クライアント C は証明する他クライアント不足によりアクセス

不許可とされ、アクセスが妨害されることになる。

そこで、相互の証明に着目する。2つのクライアントの間で互いに対する証明が異なる場合、片方が嘘の証明を行った、もしくは移動等を行ったクライアントであるため、本来なら正しい証明を選ぶ必要があるが、これを判断するのは難しい。

2つの証明のうち正しい方が全く分からない状況を仮定した時、正しい証明を選択できる確率は50 %である。そこで、近傍リスト受信時に相互で証明が一致しない場合、存在するという証明を無効とする。他クライアントからの証明を1ポイントとした場合、相互で不一致だった証明全てを0.5ポイントの証明と置き換えて近傍リストを再計算する。

| | |
|---|---|
| **A** | A B |
| **B** | A B C E |
| **C** | B C (E) F |
| **D** | D |
| **E** | B (C) E F G H |
| **F** | C E F H |
| **G** | E G H |
| **H** | E F G H |

| | スコア | 判定 |
|---|---|---|
| **A** | 2 | 不許可 |
| **B** | 4 | 許可 |
| **C** | 3.5 | 不許可 |
| **D** | 1 | 不許可 |
| **E** | 5.5 | 許可 |
| **F** | 4 | 許可 |
| **G** | 3 | 不許可 |
| **H** | 4 | 許可 |

図 3.8: 相互証明の不一致の解決

図3.8は、図3.7のクライアント間の証明が一致しない場合を、前述の通り再計算した結果である。クライアントCは近傍にクライアントEを確認できるが、その逆は証明できないため、Cの近傍リストからEを除外する。その後、クライアントCでのEの証明と、これに対応するクライアントEでのCの証明を全て0.5ポイントとして再編成する。この結果、クライアントEが故意にCの証明を行わない場合でも、Cはアクセス許可を得ることができる。

### 3.2.3 悪意ある外部クライアントを考慮に入れた判定

悪意のあるクライアントが外部から近傍リストの虚偽報告によりファイルにアクセスする場合を考える。前小節で、2つのクライアント間で片方からのみ近傍にあるという証明がある場合について、アクセス妨害を回避する解決案を提示した。しかし、この手法を用いることで、逆に集団から離れた場所にいるクライアントにアクセス許可が出されてしまう可能性がある。例えば、本来は近傍に他クライアントが存在しないクライアントが、幾つかのクライアントを近傍リストに追加した虚偽報告を行った場合を考える。

| | | |
|---|---|---|
| **A** | A B (C) (F) (H) | |
| **B** | A B C E | |
| **C** | B C E F (A) | |
| **D** | D | |
| **E** | B C E F G H | |
| **F** | C E F H (A) | |
| **G** | E G H | |
| **H** | E F G H (A) | |

| | スコア | 判定 |
|---|---|---|
| **A** | 3.5 | 許可 |
| **B** | 4 | 許可 |
| **C** | 4.5 | 許可 |
| **D** | 1 | 不許可 |
| **E** | 6 | 許可 |
| **F** | 4.5 | 許可 |
| **G** | 3 | 不許可 |
| **H** | 4.5 | 許可 |

図 3.9: 虚偽報告による間違ったアクセス許可

図 3.9 は集団の外部に存在するクライアント A が、近傍に存在しないはずのクライアント C,F,H を近傍リストに追加した場合に、前述の相互証明の不一致における処理を行った結果である。先と同様、証明数が 3 より多ければアクセス許可が与えられる場合、外部のクライアントであるはずのクライアント A のスコアは 3.5 となり、ファイルアクセスが可能となってしまう。

これを回避するために、前述の証明の不一致処理回数に上限を設ける。本研究では、仮定として、他クライアントの半数以上が正しいリストをサーバに送信するものとしている。この仮定より、正しいリストと見做せる相互証明を少なくとも全証明の半数

だけ確保する必要ものとする。

不一致処理を行う回数の上限を相互証明の個数とする。例えば、図 3.9 でクライアント A は相互証明はクライアント B のみしか存在しないため前述の不一致処理は最大 1 つまで、クライアント B は相互証明を行うクライアントが 3 つあるため不一致処理は最大 3 まで可能とする。この結果、クライアント A は相互証明を 1 つしか持たないため、前述の不一致処理を 3 クライアント分行うことができず、スコアは最大で 2.5、判定もアクセス不許可となる。

### 3.2.4 集団外でアクセス条件を満たすクライアントを考慮した判定

集団の外部にあるクライアントが、ファイルへのアクセス許可を得ることが可能な場合について考える。3.1 節で説明したように、サーバはクライアントから受信した近傍リストによって判定を行う。サーバは、近傍リストによって多くの他クライアントに証明されているクライアントについてアクセス許可を出せば良いが、以下のようなクライアントにもアクセス許可が与えられる可能性がある。

- 集団から僅かに離れた場所にいるクライアント
- 集団とは別の場所に集まっているクライアント

前者は、集団の外部にいながら集団の端に存在する幾つかのクライアントから証明されてしまう場合に、後者は、別の場所で集まったクライアント同士で証明してしまう場合に、それぞれ起りうる。以上から、集団内のクライアントには最大限アクセス許可を出し、外部のクライアントには確実にアクセス不許可を行うアルゴリズムを提案する。

#### 中心クライアントによる証明を用いた判定

一般にアドホックモードによる通信は規格上 50m から 100m 程度において可能である。本研究では、参加者は 1 つの場所、例えば講堂など、に集まるものを想定しており、参加クライアントは全て中心にあるクライアントと電波が届く距離に存在するも

のと見ることができる。よって、先に求めた中心クライアントの電波を受信できる範囲内にいるものを参加者としてアクセス許可を与えるものとする。そこで、中心クライアントを確定する手法について述べる。

まず、確実に外部のクライアントのものと判断できるクライアントを削除する。他のどのクライアントからも全く証明されないクライアントは、外部のクライアントと見做すことができる。外部クライアントによる証明は、3.2.2 で述べた通り嘘の証明である可能性が高いため、以降のアクセス許可の判定から除外する。次に、少なくとも 1 つ以上の他クライアントに証明されてるクライアントについては、スコアの初期値として 0 を設定し、各クライアントについて近傍リストで証明される毎にそのクライアントのスコアを 1 ポイントずつ加算する。そして、各クライアントのスコアの加算結果を比較し、全スコア中最も大きいスコアを求める。

ただし、最大スコアを持つクライアントが必ずしも集団の中心であるとは限らない。集団の形態や、3.2.2 小節で述べたような実際とは異なる証明によって、集団の中心から離れたクライアントが最大スコアを持つことも考えられる。各クライアントの理想スコアは、中心ほどスコアが高く、そこから離れるにつれてスコアが低くなっていく。先に述べたように中心クライアントを確定することは難しいが、中心から離れたクライアントを特定するのはスコアを参照することにより容易であり、これらの離れた場所にあるクライアントのスコアを下げることで、相対的に中心クライアントを見つける。

クライアントの位置によっては、集団の近くにあるだけで集団内には存在しない可能性もあり、これら集団に近いクライアントを区別するために、**信頼度**というパラメータを用意する。信頼度とは、証明を行っているクライアントが集団内に存在している可能性を表す指標とし、他クライアントからの証明が少なかったものに対して低くく設定する。他クライアントからの証明をあまり得られなかったクライアントは信頼度が低い、つまり、集団の中心から離れた位置に存在するクライアントであると言い替えられる。信頼度を下げたクライアントによる証明の重みを減らすことで、信頼度の高い中心付近のクライアントと差別化を行う。

集団の近くにある外部クライアントは、集団の端にあるクライアントにしか証明されない。信頼度の低いクライアントの多くは集団の端にあるクライアントであり、主

にこれらによって存在を証明される外部クライアントは、証明の重みが減ることにより外部クライアント自身のスコアが大幅に低下する。端に位置するクライアントによる証明の低下は集団内のクライアントのスコアにも関わるが、集団内にあるクライアントならば信頼度の高い中心付近のクライアントからの証明があり、スコアの低下は少ない。結果、集団付近にある外部クライアントを削除しやすく、また中心クライアントは最も影響が少なくなり特定しやすくなる。

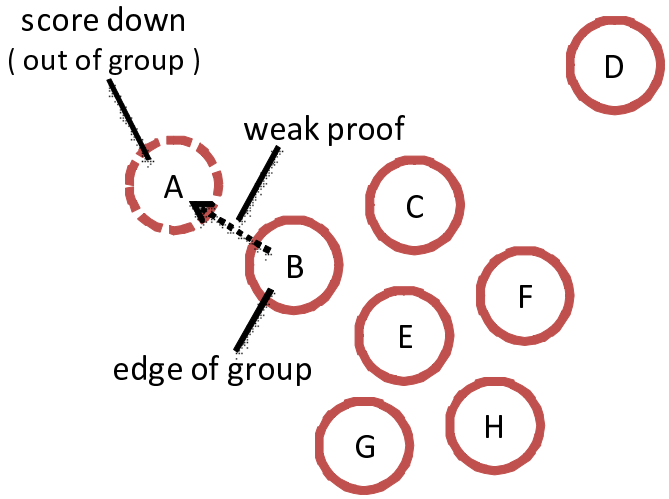



図 3.10: 信頼度と証明の重み付け

図 3.10 では、A が集団近傍にある外部クライアント、B が参加者集団の端に位置するクライアントとした時、クライアント B による証明の重みを減らすことでクライアント A のスコアを相対的に下げる。各クライアントに対してこの処理を行った場合、近傍リストによる証明には自分自身に対してのものも含まれるため、集団の中心にあるクライアントは少なくとも中心から離れたクライアントよりもスコアが減少せず、中心の特定が行いやすくなる。

本研究では、各クライアントのスコアが最大スコアの半分以下であるならば、そのクライアントによる証明の信頼度を下げる。信頼度を下げたクライアントによる他クライアントの証明はその重みを半分にする。

具体的には、本来前述のように近傍リストの証明では、クライアントのスコアに 1 ポイントを加算する所を、信頼度を下げたクライアントの近傍リストの証明では、各

クライアントのスコアに 0.5 ポイントを加算するものに変更する。その後、信頼度の重みを考慮してスコアの再計算を行う。得られたスコアは、集団の中心近くに存在するクライアントほどスコアが大きく、逆に中心から離れるほどスコアが小さくなるため、最大のスコアを持つクライアントが集団の中心であると近似できる。

ここで求めた中心クライアントと相互証明できるものを範囲内と見做す。しかし、この方法では中心クライアントの一存でアクセス許可が出てしまい、仮に中心クライアントに悪意があった場合、もしくは証明情報が既に古いものであった場合に参加者以外のクライアントのアクセスが許可される可能性がある。

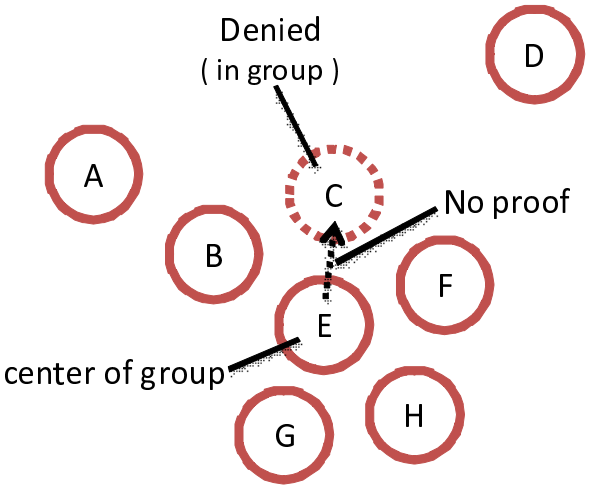



図 3.11: 中心クライアントの誤証明によるアクセス妨害

図 3.11 では、中心クライアントである E が、近傍に存在するはずであるクライアント C について近傍リストで証明を行わなかった場合、本来はアクセス許可が与えられるべき C にアクセス許可が与えられない。このように、中心に選ばれたクライアントが確実に正しい証明を行うという証明が無い状況において、このような単一クライアントによる判定には多くの問題が残る。

### 中心付近の他クライアントによる証明を用いた判定

前述の問題を回避するために、中心クライアントの無線通信圏内に存在するクライアントを、中心クライアント以外の他クライアントによって証明することでアクセス

許可判定を行う手法を提案する。つまり、中心クライアントと相互証明できるクライアントは全て無線通信圏内であり、これらのクライアント群の中に判定対象のクライアントが含まれていることが証明できればよい。

まずは先と同様に、中心クライアントと相互証明できるものに着目する。ただし、これらのクライアントはそのままアクセス許可は得られず、参加者である可能性が高いものとして**メインクライアント**とする。メインクライアントは、近傍リストによる証明においてスコアの最も高かった中心クライアントと相互証明が可能なクライアント、つまり中心付近に存在するクライアントを指す。基本的には、中心クライアントと無線通信が可能な位置にあるクライアントがこのメインクライアントとなり得るが、先に述べたように中心クライアントの近傍リストが正しいものとは限らないため、必ずしも中心付近のクライアントが全てメインクライアントに当てはまるわけではない。そして、このメインクライアントのうち、どれだけ多くと相互証明が行えているかによってアクセス許可について判定する。

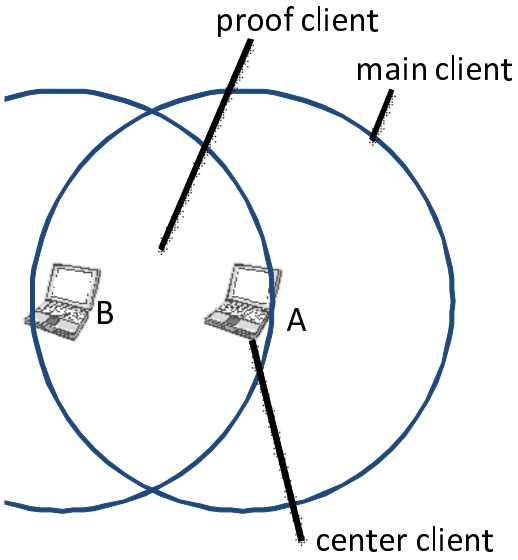



図 3.12: 電波が届く範囲の判定

図 3.12 の円は中心クライアント A の電波が届く範囲を示しており、円の中にあるクライアントはメインクライアントとする。中心クライアントの電波を受信できる最も

離れたクライアントはＢであり、このクライアントがアクセス許可を受けることができればよい。クライアントＢが相互証明を行えるのはＢが中心の円の内部であり、この円とＡを中心とした円の重なる部分がA,Bの双方が相互証明を行える範囲である。つまり、メインクライアントの内この範囲にある各クライアントと相互証明が行える位置に存在するクライアントならば、それは中心クライアントから電波の届く場所、即ち集団内に存在するクライアントと近似できる。

ここで、上図の円が重なる範囲が最小となるのは、クライアント同士が最も離れた時、即ち片方の円の中心がもう片方の円の辺と重なる時であり、この時の面積は１つの円の面積の約１／３に近似できる。円の中にクライアントが均等に存在するとすると、メインクライアントのうち１／３以上と相互証明が行えるクライアントについてアクセス許可を与える。

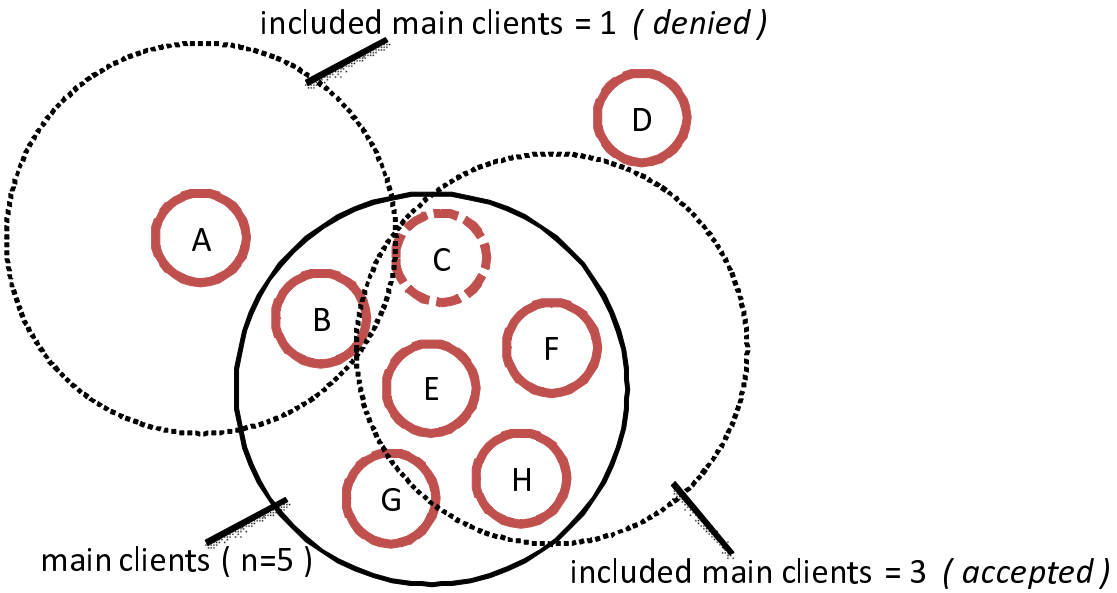



図 3.13: メインクライアントによる許可判定

図3.13は図3.6に対してこの手法を用いた結果である。参加者の集団はクライアントB,C,E,F,G,Hの６つ、クライアントを中心とした実線円と破線円はそれぞれのクライアントがアドホックモードで無線通信が可能な範囲、集団の中心をクライアントＥとする。クライアントＥを中心とした実線円の内部に存在するクライアントがＥと相

互証明可能なメインクライアントとなるが、破線で表したクライアント C については E からの証明が得られないためメインクライアントには含まれない。

クライアント E との相互証明により、メインクライアントは中心である E 自身を含めて B,E,F,G,H の 5 つであるため、アクセス許可を与えるには 2 つ以上のメインクライアントと相互証明を行う必要がある。外部クライアントである A について考えると、相互証明を得られるメインクライアントは B だけであり、アクセス許可は与えられない。集団内にあるクライアント F に関しては、相互証明となるメインクライアントは自身を含めて E,F,H の 3 つとなりアクセス許可を与えられる。

また、中心クライアントである E と相互証明できるにも関わらず、E からの証明が得られないクライアント C では、相互証明が得られるメインクライアントが B,F の 2 つ (C 自身はメインクライアントではなく、E との相互証明は得られていないため) 存在することにより、アクセス許可を得ることができる (図 3.14)。

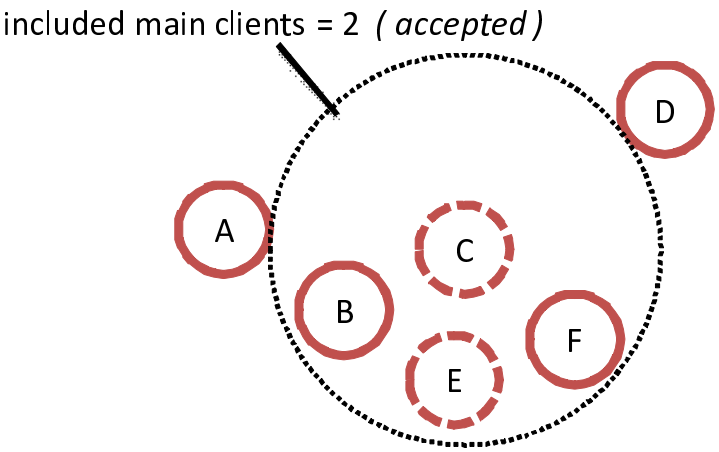



図 3.14: 中心クライアントの証明なしでの許可

この手法を用いることで、情報漏洩の危険度や参加者の分散程度を考慮に入れて閾値を決める必要がなくなり、且つ、アクセス許可判定をあるクライアントの一存で決定するとこを防ぐ。

## 3.3　クライアント情報の管理

サーバでのクライアント情報の管理について説明する。本研究では、クライアントは無線端末を想定しているため、端末が移動することによりアクセスの途中でクライアントが集団から抜ける、もしくは集団に入る、といったことが頻繁に起こり得る。そこで、サーバの保持するクライアントリストは常に最新のものである必要があり、同時に、サーバはクライアントから最新の近傍リストを入手しなければならない。サーバはクライアントと一定時間毎に 3.1 節と同様の通信を行い、その際に 3.2 節で述べたアクセス許可判定の結果から全クライアントに対して更新後のアクセス許可を送信しなければならない。よって、通信負荷が非常に大きくなるため対策が必要である。

まず、全クライアントに対する判定結果の送信は毎回行う必要はない。サーバがクライアントごとのアクセス許可を記憶しておくことで、アクセス許可判定前後でその判定結果に変化があったクライアントに対してのみ、結果を送信すれば良い。

次に、近傍リストの更新間隔について説明する。近傍リストの更新は、それまで存在していたクライアントが消失した場合や、新たなクライアントが加入した場合に、アクセス許可についての最新の判定を行うために必要となる。つまり、クライアントが消失および加入した場合に、そのクライアントとその近傍のクライアントが近傍リストを更新してサーバに送信すればよいことになる。

ここで、集団の中心に近いクライアントほど、近傍に他のクライアントが存在することに着目する。全クライアントの近傍リストを更新する時間間隔がそれぞれ一定とした場合、近傍に多くのクライアントを持つものは、そのクライアントについての証明がより多く得られる。よって、図 3.15 のように、中心近くのクライアントの存在証明は短い間隔で更新されることになる。逆に、集団の端の方にあるクライアントは、その存在を証明する他クライアントが少ないために、証明の更新が行われにくい。

3.2 節で述べた判定方法において、スコアの高いクライアントを中心近くにあるクライアントとし、これらのクライアントについては近傍リストの更新間隔を長くとる。集団の中心ほどクライアントが密集している可能性が高く、これらのクライアントの通信回数を少なくすることで、システム全体の通信コスト低下させることができる。ただし、中心近くのクライアントと見做すためのスコアの閾値はヒューリスティック

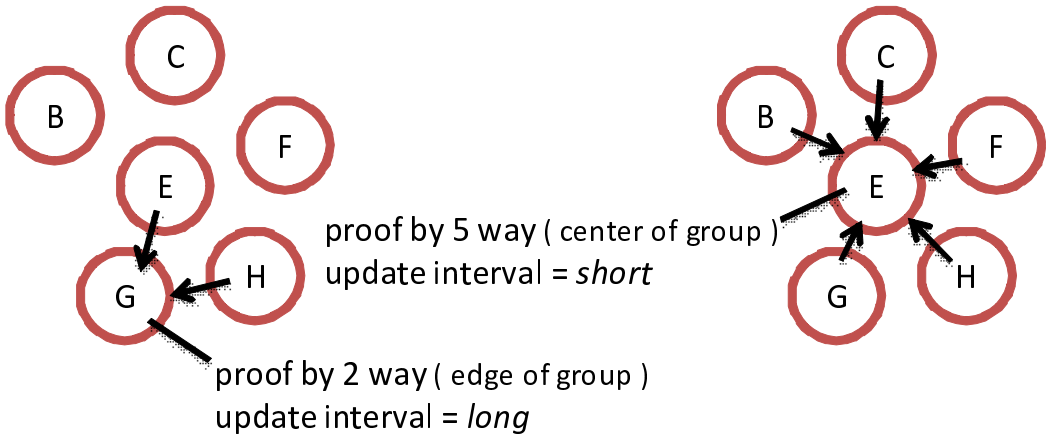



図 3.15: 近傍リストの更新間隔

に決定する必要があり、閾値をいくつに設定すればよいかは今後の課題となる。

上記の更新によって、クライアントから近傍リストが受信できない場合、もしくは再判定によりクライアントが不許可となった場合、そのクライアントにエラーを返してクライアントリストから除外する。

## 3.4 アクセス制御

本節では、クライアントが入手した情報の扱いに対してのアクセス制御について説明する。3.1 節で、他クライアントからの証明により、ファイルへのアクセス許可を得る手法について説明した。この手法により、集団外のクライアントは情報にアクセスすることは難しくなっている。

しかし、例えば、集団内のクライアントが情報を入手したまま集団から離れ、外部に情報を持ち出すことで、情報の漏洩は可能となる。また、集団内のクライアントから外部のクライアントへ情報を転送することで、集団外のクライアントが情報を入手することも考えられる。本提案手法では、ファイルシステムを利用することでアクセス制御を実現する。

### 3.4.1 アクセス許可に対する制御

ファイルアクセス中のクライアントが、サーバからアクセス不許可とされた場合の処理について説明する。集団から外れた等の理由によりサーバからのアクセス許可を失った場合、クライアントの目的ファイルへのアクセスを禁止する必要がある。しかし、情報はクライアントに送信済みであるため、仮にサーバとの通信を終了したとしても、ファイルへのアクセスは可能である。

そこで、クライアント上で独自のファイルシステムを使用し、ファイルシステム上の目的ファイルに対して、アクセスの許可もしくは禁止を行う。クライアントはあらかじめファイルシステムを起動させ保存領域を確保する。その後、サーバとの間で 3.1 章で述べた位置証明を行う。サーバからアクセス許可を得た時、目的ファイルをサーバから受信してファイルシステムの領域上に保存し、クライアントはこの領域上のファイルにアクセスする。

クライアントが集団から離れた場合、サーバからアクセス不許可と判定される。この不許可判定を受信した時、ファイルシステムの領域全体へのアクセスを不許可とする。

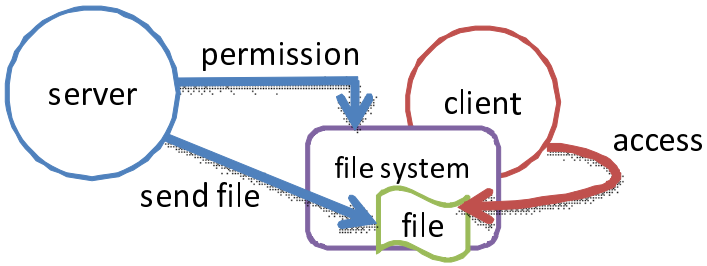



図 3.16: ファイルシステムへのアクセス

図 3.16 は実際にファイルへアクセスする様子を示したものである。アクセス許可を得たクライアントは目的ファイルをサーバから受信し、ファイルシステムの領域上に一時保存する。アクセス許可は一定時間ごとにサーバとの通信によって更新され、不許可の場合はファイルシステムに対してアクセスが禁止される。クライアントが集団の中に入る等、再びアクセス許可が得られた時、ファイルシステム及びその中にあるファイルへのアクセスが可能となる。

本研究において目的ファイルへのアクセスは、全てこの領域上に受信したファイルへのアクセスと同義であるため、複数のファイルへアクセスする場合でもクライアントが集団から離れている間、これらのファイルへのアクセスを禁止することが可能である。また、領域上のファイルはファイルシステムによる操作が可能であるため、ファイルシステムを終了する際、これらアクセスしたファイルを全て削除することができる。

### 3.4.2 アクセス可能な実行ファイルの制限

目的のファイルへアクセス可能な実行ファイルの制限について説明する。クライアントがファイルの内容を外部に持ち出さないよう、目的ファイルの保存やキャッシュを禁止する必要がある。本提案手法では、3.4.1 で述べたファイルシステムに加え、ファイルにアクセスする専用実行ファイルを用いることで実現する。

図 3.17 に手法の略図を示す。図 3.17 中の番号は処理の順番を示しており、以下の本文中の番号に対応する。

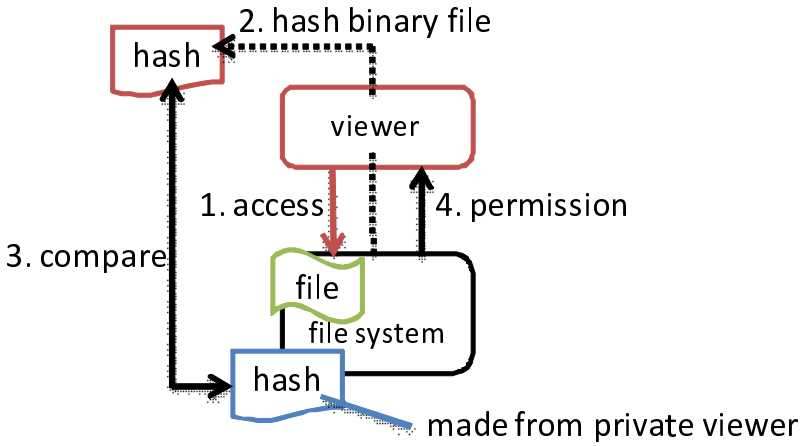



図 3.17: 専用ビューアを用いたファイルへのアクセス

目的ファイルにアクセスする実行ファイルとして、専用ビューアを用意する。この専用ビューアには保存機能を持たせず、ファイルの閲覧や更新のみを行うものとし、目的ファイルを外部に持ち出せないようにする。

目的ファイルにアクセスする際、まず (1) ファイルシステムにアクセスする。ファイ

ルシステムは、アクセスを試みた実行ファイルが前述の専用ビューアであるかを確認する。実行ファイルの確認方法として、(2) アクセスした実行ファイルのバイナリハッシュを求める。ファイルシステムはあらかじめ専用ビューアのバイナリキャッシュを保持しておく。ファイルシステムは、(3) 自身の持つバイナリハッシュと求めたバイナリハッシュとを比較し、(4) 一致したものを専用ビューアと認識する。以上のように、専用ビューアの場合は領域上の目的ファイルへのアクセスを許可し、そうでない場合、その実行ファイルが領域内のファイルにアクセスすることを禁止する。

# 第4章

# 実装

本章では、3 章で述べた提案手法の実装について述べる。本研究では、サーバとクライアントの両方について実装を行い、4.1 節でサーバについての実装を、4.2 節でクライアントについての実装をそれぞれ説明する。

## 4.1 サーバにおける実装

本節ではサーバが行う処理の実装について説明する。本研究では通信プロトコルとして TCP を用い、全ての通信のタイムアウト時に何度か再試行を行う。

### 4.1.1 クライアントリストの作成

3.1 節で述べた近傍クライアントの証明を行うために、クライアントリストを作成し、各クライアントに送信する。上記の実装モデルを図 4.1 に示す。クライアントリスト作成のため、サーバは接続要求のあったクライアントからアドレスを入手する必要がある。

サーバはマルチスレッドを用いて、常に受信パケットを監視するモニタスレッドを用意する。クライアントとの通信は通信スレッドで行う。モニタには libpcap によるパケットキャプチャを用い、通信デバイスに対してインタフェース情報を取得してモニタを開始する。ただし、モニタ対象を限定するため、ソケット通信に用いるポートを残してトラフィックフィルタをかける。これにより、クライアントがサーバにコネクト

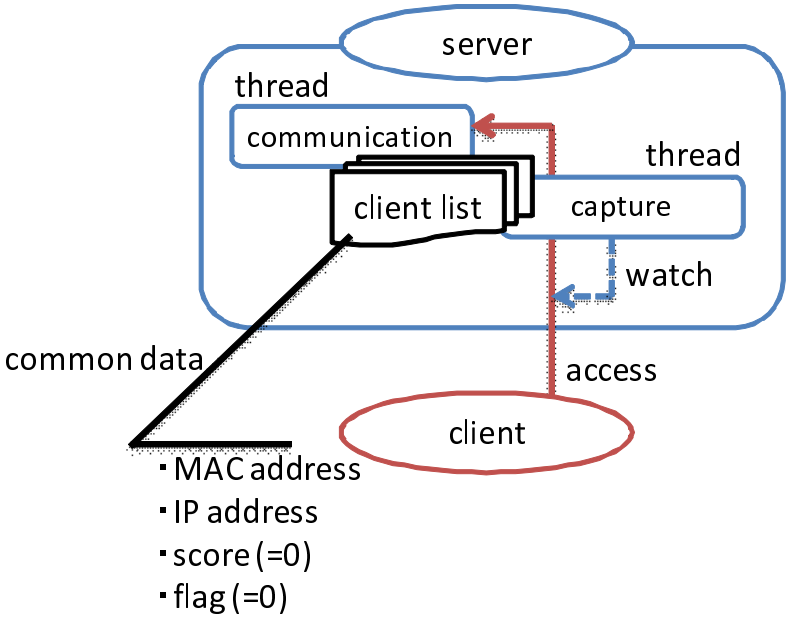



図 4.1: クライアントリストの作成

した際、通信相手であるクライアントのパケットをモニタリングすることができる。

パケットヘッダにはそのパケットを送信した IP アドレスや MAC アドレス、送信先のそれらのアドレスが記述されている。そこで、ヘッダから入手したクライアントの IP アドレスと MAC アドレスを特定し、MAC アドレスをキーとしてクライアントリストを作成する。パケットのモニタ時にそれまでのリストを探索し、入手した MAC アドレスがリストに存在しなければ追加する。ただし、本研究では TCP による通信を行っており、ハンドシェイクによるサーバのパケットもモニタリングすることになるため、送信元がサーバであるパケットは無視するものとする。

クライアントリストにクライアント情報を追加する際、クライアント毎に、MAC アドレスと IP アドレスの他に、他クライアントによる証明を表すスコア、アクセス許可を与えた後に情報を更新する際に用いるフラグを用意し、それぞれを 0 に初期化する。スコアやフラグはアクセス許可の判定で用いるものであり、これらについては後述の 4.1.2 で説明する。

クライアントリストは作成するモニタスレッド、実際にリストを用いる通信スレッドの双方からのアクセスが可能なものとし、モニタスレッドにおける最新のクライア

ントリストが各クライアントとの通信に用いられる。クライアントとの通信が終了した場合、クライアントリストから除外する。

### 4.1.2 アクセス許可の判定

サーバは近傍リストを基に、3.2 で述べたアクセス許可の判定を行う。まず、全クライアント（クライアント数 N）のスコアで N × N の二次元配列を作成する (図 4.2 参照)。配列の横軸をクライアントの行った証明（以下、**証言**）とし、初期値は全て 0 である。

近傍リストを受信した際、そのリストにある各クライアントに対し、配列の証言をそれぞれ 1 とする。ただし、近傍リストの受信は、アクセス許可後のリスト更新にも用いられるため、受信時に証言を一旦初期化する必要がある。他の近傍リストについても同様に行うことで、配列の縦列の合計がそのクライアントに対するスコアとなる。

次に、作成した二次元配列を使って以下のように判定を行う。図 4.2 は 3.2 節で用いたクライアント群のモデルについて、二次元配列を作成したものである。図中の空白部分は 0 を表すものとする。以下では、このモデル配列を用いて説明する。

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| A | 1 | 1 | | | | | | |
| B | 1 | 1 | 1 | | 1 | | | |
| C | | 1 | 1 | | 1 | 1 | | |
| D | | | | 1 | | | | |
| E | | 1 | | | 1 | 1 | 1 | 1 |
| F | | | 1 | | 1 | 1 | | 1 |
| G | | | | | 1 | | 1 | 1 |
| H | | | | | 1 | 1 | 1 | 1 |
| | **2** | **4** | **3** | **1** | **6** | **4** | **3** | **4** |

score

図 4.2: クライアント群のモデル配列

### 証明が得られないクライアントの除外

配列の M 列目のクライアントのスコアが 1 以下の場合、M 行目の全要素、M 列目の全要素を 0 とすることで、そのクライアントを判定から除外する。図 4.3 において、D のクライアントは他のどのクライアントからも証明を受けておらず、スコアは自身の証明による 1 のみであるため、D の証明を配列から取り除くとともに、そのスコアを 0 とし除外する。

### 不一致の証明の処理

配列の (s,t) の要素と (t,s) の要素を比較することで、2 つのクライアントの証明が不一致となっているものを見つけ、その 2 つの要素をどちらも 0.5 とする。図 4.3 中の (C,E) と (E,C) は相互の証明であるが、C から E への証明しか行われていない。このような証明の不一致がある場合、2 つの成分をそれぞれ 0.5 としてスコアを計算する。ここで値を一致させたことにより不一致成分では無くなってしまうが、要素としての証明が 0(存在しない) でも 1(存在する) でもないため、証明が一致していないことが確認できる。

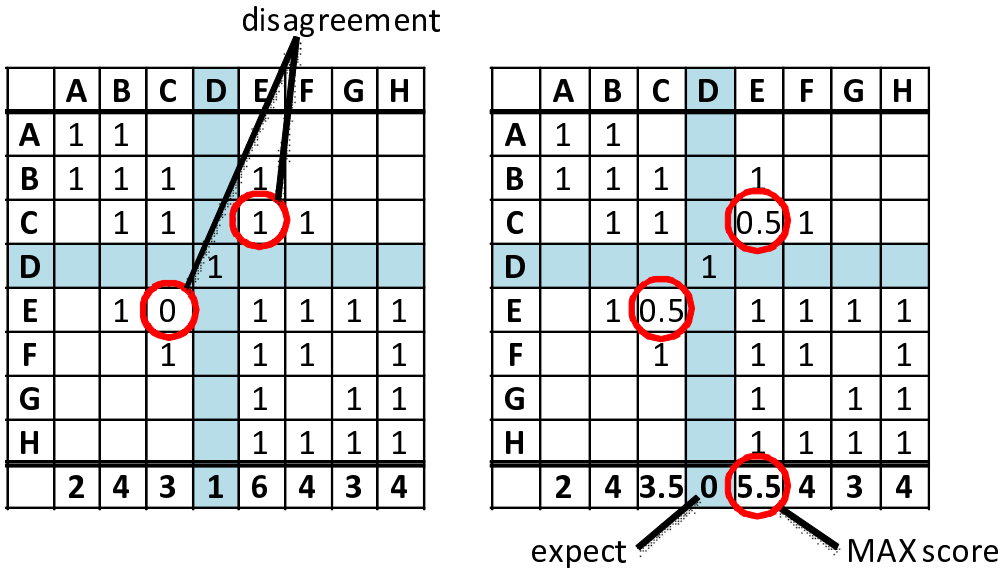



図 4.3: 無証明クライアントの除外と証明の一致

### 信頼度の変更

各列の要素を合計して全クライアントのスコアを求める。全クライアント中最大のスコアと他の各スコアを比較し、M 番目のスコアが最大スコアの半分以下ならば M 行目の全要素を 0.5 倍する。

図 4.4 左より、クライアント A のスコアが最大スコアである E のスコアの半分以下であることが分かる。A の信頼度を下げるために A による証明の重みを半減した配列を作成する (図 4.4 右参照)。ただし、ここで作成する二次元配列は、これまでに作成してきたものとは別のものとする。これは、ここでの作業は中心クライアントを見つけるためであり、相互証明の有無を前の配列と混同しないためである。

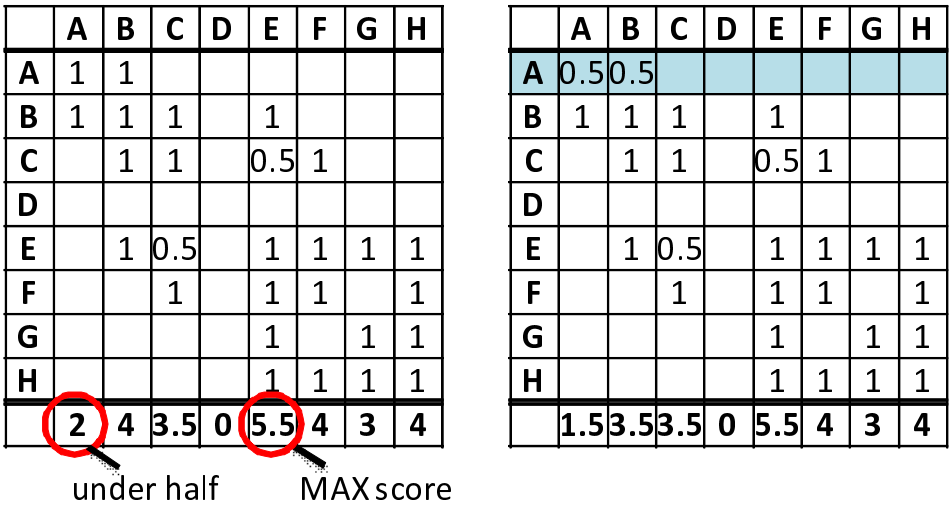
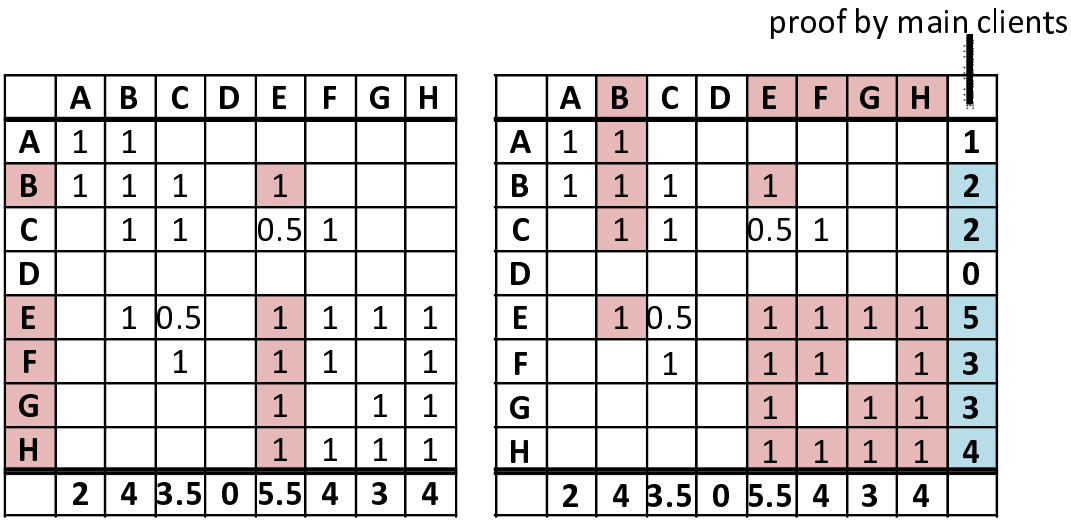



図 4.4: 信頼度の変更

### メインクライアントの決定

全スコアを再計算する。最大のスコアを持つクライアントを中心クライアントとし、この中心クライアントと相互証明が得られるクライアントをメインクライアントとする。サーバはメインクライアントのアドレスをまとめた表を別に作成する。クライアントリストの M 番目がメインクライアントである場合、この表に M を追加する。メインクライアント表は近傍リストの受信の度に更新される。

図 4.5 左より、最大スコアをもつのはクライアント E であることが分かる。クライアント E と相互証明を持つものは、配列の E 列目の要素が 1 となっているものであり、ここではクライアント B,E,F,G,H が該当するため、この 5 つよりメインクライアント表を作成する。




| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| A | 1 | 1 | | | | | | |
| B | 1 | 1 | 1 | | 1 | | | |
| C | | 1 | 1 | | 0.5 | 1 | | |
| D | | | | | | | | |
| E | | 1 | 0.5 | | 1 | 1 | 1 | 1 |
| F | | | 1 | | 1 | 1 | | 1 |
| G | | | | | 1 | | 1 | 1 |
| H | | | | | 1 | 1 | 1 | 1 |
| | 2 | 4 | 3.5 | 0 | 5.5 | 4 | 3 | 4 |

| | A | B | C | D | E | F | G | H | |
|---|---|---|---|---|---|---|---|---|---|
| A | 1 | 1 | | | | | | | 1 |
| B | 1 | 1 | 1 | | 1 | | | | 2 |
| C | | 1 | 1 | | 0.5 | 1 | | | 2 |
| D | | | | | | | | | 0 |
| E | | 1 | 0.5 | | 1 | 1 | 1 | 1 | 5 |
| F | | | 1 | | 1 | 1 | | 1 | 3 |
| G | | | | | 1 | | 1 | 1 | 3 |
| H | | | | | 1 | 1 | 1 | 1 | 4 |
| | 2 | 4 | 3.5 | 0 | 5.5 | 4 | 3 | 4 | |

図 4.5: メインクライアント証明による許可判定

**アクセス許可の判定**

先に作成した二次元配列とメインクライアント表を用いてアクセス許可の判定を行う。N がメインクライアントのとき、配列上で (M,N) 及び (N,M) がともに 1 であるならば、M 行目のクライアントはスコア 1 を得る。ただし、このスコアは前述の他クライアントからの証明によるスコアとは別に用意する必要がある。以降、**証明スコア**と呼ぶ。行列内の全クライアントに対してメインクライアントと相互証明を持つか確認し、得られた証明スコアの合計がメインクライアント表の要素数の 1／3 以上ならば、M 行目のクライアントに対してアクセス許可を与える。

図 4.5 右において、色つきの要素がメインクライアントに相互証明により証明されているものを示す。各クライアントについて、これまでのスコアとは別に、上記の要素の和として証明スコアを求める。今、メインクライアント数は中心クライアント E

の証明スコアと等しく、メインクライアントは5つである。よって、各クライアントは5の1／3以上の証明スコアを所持していればアクセス許可を得られる。図4.5ではクライアントB,C,E,F,G,Hに対してアクセス許可を与える。

クライアントにアクセス許可／不許可の判定をクライアントに送信し、それが初めて許可を得たクライアントの場合、続いて目的ファイルを送信する。不許可であった場合、再びクライアントからのコネクトを待つ。数回の再試行の結果アクセス許可が全く得られない場合、通信を終了しクライアントリストからアクセス許可が得られなかったクライアントを破棄する。

### 4.1.3 ファイルを受信した後のアクセス制御

アクセス許可を得られた場合、3.3で述べたように近傍リストを更新し、これによりアクセス許可の更新を行う。近傍リストの更新は4.1.1で行った処理を再び行うことにより実現する。近傍リストの更新間隔については、次のように中心近くに存在するクライアントの更新間隔を長くすることで、サーバクライアント間の通信トラフィックを低減する。

メインクライアント決定時のスコアを再計算した結果のうち、最大スコアの一定割合を閾値qとし、q以上のスコアを持つクライアントについてクライアントリストにフラグを立てる。クライアントごとに近傍リストの更新をそれぞれ一定の時間間隔tで行う時、クライアントにフラグが立っていれば、そのフラグを下ろし、再び時間間隔tの後に更新を行う。例えば、図4.4左の結果において、最大スコアの2／3以上を中心近くに存在するクライアントとした場合、クライアントE,F,Hにフラグを立てて、近傍クライアントの更新間隔を長くとる。上記の処理により近傍リストが更新され、目的ファイルへのアクセス許可が変化した場合、クライアントにはその判定結果のみを送信する。

## 4.2 クライアントにおける実装

本節では各クライアントが行う処理の実装について説明する。本論文における全てのクライアントは同様の実装をしているものとする。また、サーバと同様に通信のタイムアウト時に再試行を行う。なお、ファイルシステムの起動はサーバとの通信を開始する前に行うものとする。

### 4.2.1 近傍リストの作成

3.1 節で述べた証明を行うために、各クライアントは近傍リストを作成する。クライアントはコネクトしたサーバからクライアントリストを受信する。このリストの各アドレスに対して、アドホックモードでの通信により、PING を発信する。PING による通信は端末から 1 ホップのみ有効であり、対象のクライアントが電波が届く範囲 (1 ホップ圏内) の場合に PING が返る。PING が返ったクラアイントのアドレスをこのリストに追加する。

作成した近傍リストをサーバに送信し、アクセス許可の判定を待つ。許可が得られた場合は、続いてサーバから目的ファイルを受信する。許可が得られなかった場合は、そのまま処理を終了する。

近傍リストはファイル受信後の目的ファイルについて、アクセス許可の更新ごとに再び作成する。アクセス許可更新の際、前述と同様にサーバからクライアントリストを受信して再び近傍リストを作成する。この更新作業では、クライアントリストの受信と同時に近傍リストを破棄する。

### 4.2.2 FUSE によるアクセス制御

本研究では、FUSE（Filesystem in Userspace）というカーネルモジュールを用いてファイルシステムを作成する [8]。今回は fuse-2.7.4 を用いてファイルシステムに実装を行った。

### FUSE について

FUSE カーネルモジュールについて簡単に説明する。FUSE を使用すると、ファイルシステムの内部を理解したりカーネルモジュールによるプログラミングを理解しなくても、ユーザ空間でファイルシステム／フレームワークを開発することができるシステムである。FUSE を用いることでフル機能のファイルシステムを開発することができ、このファイルシステムはシンプルな API ライブラリを備え、非特権ユーザのアクセスが可能でセキュア実装を提供する。

本研究では、FUSE によるファイルシステムは前述の通信プログラムとは別のプログラムとして作成し、このプログラムをあらかじめ起動させておくものとする。

### 許可の変化におけるアクセス制御

3.4.1 で述べたように、クライアントに対するアクセス許可の判定が変化した場合に、ファイルシステム全体に対して処理を行う。通信プログラムとファイルシステムとの間に共有引数を用意する。通信プログラムはサーバから受信した判定結果を共有引数に渡し、ファイルシステムはこの引数を参照することで、ファイルシステムの領域全体へのアクセス許可／不許可を行う。

本研究では、ファイルの保存領域の管理を行うファイルシステムとは別プロセスで、この共有変数による領域へのアクセス制御を行う。アクセス許可更新時にサーバから受信した判定が不許可ならば、領域全体に shmod コマンドによってアクセスを遮断する。ここで用いる判定は共有引数として通信プログラムと共有しているものとする。

ファイルシステム及び通信プログラムの終了時、もしくはアクセス不許可状態が十分な時間続いた場合、ファイルシステムは領域内のファイルを消去した後終了する。

### ファイルへのアクセス制御

目的ファイルへのアクセスを特定の実行ファイルに限定する。これにより、3.4.2 で述べた保存やキャッシュによる情報の持ち出しを防ぐ。本研究では、ファイルにアクセスする実行ファイルを、専用のビューアに限定する。今回は X Window System 用

の PDF ビューアである、xpdf[9](xpdf-3.02) を用いて、このビューアから保存機能を取り除いたものを、ファイルにアクセス可能な専用ビューアとする。

FUSE ファイルシステムはあらかじめ、専用ビューアのプログラムのバイナリハッシュを計算し、その値を保持する。ビューアがアクセスした際、以下の手順でビューアのバイナリハッシュを求める。

1. fuse_get_context()->pid により、ビューアのプロセス ID を得る。
2. /proc/(プロセス ID)/exe に実行ファイルへのリンクがあり、これからプログラムのバイナリを得る。
3. プログラムのバイナリからハッシュを計算する。

FUSE ファイルシステムのファイルオープン関数内でハッシュの計算、及び保持していたハッシュとの比較を行う。一致しなかった場合、つまり他のビューアによるアクセスだった場合、ファイルオープンに対してエラーを返し、そのアクセスを禁止する。

# 第5章

# 実験と考察

本章では、本研究の動作実験及びその評価を行う。実験はサーバとしてのデスクトップと、クライアントとしての無線端末を複数にそれぞれ実装して、実機により動作を行った。サーバ、クラアイント共に Linux 上で実装した。

## 5.1 実験環境

今回実験に用いた環境を以下に示す。クライアント端末は全て同じ環境で実験を行った。

### サーバ

- OS: Fedora Core 8
- kernel: 2.6.24.5-85.fc8
- CPU: AMD Athlon(tm) 64 Processor 3500+

### クライアント

- OS: Ubuntu
- kernel: 2.6.24-19-generic
- CPU: Celeron(R) 2.00GHz

- 無線 LAN: PCi GU-US54GXS
- ファイルシステム: fuse-2.7.4
- ビューア: xpdf-3.02

## 5.2 近傍リストによるアクセス許可判定

近傍リストを用いた相互証明によって実際にアクセス判定を行うプロセスについて検証した。今回は 5 台のクライアントを図 5.1 のように配置して実験を行った。

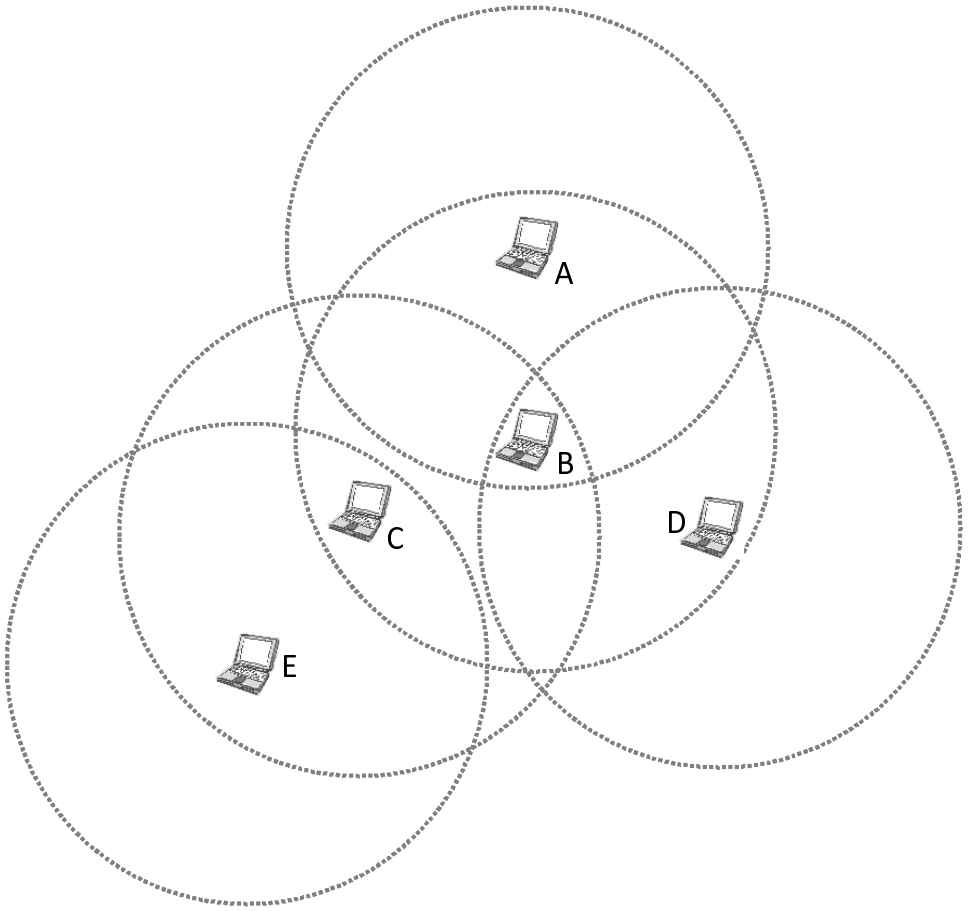



図 5.1: 実験クライアントの配置

図 5.1 中のクライアント A~D を参加者の集団、クライアント E は集団から離れた外部のクライアントである。また、クライアント B は集団の中心に位置しており、他の

3つのクライアントとは無線通信が可能である。クライアント A,C,D については、それぞれ十分な距離だけ離してあり、これらが互いに直接通信することはできない。クライアント E は集団の外部にあるがクライアント C との通信は可能な位置にある。ただし、いずれのクライアントもサーバとの通信は常に可能である。図 5.2 に各クライアントのアドレスをまとめた。本節では以降、クライアント A∼E は図 5.1 および図 5.2 中のクライアントを指すものとする。

| | MACアドレス | IPアドレス |
|---|---|---|
| A | 000A797BEACC | 192.168.11.205 |
| B | 000A7977CAA9 | 192.168.11.206 |
| C | 000A797BEAE8 | 192.168.11.208 |
| D | 000A79779F27 | 192.168.11.210 |
| E | 000A7977CABA | 192.168.11.209 |

図 5.2: 実験クライアントのアドレス

### 5.2.1 近傍リストの更新によるアクセス許可

参加者であるクライアント A∼D がサーバに対してアクセス許可を求めた場合について実際に近傍リスト用いて判定を行った。ただし、クライアントは A,B,C,D の順にサーバへアクセス要求を出す。また、全ての結果を本論文に掲載するのは難しいため、本節ではクライアント D がアクセス判定を得るまでについて述べる。図 5.3 はクライアント D が初めてサーバに接続した際のサーバが行った判定とその過程である。

まず、サーバはクライアント D から近傍リストを受信する。クライアント A∼C が既にサーバへアクセス済みであり、サーバが D に送信するクライアントリストには A∼D のアドレスが記載されてるため、サーバがクライアント D から受信する近傍リスト (Recieve Neighbor_list) ではクライアント B,D が近傍にあることが証明されている。クライアント A,C については D の電波範囲外であるため、存在を証明することはできない。

次に、近傍リストによる証明を二次元配列に反映する (list before process)。今回この

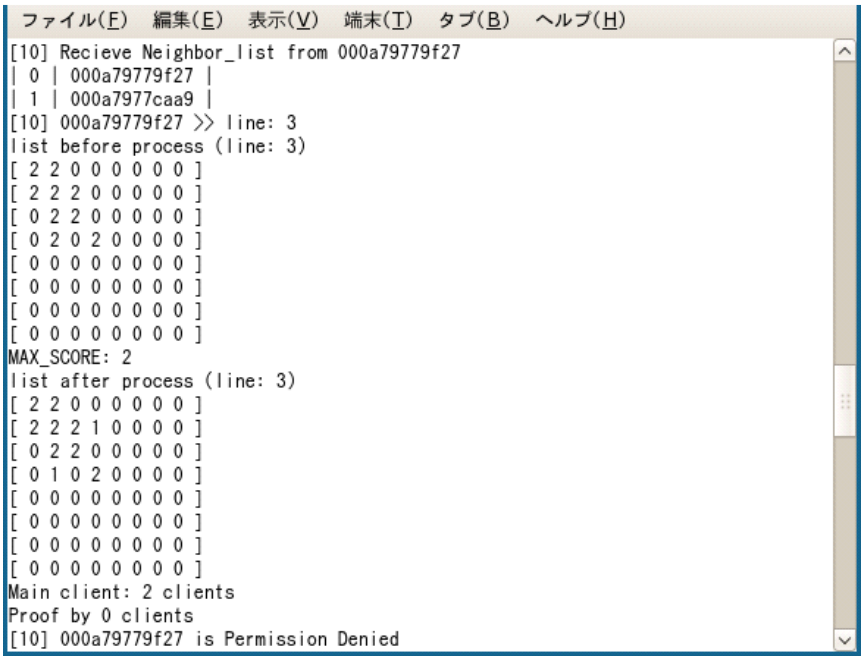



図 5.3: クライアント D のアクセス要求に対する許可

二次元配列では行、列ともに A,B,C,D の順に並んでおり、クライアント D の近傍リストによる証言は 3 行目に示される。ここで、全クライアントの最大スコア (MAX_SCORE) が 2 となっているが、これはクライアント C の近傍リストがまだ結果に反映されていないためである。相互証明である (3,1) と (1,3) が不一致であるため、相互証明の不一致の解決 (3.2.2 参照) を行ったものが list after process である。列ごとに得られた証明の合計がスコアとなるため、最大スコアは 1 列目のクライアント B であることが分かる (スコアは 7)。

最大スコアを持つクライアント B と相互証明を持つものがメインクライアント (Main client) となるはずであるが、ここでは 2 つしかメインクライアントとなっていない。これは先に最大スコアを求める際に他の近傍リストが反映されていないために齟齬が生じたのと同様、他のクライアントとの通信を行っているスレッドの処理によって値が書き換えられたためである。しかし、いずれの場合においてもクライアント D を証明するクライアントは存在せず、D 自身もメインクライアントではないため、クライア

ントDの証明数(Ploof by)は0となりアクセスは不許可(Permission Denied)である。

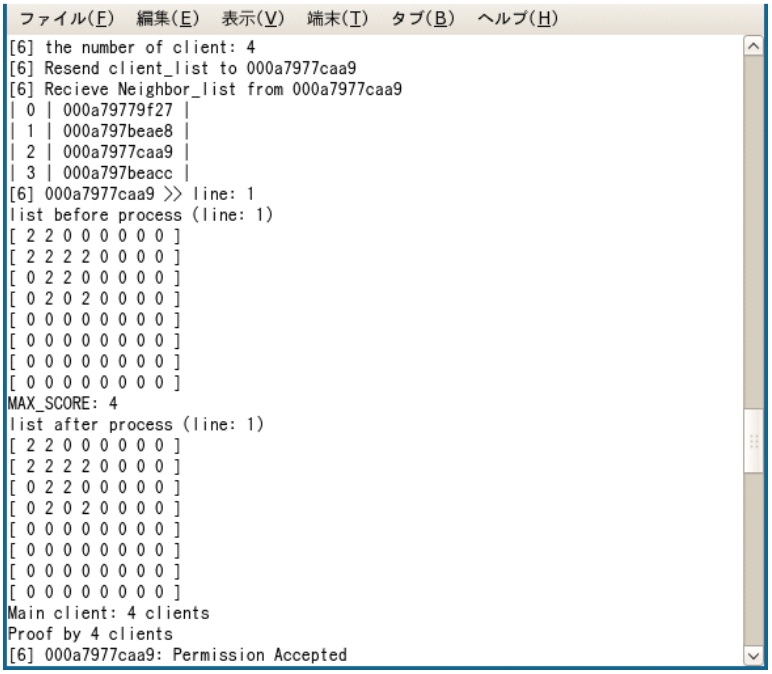



図 5.4: クライアントDのアクセス許可の再判定

図5.4は図5.3の結果を得た後に、クライアントBが近傍リストを更新した際の結果である。クライアントDがアクセス要求を出した後であるため、Bの近傍リストはクライアントD(000a79779f27)を含むA∼Dの全クライアント証明する。二次元配列に反映した際、図5.3では証明が得られなかった(1,3)が証明されたことにより相互証明おける不一致が解消され、list after processの二次元配列ではlist brfore processの配列を維持することができる。最大スコアのクライアントBと相互証明をもつクライアントはA∼Dのクライアント全てとなるためメインクライアントは合計4つであり、メインクライアント全体の1／3と証明を得たことによりクライアントDはアクセス許可を得られる。

図5.5は上記の処理の間のクライアント側の動作結果である。図の上部が最初のアクセス判定、下部が5.4の後に再びアクセス判定を行った際の結果である。上部ではサーバから受信したクライアントリスト(Client_list)について近傍リスト(Neighbor_list)を


図 5.5: クライアント D の動作結果

返信しているがアクセス判定は不許可 (NOT ACCEPTED) となっているのに対し、下部では全く同じ動作をしてもアクセス判定の結果が許可 (ACCEPTED) となっていることが分かる。これは一度アクセス要求を出したことで他のクライアントの近傍リストが更新され、D に対する証明が得られたことによる結果だと言える。

### 5.2.2 外部クライアントに対するアクセス許可

参加者であるクライアント A~D の他に集団から離れた位置にあるクライアント E がサーバに対してアクセス許可を求めた場合について実際に近傍リスト用いて判定を行った。図 5.6 はクライアント E がサーバに接続した際、サーバが行った判定とその過程である。ただし、クライアント E がアクセス要求を行う前に、A~D のはサーバへアクセス要求を既に出してあるものとし、二次元配列の 0~3 番目の要素については 5.2.1 と同じ結果である。

既に全クライアントのアドレスはクライアントリストで把握されているため、クライアント E は近傍リストによって C,E の存在を証明する。外部クライアント E は C,E から証明を得ることができるため、相互証明を得る数はクライアント D と同様に 2 つ

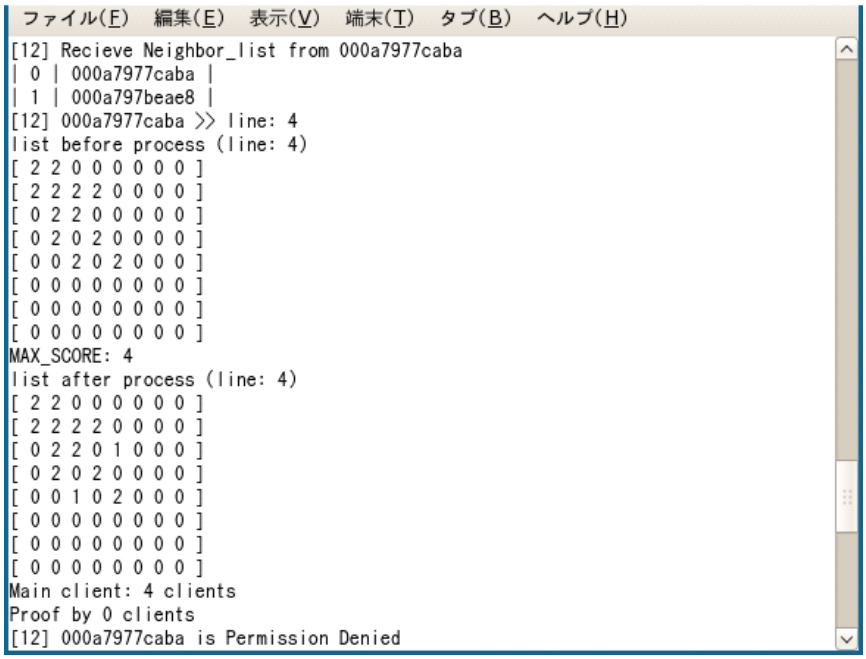



図 5.6: 外部クライアント E のアクセス許可判定

である。しかし、下部の配列より中心クライアントは B(1 列目)、メインクライアントになり得るクライアントは B と相互証明を持つ A~D の 4 つであり、クライアント E はこのうちクライアント C としか相互証明を行い得ない。よって、外部クライアントである E が如何に嘘の近傍リストを作成したとしても、全メインクライアントの 1／3 の相互証明を得ることができず、アクセス許可が得られない。

# 第6章

# まとめ

本論文では、会議など1箇所に集まった無線端末にのみ資料等を閲覧させることを目的として、これらのクライアントにサーバ上のファイルへのアクセスを許可する手法を提案した。同時に、アクセスを許可した後にクライアントが集団から離れた場合に、アクセスを終了させるアクセス管理についても実現した。

提案手法として、近傍クライアントの存在証明を、参加者である他クライアントと相互に行うことによる相対的な位置証明を提案した。近傍クライアントの証明にはアドホックモードによるPINGを使って近傍リストを作成し、このリストの情報をサーバで統合してアクセス許可の判定を行う。

また、アクセス管理についてはFUSEによるファイルシステムを利用して、受信したファイルをファイルシステムにより管理する。一定時間間隔で常時行うアクセス判定によりアクセス不許可となった場合、ファイルの保存領域へのアクセスを禁止し、ファイルにアクセス可能なプログラムを保存機能のないビューアに限定することで、情報の外部への持ち出しを防止する。

実験の結果から、サーバは各クライアントの近傍リストを統合することによって、参加者であるクライアントに対してアクセス許可を出すことを確認した。ここで、本来許可を得られるクライアントがアクセス不許可だった場合、他の近傍リストを随時更新することでアクセス許可を与えることができることを確認した。また、外部からアクセスを試みたクライアントに対して、他のクライアントの証明を利用することでアクセスを不許可とすることも確認した。しかし、今回の実験ではクライアント間、

サーバクライアント間の通信量が増えた結果、プログラムの動作が不安定になる場面が多く見られた。今後の課題として、クライアントが増えた場合にも通信量をできるだけ低減し、安定した動作結果を得られる手法を考えたい。

# 参考文献

[1] 佐々木直志: 近傍端末間の相互保証による位置情報改ざん検知機構の設計と評価, 電気通信大学大学院修士論文 (2005)

[2] 石原孝通, 西尾信彦: GPS と無線基地局検出ツールを排他利用する位置情報システム, 情報処理学会研究報告書 第 6 回ユビキタスコンピューティング研究発表会 (2004)

[3] 牛島直記, 大谷誠, 渡辺健次: Opengate との連携による無線 LAN 利用者の位置推定システムの構築, 情報処理学会研究報告書 分散システムインターネット運用,Vol2007,No.24,pp.56-64(2004)

[4] 伊藤誠悟, 吉田廣志, 川口信夫: 無線 LAN を用いた広域位置情報システム構築に関する検討, 情報処理学会論文誌,Vol47,No.12,pp.3124-3136(2006)

[5] 朝長康介, 太田昌孝, 荒木啓二郎: エニキャストを用いた位置依存サービス, 情報処理学会論文誌,Vol49,No.5,pp.1713-1726 (2008)

[6] 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 (2005)

[7] アクセス制御リスト ACL とは?
http://www.atmarkit.co.jp/fwin2k/win2ktips/700whatisacl/whatisacl.html

[8] FUSE による独自ファイルシステムの開発
http://www.ibm.com/developerworks/jp/linux/library/l-fuse/index.html

[9] Xpdf

http://www.foolabs.com/xpdf/

# 謝辞

本研究のために、多大な御尽力を頂き、御指導を賜った名古屋工業大学の松尾啓志教授、齋藤彰一准教授、津邑公暁准教授、松井俊浩助教に深く感謝いたします。

また、本研究の際に多くの助言、協力をして頂いた松尾研究室の方々に深く感謝いたします。