

分散ハッシュを用いた P2P 型センサデータストリーム配信システムにおける 耐障害性向上法の評価

川上 朋也¹ 石 芳正² 義久 智樹² 寺西 裕一^{3,2}

概要：ライブカメラや環境モニタリングなど，センサデータを周期的に収集および配信するセンサデータストリーム配信が近年注目されている．センサデータストリーム配信では，配信先は用途によって異なる周期で要求することが考えられる．我々のグループでは，センサデータストリームを中継するコンピュータ（ノード）を分散ハッシュによって決定し，配信にかかる負荷を複数のノードで分散する負荷均等化手法を提案した．また，分散ハッシュを用いた手法では，Successor List によって担当ノードを冗長化し，耐障害性の向上を検討している．本研究では，ノードが消失する環境において，分散ハッシュを用いた手法の Successor List による耐障害性について述べる．

An Evaluation of Robustness Enhancement Technique on P2P Sensor Data Stream Delivery System Using Distributed Hashing

TOMOYA KAWAKAMI¹ YOSHIMASA ISHI² TOMOKI YOSHIHISA² YUUICHI TERANISHI^{3,2}

1. はじめに

ライブカメラや環境センサといったセンサデータを周期的に収集して，収集するたびに利用者に対して配信するセンサデータストリーム配信が近年注目されている．センサデータストリーム配信では，センサデータの収集周期より，配信元の送信や配信先の受信といった配信にかかる処理時間が長くなると，配信の遅れが蓄積されるため，収集周期より通信時間が長くないようにすることが重要になる．

配信元や配信先の通信負荷を分散させることで通信時間を短縮できるため，センサデータストリーム配信において，通信負荷を分散させる手法が研究されている [1–10]．これらの既存研究では，複数の配信先に同じセンサデータストリームを配信する場合に，センサデータを受信した配信

先がさらに他の配信先へ送信することで，配信元の通信負荷を分散させている．しかし，センサデータストリーム配信に関しては，同じセンサデータストリームを異なる周期で配信する以下の状況などが考えられる．例えば，日食のライブカメラの映像を配信する場合，有線でインターネットに繋がったパソコンの利用者には 30fps で配信し，移動中に 3G 回線で繋がったパソコンの利用者には 10fps で配信する．

センサデータストリームを異なる周期で配信する場合においても，配信周期が倍数関係になっている，もしくは倍数関係で近似できれば，配信周期の最も短いセンサデータストリームをすべての配信先に配信し，配信先側で間引くことで，要求される配信周期を再現できる．しかし，冗長なセンサデータを配信することになり，配信元および配信先の通信負荷が大きくなる．

我々のグループでは，センサデータストリームを異なる周期で配信する場合に，配信元および配信先となる各コンピュータ（ノード）が P2P 型のオーバーレイネットワークを構成し，配信先の配信周期を考慮することで通信負

¹ 神戸大学大学院工学研究科
Graduate School of Engineering, Kobe University
² 大阪大学サイバーメディアセンター
Cybermedia Center, Osaka University
³ 独立行政法人情報通信研究機構
National Institute of Information and Communications Technology

荷を分散する LCF (Longest Cycle First; 最長周期優先) 法や LLF (Lowest Load First; 最小負荷優先) 法を提案した [11]. これら従来手法では, 異なる配信周期のセンサデータストリームに含まれる同じ配信時刻のセンサデータを配信先間で送受信することで, 配信元および配信先の通信負荷を分散する. また, 複数のセンサデータストリームが混在する環境での配信システムにおいて, 配信にかかる負荷を分散ハッシュに基づいて複数のコンピュータで分散し, 配信元からのセンサデータストリームを中継する負荷均等化手法を提案した. 分散ハッシュを用いた手法では, センサデータストリームごとに環状のハッシュ空間上にノードを配置し, 短い周期グループほどノード数が増えるように, 各周期の部分ハッシュ空間に分割する. 分割された部分ハッシュ空間は同様に環状に扱い, その部分ハッシュ空間上のノードで時刻ごとの担当ノードを決定する. さらに, 分散ハッシュを用いた手法では, Successor List によって担当ノードを冗長化し, 耐障害性の向上を検討している [12]. Successor List は, 環状構造のオーバーレイネットワーク構築手法である Chord などにおいて, 担当ノードを冗長化する方法である. ノードの通信負荷をシミュレーションにより計算し, Successor List が特定の長さ以下ならば, 通信負荷の大きさやノード間の分散は大きく低下しないことを確認した.

本研究では, ノードが消失する環境において, 分散ハッシュを用いた手法の Successor List による耐障害性について述べる.

2. 問題設定

2.1 想定環境

本研究で想定するセンサデータストリーム配信システムでは, センサデータストリームの管理や配信を行う複数のコンピュータ (ノード) によって P2P 型のオーバーレイネットワークが構成され, センサデータストリーム配信における処理を分散させることで, センサデータストリームの種類や配信先が膨大な環境においても高いスケーラビリティが維持される. センサデータストリームはインターネット経由で配信元が配信システムへ周期的に送信し, ノードを経由して複数の配信先へ配信される. センサデータストリームの配信を希望する端末は同様にインターネット経由で配信システムへ依頼し, センサデータの種類および配信周期を指定する. このとき, 配信システム内に存在しないセンサデータは配信できないため, 指定される配信周期はセンサデータストリームの配信元が配信システムへ送信する周期の倍数で近似できるものとする. オーバーレイネットワーク上のノードは任意の周期および時刻のセンサデータを他のノードと送受信できるものとし, センサデータはノード間で種類および時刻ごとに分散管理されることになる.

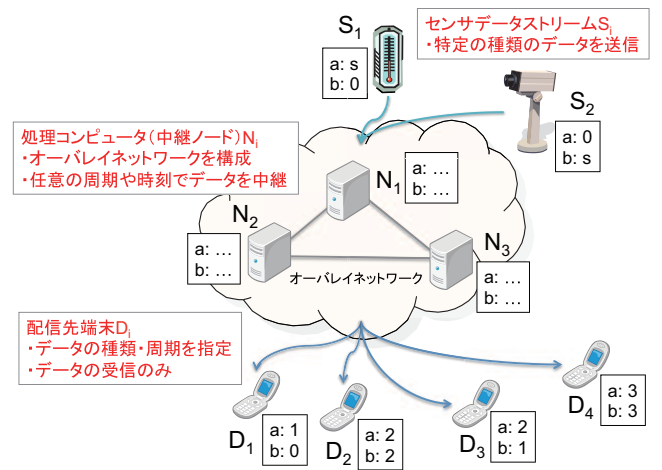


図 1 システムモデル

Fig. 1 System model

l 種類のセンサデータストリームを S_i ($i = 1, \dots, l$), m 台の配信先端末を D_i ($i = 1, \dots, m$), n 台のノードを N_i ($i = 1, \dots, n$) とし, 配信システムのモデルを図 1 に示す. 図 1 の例では, センサデータストリームの種類数 $l = 2$, 配信先数 $m = 4$, ノード数 $n = 3$ である. 図 1 の a, b はそれぞれ S_1, S_2 が配信するセンサデータストリームを表す. センサデータストリームの配信元およびノード, 配信先の付近に示しているのはセンサデータストリーム a, b に対する周期であり, s はセンサデータストリームの配信元を, 数字は希望する配信周期を表している. 配信周期が 0 は配信を希望しないことを表しており, 例えば, ライブカメラの画像に対して, D_1 は配信を希望せず, D_2 が 2 秒に 1 回, D_3 が 1 秒に 1 回, D_4 が 3 秒に 1 回視聴している場合に相当する. 本研究では配信先が指定可能な周期はあらかじめ決まっているものとし, C_i ($i = 1, 2, \dots$) の集合で表す. ノードには任意かつ複数の配信周期や時刻が割り当てられ, 他のノードと任意の種類および時刻のセンサデータが送受信される.

2.2 負荷の定義

センサデータストリーム配信では, 配信にかかる処理時間が長くなると配信の遅れが蓄積されるため, 特定のコンピュータやネットワークに処理や負荷を集中させないことが重要となる. そのため本研究では, センサデータストリーム配信システムにおいてセンサデータの配信および中継を行うノードの通信負荷を分散させることを目的とする.

本研究では, ノード N_i の通信負荷を L_i とし, センサデータストリームの受信および送信による通信負荷の合計で与える. 受信による通信負荷は受信負荷と呼び, N_i の受信負荷を I_i で示す. 送信による通信負荷は送信負荷と呼び, N_i の送信負荷を O_i で示す. 多くの場合, 受信負荷および送信負荷は送受信するセンサデータの単位時間あたりの数に比例する. N_i が N_j および S_k ($i \neq j, i, j = 1, \dots, n$,

$k = 1, \dots, l$ から受信する単位時間あたりのセンサデータ数を $R(N_j, N_i)$ および $R(S_k, N_i)$ で、 N_j および D_k ($i \neq j$, $i, j = 1, \dots, n$, $k = 1, \dots, m$) へ送信する単位時間あたりのセンサデータ数を $R(N_i, N_j)$ および $R(N_i, D_k)$ で表し、 L_i , I_i , O_i はそれぞれ以下で与える.

$$L_i = I_i + O_i \quad (1)$$

$$I_i = \alpha \sum_{j=1}^n R(N_j, N_i) + \alpha \sum_{k=1}^l R(S_k, N_i) \quad (2)$$

$$O_i = \beta \sum_{j=1}^n R(N_i, N_j) + \beta \sum_{k=1}^m R(N_i, D_k) \quad (3)$$

α および β はそれぞれ、1つのセンサデータの受信および送信による負荷である.

センサデータストリーム配信システム全体の通信負荷は、 n 台のノードの通信負荷 L_i を用いた以下で与える.

$$SL = \sum_{i=1}^n L_i \quad (4)$$

また、負荷分散の指標として、Fairness Index (FI) を用いる. Fairness Index は公平性の指標としてよく用いられ、以下で表される.

$$FI = \frac{(\sum_{i=1}^n L_i)^2}{n \sum_{i=1}^n L_i^2} \quad (5)$$

$0 \leq FI \leq 1$ であり、 $FI = 1$ であれば、 $L_1 = \dots = L_n$ となる. FI が 1 に近いほど偏りが小さく、通信負荷が分散されていることを示す. 本研究の目的は、システム全体の通信負荷を抑えつつ、ノード間で分散させることである. よって、目的関数は SL および $1 - FI$ となり、これらの値が最小となるようにノードの配信経路を決定する.

3. センサデータストリーム配信システムのための負荷均等化手法

3.1 分散ハッシュを用いた手法

我々のグループでは、センサデータストリーム配信におけるシステム全体の通信負荷を抑えつつ、ノード間で負荷を分散させることを目的に、LCF 法および LLF 法と呼ばれる負荷均等化手法を提案した [11]. また、複数のセンサデータストリームが混在する環境での配信システムにおいて、配信にかかる負荷を分散ハッシュに基づいて複数のコンピュータで分散し、配信元からのセンサデータストリームを中継する負荷均等化手法を提案した [12]. 各ノードが自律的に配信経路を構築するためには、ハッシュ空間上にノードを配置し、以下などの分散ハッシュに基づいて担当ノードを決定することが考えられる.

- センサデータストリーム
- センサデータストリームおよび周期
- センサデータストリームおよび時刻

従来手法では、センサデータストリームおよび周期によってノードをグループ化し、各周期グループで時刻ごとの担当ノードを決定している. しかし、従来手法では特定のデータを単一のノードが担当する可能性があり、ノードの障害時に新たな担当ノードを探索する必要がある.

3.2 Successor List を用いた担当ノードの冗長化

分散ハッシュを用いた従来研究では、センサデータストリームごとにハッシュ空間上にノードを配置し、短い周期グループほどノード数が増えるように、各周期ごとの部分ハッシュ空間に分割する. 例えば、指定可能な周期 $C_i = i$ ($i = 1, 2, 3$) では、 $1/C_1 : 1/C_2 : 1/C_3 = 1/1 : 1/2 : 1/3 = 6 : 3 : 2$ となる. 分割された部分ハッシュ空間は環状に扱い、その部分ハッシュ空間上のノードで時刻ごとの担当ノードを決定する. 周期の部分ハッシュ空間上にノードが存在しない場合には、その前の部分ハッシュ空間上の最近傍ノードがその周期グループも担当する. また、センサデータストリームの配信元からデータを受信するノード (配信元ノード) については、センサデータストリームの分散ハッシュに基づいて決定する.

我々のグループでは現在、担当ノードの消失などに対する耐障害性を向上するため、時刻ごとの担当ノードを Successor List を用いて冗長化する手法を検討している [12]. Successor List は、同様に環状構造のオーバーレイネットワークである Chord [13] などにおいて、担当ノードを冗長化する方法である. 本研究では Successor List の長さをあらかじめ設定し、本来の担当ノードがすべての Successor にセンサデータを中継する. 本来の担当ノードが消失した場合には、分散ハッシュ上で Successor が次の担当ノードとなるため、配信経路の再構築処理の一部を効率化できる. ノード数 $n = 8$, 周期 $C_i = i$ ($i = 1, 2, 3$), ハッシュ空間の大きさ 2^p , Successor List の長さが 2 の例を図 2 に示す. 図 2 では、各周期の部分ハッシュ空間の開始値はそれぞれ、 $2^p \times 0/11$, $2^p \times 6/11$, $2^p \times 9/11$ となる. 例えば、ノード N_4 の Successor List にはノード N_1 , N_2 が登録され、ノード N_4 は自身が担当する時刻のデータを送信する.

各ノードがセンサデータストリームごとの配信経路を構築するアルゴリズムの擬似コードを図 3 に示す. 各ノードは、指定可能な周期の最小公倍数を計算し、自身が属するグループの周期で時刻の分散ハッシュを計算する. 計算した分散ハッシュが自身の担当範囲の場合、LCF 法 [11] と同様、その時刻の最長周期のノードにセンサデータの送信を要求する. その際、最長周期ノードの Successor List のうちで、実際に受信するノードをランダムに決定する. その時刻の最長周期のノードは、配信元ノードにセンサデータの送信を要求する. また、配信先がセンサデータストリームごとの配信を依頼するアルゴリズムの擬似コードを図 4 に示す. 配信先は、センサデータを受信する時刻を自身の

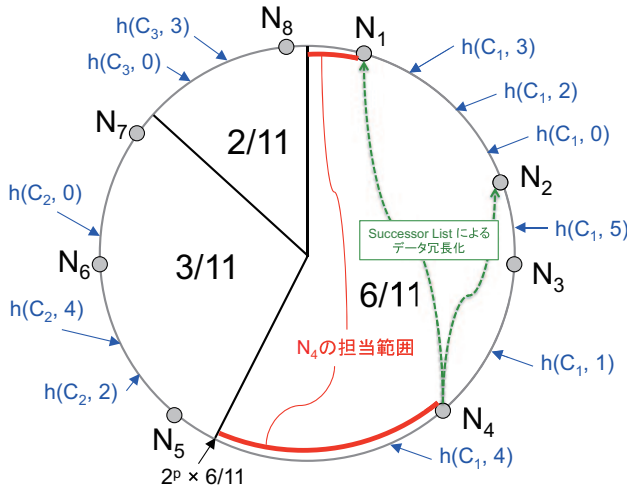


図2 ノードのグループ分けおよび担当ノードの割り当て
Fig. 2 Grouping and assignment of nodes

指定周期から計算し、各時刻のセンサデータを送信する周期グループのうちから、それぞれランダムに1グループを選択する。さらに、選択したグループにおいてその時刻を担当するノードのSuccessor Listに対して、実際にセンサデータの送信を要求するノードをランダムに決定する。

4. 評価

本研究では、3章のSuccessor Listを用いた耐障害性向上法をシミュレーションにより評価した。

4.1 シミュレーション環境

本シミュレーションでは、ノード数 $n = 2^7 = 128$ 、配信先数 $m = 1000$ の環境で評価した。また、単一のセンサデータストリームによる結果を確認するため、センサデータストリーム数については1とした。配信先が要求する周期 $C_i = i$ ($i = 1, \dots, 10$) とし、各配信先に1~10の範囲でそれぞれランダムに決定した。各ノードは分散ハッシュに基づいて各周期グループに割り当てられ、本シミュレーションでは最小公倍数である2520が長さとなる時刻表の配信先経路をノード間や配信先に構築する。

各シミュレーションの結果は、各手法および後述するパラメータごとに10回の試行を行い、その平均値を算出した。

4.2 ノードの消失が影響する配信先数

時刻0~2519において、各時刻の最長周期グループのノードが消失した場合に配信できなくなる配信先数を図5に示す。

分散ハッシュを用いた手法では、各時刻の最長周期グループのノードが他の周期グループのノードへ中継する。周期グループ内で各時刻を特定のノードが担当するため、最長周期グループ内の担当ノードが消失すると、他の周期

Require:

cycles: 選択可能な配信周期を昇順に並べた配列
ownId: 自身を示す識別子 (ノード)
assignedCycleIndex: 配列 *cycles* で自身に割り当てられた周期のインデックス
succCount: Successor List の長さ

```

1: cycleLcm ← calculateLCM(cycles); // 選択可能な配信周期の最小公倍数を計算
2: if assignedCycleIndex ≠ 0
   or searchNode(0, cycleLcm, 0) ≠ ownId then
3:   // 自身が最短周期グループではない、または、ルートノードではない場合
4:   time ← 0;
5:   while time < cycleLcm do
6:     assignedNode
       ← searchNode(assignedCycleIndex, time, 0);
7:     if assignedNode = ownId then
8:       // 自身が周期グループ内で配信時刻 time に割り当てられている場合
9:       longCycleIndex
       ← calculateLongestCycleIndex(cycles, time, 0);
10:      relayNode;
11:      if longCycleIndex = assignedCycleIndex then
12:        // 自身の周期グループが時刻 time で最長の場合
13:        relayNode ← searchNode(0, cycleLcm, 0); // 時刻 time にセンサデータを受信するため、ルートノードを探索
14:        succList;
15:        succNode ← ownId;
16:        for i ← 0 to succCount do
17:          succNode ← successor(succNode); // Successor ノードを探索
18:          succList.add(succNode); // 探索されたノードを Successor List に追加
19:        end for
20:      else
21:        succNodeIndex
       ← random(0, succCount + 1); // センサデータを受信する Successor ノードをランダムに決定
22:        relayNode ← searchNode(longCycleIndex, time, succNodeIndex); // succNodeIndex 番目の Successor ノードを探索
23:      end if
24:      requestToSend(relayNode, ownId, time); // 時刻 time における relayNode からの配信経路を構築
25:    end if
26:    time ← time + cycles[assignedCycleIndex];
27:  end while
28: end if

```

図3 ノードが配信経路を構築する擬似コード

Fig. 3 Pseudocode to construct delivery paths by nodes

グループのノードやそれらの配信先へも配信できなくなる。図5では、時刻0ですべての周期グループが配信先へ配信し、最長周期グループは配信周期10である。そのため、配信周期10のグループで時刻0を担当するノードが消失すると、1000台すべての配信先へ配信できなくなる。同

Require:

cycles: 選択可能な配信周期を昇順に並べた配列
ownId: 自身を示す識別子 (ノード)
requestCycleIndex: 配列 *cycles* で要求する周期のインデックス
succCount: Successor List の長さ

```

1: cycleLcm ← calculateLCM(cycles); // 選択可能な配信周期の最小公倍数を計算
2: time ← 0;
3: while time < cycleLcm do
4:   targetCycIndex
     ← getRandomCycleIndex(cycles, time); // 時刻 time
     にセンサデータを受信する周期グループをランダムに決定
5:   succNodeIndex
     ← random(0, succCount + 1); // センサデータを受信する
     Successor ノードをランダムに決定
6:   relayNode
     ← searchNode(targetCycIndex, time, succNodeIndex);
     // succNodeIndex 番目の Successor ノードを探索
7:   requestToSend(relayNode, ownId, time); // 時刻 time
     における relayNode からの配信経路を構築
8:   time ← time + cycles[requestCycleIndex];
9: end while

```

図 4 配信先が配信経路を構築する擬似コード

Fig. 4 Pseudocode to construct delivery paths by destinations

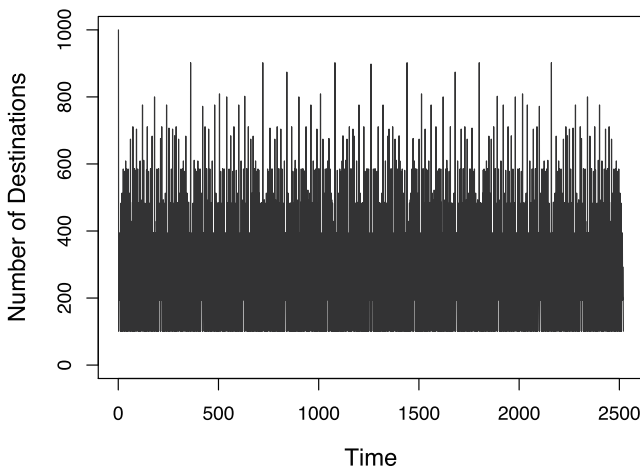


図 5 各時刻における最長周期グループの配信先数

Fig. 5 Number of destinations for the longest cycle group at each time

様に、各時刻での配信周期の重なりによって配信先数や最長周期グループが変わり、図 5 に示した数の配信先へ影響する。ただし、我々のグループが検討している Successor List を用いることで、これらの 1 つのノードの消失が与える影響を軽減できる。

各周期グループにおける単位時間あたりの最大配信先数を図 6 に、各周期グループにおける総配信先数を図 7 に示す。

図 6 より、多くの周期グループが関係する時刻で最長周期グループとなる可能性が高いため、配信周期が長いグ

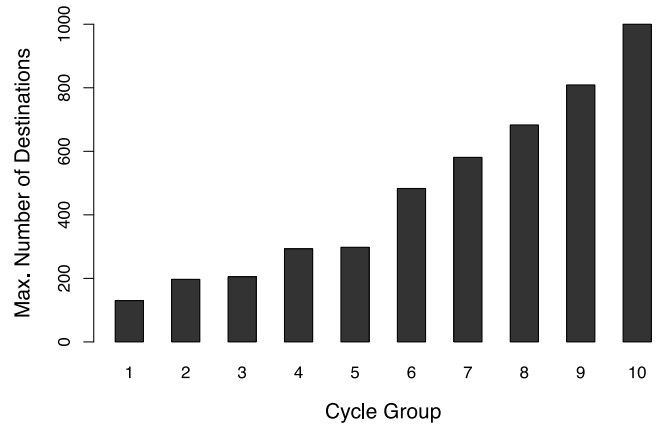


図 6 各周期グループにおける単位時間あたりの最大配信先数

Fig. 6 Maximum instantaneous number of destinations for each cycle group

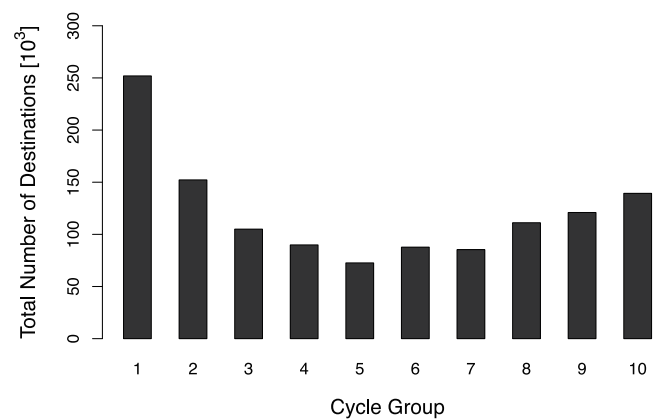


図 7 各周期グループにおける総配信先数

Fig. 7 Total number of destinations for each cycle group

ループほど、単位時間あたりに影響する配信先数が多い。一方、図 7 より、配信周期 1 のグループなどの配信周期が短いグループでは、多くの時刻で配信先へ配信するため、すべての時刻における合計である総配信先数が多い。

4.3 Successor List の長さによるホップ数

Successor List の長さを $0, \dots, 4$ で変化させた場合において、時刻 0 で各ノードがデータを受信するまでのホップ数の割合を図 8 に示す。横軸はホップ数で、縦軸はそのホップ数まででデータを受信しているノードの割合を累積で表している。横軸が「N/A」としているノードは、同じ周期グループ内の別のノードが時刻 0 を担当するなどによって、時刻 0 でデータを受信しないノードである。

図 8 より、時刻 0 においてデータを受信するノードが増えるため、Successor List が長くなるほど「N/A」で示したノードが減少する。また、Successor List が長いほど、長いホップ数でデータを受信するノードの割合が大きくなるが、すべてのノードが 4 ホップ以内にデータを受信できている。

同様に、Successor List の長さを $0, \dots, 4$ で変化させた

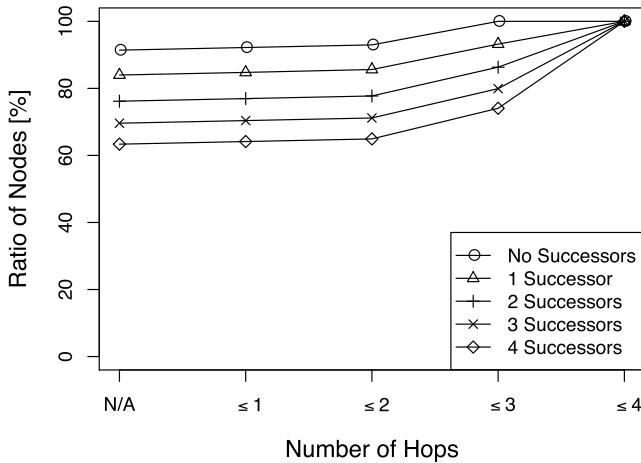


図 8 各ホップ数で配信されるノードの割合

Fig. 8 Ratio of nodes by the number of hops to delivery

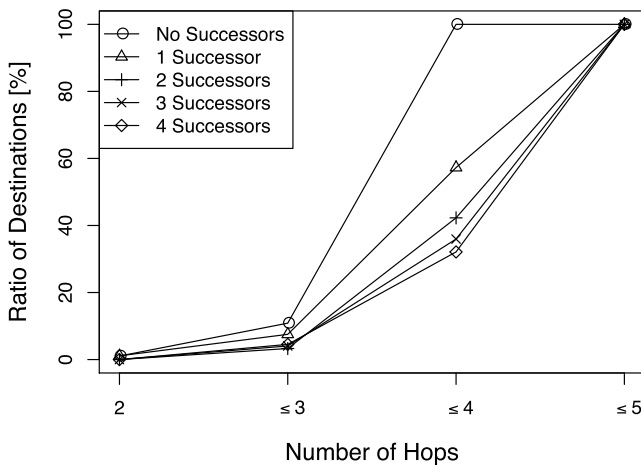


図 9 各ホップ数で配信される配信先の割合

Fig. 9 Ratio of destinations by the number of hops to delivery

場合において、時刻 0 で各配信先がデータを受信するまでのホップ数の割合を図 8 に示す。横軸はホップ数で、縦軸はそのホップ数まででデータを受信している配信先の割合を累積で表している。

図 9 より、図 8 と同様、Successor List が長いほど、長いホップ数でデータを受信する配信先の割合が大きくなるが、すべての配信先は 5 ホップ以内にデータが配信されている。

5. 関連研究

ストリーム配信において、通信負荷を分散させる様々な手法が研究されている。

通信負荷をネットワーク上の端末に分散させるために、サーバを介さずに端末間でデータを送受信する P2P (Peer-to-Peer) 技術を用いてストリーム配信する P2P ストリーム配信手法が提案されている [1–5]。P2P ストリーム配信手法は、プル型の手法とプッシュ型の手法に分けられる。

PPLive^{*1}, DONet [1], SopCast^{*2} といったプル型の手法では、データの受信側となる受信端末の方から他の端末にデータを要求して取得する。AnySee などのプッシュ型の手法では、データの送信側となる送信端末の方から他の端末にデータを配信する [2]。また、PRIME のようにプル型とプッシュ型を組み合わせた手法や、オーバーレイネットワーク技術を用いてセンサデータの収集や加工、配信を行う手法も提案されている [3]。これらの手法を異なる配信周期があるセンサデータストリーム配信に適用する場合、配信周期が異なるセンサデータストリーム毎に別のデータストリームとして配信することになり、通信負荷を効率的に削減できない。

マルチキャスト木と呼ばれる、データの配信経路をあらかじめ構築してデータストリームを配信し、特定の端末に通信負荷が集中することを防ぐ手法がいくつか提案されている [6–10]。ZIGZAG 法では、端末の集合であるクラスタでマルチキャスト木を構築し、マルチキャスト木における各深さに含まれるクラスタの数を平衡にすることで負荷分散を行っている [6]。アプリケーション層で得られる情報のみでマルチキャスト木を構築しており、物理的なネットワーク構造を把握する必要がない。LAC (Locality Aware Clustering) では、一部の端末間の通信遅延のみを考慮することで、ZIGZAG 法と比べて高い負荷分散を達成している [7]。MSMT/MBST 法では、物理的なネットワーク構造を把握できる場合に、端末間の通信遅延を考慮することで、ZIGZAG 法と比べて特定の端末に通信負荷が集中することを避けている [8]。しかし、上記の P2P ストリーム配信手法と同じく、センサデータストリームの配信周期が異なる場合、配信周期が異なるセンサデータストリーム毎に別のデータストリームとして配信することになり、通信負荷を効率的に分散できない。

6. まとめ

本研究では、分散ハッシュを用いて各ノードが自律的に配信経路を構築するセンサデータストリーム配信システムにおいて、担当ノードを冗長化することで耐障害性を向上する方法についての評価した。シミュレーション結果より、分散ハッシュを用いた手法では 1 つのノードの消失によって多くの配信先へ配信できなくなるが、Successor List を用いることでノードの消失による影響を軽減できると考えられる。また、Successor List を用いた場合でも、ノードや配信先までのホップ数は大きく変化しないことを確認した。

今後の課題としては、担当ノードが動的に追加や離脱、消失する環境において、提案手法の耐障害性を検証することが必要である。

*1 <http://www.pplive.com/>

*2 <http://www.sopcast.com/>

謝辞 本研究の一部は、科学研究費補助金（若手研究 A）「次世代オンデマンド型視聴形態のためのコンテンツ配信方式」（課題番号：23680007）および（挑戦的萌芽研究）「移動型カメラを用いた任意地点ライブビューの実現」（課題番号：26540045）、NICT・大阪大学共同研究「大規模分散コンピューティングのための高機能ネットワークプラットフォーム技術の研究開発」、独立行政法人科学技術振興機構、京都地域スーパークラスタープログラム「クリーン・低環境負荷社会を実現する高効率エネルギー利用システムネットワークの構築」による成果である。

参考文献

- [1] Zhang, X., Liu, J., Li, B. and Yum, T.-S. P.: Cool-Streaming/DONet: A Data-Driven Overlay Network for Peer-to-Peer Live Media Streaming, *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2005)*, Vol. 3, pp. 2102–2111 (2005).
- [2] Liao, X., Jin, H., Liu, Y., Ni, L. M. and Deng, D.: AnySee: Peer-to-Peer Live Streaming, *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM 2006)*, pp. 1–10 (2006).
- [3] Magharei, N. and Rejaie, R.: PRIME: Peer-to-Peer Receiver-Driven Mesh-Based Streaming, *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM 2007)*, pp. 1415–1423 (2007).
- [4] Yu, L., Liao, X., Jin, H. and Jiang, W.: Integrated Buffering Schemes for P2P VoD Services, *Peer-to-Peer Networking and Applications*, Vol. 4, No. 1, pp. 63–74 (2011).
- [5] 坂下 卓, 義久智樹, 原 隆浩, 西尾章治郎: P2P ストリーミング環境における分割データの重要度を考慮した視聴中止端末数削減手法, *情報処理学会論文誌*, Vol. 52, No. 11, pp. 3008–3017 (2011).
- [6] Tran, D. A., Hua, K. A. and Do, T.: ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming, *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*, Vol. 2, pp. 1283–1292 (2003).
- [7] Silawarawet, K. and Nupairoj, N.: Locality-Aware Clustering Application Level Multicast for Live Streaming Services on the Internet, *Journal of Information Science and Engineering*, Vol. 27, No. 1, pp. 319–336 (2011).
- [8] Jin, X., Yiu, W.-P. K., Chan, S.-H. G. and Wang, Y.: On Maximizing Tree Bandwidth for Topology-Aware Peer-to-Peer Streaming, *IEEE Transactions on Multimedia*, Vol. 9, No. 8, pp. 1580–1592 (2007).
- [9] Banerjee, S., Bhattacharjee, B. and Kommareddy, C.: Scalable Application Layer Multicast, *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM 2002)*, pp. 205–217 (2002).
- [10] Le, T. A. and Nguyen, H.: Application-Aware Cost Function and Its Performance Evaluation over Scalable Video Conferencing Services on Heterogeneous Networks, *Proceedings of the IEEE Wireless Communications and Networking Conference: Mobile and Wireless Networks (IEEE WCNC 2012 Track 3 Mobile and Wireless)*, pp. 2185–2190 (2012).
- [11] Kawakami, T., Ishi, Y., Yoshihisa, T. and Teranishi, Y.: A P2P-Based Sensor Data Stream Delivery Method to Accommodate Heterogeneous Cycles, *Journal of Information Processing (JIP)*, Vol. 22, No. 3 (2014).
- [12] 川上朋也, 石 芳正, 義久智樹, 寺西裕一: 分散ハッシュを用いた P2P 型センサデータストリーム配信システムにおける耐障害性向上法の検討, *情報処理学会研究報告 (マルチメディア通信と分散処理研究会)*, Vol. 2014-DPS-158, No. 3, pp. 1–7 (2014).
- [13] Stoica, I., Morris, R., Liben-Nowell, D., Karger, D. R., Kaashoek, M. F., Dabek, F. and Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications, *IEEE/ACM Transactions on Networking*, Vol. 11, No. 1, pp. 17–32 (2003).