5U-01

センサネットワークにおける時刻同期の省資源性と計算時間に関する検討

秋本 ゆり † 黒木 琴海 † 倉田 成人 ‡† 渡辺 尚 ‡ 猿渡 俊介 ‡
† 静岡大学 ‡ 大阪大学 ‡† 筑波技術大学

1 はじめに

本研究では，BEMS (Building Energy Management System) によるエネルギーモニタリング [1] と構造モニタリング [2] との融合を目的として，多種多様なセンサから必要なデータを取得する仕組みを検討する．異なる種類のセンサからデータを取得するためには，サンプリングタイミングの同期と計算資源を少なくすることが必要である．そこで本稿では，多種多様なセンサから必要なデータを取得するための時刻同期方式として XenoSync を提案する．XenoSync を実装して性能を評価した結果，XenoSync が省資源かつ短い計算時間で高い時刻同期精度を達成していることが分かった．

2 異種センサネットワークの課題

筆者らは，BEMS と構造モニタリングの融合を目指している [1]．エネルギーモニタリングと構造モニタリングを融合させるためには多種多様なセンサを設置する必要がある．また，建物内に存在するセンサは，イーサネット，RS232C，無線 LAN，IEEE802.15.4 など多様なネットワークによって接続される．各ノードが具備している CPU も x86 などの PC 向けのものから，ARM やマイコンなど処理能力の幅が広い．本稿では，このような多種多様なネットワーク環境，計算資源によって構築されるセンサネットワークを異種センサネットワークと呼ぶこととする．異種センサネットワークを実現するためには 1. サンプリングタイミングを同期すること，2. 多様な遅延環境に対応できること，3. 少ない計算資源で実装可能なことの 3 つを同時に満たす必要がある．

3 提案手法

2 節で述べた要件を同時に実現するための手法として，時刻同期方式 XenoSync を提案する．通常のセンサネットワークにおける同期サンプリングは，タイムスタンプのサンプリング，時刻補正式の算出，センシングタイミングの同期の 3 つから構成される．それに対して，XenoSync では，通常の同期サンプリングのプロセスに対して新たに 1. 4 way ハンドシェイク，2. オフセットベースの軽量時刻補正式の算出，3. 絶対時刻アライメントによるオーバフロー処理の 3 つの仕組みを追加する．

3.1 4 way ハンドシェイクによる時刻誤差軽減

時刻同期では，一般的にリファレンスとなる絶対時刻をセンサノードで受け取った時のセンサノードの内部時刻のタイムスタンプを元に，絶対時刻と内部時刻との関係から補正式を算出する．FTSP[3] や NTP などの既存のセンサネットワーク向けの時刻同期プロトコルでは，通信遅延時間が少ないことや送受信の遅延が等しいことを前提に誤差の最小化を図っている．XenoSync では異種センサネットワークでの送受信の遅延が必ずしも等しくないことに着目して，4 way ハンドシェイクを行うことで誤差を軽減する．

図 1 に XenoSync におけるタイムスタンプサンプリングのシーケンス図を示す．XenoSync ではクライアントからの pull 型でサーバからリファレンスとなる絶対時刻を受け取る．具体的には，XenoSync クライアントから XenoSync サーバに対して probe を送信した後，probe を受け取った XenoSync サーバが probe を受け取った際の絶対時刻を計測して，ACK を返信する．この時，ACK にはタイムスタンプは含めない．ACK のサイズを小さくする事で RTT の精度を高める事ができる．ACK を受け取った XenoSync クライアントは内部時刻を計測して，time request を XenoSync サーバに送信する．time request を受け取った XenoSync サーバは time response を XenoSync クライアントに返信する．XenoSync クライアントは時刻計測パケットを受け取った際の内部時刻と絶対時刻を元に時刻補正式を算出する．

時刻計測用パケットは 1 バイトで構成されており，通信経路のハードウェア的な制約をできる限り受けないように最小限の情報で通信を行っている．例えば，USB 上ではフレーム単位で通信が行われており，フレームの送信は 1 ms 周期で行われている．USB が通信経路上にある場合，データが複数のフレームに分割されると最低でも 1 ms の誤差が出てしまう．送信するデータを最小限にした場合ではフレームに分割されないため誤差を抑制できる．

3.2 オフセットベースの軽量時刻補正式

オフセットベースの軽量時刻補正式は，センシングタイミングの同期を行う際に利用する．FTSP や PulseSync などの IEEE 802.15.4 を前提とした既存のセンサネットワーク向けの時刻同期方式では，リファレンスとなる絶対時刻と各クライアントが具備するローカルの時刻の関係を最小二乗法によって求めた補正式でモデル化して時刻同期を行っている．絶対時刻を G，センサノードが具備するローカルの時刻を L とすると，補正式は

$$G = aL + b \tag{1}$$

となる．a は傾き，b は切片であり，複数のタイムスタンプから最小二乗法を用いて算出される．例えば CPU の 2 MHz の内部タイマを使って内部時刻を生成していた場合，a は内部時刻が進む速度であるため 0.5 μs となる．

しかしながら，最小二乗法によって傾き a と切片 b を求める場合，次の 2 つの問題が発生する．1 つ目は，過去のタイムスタンプを記録するためにメモリを消費することである．一般的に，最小二乗法を用いた時刻同期では過去のタイムスタンプが多ければ多いほど精度が高くなるため，精度を求めるとメモリを多く消費してしまう．2 つ目は，最小二乗法で傾き a と切片 b を求める際の計算コストがかかることである．最小二乗法では多くの乗算が生じるため，省資源のセンサノードの場合には実装が困難となる．

このような観点から，XenoSync では，最小二乗法を用いない代わりにタイムスタンプの交換頻度を高くする事で同期精度を維持する．傾き a の短期間での変動は小さいという前提の下に，傾き a を既知の固定値として，切片 b をタイムスタンプを受信するたびに更新する．具体的には，タイムスタンプを受信した際に

$$b = G - aL \tag{2}$$

によって切片 b を算出する．

3.3 絶対時刻アライメントによるオーバフロー処理

絶対時刻アライメントによるオーバフロー処理では，省資源で同期サンプリングを行う機構を提供する．全てのセンサノードは絶対時刻を基準に同期のタイミングをそろえる．例えば 100 Hz で同期サンプリングを行う場合には，絶対時刻のミリ秒部分が 10 ms の倍数の時にサンプリングを行う．現在のサンプリングのタイミング G_{current} が 18 時 03 分 0.02 秒だとすると，次のサンプリングのタイミング G_{next} は G_{current} に 0.01 秒を加算して算出する．絶対時刻 G_{next} を補正式を用いてセンサノードの内部時刻 L_{next} に変換した後に，L_{next} をセンサノードの内部タイマに設定して次のサンプリングのタイミングで割り込みが発生するように設定する．

このような同期サンプリングを小さなサイズの変数を用いて実現した場合，オーバフローが問題になる．オーバフローしないように

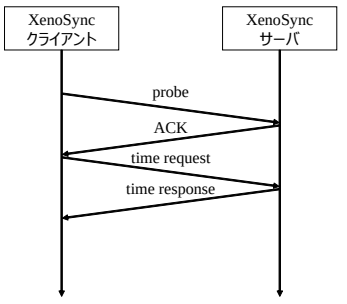

図 1: タイムスタンプサンプリング

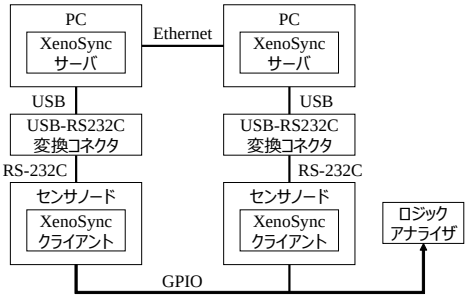

図 2: 同期性能評価環境

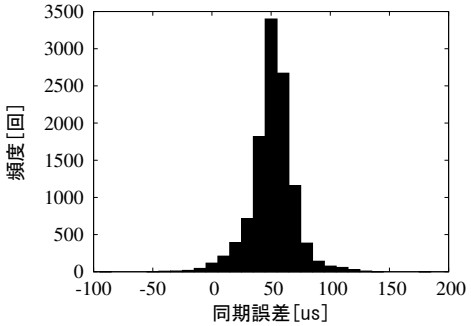

図 3: 同期誤差

表 1: 各モジュールの footprint

モジュール名	RAM [B]	ROM [B]
ハードウェア初期化	2	242
加速度サンプリング	6	226
デバッグ用の関数群	266	1,924
XenoSync	140	2,887
合計	414	5,279

大きなサイズの変数を用いると，メモリを多く消費するだけでなく演算コストも大きくなる．例えば，絶対時刻を μ 秒単位で扱った場合，UNIX 時間を 32 bit を基準とすると最低でも 52 bit の変数で扱う必要が生じる．このような同期サンプリングを省資源で実現する場合の問題に対して，XenoSync では絶対時刻の補正と切片 b の補正の 2 つの仕組みを導入する．

1 つ目の絶対時刻の補正では，同期サンプリングの間隔の倍数でアライメントを行う．絶対時刻をアライメントする際の絶対時刻の最大値を $G_{\max}$ とする．例えば，100 Hz の同期サンプリングを行っている場合，サンプリング間隔は 10 ms となる．PC から絶対時刻を送る際にサンプリング間隔の倍数である 1,000 ms で絶対時刻をアライメントすることで，絶対時刻を μ 秒単位で扱って 32 bit 変数で絶対時刻がオーバフローした場合でも正しいタイミングで同期サンプリングをすることができる．

2 つ目の切片 b の補正では，次のサンプリングタイミングの絶対時刻が $G_{\max}$ を超えた場合に絶対時刻と切片 b の値に矛盾が生じないように，切片 b を 0 に補正する．具体的には，更新後の切片を b'，内部時刻の最大値を $L_{\max}$ とおくと，

$$b' = b - \left(G_{\max} - aL_{\max} \lfloor \frac{G_{\max}}{aL_{\max}} \rfloor \right) \quad (3)$$

として絶対時刻と切片 b の矛盾を補正する．

4 評価

3 節に示した XenoSync のクライアントとサーバをそれぞれセンサノードと PC 上に実装した．図 2 に実装した評価環境を示す．センサノードは，RAM が 4 KB，ROM が 48 KB，動作周波数が 14.7456 MHz の Microchip 社の PIC18LF8527 を CPU として用いて，加速度センサを具備した加速度センサノードである．PC は，Panasonic 社の Let's note CF-B11QWBR である．PC 上のオペレーティングシステムとしては Linux のディストリビューションである Ubuntu Server 14.04.3 LTS を用いた．センサノードと PC は USB-RS232C 変換コネクタである BUFFALO 社の BSUSRC0610BSRS232C を介して接続されている．各センサノードが GPIO 経由で出力するサンプリングタイミングをロジックアナライザである TechTools 社の DV1-100 で取得した．XenoSync の性能を相対的に評価するため，最小二乗法を用いた時刻同期方式を実装して比較対象として用いた．

4.1 時刻同期精度

XenoSync の時刻同期性能を評価するために，センサノード間の同期誤差を計測した．図 3 に同期誤差の度数分布を示す．横軸が同期誤差 [μs]，縦軸が頻度を表している．図 3 から，以下の 2 つのことが分かる．

1 つ目は，同期誤差が± 200 μs 以内に収まっていることである．構造モニタリングの観点では，100 Hz のサンプリング時には全てのセンサのサンプリングが 1 ms 以下のずれに収まっていることが望ましいため，XenoSync は十分な同期精度を達成していると言える．

2 つ目は，オフセットが発生していることである．図 3 を見ると正規分布に従った形で約 50 μs のずれが生じており，平均値も 46 μs となっている．このようなオフセットが生じたのは，クライアントとサーバ間の通信遅延が原因の 1 つとなっていると考えられる．

4.2 省資源性

XenoSync の省資源性を評価するために，実装した XenoSync クライアントの各モジュールと比較対象である最小二乗法を用いた時刻同期方式の footprint を計測した．最小二乗法を用いた時刻同期方式では過去のタイムスタンプのデータを使用するため，最小二乗法を用いた時刻同期方式に関しては，計算に使用するデータ数を 4 から 256 に変化させてデータ数ごとの RAM 使用量を計測した．

表 1 に XenoSync を実装したセンサノードのハードウェア初期化，加速度サンプリング，デバッグ用の関数群，XenoSync の各要素におけるデータメモリ (RAM) とプログラムメモリ (ROM) の使用量を示す．ハードウェア初期化は RAM が 2 B，ROM が 242 B，加速度サンプリングは RAM が 6 B，ROM が 226 B，デバッグ用の関数群は RAM が 266 B，ROM が 1,924 B，XenoSync は RAM が 140 B，ROM が 2,887 B であった．合計すると RAM が 414 B，ROM が 5,279 B 必要となる．センサノードが 1 KB 程度の RAM しか持たない場合でも XenoSync は実装できるため，少ない計算資源で実装可能であることが分かる．

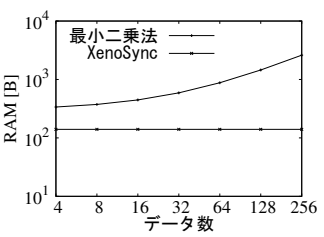

図 4: データ数ごとの RAM 使用量

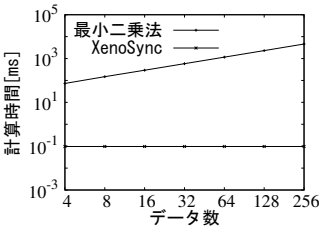

図 5: データ数ごとの計算時間

図 4 に XenoSync と最小二乗法を用いた時刻同期方式のデータ数ごとの RAM 使用量を示す．横軸がデータ数，縦軸が RAM 使用量 [B] を表している．XenoSync の値は表 1 における XenoSync の RAM 使用量と同様である．図 4 より，以下の 2 つのことが分かる．

1 つ目は，最小二乗法を用いた時刻同期方式と比べて XenoSync の RAM 使用量が小さくなっていることである．例えば，データ数が 4 の場合の最小二乗法を用いた時刻同期方式と比べると，XenoSync は RAM 使用量を 59 %削減できている．これは，最小二乗法を用いた時刻同期方式では，計算コストが高く演算時に変数として確保している RAM の領域が多いのに対して，XenoSync では式 (2) で示した単純な式を用いて演算が行われており，演算時に多くの変数を確保する必要がないからだと考えられる．

2 つ目は，最小二乗法を用いた時刻同期方式ではデータ数が増えるにつれて RAM 使用量が増えていることである．例えば，データ数が 4 の場合は 338 B，データ数が 256 の場合は 2,606 B となっている．これは，データ数が増えると RAM に変数として確保する必要のある領域が増えるからだと考えられる．

4.3 計算量

XenoSync の計算量を相対的に評価するため，最小二乗法を用いた時刻同期方式と XenoSync の計算時間を計測した．4.2 節と同様に，最小二乗法を用いた時刻同期方式に関しては計算に使用する過去のタイムスタンプのデータ数を 4 から 256 の範囲で変化させた場合のデータ数ごとの計算時間を計測した．データ数ごとの計算時間の計測結果を図 5 に示す．横軸がデータ数，縦軸が計算時間 [ms] を表している．XenoSync は過去のデータを用いないため，データ数に関わらず同じ計算時間となっている．図 5 より，以下の 2 つのことが分かる．

1 つ目は，最小二乗法を用いた時刻同期方式と比べて XenoSync の計算時間が短くなっていることである．例えばデータ数が 4 の場合，最小二乗法を用いた時刻同期方式の計算時間が 73.774 ms であるのに対して XenoSync の計算時間は 0.097 ms となっており，最小二乗法を用いた時刻同期方式の 99.867 %の計算時間を削減できている．これは，最小二乗法を用いた時刻同期方式では過去のデータを用いた複雑な計算が必要であるのに対して，XenoSync では式 (2) で示した単純な式を用いて計算を行っているからだと考えられる．

2 つ目は，最小二乗法を用いた時刻同期方式ではデータ数が増えるにつれて計算時間が長くなっていることである．例えば，データ数が 4 の場合の計算時間は 73 ms，データ数が 256 の場合の計算時間は 4,601 ms となっており，データ数に応じて線型的に計算時間が増加している．これは，データ数の増加に伴って計算回数も同様に増加するからだと考えられる．

5 おわりに

本稿では，多種多様なセンサから必要なデータを取得するための時刻同期方式「XenoSync」が省資源かつ短い計算時間で高い時刻同期精度を達成していることが示された．現在，実際の BEMS に適用した実証実験を進めている．

謝辞

本研究は JSPS 科研費 JP16H01718 の助成を受けたものである．本研究の遂行をサポートして下さった東京大学の馬郡文平氏，塩野禎隆氏，朝日機器株式会社の光山義紀氏に感謝致します．

参考文献

[1] 馬郡文平：コンビニエンスストアーにおける環境・省エネルギーのための快適 AI 制御，計測と制御， Vol. 52, No. 11, pp. 999–1002 (2013).

[2] Kurata, N., Suzuki, M., Saruwatari, S. and Morikawa, H.: Actual Application of Ubiquitous Structural Monitoring System using Wireless Sensor Networks, *Proceedings of the 14th World Conference on Earthquake Engineering (14WCEE)*, pp. 1–9 (2008).

[3] Maróti, M., Kusy, B., Simon, G. and Lédeczi, Á.: The Flooding Time Synchronization Protocol, *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (ACM SenSys'04)*, pp. 39–49 (2004).