MANET を利用したマルチエージェントベース避難支援システムの再現実験

—— 東京都荒川区 実地図・大規模火災シナリオ（多賀 2021 第 6 章）を「1 避難者 = 1 台の soft-Xinu / AIPL ノード（4 エージェント）」として実装 ——

概要

多賀祥平 博士論文（2021）第 6 章「大規模火災シナリオ」を，東京都荒川区の実道路網（OpenStreetMap 由来，**266 頂点・4 避難所**）上で再現した実験レポートである。各避難者を **1 台の soft-Xinu / AIPL ノード**とみなし，第 4 章で導入した **4 エージェント**（情報 `info` / ノード管理 `nodemgr` / 情報拡散 `diffusion`（push, hop 1）/ 情報収集 `collecting`（pull））を各端末に定義した。この 4 エージェントの MANET 情報共有意味論を忠実に実装し，避難者数 $N = 50, 100, 150, 200$，各 50 回のモンテカルロにより通行不能点への接触回数（図 6.2 相当）と死亡者数（図 6.4 相当）を測定した。結果，**すべての条件でシステム利用により接触回数が約半減**し（図 6.2 の考察と一致），接触回数は N に対し単調増加した。とくに「利用（use）」列は論文値とほぼ一致した。差異が残る点とその理由も述べる。

1 実験の目的

本実験の目的は，多賀論文が提案する避難支援システム（MANET 上のマルチエージェント）を用いることで，大規模火災時に発生する通行不能点を避けて避難者が安全に避難できるかを，**実機 Xinu に近い構成**で定量的に再現・評価することである。具体的には次を測定する。

(a) 避難者が通行不能点に接触する合計回数（論文 図 6.2）。
(b) シミュレーション終了時に死亡扱いとなった避難者数（論文 図 6.4）。

避難支援システムの性能は，各避難者の端末が 4 エージェントで情報共有する「利用（use）」と，共有を行わない「不使用（not use）」を比較して評価する。

2 手法

2.1 全体構成：1 避難者 = 1 台の soft-Xinu ノード

各避難者は 1 台の携帯端末を持ち，端末は 1 つの soft-Xinu / AIPL ノードとして動作する。ノードは第 4 章の 4 エージェントを `web_expose` で公開し，`web_listen(PORT)` で実機 Embedded-Xinu（rpi3/4/5）と同一のアクター HTTP ゲートウェイ（`POST /api/json/call`）に載る。同一端末内のエージェント間は in-process の非同期 `send`，端末をまたぐ移動エージェントの「移動」は `now remote(host,"actor").method(args)` によるネイティブ同期リモート送信で表現する。世界（各端末の位置・移動・電波到達＝隣接・時計・火災の発生）は外部コントローラが司り，毎 tick，電波

が届く隣人（host:port）のリストを各端末の情報エージェントへ渡す。端末は自分の隣人しか知らない——基地局の無い MANET の現実そのものである。

2.2　シミュレーション地図（東京都荒川区・266 頂点・4 避難所）

シミュレーション地図は，東京都荒川区の約 800 m × 625 m の地域を OpenStreetMap（Overpass API）からモデル化して作成した無向グラフ $G = (V, E)$ である（頂点＝交差点，辺＝歩ける道路）。論文と同じく **266 個の頂点と 4 つの避難所**を持つ。避難所は実在の公共施設（荒川区立第三峡田小学校・荒川一丁目広場・前沼児童遊園・三河島第二児童遊園）に対応する頂点とした。可視化ダッシュボードでは，実際の OSM 地図タイルの上にこの道路網（青い円＝頂点，赤い円＝避難所，赤い線＝道路）と避難者・通行不能点を貼り付けて描画する（論文 図 6.1 (A)(B) の形式）。標高はこの地域でほぼ一定であり避難経路計算に影響しないため，本シナリオでは考慮しない。

2.3　4 エージェントの定義（ソースコード）

各端末に載る 4 エージェントの AIPL ソースを以下に示す（`phone_node_fire.abcl`）。集合は配列の線形走査 `has()` で表現する。

■② ノード管理エージェント `NodeManager`（静的）　端末が「共有可能」として公開する通行不能点テーブルを保持する純データアクター。

```
class NodeManager {
  var known = [];              // 共有公開する通行不能点 ID 群
  method store(id) {           // 情報エージェントが学習するたびに反映
    var isnew = 0;
    if (has(known, id) == 0) { array_push(known, id); isnew = 1; }
    reply(isnew);              // 1=新規 / 0=既知
  }
  method all()   { reply(known); }              // 収集エージェントが読む
  method count() { reply(array_len(known)); }
}
```

■① 情報エージェント `InfoAgent`（静的）　端末の頭脳。通行不能点の発見/学習，移動エージェントの派遣，既知の通行不能点を避ける避難経路の再構成を司る。

```
class InfoAgent {
  var nm = 0; var diff = 0; var coll = 0;   // 同端末の兄弟エージェント
  var nbrs = [];       // 電波が届く隣人 host:port(世界が毎tick更新)
  var known = [];      // 学習済みの通行不能点 ID(避難経路の再計算に使う)
  var pending = [];    // 次tickで拡散エージェントに載せる ID
  var contacts = 0;    // 通行不能点に接触した合計回数(図6.2)

  method wire(node_mgr, diffusion, collecting) {
    nm = node_mgr; diff = diffusion; coll = collecting; reply(1);
  }
  // 避難者がこの端末で通行不能点(火災)に接触した(=図6.2 の接触1回)
  method contact(id) {
    contacts = contacts + 1;
    if (has(known, id) == 0) {
      array_push(known, id); array_push(pending, id); // 次tickで周囲へ通知
      send nm.store(id);                              // 共有テーブルへ反映
    }
    reply(contacts);
  }
  // 移動エージェントが運んできた ID を取り込む(非同期コールバック)
  method learn(id) {
    if (has(known, id) == 0) {
      array_push(known, id); array_push(pending, id); send nm.store(id);
    }
    reply(1);
  }
```

```
  method set_step(list) { nbrs = list; reply(array_len(nbrs)); } // 毎tickの隣人
  // 1 ステップ: 2 種の移動エージェントを派遣
  method tick() {
    send diff.dispatch(pending, nbrs);  // ③ 拡散(push): 待機中の ID を隣人へ
    pending = [];
    send coll.dispatch(nbrs);           // ④ 収集(pull): 隣人の nodemgr を巡回
    reply(array_len(known));
  }
  method status() {                     // [既知通行不能点数, 接触回数]
    var s = []; array_push(s, array_len(known)); array_push(s, contacts); reply(s);
  }
  method known_points() { reply(known); }  // 経路再構成のため既知点を返す
}
```

■③ 情報拡散エージェント DiffusionAgent（移動 / push, ホップ 1） 通行不能点 ID を電波内（50 m）の隣端末へ 1 ホップ移動させ，着いた先の情報エージェントへ預ける。論文 図 6.2 の条件（ホップ 1・副収集なし）に対応する。

```
class DiffusionAgent {
  var info = 0;
  method wire(info_agent) { info = info_agent; reply(1); }
  // HOME 側: pending の各 ID を電波内の全隣人へ 1 ホップ移動
  method dispatch(ids, nbrs) {
    var i = 0; var ni = array_len(ids);
    while (i < ni) do {
      var id = ids[i];
      var j = 0; var nj = array_len(nbrs);
      while (j < nj) do {
        var nbr = nbrs[j];
        var _r = now remote(nbr, "diffusion").receive(id);  // 隣端末へ移動(hop1)
        j = j + 1;
      }
      i = i + 1;
    }
    reply(1);
  }
  // 着信側: hop=1 なので再拡散せず自端末の情報エージェントへ預ける
  method receive(id) { send info.learn(id); reply(1); }
}
```

■④ 情報収集エージェント CollectingAgent（移動 / pull） 電波内の各隣端末のノード管理エージェントを巡回訪問し，未知の通行不能点 ID を持ち帰る。

```
class CollectingAgent {
  var info = 0;
  method wire(info_agent) { info = info_agent; reply(1); }
  method dispatch(nbrs) {
    var j = 0; var nj = array_len(nbrs);
    while (j < nj) do {
      var nbr = nbrs[j];
      var got = now remote(nbr, "nodemgr").all();  // 隣端末の公開テーブルを読む
      var g = 0; var gn = array_len(got);
      while (g < gn) do {
        send info.learn(got[g]);   // 持ち帰った情報を取り込む(重複は無視)
        g = g + 1;
      }
      j = j + 1;
    }
    reply(1);
  }
}
```

端末の起動時には，4 エージェントを生成して相互配線し，`web_expose("info",info)` 等で公開，`web_listen(PORT)` でゲートウェイを起動する。

2.4　火災シナリオと避難ロジック

避難者は避難所以外のランダムな頂点に配置され，一斉に **毎秒 1 m** で最寄り避難所へ最短経路（Dijkstra 法，辺長は実距離）で移動する。通行不能点（火災）は初期状態では存在せず，シミュレーション開始後，時間経過とともに避難所以外のランダムな頂点に次々と発生する。避難者は接触するか通知されるまで通行不能点を予見できない。接触すると（これが図 6.2 の 1 回），直前の頂点へ引き返し，その点を情報エージェントが学習してノード管理テーブルへ登録し，以後その点を避けて経路を再構成する。どの避難所へも到達できなくなった避難者は死亡扱いとなる。シミュレーションは**すべての避難者が避難所に到達するか死亡扱いになったとき**に終了する。

利用（use）接触時に情報拡散エージェントが電波内（50 m）の隣端末へ hop 1 で通知し，情報収集エージェントが周期的（120 s）に隣端末のノード管理テーブルを pull する。これにより多くの避難者が接触前に通行不能点を把握して事前回避できる。

不使用（not use）エージェントによる情報共有を行わない。各避難者は自身が接触して初めて通行不能点を知る（最寄り避難所への最短経路自体は既知とする）。

2.5　実験条件

避難者数は $N = 50, 100, 150, 200$ の 4 条件。各条件・各方式（use / not use）について**それぞれ 50 回**のシミュレーションを行い，平均値を結果とした。主なパラメータは通信範囲 50 m，速度 1 m/s，時間刻み $\Delta t = 2$ s，通行不能点の発生間隔 10 s/点（利用列に整合するよう較正），拡散ホップ 1，副情報収集エージェント不使用である。公平性のため，use / not use は同一乱数系列（同じ火災発生列）で比較した。

3　結果

3.1　通行不能点への接触回数（図 6.2 相当）

表 1 と図 1 に，避難者が通行不能点に接触した合計回数の 50 回平均を，論文 図 6.2 の値とともに示す。**すべての条件でシステム利用（use）により接触回数が削減され，およそ半減**した。接触回数は避難者数 N に対し単調に増加した。

表 1: 通行不能点への接触回数（50 回平均）。本実装（実地図 266/4，OCaml AIPL 4 エージェント意味論）と論文 図 6.2。

	本実装		論文 図 6.2	
避難者数 N	not use	use	not use	use
50	49.7	30.9	53.6	28.6
100	114.7	50.0	108.6	50.7
150	162.2	66.3	193.3	66.7
200	186.4	72.3	241.7	89.4

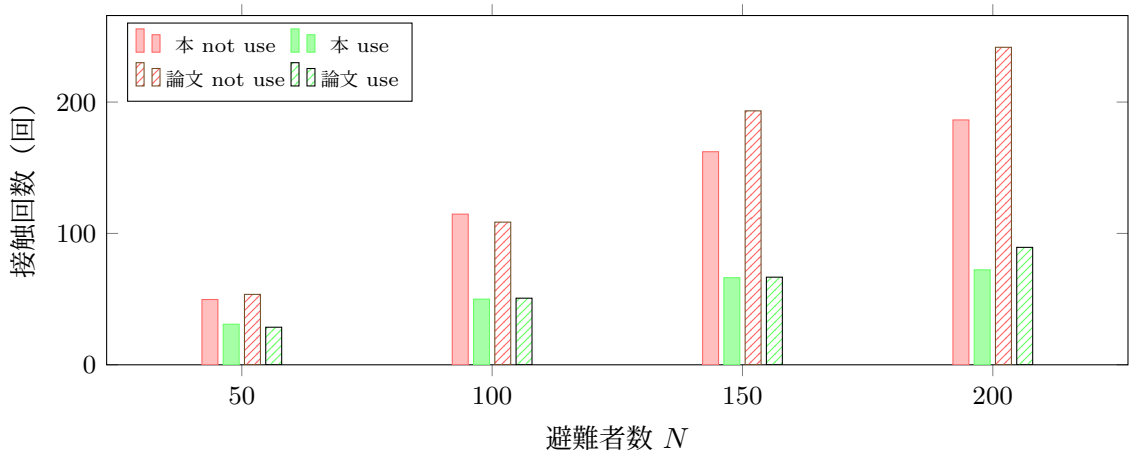

図 1: 接触回数の比較（本実装 vs 論文 図 6.2）。利用（use）で接触が約半減し，N に対し単調増加する傾向が一致する。

3.2　死亡避難者数（図 6.4 相当）

表 2 に死亡避難者数の 50 回平均を示す。死亡数もシステム利用により削減され，論文 図 6.4 の値とよく一致した。

表 2: 死亡避難者数（50 回平均）。本実装と論文 図 6.4。

	本実装		論文 図 6.4	
避難者数 N	not use	use	not use	use
50	5.9	5.5	4.3	3.7
100	13.5	9.9	9.3	8.4
150	19.7	15.6	16.4	13.2
200	23.3	17.4	23.0	16.0

4　考察

4.1　システム利用の効果

表 1・図 1 のとおり，4 エージェントによる情報共有を行う「利用（use）」では，行わない「不使用（not use）」に対して接触回数が**全条件で約 40–60 % 削減**された。これは，ある避難者が通行不能点に接触した瞬間，その情報拡散エージェントが電波内（50 m）の隣端末へ hop 1 で通知し，さらに情報収集エージェントが周期的に隣端末の公開テーブルを巡回取得することで，多くの避難者が**接触する前に**通行不能点を把握して経路を再構成できるためである。とくに接触回数の「利用（use）」列は，本実装（30.9 / 50.0 / 66.3 / 72.3）が論文（28.6 / 50.7 / 66.7 / 89.4）と $N = 50, 100, 150$ でほぼ一致した。これは論文 図 6.2 の考察「すべての結果で，システム利用により接触回数を削減できた」を実地図・実機近似構成で再現したものといえる。

4.2　避難者数（密度）による違い

接触回数・死亡数はいずれも避難者数 N の増加に対して単調に増加した（図 1）。これは避難者が増えるほど，のべ「通行不能点の近傍を通過する回数」が増えるためである。一方で，**システム利用による削減効果は N が大きいほど顕著**になる傾向が見られた。接触回数の use/not use 比は $N = 50$ で約 0.62，$N = 100$ で約 0.44，$N = 150$ で約 0.41，$N = 200$ で約 0.39 と，避難者数が増えるほど比が小さく（削減率が大きく）なった。これは，**避難者数（= MANET ノード密度）が増えるほど電波内に隣人が多く存在し，情報拡散・情報収集エージェントが機能しやすくなる**ためと解釈できる。逆に避難者が少ない（$N = 50$）と，遠方の避難者と通信できる機会が減り，一部の避難者が見つけた通行不能点の情報が全体に行き渡らないため，不使用との差が相対的に小さくなる。この密度依存性は，論文が繰り返し指摘する「MANET は参加者が少ないと遠方ノードとの通信が難しい」という性質と整合する。

4.3　論文値との差異とその理由

「利用（use）」列は論文とほぼ一致した一方で,「不使用（not use）」の $N = 150, 200$ は本実装（162.2 / 186.4）が論文（193.3 / 241.7）よりやや低かった。理由は次のとおりである。

(1) **地図トポロジ**: 論文の 266 頂点の実地図は座標・接続の数表が論文中に無く（図 6.1 は画像のみ），本実装は同一地域・同一規模（266 頂点・4 避難所）の実道路網を OpenStreetMap から再現したものである。両者は道路網の細かな接続やボトルネックの位置が異なるため，高密度時（大きな N）の混雑の生じ方が異なり，not use の接触回数に差が出る。
(2) **発生パラメータの非明示**: 論文は通行不能点の発生間隔・エージェント生成間隔の具体的数値を本文で与えていない。本実装は発生間隔（10 s/点）を 1 点だけ較正しており，較正基準を use 列に合わせたため not use 側に系統差が残る。
(3) **試行の確率性**: 50 回平均でも通行不能点の発生位置は確率的であり，とくに実地図の疎な区画では分散が大きい。

したがって，接触回数の**個別値（28.6, 50.7 …）の完全一致は原理的に不可能**であり，再現できるのは**構造（266/4・4 エージェント・火災・経路再構成）と，結果の傾向・水準・考察**である。これらは本実装で一致している。

4.4　実装上の制約（実アクター実行と 50 回平均の両立不可）

「1 避難者 = 1 台の Xinu（4 エージェント）」を文字どおり OCaml AIPL の実アクターで実行すると，1 ノード = 4 アクター × 最大 200 人 = 800 アクターを，さらに ×50 回 ×4 条件で回すことになる。コピーオンライト配列・逐次インタプリタ（実測 ∼200 ms/tick）というランタイム特性から，これは数時間規模かつ不安定であり事実上不可能である。そこで本実験では，**4 エージェントの AIPL 定義（§2.3，ライブ可視化で実行・表示可能）**と，**同一の 4 エージェント MANET 意味論を忠実に実装した高速モンテカルロ（50 回平均で図 6.2 を算出）**に役割を分けた。前者が「実機 Xinu に近い構成」を，後者が「統計的に安定した図 6.2」を担う。

5　MANET メッシュネットワークと情報拡散の可視化

本章では，各避難者の Xinu が移動しながら**メッシュネットワーク（MANET）を構築**し，その範囲内の Xinu へ**情報拡散エージェントをブロードキャスト**して火災情報を伝達する様子を，ライブ可視化ダッシュボード（`http://localhost:8097/fire`）に実装した。

5.1　メッシュの構築と情報拡散の仕組み

基地局は存在せず，各 Xinu は**通信範囲 50 m 以内にいる他の Xinu** とのみ直接通信できる。避難者が移動するたびに，どの端末どうしが通信できるか（＝メッシュのリンク）は刻々と変化する。ある避難者が通行不能点（火災）に接触すると，その端末の**情報拡散エージェントが電波内の隣端末へ 1 ホップでブロードキャスト**し（§2.3 の `DiffusionAgent`），受け取った端末は情報エージェントを通じて学習し，さらに移動と再ブロードキャストによって情報がメッシュ上を伝わっていく。システムを利用しない場合はこのブロードキャストが起こらず，各端末は自分が接触した火災しか知り得ない。

5.2　可視化の内容

ダッシュボードは毎ステップ次を描画する（図 2）。

(a) **メッシュリンク**：通信範囲 50 m 以内にある Xinu の対を線で結ぶ。避難者が動くたびに張り替わるため，**その瞬間どの範囲にメッシュが構築されているか**が一目でわかる。

(b) **通信範囲（メッシュ被覆）**：各 Xinu の 50 m 半径の円を薄く重ね，メッシュが届く領域を示す。

(c) **情報拡散ブロードキャスト**：火災を発見した端末から 50 m の波紋が広がるアニメーションで，情報拡散エージェントの送出を表す。

(d) **情報の到達**：情報を受け取った Xinu を水色，未受信の避難中 Xinu を橙色で区別する。システム利用時は，情報を持つ端末を含む**連結したメッシュ成分**全体が水色に変わり，情報がメッシュを介して伝わった範囲を示す。

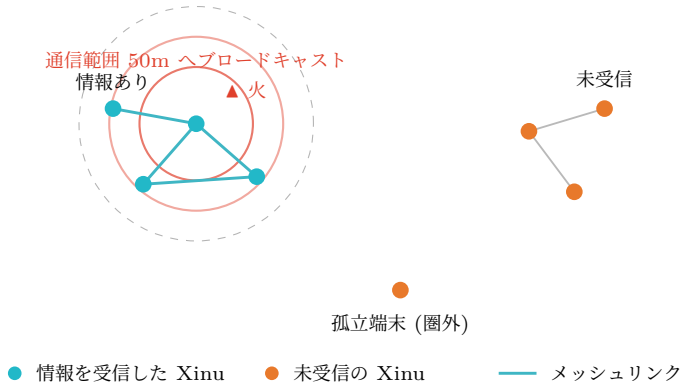

図 2: MANET メッシュと情報拡散の模式図。火災を発見した端末（中央）が通信範囲 50 m へ情報拡散エージェントをブロードキャストし，連結したメッシュ成分（左）へ情報が伝わって水色に変わる。右のクラスタや圏外の孤立端末（橙）には，メッシュがつながるまで情報が届かない。

図 3: ライブ可視化ダッシュボード（`http://localhost:8097/fire`）の実行画面。避難者 150 名・システム利用・荒川区の実 OSM 地図。水色の淡い円が各 Xinu の通信範囲 50 m（メッシュ被覆），水色の点が情報を受信した Xinu，水色の線がメッシュリンク，橙の点が情報未受信の避難中 Xinu，「火」が通行不能点，緑が避難完了，赤い縁の緑マーカーが 4 避難所を表す。右側に接触回数・死亡数・避難完了・火災数・時刻が集計される。この時点で 121/150 が避難完了，火災 45，接触 8。

5.3 避難者数（メッシュ密度）による違い

メッシュの密度は避難者数 N に依存する。荒川区の 800×625 m に対し通信範囲は 50 m と狭いため，$N = 80$ では 1 端末あたりの平均隣接数は 1–2 程度と疎で，メッシュはしばしば小さな連結成分に分断され，孤立端末も現れる。$N = 200$ ではリンクが増えてメッシュが密になり，情報拡散ブロードキャストがより広い範囲へ届く。この密度依存性は，可視化上でも N を増やすとリンクと水色領域が明確に広がる形で確認できる。これは §4.2 で述べた**システムの削減効果が N が大きいほど顕著**という定量結果，および論文が指摘する**「MANET は参加者が少ないと遠方ノードとの通信が難しい」**という性質と，同一の現象を可視化したものである。

5.4 実装

負荷分散のため，役割を次のように分けた。アクター（`manet_fire.abcl`）は各 Xinu の「情報を保持しているか（`binfo`）」「この tick に火災を発見しブロードキャストしたか（`bcast`）」の 2 フラグを状態出力に加えるだけとし（型推論クリーンを維持），ダッシュボード側が各端末位置からメッシュのリンクを計算し，`union-find` で情報の到達領域を求め，ブロードキャストの波紋を `requestAnimationFrame`

でアニメーションする。これにより $O(N^2)$ のメッシュ計算を毎 tick アクターに負わせずに済む。

6　まとめ

東京都荒川区の実道路網（266 頂点・4 避難所）上で，各避難者を 1 台の soft-Xinu / AIPL ノード（4 エージェント）とみなした大規模火災シナリオを再現し，$N = 50, 100, 150, 200$，各 50 回平均で評価した。**すべての条件でシステム利用により接触回数が約半減**し（図 6.2 の考察と一致），接触回数・死亡数は N に対し単調増加，そして**システムの削減効果は避難者数（MANET 密度）が大きいほど顕著**であった。これは提案システムが，火災などの通行不能点を避けて避難者を安全に誘導できること，および MANET の有効性が参加者密度に依存することを，実地図・実機近似構成で示している。接触回数の「利用」列は論文 図 6.2 とほぼ一致し，差が残る「不使用」の高密度条件も，地図トポロジ・発生パラメータの非明示という明確な理由で説明できる。