IoT環境におけるストリーミング処理時間短縮手法

義久 智樹[1,a)]　原 隆浩[2]

概要：センサなどのモノがインターネットに接続される流行は，IoT（Internet of Things）と呼ばれ，近年非常に注目されている．これらのモノから継続的に発生するデータを入力として，あらかじめ登録された問合せ（連続問合せ）を実行するストリーミング処理技術が研究されている．これらの研究では，データが入力されてから連続問合せの結果を得るまでのストリーミング処理にかかる時間を短縮している．近年の IoT 環境の普及により，高頻度なデータの発生源が分散してインターネットに接続されており，さらなるストリーミング処理時間の短縮が求められている．そこで本研究では，IoT 環境におけるストリーミング処理時間短縮手法を提案する．提案手法では，センサが接続された処理サーバが分散していると想定したうえで，各連続問合せを個別に処理できる幾つかの部分条件に分割してストリーミング処理を行う．評価の結果，提案手法は従来手法よりも最大通信ホップ数と平均通信量を削減でき，ストリーミング処理時間を短縮できることを確認した．

1. はじめに

センシング技術の発達に伴って人感センサやカメラなどの様々なセンサがインターネットに多数接続されている．センサなどのモノがインターネットに接続される流行は，IoT（Internet of Things）と呼ばれ，近年非常に注目されている．これらのセンサには，設置されてから継続的にデータを計測しているセンサがあり，日々大量のデータがインターネット内で発生している．これらの継続的に発生するデータを入力として，あらかじめ登録された問合せ（連続問合せ）を実行するストリーミング処理技術が研究されている．例えば，警備会社が契約先の一人暮らしの家の家電の利用状況をセンシングしておき，普段は利用するはずの複数の家電が利用されていなければ通知する連続問合せをストリーミング処理システムに登録しておく．家電が利用されていなければ，倒れていたり事件に巻き込まれて家に帰っていないといった可能性がある．また，人感センサでビルの人の出入りをセンシングしており，開錠時間外にセンサが反応した場合にその付近のカメラ映像の変化が激しければ人が侵入していると判断し，盗難の危険を発見することが考えられる．

データが入力されてから連続問合せ結果を得るまでのストリーミング処理にかかる時間を短縮することで，事件や火災の危険を即座に発見するなど応答性を高められる．また，ストリーミング処理時間を短縮することで，継続的に入力されるデータに対する単位時間あたりの最大の処理回数も向上できる．このため，ストリーミング処理時間を短縮する幾つかの手法が提案されている（[1], [2], [3]）．これらの手法は，2 種類に大別できる．

通信時間短縮型　データの発生源が複数の端末に分散している環境において，データをある端末に集約してから入力することで，個々の通信に伴うオーバーヘッドを削減する．削減されたオーバーヘッドの送受信にかかる通信時間の削減によりストリーミング処理時間を短縮する手法

処理時間短縮型　連続問合せを処理する処理サーバが分散している環境において，処理負荷を分散させて連続問合せ処理を並列に行うことで処理にかかる時間を短縮してストリーミング処理時間を短縮する手法

データの発生源と処理サーバの両者が分散している環境では，通信時間短縮型と処理時間短縮型を組み合わせて適応することもできる．一方で，近年の IoT 環境の普及により，データの発生頻度が大きいデータ発生源が分散してインターネットに接続されており，さらなるストリーミング処理時間の短縮が求められている．上記したような警備会社の契約数は約 90 万件（[4]）や約 200 万件（[5]）ある．例えば，100 万件の契約先のビルから，単位時間あたりにセンサデータを 1 個集めるとしても百万個のデータを処理することになる．人感センサ程度でデータサイズが比較的小

[1] 大阪大学サイバーメディアセンター
Cyberemedia Center, Osaka University, Ibaraki, Osaka 567–0047, Japan
[2] 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University, Suita, Osaka 565–0871, Japan
[a)] yoshihisa@cmc.osaka-u.ac.jp

©2015 Information Processing Society of Japan

さく，センサデータ 1 個当たり 20 バイト（IP パケットの最小データサイズ）の場合，160Mbps のデータ発生頻度になる．映像データのようにデータサイズが比較的大きく，センサデータ 1 個当たり 100K バイト（MPEG 形式の 1 フレーム相当）の場合では，800Gbps のデータ発生頻度になる．ストリーミング処理時間が長くなると，遅延が重なってさらに処理完了までの時間が長くなってしまう．センサが接続された処理サーバ（データの発生源と処理サーバ）が分散していると想定したうえで，通信時間と処理時間の両者を短縮することで，ストリーミング処理時間をより短縮できる．

そこで本研究では，IoT 環境において，センサが接続された処理サーバが分散してインターネットに接続されている点を考慮して，ストリーミング処理時間短縮手法を提案する．提案手法では，各連続問合せを個別に処理できる幾つかの部分条件に分割し，処理結果が偽になる確率の高い部分条件から順番に各処理サーバが処理する．部分条件とは，温度データが異常に高い，カメラに炎が映っているといった連続問合せの各条件を指す．部分条件が満たされない場合には処理を行わないことで，処理時間を短縮できる．さらに，本研究では，提案手法の評価，議論を行う．

本研究の貢献は以下の通りである．1）IoT 環境におけるストリーミング処理システムの明確化，2）連続問合せに含まれる条件の部分条件への分割方法の考案，3）IoT 環境におけるストリーミング処理時間短縮手法の確立．以下，2 章で関連研究について説明し，3 章で本研究で想定する IoT 環境におけるストリーミング処理システムを説明する．4 章でストリーミング処理時間短縮手法を提案し，5 章で提案手法の評価，6 章で考察を行う．最後に 7 章で本稿をまとめる．

2. 関連研究

文献 [6] のように，発生したセンサデータを保存しておいて，後で連続問合せを行う場合，幾つかのデータが保存されるまで時間がかかり，データの発生から処理結果を得るまでの時間が長くなる．この場合と比べて，連続的に発生するデータに対してストリーミング処理することで，素早く結果を得られるため，ストリーミング処理に関する研究が多数行われている．

文献 [1] では，ストリーミング処理において，Streaming Quotient Filter (SQF) と呼ぶデータ構造を用いて重複する処理結果を 1 個にまとめて通信頻度を抑える手法を提案しているが，対象とするデータストリームが 1 個であった．このため，センサデータサーバが複数あってデータストリームが複数個ある場合には，各データストリームを順番に処理することになるため，通信量を十分に低減することができない．

文献 [2] では，順序のある処理を複数のセンサデータサーバで逐次処理する場合に，高速に処理するための処理割り当て手法を提案している．上記と同じく，文献 [2] の提案手法では，複数のデータストリームがある場合には順番に処理することになるため，処理を十分に高速化できない．

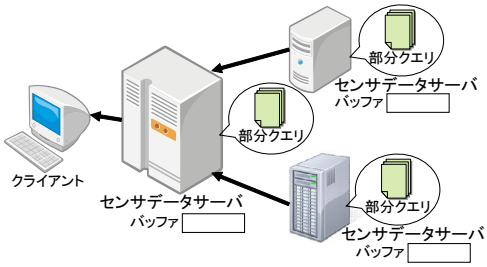


図 1 システム構成の一例
Fig. 1 An example of the system image

文献 [3] では，複数のデータストリームをまとめて 1 個のデータストリームとして扱うことで，通信や処理に伴う負荷を軽減する手法を提案している．しかし，本研究のように，複数のセンサデータサーバに処理が分散している環境を想定していない．

文献 [7] では，複数のデータストリームの結合処理において，過去の複数のデータをスケッチとして高速な記憶領域に貯えておくことで，結合処理を高速に行える手法を提案している．本研究においても，過去のセンサデータが処理に含まれる場合には，この手法を適応できる．

3. 想定環境

本章では，本研究で想定している IoT 環境におけるストリーミング処理システムを明確にし，システム構成等について説明する．

3.1 システム構成

クライアントは，ユーザからの連続問合せを受け付け，その実行結果をユーザに提示する．センサデータサーバは複数あり，各センサデータサーバにはセンサが接続されている．各センサデータサーバは，センサからセンサデータを取得して，ユーザが入力した連続問合せのうち，取得したセンサデータを対象としている部分を実行する．実行結果を送信するためのバッファを備えている．クライアントおよび各センサデータサーバはインターネット等の情報ネットワークで接続されており，互いに通信可能とする．IoT 環境におけるシステム構成の一例を**図 1** に示す．以上のシステム構成において，下記の環境を想定する．

- センサデータサーバは，センサから継続的にデータを取得する．
- クライアントはセンサデータサーバの識別子を取得している．
- クライアントは，登録された連続問合せを部分条件に分解できる．

©2015 Information Processing Society of Japan

- 部分条件の実行に必要なセンサデータはいずれかのセンサデータサーバがもっている．
- センサデータサーバは，実行結果をバッファに貯えて順次送信する．

具体例として，1 章に記述した通り，警備会社が多数の契約先のビルに設置したセンサからセンサデータを取得し，複数のセンサデータサーバを用いてストリーミング処理を行う場合が考えられる．後の評価では，この例を参考にして評価環境を設定している．

3.2 部分条件

本節では，連続問合せに含まれる条件の部分条件への分割方法を提供する．

N 個の連続問合せを Q_n（$n = 1, \cdots, N$）で示す．一般に，連続問合せは記述された複数の条件（式だけでなく射影や選択も含まれる）が満たされた場合に実行結果をユーザに提示する．条件が満たされるかどうかの真偽は論理関数で求められるため，Q_n が和積標準形の論理関数で表現できると考えても一般性を損なわない．射影や選択演算であっても，ある和項に含めることで和積標準形で表現できる．具体的な例は次節で説明している．この和積標準形で記述された Q_n の条件の I_n 個の和項を $C(n,i)$（$i = 1, \cdots, I_n$）で示し部分条件と呼ぶ．部分条件が真の場合 1，偽の場合 0 とする．下記が満たされると Q_n の実行結果がユーザに提示されることになる．

$$\Pi_{i=1}^{I_n} C(n,i) = 1 \tag{1}$$

$C(n,i)$ の真偽判定に必要なセンサデータをもつセンサデータサーバの識別子の集合を $Servs(n,i)$ で示し，$Servs(n,i)$ に含まれるあるセンサデータサーバを部分条件管理センサデータサーバと呼び $MServ(n,i)$ で示す．$MServ(n,i)$ は，$C(n,i)$ の真偽判定を行うために，$Servs(n,i)$ に含まれる複数のセンサデータサーバからセンサデータを取得する．$p, q \in Servs(n,i)$ に関して，$C(n,i)$ の真偽判定のために必要な p から q への通信の頻度を $ComF_{p\to q}(n,i)$ で示す．センサデータが連続的に発生するため，本研究では通信の頻度を用いる．

また，$C(n,i) = 1$ となる確率を $TProb(n,i)$ で示す．$TProb(n,i)$ の予測値は，システムメンテナンス時や定期的に，クライアントが $C(n,i)$ の真偽を判定したセンサデータサーバから，$C(n,i) = 1$ となった割合を収集することで算出できる．

各センサデータサーバは，各連続問合せにおいて，複数の部分条件の $Servs$ に含まれることがある．各センサデータサーバはセンサデータを連続的に取得しており，各部分条件の真偽判定には保持する最新のセンサデータを用いる．

センサデータサーバA

センサデータ	時刻	部分条件	データ
Aの温度データ	12:02	1, 2, 5	110
Aの映像データ	12:02	1, 2, 5	a.jpg
Bの温度データ	12:00	3, 4, 5	80
Bの映像データ	12:00	3, 4, 5	b.jpg

図 2 センサデータサーバ A が保持するセンサデータ
Fig. 2 Sensor data that the sensor data server A has

3.3 連続問合せの例

警備会社が火事を素早く発見するために，温度センサとカメラを接続したセンサデータサーバを用意する．温度が高くカメラに炎が映っていれば火事と判断し，その時の温度と画像をユーザに提示して注意喚起する．誤認識を防ぐために，センサデータサーバを 2 台（A と B）用意し，これらを近い場所に設置してセンシングする．上記条件に加えてセンサデータサーバ A，B の温度データの差が小さい場合に火事と判断する．温度センサの計測頻度は 1Hz，カメラの撮影頻度は 30Hz とする．一例として，上記のセンサデータサーバ A が**図 2** に示すような形式でセンサデータを保持することが考えられる．

この場合，連続問合せは $N = 1$ 個で，Q_1 の部分条件は $I = 5$ 個になってそれぞれ，$C(1,1)$「センサデータサーバ A の温度データが高い」，$C(1,2)$「センサデータサーバ A のカメラに炎が映っている」，$C(1,3)$「センサデータサーバ B の温度データが高い」，$C(1,4)$「センサデータサーバ B のカメラに炎が映っている」，$C(1,5)$「センサデータサーバ A と B の温度データの差が小さい」となる．$C(1,1)$，$C(1,2)$ の真偽の判定に必要なセンサデータをもつセンサデータサーバは A のみであるため $Servs(1,1)$ および $Servs(1,2)$ は $\{A\}$，また，$Servs(1,3)$ および $Servs(1,4)$ は $\{B\}$，$Servs(1,5) = \{A, B\}$ となる．$C(1,5)$ の判定を A で行う場合（$MServ(1,5) = A$），B のカメラで撮影された映像を A に送信する．単純に常に部分条件を判定し，B のカメラで撮影する毎にセンサデータを A に送信する場合，$ComF_{A\to B}(1,5) = 0$，$ComF_{B\to A}(1,5) = 30$ となる．

4. 提案手法

式（1）より，いずれかの部分条件が偽であれば連続問合せの条件は満たされない．満たされない場合にはセンサデータを送信しないことで，通信量を削減できる．提案手法では，連続問合せ Q_n（$n = 1, \cdots, N$）に関して，$TProb(n,i)$（$i = 1, \cdots, I_n$）が小さく偽になる確率が大きい部分条件から順番に真偽の通知を行う．すなわち，$TProb(n,i_1) < \cdots < TProb(n,i_I)$ とすると，$C(n,i_1), \cdots, C(n,i_I)$ の順番に通知する．$MServ(n,i_j)$（$j = 1, \cdots, I-1$）は $C(n,i_j) = 1$ であれば，$MServ(n,i_{j+1})$ に通知する．そこで，提案手法では，トリガグラフを用いて部分条件の判定順序を決定する．さらに，トリガグラフにおける部分条件判定のタイミングを 2 種類提案する．

©2015 Information Processing Society of Japan

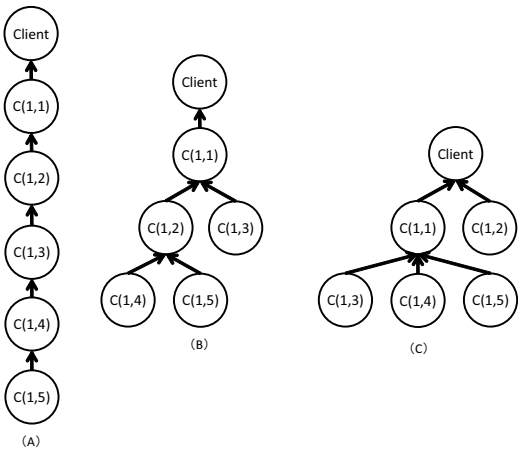


図 3 単純なトリガグラフの例
Fig. 3 An example for a simple trigger graph

4.1 部分条件判定のトリガグラフ

トリガグラフとは，ある部分条件が満たされた場合に次に判定する（トリガをかける）部分条件を示すグラフである．通知先クライアントが根となり，各部分条件が節点の木構造のトリガグラフとなる．トリガグラフの平均通信量は，$ComF$ と $TProb$ から算出できる．最大通信ホップ数はトリガグラフの形状から算出できる．真偽判定の順番に制約がなければ，ストリーミング処理を行う様々な形状のトリガグラフが考えられる．

例えば，3.3 節の例の場合，クライアントおよび $C(1,i)$ $(i = 1, \cdots, 5)$ が節点となり，単純なトリガグラフとして**図 3**(A) が考えられる．図 3(A) では，$C(1,5)$ が真であれば $C(1,4)$ の部分条件管理センサデータサーバに通知され，$C(1,4)$ が真であればさらに $C(1,3)$ の部分条件管理センサデータサーバに通知されるといったように，順番に通知される．最大通信ホップ数は 5 ホップになる．1 回の通知の通信量を N とすると，部分条件管理センサデータサーバ間の平均通信量は下記で与えられる．

$$ComF_{MServ(1,i)\to MServ(1,i-1)}(1,i) = N \times \Pi_{j=i}^{5} TProb(1,j) \quad (2)$$

ただし，$MServ(1,0)$ をクライアントと考える．

また，図 3(B) のような 2 分木のトリガグラフも考えられる．この場合の最大通信ホップ数は 3 ホップになる．さらに，図 3(C) のようなトリガグラフも考えられる．平均通信量は (B) より大きくなるが，最大通信ホップ数を抑えられる．図 3 の (A) のように順番に処理することで平均通信量を最も削減できるが，最大通信ホップ数が大きくなって通信遅延が長くなってストリーミング処理時間も長くなる．

部分条件が満たされた場合，深さが深く葉に近い部分条件から根の方に向かって通知されるため，真になる確率の小さい部分条件を葉の方に配置することで通信量を削減できる．そこで，提案手法では，真になる確率の小さい部分条件をトリガグラフの葉の方へ配置したトリガグラフを作成する．

4.2 部分条件判定のタイミング

各部分条件の判定を行うタイミングによってストリーミング処理時間が異なる．本研究では，部分条件を判定するタイミングとして，以下の 2 種類を考える．

4.2.1 独立判定法

$MServ(n,i_j)$ は他の部分条件の真偽に関わらず常に $C(n,i_j)$ の判定を行う．$MServ(n,i_j)$ が部分条件を判定するための通信が常に発生するが，あらかじめ真偽を判定しているため，次の順次判定法と比べて $MServ(n,i_j)$ が $MServ(n,i_{j+1})$ に通知するまでの時間を短くできる．各センサデータサーバは，$Servs$ に自身が含まれる部分条件の部分条件管理センサデータサーバに，センサデータが発生する度にセンサデータを送信する．

各部分条件管理センサデータサーバは，判定結果が出る度に真偽結果を，トリガグラフ上で親となる部分条件管理サーバに通知する．

4.2.2 順次判定法

$MServ(n,i_j)$ はある子ノードからトリガを受け取ってから $C(n,i_j)$ の判定を行う．独立判定法と比べて通知するまでの時間が長くなるが，$MServ(n,i_j)$ は通知を受け取った時のみ部分条件を判定するため，判定に必要な通信を削減できる．各センサデータサーバは，$Servs$ に自身が含まれる部分条件の部分条件管理センサデータサーバから確認のメッセージが送信された場合にセンサデータを送信する．

各部分条件管理センサデータサーバは，自身の真偽判定が偽であれば他の部分条件の真偽結果を確認する必要がないため，トリガを受けると，自身の真偽判定を行い，真であれば，トリガグラフ上で，トリガをかけた子ノード以外の子ノードの部分条件管理サーバの真偽判定の結果を確認する．確認の回数を確率的に減らすため，子が複数ある場合，真になる確率が低い部分条件の部分条件管理センサデータサーバから順番に確認を行う．すなわち，$C(n,i_j)$ の K 個の子ノードが $C(n,k_1), \cdots, C(n,k_K)$ で $TProb(n,k_1), \cdots, TProb(n,k_K)$ とすると，$C(n,k_1)$ が $C(n,i_j)$ にトリガをかける子ノードで，$C(n,k_2), \cdots, C(n,k_K)$ の順に確認を行う．トリガグラフの作成時に親ノードにトリガをかける子ノードを決定し，他の子ノードは確認された時のみ真偽を親ノードに伝えるようにすることで，親ノードが異なる複数の子ノードからトリガをかけられることを防げる．冗長な通信を削減するため，真偽判定結果が偽であれば，他の子への確認を中止する．

各方法の処理開始のタイミングを**図 4** を用いて説明する．3.3 節の例と同様に部分条件 $C(1,5)$ の部分条件管理センサデータサーバ $MServ(1,5)$ が A で，$Servs(1,5) = \{AB\}$

©2015 Information Processing Society of Japan

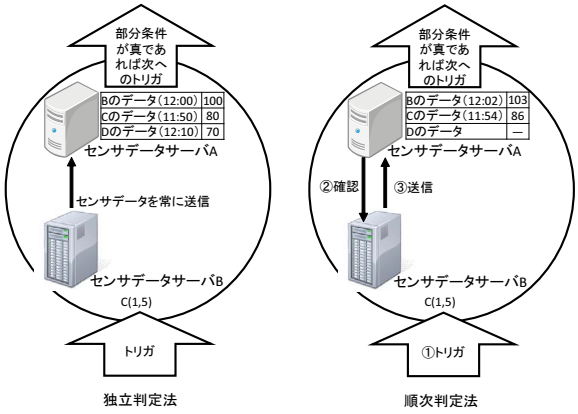


図 4 処理開始のタイミング
Fig. 4 Timing for staring sensor data processing

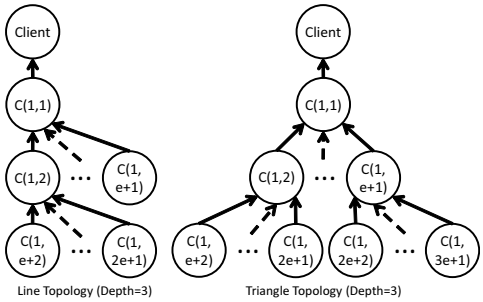


図 5 評価に用いたトポロジ
Fig. 5 Topologies for evaluation

の場合を考える．独立判定法では，センサデータサーバ B がセンサデータが発生する度にセンサデータサーバ A にセンサデータを送信する．センサデータサーバ A が他の部分条件の判定のためにセンサデータサーバ C や D からセンサデータを受信している場合には，それらの最新のセンサデータを保持している．部分条件管理センサデータサーバである A がトリガグラフに沿ってトリガを受信すると，保持しているセンサデータを用いて部分条件を判定し，結果が真であれば次の部分条件へのトリガを送信する．

順次判定法では，センサデータサーバ A がトリガを受信すると，$Servs(1,5)$ に含まれる自身以外のセンサデータサーバに確認を行い，最新のセンサデータを受信する．センサデータサーバ A が保持するセンサデータは，トリガを受信した時点で各センサデータサーバが保持する最新のセンサデータとなる．部分条件管理センサデータサーバである A が部分条件の判定に必要なすべてのセンサデータを受信すると，部分条件を判定し，結果が真であれば次の部分条件へのトリガを送信する．

5. 評価

本章では，提案手法の評価を行う．文献 [8], [9], [10] といったセンサデータが公開されているが，ストリーミング処理の連続問合せ対象となるセンサデータは様々であるため，ここでは評価用に人工的に発生させたストリーミングデータを用いる．また，ストリーミング処理時間はセンサデータサーバの処理能力等に依存するため，最大ホップ数および平均通信量で提案手法を評価する．

最大ホップ数には，通知メッセージのホップ数も含まれており，例えばトリガグラフが図 3(B)（評価で用いる列トポロジで深さ 2，枝の数 2）で各部分条件のセンサデータサーバが 1 個の場合，独立判定法では，葉の部分条件管理センサデータサーバがセンサデータサーバからセンサデータを受信するのに 1 ホップ，根の部分条件管理センサデータサーバまで通知するのに 2 ホップ，クライアントに通知するのに 1 ホップで最大ホップ数は 4 ホップになる．逐次判定法では，各部分条件管理センサデータサーバがセンサデータサーバに真偽判定の結果を確認するのに 2 ホップずつかかり，部分条件管理センサデータサーバ間の通知に 4 ホップ（葉ノードがセンサデータを受信するのに 1 ホップ，各部分条件管理サーバが自身のセンサデータを確認して次の部分条件管理サーバに通知するのに 3 ホップ），クライアントの通知に 1 ホップかかるため，最大ホップ数は 15 ホップになる．

本評価では，提案手法をトリガグラフを用いない単純な従来手法と比較する．従来手法では，あるセンサデータサーバがセンサデータを取得する度に，問合せの処理に必要な真偽判定の結果を全てのセンサデータサーバに確認する．1 章で述べた通信時間短縮型と処理時間短縮型を組み合わせた手法に相当する．

5.1 評価環境

評価では，**図 5** に示す 2 種類のトリガグラフを用いる．列トポロジ（Line Topology）では，根意外で各深さの部分条件の数が等しい．各評価で枝の数 e を変化させて評価を行う．三角トポロジ（Triangle Topology）では，葉以外で各部分条件に同じ数の子の部分条件がある．列トポロジと同様に枝の数 e を変化させて評価を行う．各部分条件のセンサデータサーバは 1 台とした．IoT 環境の一例として，1 章で述べた警備会社の契約数を参考にし，独立判定法では，各センサデータサーバは 1 単位時間で 1M メッセージを部分条件管理サーバに送信する．各部分条件管理サーバも，1 単位時間で 100 万（1M）メッセージをトリガグラフの親ノードに送信することになる．逐次判定法においても，各センサデータサーバは 1 単位時間で 1M メッセージを部分条件管理サーバに送信する．トリガやセンサデータ，真偽判定の確認結果は 1 メッセージで送受信できるとし，逐次判定を開始する末端ノードは 1 単位時間で 1M メッセージを送信するものとした．$TProb(n,i)$ ($n = 1, \cdots, N$, $i = 1, \cdots, I_n$) は，評価で変化させる場合を除き，0.7 とした．

©2015 Information Processing Society of Japan

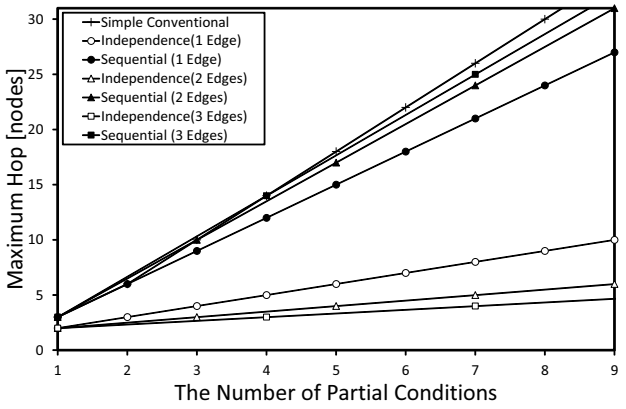


図 6 列トポロジの最大ホップ数
Fig. 6 Maximum hop for line topology

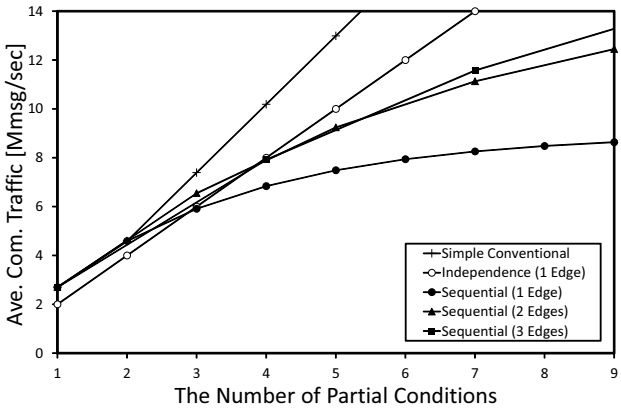


図 7 列トポロジの通信量
Fig. 7 Average traffic for line topology

5.2 列トポロジ

各深さの枝の数と深さを変化させ，列トポロジの最大ホップ数をコンピュータシミュレーションにより計測した．深さの基準は $C(1,1)$ であり，図 5 の列トポロジでは深さは 2 になる．結果を図 6 に示す．横軸は部分条件の数であり，例えば枝の数が 2，深さが 2 の時 5 となる．縦軸は最大ホップ数であり，逐次判定法では，判定を開始するノードからクライアントに結果が通知されるまでの最大の通信ホップ数となる．凡例中，‘Independence’ が独立判定法，‘Sequential’ が逐次判定法を示す．‘Simple Conventional’ は従来手法である．従来手法はトリガグラフを用いないため，枝の数を定義できない．

この結果より，独立判定法が，逐次判定法よりも最大ホップ数を抑えられ，ストリーミング処理にかかる時間を短縮できることが分かる．これは，独立判定法では，各部分条件管理サーバが，センサデータが発生する度に部分条件を判定しており，他の部分条件管理サーバからトリガを受信するとすぐに上位ノードに通知を送信できるためである．一方，逐次判定法では，トリガを受信してから部分条件を判定するため，部分条件判定に必要なセンサデータサーバとの通信が発生し，最大ホップ数が大きくなる．また，部分条件の数に比例して判定のための通信が発生してストリーミング処理時間が長くなっている．従来手法では，真偽判定の結果を全てのセンサデータサーバと通信して取得しているため，ホップ数が最も大きくなっている．

独立判定法では，部分条件の数が同じであれば枝の数が増えるほど深さが浅くなり，最大ホップ数が減少する．逐次判定法では，枝の数が増えるほど木構造の兄弟ノードとなる部分条件が多くなり，条件判定の通信が多くなって最大ホップ数も大きくなる．

次に，通信量を図 7 に示す．横軸は部分条件の数，縦軸は平均通信量であり，逐次判定法では，$TProb$ を考慮したうえで平均的な通信量を示している．独立判定法の通信量は枝の数に依存しないため，枝が 1 本の時のみ示している．

この結果より，部分条件の数が少ない場合には独立判定法の平均通信量が逐次判定法の平均通信量よりも少ないことが分かる．これは，逐次判定法では，判定に必要なデータをセンサデータサーバに要求する際，1 メッセージの通信が発生するためである．独立判定法では，継続的に部分条件管理サーバにセンサデータが送信されているため，この逐次判定法のような通信が発生しない．しかし，部分条件の数が多くなると，逐次判定法の平均通信量が独立判定法の平均通信量よりも少なくなっている．これは，各部分条件管理サーバで部分条件が満たされない場合に，トリガグラフで上位のノードにおいて部分条件を判定する必要がなく，これに伴う通信が発生しないためである．枝の数が多いほど発生する通信も増加するため，平均通信量が多くなっている．

従来手法はトリガグラフを用いておらず，提案手法のように部分条件の真偽判定が偽であれば通信を行わないといった工夫をしていないため，平均通信量が最も多くなっている

例えば，1 章で述べた例のように枝の数が 2，深さが 2 の場合，独立判定法の最大ホップ数は 3，平均通信量は 6M メッセージ/秒，逐次判定法の最大ホップ数は 10，平均通信量は 6.55M メッセージ/秒となり，独立判定法のストリーミング処理時間，通信量ともに逐次判定法よりも小さい値を与えている．部分条件の数が増えて 7（深さ 4）の場合，独立判定法の最大ホップ数は 5，平均通信量は 14M メッセージ/秒，逐次判定法の最大ホップ数は 24，平均通信量は 11.1315M メッセージ/秒となり，独立判定法のストリーミング処理時間が逐次判定法よりも短くなるが，通信量は逐次判定法の方が少なくなる．逐次判定法の方が通信量が少なるなる部分条件の数は，トリガグラフの形状に依存する．

5.3 三角トポロジ

各深さの枝の数と深さを変化させ，三角トポロジの最大

©2015 Information Processing Society of Japan

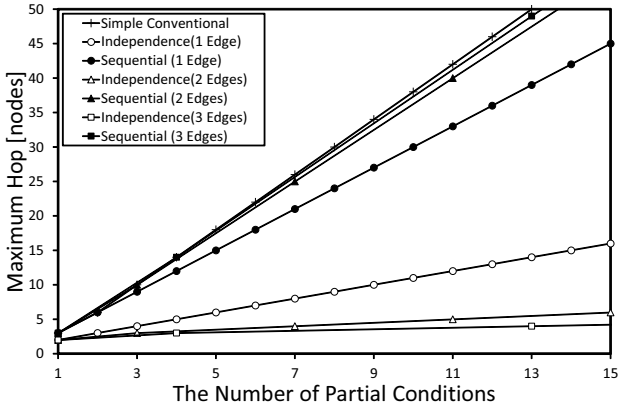


図 8 三角トポロジの最大ホップ数
Fig. 8 Maximum hop for triangle topology

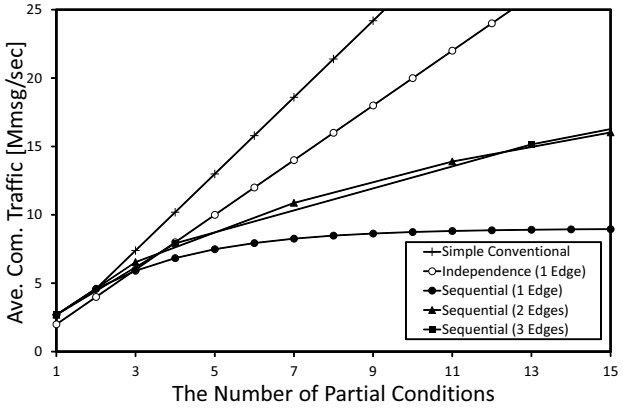


図 9 三角トポロジの通信量
Fig. 9 Average traffic for triange topology

ホップ数をコンピュータシミュレーションにより計測した．結果を図 8 に示す．横軸は部分条件の数であり，例えば枝の数が 2，深さが 2 の時 7 となる．縦軸は最大ホップ数である．凡例中，‘Independence’ が独立判定法，‘Sequential’ が逐次判定法，‘Simple Conventional’ が従来手法を示す．

この結果より，独立判定法が逐次判定法よりも最大ホップ数を抑えられ，ストリーミング処理にかかる時間を短縮できることが分かる．これは，独立判定法では，部分条件管理サーバからトリガを受信するとすぐに上位ノードに通知を送信できるが，逐次判定法では，トリガを受信してから部分条件を判定しているためである．列トポロジと同じく，独立判定法では枝の数が増えるほど最大ホップ数が小さくなり，逐次判定法では増加している．

次に，通信量を図 9 に示す．横軸は部分条件の数，縦軸は平均通信量であり，列トポロジの場合と同じく，逐次判定法では，$TProb$ を考慮したうえで平均的な通信量を示している．独立判定法の通信量は枝の数に依存しないため，枝が 1 本の時のみ示している．

この結果より，部分条件の数が少ない場合には独立判定法の平均通信量が逐次判定法の平均通信量よりも少ないことが分かる．これは，逐次判定法では，判定に必要なデータをセンサデータサーバに要求する際，1 メッセージの通信が発生しているためである．枝の数が多いほど発生する通信も増加するため，通信量が多くなっている．

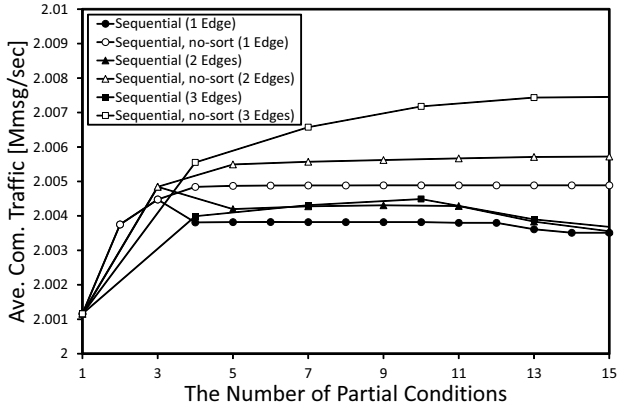


図 10 真になる確率の考慮の有無による差異
Fig. 10 Difference caused by true probability

5.4 真になる確率を考慮する効果

提案手法では，$TProb$ が小さい部分条件をトリガグラフの葉の方に配置させることで，通信量を削減させている．この効果を調べるために，提案手法とランダムに配置する手法を比較した．独立判定法については，真偽判定の通知を定期的に行っていて $TProb$ の影響がないため，逐次判定法で調べた．真になる確率は最大ホップ数に影響を及ぼさないため，トリガグラフの形状が同じであれば，配置が変わっても変化はない．

通信量をシミュレーションで求めた結果を図 10 に示す．横軸は部分条件の数，縦軸は平均通信量であり，”no-sort” が $TProb$ の値を考慮しない場合である．結果を見やすくするため，平均通信量の最小値を 2，最大値を 2.01 としている．トリガグラフは列トポロジを用いた．

この結果より，真になる確率を考慮することで，平均通信量を抑えられていることが分かる．これは，連続問合せの条件が満たされない場合には，センサデータを送信しないことで，通信量を削減できるためである．部分条件の数が 13 の場合に提案手法で平均通信量が小さくなっているのは，$TProb$ をランダムに与えており，$TProb$ の最小値が小さくなったためである．真になる確率を考慮することによる平均通信量の変化は，$TProb$ の差に比例する．

5.5 真の確率の影響

部分条件が真になる確率が，通信量に影響を与えるため，$TProb$ を変化させて通信量をコンピュータシミレーションにより求めた．最大ホップ数は，トリガグラフの形状にのみ依存するため，$TProb$ が変化しても影響はない．結果を図 11 に示す．トリガグラフは列トポロジを用い，深さ 3 とした．横軸は $TProb$ の値，縦軸は平均通信量である．

©2015 Information Processing Society of Japan

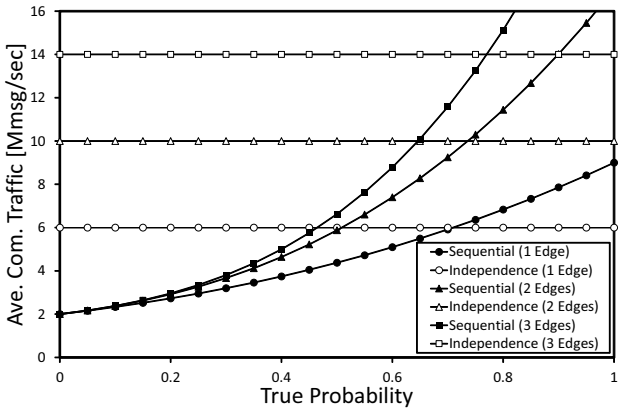


図 11 真になる確率と通信量
Fig. 11 Average traffic and *TProb*

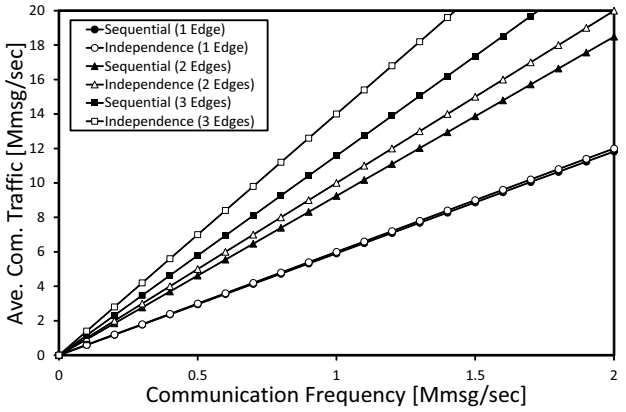


図 12 送信周期と通信量
Fig. 12 Average traffic and collection cycle

この結果より，真になる確率が小さい場合には逐次判定法の平均通信量が独立判定法より少なくなっていることが分かる．これは，前節で説明したとおり，逐次判定法では，トリガグラフの上位ノードにおける通信を削減できるためである．真になる確率が大きくなると，逐次判定法では，センサデータ取得のためのメッセージが必要なため，独立判定法に比べて平均通信量が多くなっている．独立判定法では，部分条件の真偽に関わらず継続的に各部分条件管理サーバが部分条件を判定しているため，平均通信量は一定になっている．

枝の数が多くなるほど，通知メッセージの数も増えるため，平均通信量も大きくなっている．

5.6 送信周期の影響

ストリーミング処理では，継続的にセンサデータが発生するため，発生の頻度によって通信量が変化する．本評価では，定期的にセンサデータが発生すると想定しているため，周期を変化させて通信量の変化を調査した．最大ホップ数は，送信周期が変化しても影響はない．結果を図 12 に示す．トリガグラフは列トポロジを用い，深さ 3 とした．横軸は送信周期，縦軸は平均通信量である．

この結果より，送信周期に比例して通信量が増加しており，独立判定法と逐次判定法の通信量の大小に変化がないことが分かる．これは，本評価では各部分条件の判定に必要なセンサデータをもつセンサデータサーバの数が一定であり，独立判定法，逐次判定法ともに送信周期に比例して通信が発生するためである．各センサデータサーバの送信周期に違いがある場合には，比例して増加するとは限らない．

6. 考察

6.1 ストリーミング処理時間

本研究では，最大ホップ数および平均通信量で提案手法を評価した．最大ホップ数小さくなるほど，通信遅延が短くなってセンサデータが発生してからストリーミング処理を完了するまでの時間が短くなり，ストリーミング処理時間を短縮できる．また，平均通信量が小さくなるほど通信にかかる時間が短くなってストリーミング処理時間を短縮できる．通信帯域が小さかったり通信環境が不安定で遅延や再送が発生する場合には，平均通信量を少なくすることで，これらが発生する頻度が小さくなってストリーミング処理時間に時間がかかるのを抑えることができる．

具体的に求められるストリーミング処理時間は応用等によって異なるが，1 章で述べた事件発見等の例では，短くすることで即座に発見できる．提案手法を用いることでストリーミング処理時間を短くでき，これらの応用に有効と考えられる．

6.2 列トポロジと三角トポロジ

幾つかのトリガグラフの形状で評価を行うため，本研究では列トポロジと三角トポロジを用いた．本節では，これらのトポロジの比較議論を行う．**図 6**〜**図 9** に示す通り，部分条件の数が同じ場合，列トポロジより三角トポロジの方がトリガグラフの深さを抑えられるため，独立判定法において最大ホップ数を短くできる．逐次判定法では各部分条件管理センサデータサーバが逐次的に真偽判定を行うため，大きくなる．一方，平均通信量については，三角トポロジでは，偽になる場合に子の部分条件管理センサデータサーバへの真偽判定の確認回数を削減できる割合が高くなるため，列トポロジよりも削減できる．

7. まとめ

本研究では，IoT 環境において，センサが接続された処理サーバが分散してインターネットに接続されている点を考慮し，ストリーミング処理時間短縮手法を 2 種類提案した．提案手法では，各連続問合せを個別に処理できる幾つかの部分条件に分割し，処理結果が偽になる確率の高い部分条件から順番に処理する．独立判定法では，トリガグラフにおける下位ノードの部分条件の真偽に関わらず，部分

©2015 Information Processing Society of Japan

条件管理サーバは常に部分条件の判定を行う．順次判定法では，下位のノードが真になる場合のみ部分条件の判定を行う．評価の結果，逐次判定法に比べて独立判定法がストリーミング処理時間を短縮できるが，真になる確率が大きい場合には通信量が多くなることが判明した．また，提案手法は従来手法よりも最大通信ホップ数と平均通信量を削減できることを確認した．

今後，部分条件が真になる確率を考慮してトリガグラフの一部のみ逐次判定法を用いて処理時間を短縮する手法や，実システムの開発を考えている．

謝辞

本研究の一部は，文部科学省科学研究費補助金・基盤研究 (A)(26240013)，基盤研究 (B)(15H02702)，挑戦的萌芽研究（26540045）の研究助成によるものである．ここに記して謝意を表す．

参考文献

[1] S. Dutta, A. Narang, and S. K. Bera, “Streaming Quotient Filter: A Near Optimal Approximate Duplicate Detection Approach for Data Streams,” in Proc. of the VLDB Endowment, Vol. 6, No. 8, pp. 589-600 (2013).
[2] C. Lei, E. A. Rundensteiner, and J. D. Guttman, “Robust Distributed Stream Processing,” in Proc. of the International Conference on Data Engineering (ICDE2013), pp. 817-828 (2013).
[3] Q. Zhang, J. Liu, and W. Wang, “Approximate Clustering on Distributed Data Streams,” in Proc. of the International Conference on Data Engineering (ICDE2008), pp. 1131-1139 (2008).
[4] ALSOK, Number of Contracts, http://www.alsok.co.jp/en/ir/finance/contractor.html
[5] SECOM, 会社概要, http://www.secom.co.jp/corporate/outline/about.html
[6] N. Polyzotis, S. Skiadopoulos, P. Vassiliadis, A. Simitsis, and N. Frantzell, “Meshing Streaming Updates with Persistent Data in an Active Data Warehouse,” IEEE Transactions on Knowledge and Data Engineering, Vol. 20, No. 7, pp. 976-991 (2008).
[7] O. Papapetrou, M. Garofalakis, and A. Deligiannakis, “Sketch-based Querying of Distributed Sliding-Window Data Stream,” in Proc. of the VLDB Endowment, Vol. 5, No. 10, pp. 992-1003 (2012).
[8] Stanford Stream Data Manager, http://infolab.stanford.edu/stream/
[9] Open Sensor Data, http://opensensordata.net/
[10] SPIRIT: Streaming Pattern Discovery, http://www.cs.cmu.edu/afs/cs/project/spirit-1/www/

©2015 Information Processing Society of Japan