Vol.2015-MBL-77 No.21
Vol.2015-ITS-63 No.21
2015/12/3

軍艦島センサネットワークのための タスクスケジューリングの設計と評価

黒木 琴海[1,a)]　小寺 志保[2,b)]　倉田 成人[3,c)]　濱本 卓司[4,d)]　猿渡 俊介[2,e)]

概要：本稿では，現在筆者らが進めている軍艦島モニタリングプロジェクトにおけるタスクスケジューリングについて述べる．軍艦島モニタリングプロジェクトは，現在建築構造物の崩壊が進んでいる軍艦島において映像や音声，加速度データといった建築構造物の崩壊現象のデータを収集し，建築構造解析に貢献することを目的としている．太陽光発電を電源として用いるモニタリングシステムにおいてエネルギーの利用を効率化するため，本稿では DC-LQ (Data Centric LQ-Tracker) と BLT Allocation (Battery Level Task Allocation) の 2 つの手法から構成されるデータ中心型のタスクスケジューリング方式として，BAAD scheduling (BAttery Aware Data centric scheduling) を提案する．シミュレーションによる評価の結果，BAAD scheduling が優れた性能を持っていることがわかった．

キーワード：軍艦島モニタリング，タスクスケジューリング

1. はじめに

センサネットワークの発達によって，実空間におけるありとあらゆるデータを取得することができるようになった．さらに，センサネットワークを他の分野の学問と組み合わせ，人々の生活に新たな価値をもたらすことも可能になりつつある．センサネットワークと建築の分野を融合した技術に，構造ヘルスモニタリングがある．構造ヘルスモニタリングとは，建築構造物にセンサを張り巡らせ，センサから取得したデータを用いて建築構造物の安全性を分析する技術である．構造ヘルスモニタリングが発達すると，建築構造物が崩壊する予兆を検知し，崩壊する前に人々を避難させることも可能になる．

しかしながら，現在の構造ヘルスモニタリングでは，基準となる建築構造物が崩壊する際のデータが少なく，建築構造物の崩壊現象の予兆を検知することは極めて困難である．実際の建築構造物において崩壊するデータを取得しようとすると，人命が危険にさらされてしまうため，データを取得することができない．現在はシミュレーションを用いて人為的に崩壊現象を発生させているが，経年劣化などによる複雑な崩壊現象までは網羅できない．

実際の建築構造物が崩壊するデータを取得するため，筆者らは，軍艦島で崩壊現象のビッグデータを収集する軍艦島モニタリングプロジェクト [1] に取り組んでいる．軍艦島は，今まさに建築構造物の崩壊が進んでいる環境であり，経年劣化などによる建築構造物の複雑な崩壊現象のデータが取得できる．軍艦島モニタリングプロジェクトでは，建築構造物にセンサやカメラを設置し，映像や音声，加速度などのデータを取得する．取得したデータは無線通信を用いて本土に送信する．しかしながら，軍艦島には発電所がないため，軍艦島モニタリングシステムに必要な電力は全て太陽光発電で供給しており，使用できる電力には限りがある．

本稿では，限られたエネルギーから効率良くセンサデータを取得するためのタスクスケジューリング方式を検討する．限られたエネルギーから効率良くセンサデータを取得するためには，以下の 3 つの要件を同時に満たす必要がある．1 つ目は，複数のタスクをスケジューリングすることである．各センサには，データの取得や送信などの複数のタスクが存在する．複数のタスクをどのようにスケジューリングするかによって，取得するデータや消費電力が異なる．2 つ目は，センサから取得するデータ量を最大化することである．より多くのデータが取得できれば，詳細な分析が可能になる．3 つ目は，データロス時間の最大値を最小化することである．データロス時間を短縮することができれば，取得したデータの時間のばらつきを削減することができる．

本稿では，3 つの要件を同時に満たす，データ中心型のタスクスケジューリング方式として，BAAD scheduling (BAttery Aware Data centric scheduling) を提案する．BAAD scheduling は，DC-LQ (Data Centric LQ-Tracker) と BLT Allocation (Battery Level Task Allocation) の 2 つから構成される．DC-LQ では，エネルギーが枯渇しないように，センサノードの起動時間とスリープ時間を決定する．BLT Allocation では，センシングや通信といった複数のタスクから，システムのバッテリ残量を元に実行するタスクを決定する．シミュレーションを用いた評価によって，BAAD scheduling が多くのデータを取得しつつ，データロス時間の短縮を実現していることがわかった．

[1] 静岡大学情報学部
[2] 静岡大学大学院情報学研究科
[3] 筑波技術大学
[4] 東京都市大学
[a)] kuroki@aurum.cs.inf.shizuoka.ac.jp
[b)] kodera@aurum.cs.inf.shizuoka.ac.jp
[c)] kurata@a.tsukuba-tech.ac.jp
[d)] thama@tcu.ac.jp
[e)] saru@inf.shizuoka.ac.jp

Vol.2015-MBL-77 No.21
Vol.2015-ITS-63 No.21
2015/12/3

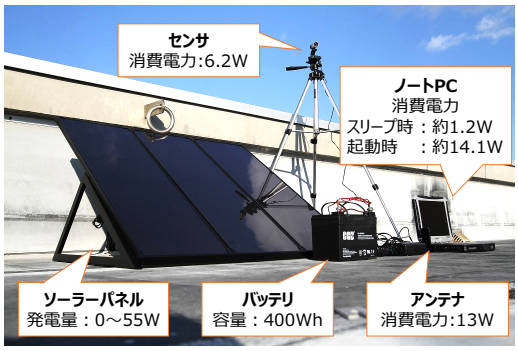

図 1　モニタリングシステム

本稿の構成は以下の通りである．2 節では，軍艦島モニタリングシステムの詳細を述べる．3 節では，タスクスケジューリングにおける課題について述べる．4 節では，提案手法である BAAD scheduling と，BAAD scheduling を構成する DC-LQ と BLT Allocation について述べる．5 節では，シミュレーションを用いて，既存タスクスケジューリング方式と BAAD scheduling を比較する．6 節では，既存研究について述べる．最後に 7 節でまとめとする．

2. 軍艦島モニタリング

軍艦島モニタリングプロジェクト [1] は，軍艦島において崩壊中の建築構造物の映像や音声，加速度といったデータを収集することで，建築構造解析に貢献することを目指すプロジェクトである．軍艦島は，長崎県にある無人島であり，現在も経年劣化による建築構造物の複雑な崩壊現象が発生している．軍艦島でデータを取得することで，建築構造物が実際に崩壊する瞬間のデータを取得することができるため，人為的に発生させた崩壊現象の検証では得られなかったデータを取得できると期待されている．

図 1 に，軍艦島モニタリングで利用するハードウェアと，それぞれのハードウェアにおける電力供給量，電力消費量を示す．軍艦島には電源もネットワークも存在しないため，太陽光発電を用いてセンシングシステムを駆動し，無線を用いて本土へとデータを送信する．電力を得るためのソーラーパネルとして公称最大出力が 55 [W] の Coleman PVS-55 W，400 [Wh] のバッテリ，ノート PC として TOUGHBOOK CF-195W1ACS を使用する．CF-195W1ACS にはセンサと無線通信用のデバイスが USB で接続されている．CF-195W1ACS の電力消費量を計測した結果，スリープ時で約 1.2 [W]，起動時で約 14.1 [W] であった．さらにセンサデータの取得に 6.2 [W]，本土への無線通信に 13 [W] の電力を消費する．

3. タスクスケジューリングにおける課題

軍艦島モニタリングに向けたタスクスケジューリングでは，センサノード上にあるセンシングや無線通信を用いたデータ送信などの様々なタスクを扱う．さらに，太陽光発電で得られた電力を利用するため，限られた電力をどのように利用するかが課題である．以上を踏まえて，軍艦島モニタリングに向けたタスクスケジューリング方式は次の 3 つの要件を同時に満たす必要がある．

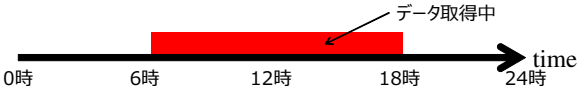

図 2　蓄積エネルギーがあれば直ちにデータを取得 (ASAP)

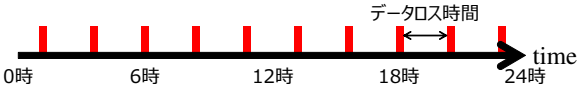

図 3　一定間隔でデータを取得

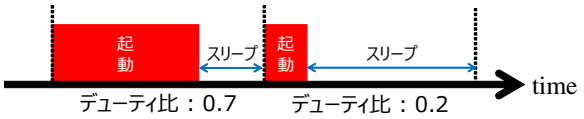

図 4　LQ-Tracker

1 つ目は，センシングや通信といった複数のタスクをスケジューリングすることである．タスクによって電力消費量が異なるため，センシングと通信をそれぞれどのようにスケジューリングするかによってエネルギーの消費量が変化する．また，センサデータの取得量やセンシングしてから本土へデータを送信するまでの遅延時間などが大幅に変化する．

2 つ目は，センサから取得するデータ量を最大化することである．取得するデータ量が少ない場合，時間経過によるデータの変化を発見できない可能性がある．例えば，センシング開始から 10 分後と 20 分後のデータしか取得できなかったとすると，15 分後に建物が崩壊するなどによってデータが大きく変化した場合，変化を知ることができない．取得するデータ量が多ければ多いほど，データの変化を発見できる可能性は高くなる．

3 つ目は，最悪データロス時間を最小化することである．データロス時間とはセンサノードが連続してデータを取得していない時間であり，最悪データロス時間とはデータロス時間の最大値である．例えば，同じデータ量において，最悪データロス時間が 1 時間のデータと 12 時間のデータを考える．最悪データロス時間が 1 時間のデータでは，最悪の場合でも，1 日のうち何時ごろその崩壊現象が起こったかを把握することができる．一方で，最悪データロス時間が 12 時間のデータでは，半日単位でしか崩壊現象のタイミングを把握することはできない．

センサから取得するデータ量を最大化する方法として，蓄積エネルギーがあれば直ちにデータを取得する方法 (ASAP:As Soon As Possible) が考えられる．図 2 に ASAP の例を示す．時間軸上に四角で示した部分でデータを取得する．蓄積エネルギーがある場合はデータを取得し続けるため，取得できるデータ量は最大となる．しかしながら，蓄積しているエネルギーが枯渇した場合，新たにエネルギーが蓄積されるまでデータを取得できないため，取得できるデータの時間帯に偏りが生じ，最悪データロス時間が大きくなる．図 2 の例では，18 時からのデータが全く取得できておらず，取得したデータはすべて日中のデータとなっている．

最悪データロス時間を小さくする方法として，一定間隔で

Algorithm 1 BAAD scheduling

1: **loop**
2: $B \leftarrow$ current battery level $\in [0, 1]$
3: $T_{\text{work}}, T_{\text{sleep}} \leftarrow \text{dclq}(B)$
4: $\text{blt}(B, T_{\text{work}})$
5: Sleep for T_{sleep} seconds
6: **end loop**

データを取得する方法が考えられる．図 3 に一定間隔でデータを取得する場合の例を示す．時間軸上に四角で示した部分でデータを取得する．一定間隔でデータを取得するため，取得したデータに時間帯の偏りは少なくなり，最悪データロス時間も小さくなる．しかしながら，指定した時間以外はデータを取得しないため，蓄積エネルギーに余裕がある場合でも取得できるデータ量は変化しない．

センサから取得するデータ量の最大化と，最悪データロス時間の最小化を達成する既存の研究として，LQ-Tracker [2] が挙げられる．LQ-Tracker は，適応制御理論を用いてセンサノードのデューティ比を決定することで，バッテリレベルを目標値に維持する．デューティ比とは，センサノードが起動とスリープを繰り返す中で 1 つのサイクルにおける起動時間の割合のことである．図 4 に LQ-Tracker の動作例を示す．デューティ比が 0.7 の場合，1 サイクルの 70%の期間はセンサノードを起動させる．1 サイクルごとにその時のバッテリ残量に従ってデューティ比を決定することで，常にバッテリ残量に応じたタスクスケジューリングが可能となる．しかしながら，LQ-Tracker はデューティ比が変わるたびに，データロス時間が変動するという課題がある．図 4 に示した LQ-Tracker では，1 サイクルの長さが固定であるため，デューティ比が 0.7 から 0.2 に変化すると，スリープ時間が急増する．スリープ時間の変動が大きくなると取得データに偏りが生じ，最悪データロス時間が増加する．

4. BAAD scheduling

3 節での議論を元に，センサから取得するデータ量の最大化と，データロス時間の最大値を両立する手法として，データ中心型タスクスケジューリング方式，BAAD scheduling を設計した．BAAD scheduling は，DC-LQ と BLT Allocation から構成される．DC-LQ は，センサノードの起動時間とスリープ時間を決定する．BLT Allocation は，センサノードが起動したときにバッテリ残量に応じてセンサノードで実行するタスクを決定し，実行する．

Algorithm 1 に BAAD scheduling のアルゴリズム，表 1 に *Algorithm* 1 で使用する関数と変数を示す．B はセンサノードの現在のバッテリレベルである．T_{work} はタスクの実行時間，T_{sleep} はセンサノードのスリープ時間である．$\text{dclq}(B)$ は B を元に T_{work} と T_{sleep} を決定する関数である．詳細は 4.1 節にて説明する．$\text{blt}(B, T_{\text{work}})$ は B を元に実行するタスクを決定し，実行する関数である．詳細は 4.2 節にて説明する．

Algorithm 1 では，まず DC-LQ を用いて，タスクの実行時間とセンサノードのスリープ時間を決定する．*Algorithm* 1 の 2 行目で現在のバッテリレベルを取得し，3 行目で T_{work} と T_{sleep} を決定する．

表 1 *Algorithm* 1 で使用する変数，関数

変数，関数	説明
B	センサノードの現在のバッテリレベル
T_{work}	タスクの実行時間
T_{sleep}	センサノードのスリープ時間
$\text{dclq}(B)$	B を元に T_{work} と T_{sleep} を決定する関数 詳細は 4.1 節
$\text{blt}(B, T_{\text{work}})$	B を元に実行するタスクを決定し，実行する関数 詳細は 4.2 節

次に，BLT Allocation を用いて，決定された実行時間とスリープ時間に応じてタスクを実行したのち，センサノードをスリープさせる．*Algorithm* 1 の 4 行目で B を元に実行するタスクを決定，T_{work} [sec] の間タスクを実行する．タスクが終了すると，6 行目で T_{sleep} [sec] の間センサノードをスリープさせる．T_{sleep} [sec] が経過するとセンサノードを再び起動させ，*Algorithm* 1 の 2 行目から繰り返す．

4.1 DC-LQ

DC-LQ では，3 節で述べた LQ-Tracker を用いて動的に決定されるデューティ比に従いながら，タスクの実行時間とセンサノードのスリープ時間を決定する．3 節で述べた通り，LQ-Tracker はデューティ比が変わるたびにスリープ時間が変動するため，提案手法である DC-LQ ではスリープ時間の変動が小さくなるように 1 サイクルの時間を調節する．

まず，センサノードがスリープ状態から復帰するたびに LQ-Tracker によってデューティ比を決定する．次に，決定したデューティ比を元にスリープ時間を算出する．スリープ時間 T_{sleep} [sec] を式 (1) に示す．

$$T_{\text{sleep}} = \begin{cases} T_{\text{cyclebase}}(1 - D) & \text{(初回)} \\ T_{\text{cyclebase}}(1 - D)\alpha + T'_{\text{sleep}}(1 - \alpha) & \text{(2 回目以降)} \end{cases} \quad (1)$$

$T_{\text{cyclebase}}$ [sec] は 1 サイクルの基準となるあらかじめ決められた時間，D は LQ-Tracker によって決定されたデューティ比を意味している．α はスリープ時間の変動をどの程度許容するかを示す定数であり，0 から 1 の範囲で決定する．初回の T_{sleep} は事前に決定する $T_{\text{cyclebase}}$ を 1 サイクルとして，デューティ比に従った場合のスリープ時間を設定する．$T_{\text{cyclebase}}$ が短いときには頻繁に起動とスリープを繰り返すためデータロス時間は短くなるが，センサノードの起動で消費する電力が大きくなるため，データを取得するための電力が少なくなりデータ取得量は小さくなる．一方，$T_{\text{cyclebase}}$ が長いときには起動とスリープの繰り返しが少なくなるためデータロス時間が長くなるが，一度センサノードが起動すると長期間データを取得できるためデータ取得量は多くなる．2 回目以降は，$T_{\text{cyclebase}}$ と新たなデューティ比から決定したスリープ時間を，前回のスリープ時間 T'_{sleep} に加算する．α と T'_{sleep} によって T_{sleep} の急激な変動を抑えることができる．α が小さいと，新たなデューティ比に

Algorithm 2 DC-LQ

Input: $B \in [0,1]$
Output: $T_{\text{work}}, T_{\text{sleep}}$
1: $D \leftarrow$ lq_tracker(B)
2: $T'_{\text{sleep}} \leftarrow$ previous sleep time
3: **if** T'_{sleep} does not exist **then**
4: $\quad T_{\text{sleep}} \leftarrow T_{\text{cyclebase}}(1-D)$
5: **else**
6: $\quad T_{\text{sleep}} \leftarrow T_{\text{cyclebase}}(1-D)\alpha + T'_{\text{sleep}}(1-\alpha)$
7: **end if**
8: $T_{\text{work}} \leftarrow \frac{T_{\text{sleep}}D}{1-D} - T_{\text{wake}}$

表 2 *Algorithm* 2 で使用する変数，関数

変数，関数	説明
B	センサノードの現在のバッテリレベル
T_{work}	タスクの実行時間
T_{sleep}	センサノードのスリープ時間
D	センサノードのデューティ比
$\text{l}_{\text{q\_tracker}}(B)$	B を元に LQ-Tracker を用いてデューティ比を決定する関数
T'_{sleep}	前回のスリープ時間
$T_{\text{cyclebase}}$	1 サイクルの基準となる時間
$\alpha \in [0,1]$	スリープの変動を許容する割合
T_{wake}	センサノードが起動にかかる時間

よるスリープ時間 $T_{\text{cyclebase}}(1-D)$ より前回のスリープ時間 T'_{sleep} に重きを置いて T_{sleep} を算出するため，T_{sleep} の変動が小さくなる．一方，α が大きいと，T'_{sleep} より $T_{\text{cyclebase}}(1-D)$ に重きを置いて T_{sleep} を算出するため，T_{sleep} の変動が大きくなる，α と $T_{\text{cyclebase}}$ に関してはそれぞれ 5.4 節，5.5 節で詳細に検証する．

式 (1) によりスリープ時間が決定されると，スリープ時間とデューティ比に応じ 1 サイクルの時間 T_{cycle} [sec] を決定する．T_{cycle} を式 (2) に示す．

$$T_{\text{cycle}} = \frac{T_{\text{sleep}}}{1-D} \tag{2}$$

$1-D$ は T_{cycle} に対するスリープ時間の割合であるため，T_{sleep} が T_{cycle} の $1-D$ を占めるように T_{cycle} を決定する．T_{cycle} が決定されると，タスクの実行時間 T_{work} [sec] を決定する．式 (3) に T_{work} を示す．

$$T_{\text{work}} = T_{\text{cycle}}D - T_{\text{wake}} \tag{3}$$

T_{wake} [sec] はセンサノードの起動にかかるシステム固有の時間を意味している．$T_{\text{cycle}}D$ はセンサノードが起動している時間全体である．センサノードが起動している時間全体から T_{wake} を引いた値が T_{work} となる．式 (2) と式 (3) より，T_{work} は式 (4) となる．

$$T_{\text{work}} = \frac{T_{\text{sleep}}}{1-D}D - T_{\text{wake}} \tag{4}$$

Algorithm 2 に DC-LQ のアルゴリズム，表 2 に *Algorithm* 2 で使用する変数を示す．B はセンサノードの現在のバッテリレベル，T_{work} はタスクの実行時間，T_{sleep} はセンサノードのスリープ時間，D はセンサノードのデューティ比である．

Algorithm 3 BLT Allocation

Input: $B \in [0,1], T_{\text{work}}$
1: **for** $i = 0$ to n **do**
2: $\quad$ **if** $B \geq B_i$ **then**
3: $\quad\quad$ invoke τ_i
4: $\quad$ **end if**
5: **end for**
6: Run tasks for T_{work} seconds

表 3 *Algorithm* 3 で使用する変数，関数

変数	説明
B	センサノードの現在のバッテリレベル
T_{work}	タスクの実行時間
n	タスクの数
$\tau = \{\tau_1, ..., \tau_n\}$	スケジューリングの対象となるタスク
B_i	タスク τ_i の実行バッテリレベル

lq_tracker(B)s はバッテリレベル B を元に LQ-Tracker を用いてデューティ比を決定する関数である．T'_{sleep} は前回のスリープ時間を表している．前述したとおり，$T_{\text{cyclebase}}$ は 1 サイクルの基準となる時間，α はスリープの変動を許容する割合となっており，それぞれ事前に決定される．T_{wake} はセンサノードが起動にかかる時間で，システム固有の値となっている．

DC-LQ はバッテリレベル B を入力とし，実行時間 T_{work} とスリープ時間 T_{sleep} を出力とする．*Algorithm* 2 の 1 行目ではバッテリレベル B を元に LQ-Tracker を用いてデューティ比 D を決定する．T'_{sleep} が存在しない場合，*Algorithm* 2 の 4 行目で式 (1) の初回の式を用いて T_{sleep} を算出する．一方，T'_{sleep} が存在する場合，*Algorithm* 2 の 5 行目から 6 行目で式 (1) の 2 回目以降の式を用いて T_{sleep} を算出する．T_{sleep} が決定すると，*Algorithm* 2 の 8 行目で式 (4) に従って T_{work} を決定する．

4.2 BLT Allocation

BLT Allocation では，センサノードがスリープから復帰した際に，バッテリレベルに応じて実行するタスクを決定して実行する．頻繁に実行する必要のないタスクはバッテリレベルの低いときには実行しないようにすることで，エネルギーの消費を抑えることができる．

Algorithm 3 に BLT Allocation のアルゴリズム，表 3 に *Algorithm* 3 で使用する変数を示す．B は現在のバッテリレベルである．n はタスクの数，τ はスケジューリングの対象となるタスクの集合を表している．B_i は τ_i の実行バッテリレベルである．センサノード上にタスク $\tau = \{\tau_1, \tau_2, ..., \tau_n\}$ が存在する場合，それぞれのタスク τ_i は，計算時間 C_i [sec] と，実行バッテリレベル $B_i \in [0,1]$ で特徴づけられる．

Algorithm 3 では，現在のバッテリレベル B と DC-LQ によって決定された T_{work} を入力として用いる．*Algorithm* 3 の 3 行目で現在のバッテリレベル B と i 番目のタスクの実行バッテリレベル B_i を比較する．現在のバッテリレベルが実行バッテリレベル以上だった場合，*Algorithm* 3 の 4 行目で i 番目のタスクを起動する．*Algorithm* 3 の 3 行目から 5 行目をタス

表 4　タスクの対応関係の例

タスク	計算時間 C_i	実行バッテリレベル B_i
τ_1	T_{work}	0
τ_2	T_{work}	0.8

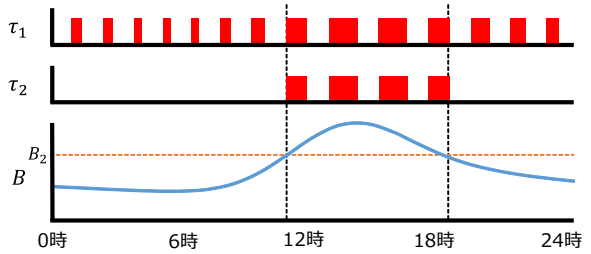

図 5　BLT Allocation

クの数だけ繰り返す．最後に 6 行目で起動したタスクを T_{work} [sec] の間実行する．

例として，軍艦島モニタリングにおける，タスク τ_i と実行バッテリレベル B_i，計算時間 C_i の対応関係を表 4 に示す．τ_1 は映像や加速度といったデータのセンシング，τ_2 は本土とのデータの通信のタスクを想定している．軍艦島モニタリングにおいてはより多くのデータを取得することが重要となるため，センシングのタスク τ_1 の実行バッテリレベル B_1 は 0 として常に実行されるように設定する．一方，データをリアルタイムで本土へ送信する必要はないため，通信のタスク τ_2 の実行バッテリレベル B_2 は 0.8 として，バッテリに余裕のある場合にのみ実行されるように設定する．

図 5 に表 4 のタスクを対象とした軍艦島モニタリングにおける BLT Allocation を示す．1 日のバッテリレベルが下のグラフの実線のように推移した場合，それぞれのタスクがどのようにスケジューリングされているかを表す．通信のタスク τ_2 は，バッテリレベルが実行バッテリレベル B_2 以上となる日中のみ実行されている．一方，センシングのタスク τ_1 は実行バッテリレベル B_1 が 0 であるため，バッテリレベルに関係なく実行される．

5. 評価

5.1 評価環境

BAAD scheduling の有効性を確認するために計算機シミュレーションにより取得データ量と最悪データロス時間を評価した．BAAD scheduling の性能を相対的に評価するために，以下の 3 つのタスクスケジューリング方式を比較した．

(1) ASAP (As Soon As Possible)
タスクスケジューリングを行わず，バッテリ残量がある限りセンシングと通信をし続ける手法である．バッテリ残量が枯渇するまでデータを取得するため，発電量におけるデータ取得量の目標値となる．

(2) LQ-Tracker
3 節で述べた，適応制御理論を用いたタスクスケジューリング手法である．30 分を 1 サイクルとしてセンシングと通信を実行する．

(3) BAAD scheduling
4 節で述べた提案手法である．DC-LQ と BLT Allocation を用いて，データ取得量の最大化と最悪データロス時間の最小化を実現する．

表 5　評価パラメータ

起動時の消費電力	14.1 W
スリープ時の消費電力	1.2 W
センシングの消費電力	6.2 W
通信の消費電力	13 W
ソーラーパネルの最大出力	55 W
バッテリの容量	400 Wh
T_{wake}	60 sec
1 秒で取得できるデータ量	1 MB

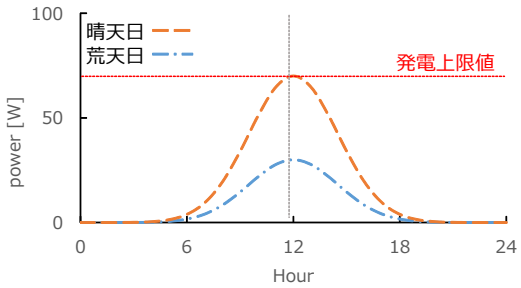

図 6　シミュレーションにおける発電量のモデル

共通のパラメータを表 5 に示す．図 1 に示した軍艦島モニタリングシステムに従って，システムの消費電力は起動時が 14.1 [W]，スリープ時が 1.2 [W] とした．システムは，センシング時に起動時の消費電力に加えてさらに 6.2 [W]，通信時に起動時の消費電力に加えてさらに 13 [W] 消費する．また，ソーラーパネルの最大出力は 55 [W]，バッテリの容量は 400 [Wh]，センサノードの起動にかかる時間 T_{wake} を 60 [sec]，1 秒で取得できるデータ量を 1 [MB] とした．BLT Allocation における通信の実行バッテリレベルは，軍艦島モニタリングを想定して 0.8 とした．

評価では，時間，季節，天候による太陽光発電の発電量を模擬するため，1 日の発電量の推移をモデル化した．図 6 に，太陽光発電による 1 日の発電量の推移のモデルを示す．時間における発電量の推移を模擬するため，発電量は午後 12 時を頂点とする正規分布に従うと仮定した．また，季節における太陽光の入射角の変化を模擬するため，発電上限値 (maximum generated power) を季節に応じて変化させた．さらに，天候による発電量の違いを模擬するため，1 日の始めに，午後 12 時時点での発電量を 0[W] から発電上限値までの範囲でランダムに決定した．図 6 の場合，晴天日は午後 12 時に発電上限値に到達するように発電量が推移するが，荒天日は発電量が少なくなるように発電量が推移する．7 日間のシミュレーションを 10000 回実行し，その平均値を評価として用いた．

5.2 発電上限値に対するデータ取得量の評価

BAAD scheduling がデータ取得量の最大化を実現できているかを評価するために，発電上限値に対するデータ取得量を取得した．発電上限値は 10 [W] から 100 [W] の範囲を 10 [W] 刻みで変化させた．また，BAAD scheduling における $T_{\text{cyclebase}}$

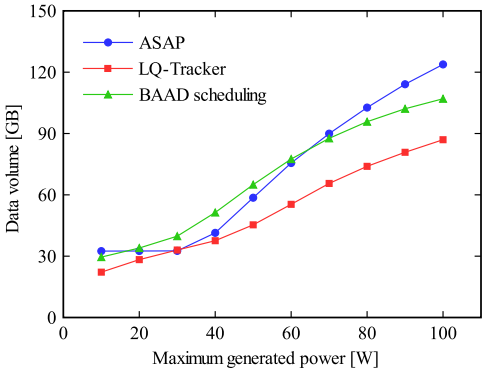

図 7 データ取得量

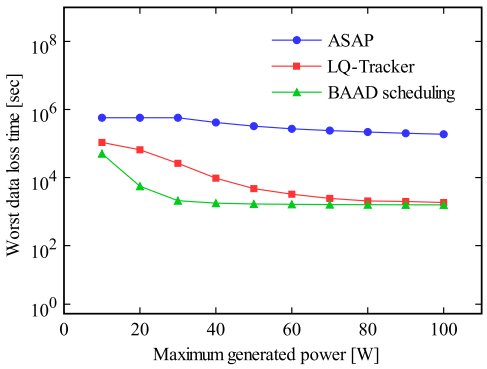

図 8 最悪データロス時間

は 1800 [sec]，α は 0.01 とした．図 7 に，発電上限値に対するデータ取得量を示す．縦軸がデータ取得量 [GB]，横軸は発電上限値 [W] である．図 7 から以下の 2 つのことがわかる．

1 つ目は，BAAD scheduling のデータ取得量が LQ-Tracker のデータ取得量を上回っていることである．例えば，発電上限値が 50 [W] のとき，BAAD scheduling は LQ-Tracker と比較して約 1.43 倍のデータ取得量を実現している．これは，BAAD scheduling が DC-LQ を用いて 1 サイクルの長さを動的に変化させることで，同じデューティ比でも多くのセンシング時間を確保できたためだと考えられる．また，LQ-Tracker は常にセンシングと通信を実行するが，BAAD scheduling は BLT Allocation によってバッテリ残量が少ない場合はセンシングと DC-LQ のみを実行するため，限られた電力でより多くのデータが取得できたと考えられる．

2 つ目は，発電上限値が高くなるにつれて ASAP のデータ取得量が BAAD scheduling のデータ取得量を上回っていることである．例えば，発電上限値が 30 [W] のときは BAAD scheduling が ASAP より約 1.22 倍のデータ取得量を実現しているが，発電上限値が 80 [W] のときには約 0.93 倍と少なくなっている．これは，発電上限値が高くなるにつれて，ASAP ではセンサノードの起動のために消費しているエネルギーが少なくなるためだと考えられる．発電上限値が低い場合，ASAP ではセンサノードを起動させるだけでエネルギーが枯渇することが多くなるため，データ取得量は少なくなる．一方，発電上限値が高い場合，センサノードを起動している期間が長くなるため，起動に消費するエネルギーが少なくなり，多くのデータを取得できる．

5.3 発電上限値に対するデータロス時間の評価

BAAD scheduling が最悪データロス時間の最小化を実現できているかを評価するために，発電上限値に対する最悪データロス時間を取得した．発電上限値は 10 [W] から 100 [W] の範囲を 10 [W] 刻みで変化させた．また，BAAD scheduling における $T_{\mathrm{cyclebase}}$ は 1800 [sec]，α は 0.01 とした．図 8 に発電上限値に対する最悪データロス時間を示す．縦軸が最悪データロス時間 [sec]，横軸が発電上限値 [W] である．図 8 から，以下の 2 つのことがわかる．

1 つ目は，BAAD scheduling が常に最も小さい最悪データロス時間を達成していることである．特に発電上限値が低い場合には，他の 2 つの手法より大幅に最悪データロス時間が小さくなっている．これは，BAAD scheduling が DC-LQ によってスリープ時間の変動を小さくしているためだと考えられる．

2 つ目は，発電上限値が高くなるにつれて，BAAD scheduling と LQ-Tracker の最悪データロス時間の差が小さくなっていることである．発電上限値が高い場合はセンサノードを何度も起動することができるため，データロス時間が短くなる．よって，LQ-Tracker において 1 サイクルの時間が固定であっても，最悪データロス時間を短縮できると考えられる．

5.4 α を変えた場合の評価

5.2，5.3 節の評価では，BAAD scheduling における α を 0.01 とした．本節では α の値によって BAAD scheduling の性能に変化が生じるかを検証した．具体的には，α を 10^{-5} から 10^{0} まで変化させた場合のデータ取得量と最悪データロス時間を取得した．α の変化による結果の変動を相対的に評価するため，データ取得量と最悪データロス時間のそれぞれについて結果の悪化率を評価した．結果の悪化率算出手順を以下に示す．

(1) 発電上限値が 10 [W]，50 [W]，100 [W] の場合において，それぞれ α を 10^{-5} から 10^{0} まで変化させて 7 日間のシミュレーションを 100000 回実行し，データ取得量と最悪データロス時間を計測する．
(2) 発電上限値が 10 [W]，α を 10^{-5} から 10^{0} まで変化させた結果の中で，データ取得量の最大値と最悪データロス時間の最小値をそれぞれ最も良い結果とする．
(3) データ取得量と最悪データロス時間の両方について，最も良い結果とそれ以外の結果の差分を算出する．差分を最も良い結果で割った値を結果の悪化率とする．
(4) 発電上限値が 50 [W]，100 [W] の場合も同様に算出する．

データ取得量と最悪データロス時間のどちらにおいても，悪化率が小さいほど高い性能であることを意味する．

図 9 に，α の値を 10^{-5} から 10^{0} の範囲で変化させた場合のデータ取得量と最悪データロス時間の悪化率を示す．$T_{\mathrm{cyclebase}}$ は全て 1800 [sec] である．横軸が α，縦軸が結果の悪化率である．図 9 から，2 つのことがわかる．

1 つ目は，α が小さくなるにつれて最悪データロス時間の悪化率が低下していることである．例えば発電上限値が 50 [W] の場合，α が 10^{-1} のとき最悪データロス時間の悪化率は約 13.8%であるが，α が 10^{-4} のとき悪化率は約 0.5%である．これは，α を小さくすることで，スリープ時間の変動が抑えられたためだ

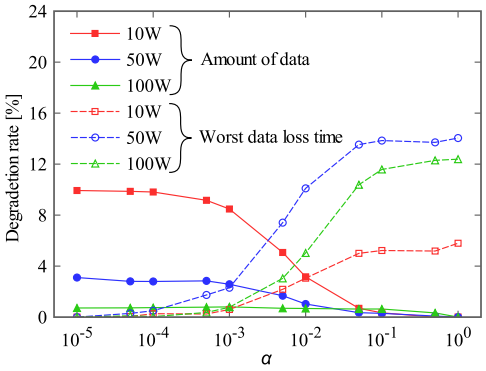

図 9 α を変化させた場合の評価

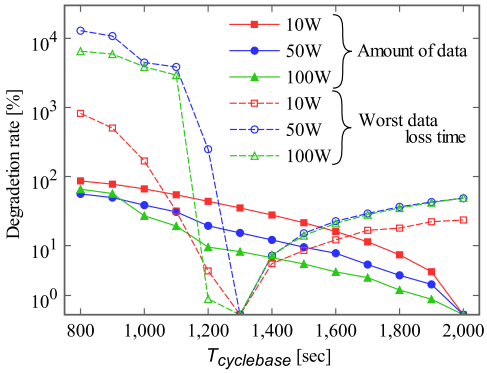

図 10 $T_{\mathrm{cyclebase}}$ を変化させた場合の評価

と考えられる．α が小さくなると，スリープ時間の算出では，新たなデューティ比によるスリープ時間より前回のスリープ時間が重視される．よって，スリープ時間の変動が小さくなり，最悪データロス時間が短縮できたと考えられる．

2 つ目は，α が小さくなるにつれてデータ取得量の悪化率が上昇していることである．例えば発電上限値が 50 [W] の場合，α が 10^{-1} のときデータ取得量の悪化率は約 0.3%であるが，α が 10^{-4} のとき悪化率は約 2.8%である．これは，α を小さくしてスリープ時間の変動を抑えると，デューティ比によってタスクの実行時間が変動するためである．説明のために，4.1 節の式 (4) を式 (5) に変形する．

$$T_{\mathrm{work}} = T_{\mathrm{sleep}} \frac{D}{1-D} - T_{\mathrm{wake}} \tag{5}$$

α が小さくなるとスリープ時間 T_{sleep} の変動も小さくなることから，式 (5) において T_{sleep} を固定値だと仮定する．センサノードの起動時間 T_{wake} は定数であるため，タスクの実行時間 T_{wake} はデューティ比 D によって変動することがわかる．D が大きくなるにつれて T_{work} は増加し，D が小さくなるにつれて T_{work} は減少する．よって，α が小さい場合，デューティ比が低くなるとセンシングのタスク実行時間が減少するため，データ取得量が減少する．一方，α が小さくてもデューティ比が高くなれば，センシングのタスク実行時間が増加するため，データ取得量は増加する．図 9 に示したデータ取得量における悪化率では，発電上限値が 10 [W] でデューティ比が低くなると考えられる場合は悪化率が高く，発電上限値が 100 [W] でデューティ比が高くなると考えられる場合は悪化率が低いことがわかる．

5.5 $T_{\mathrm{cyclebase}}$ を変えた場合の評価

5.2, 5.3 節の評価では，BAAD scheduling における $T_{\mathrm{cyclebase}}$ を 1800 [sec] とした．本節では $T_{\mathrm{cyclebase}}$ の値によって BAAD scheduling の性能に変化が生じるかを検証する．具体的には，$T_{\mathrm{cyclebase}}$ を 800 [sec] から 2000 [sec] まで変化させた場合のデータ取得量と最悪データロス時間を取得する．$T_{\mathrm{cyclebase}}$ の変化による結果の変動を相対的に評価するため，データ取得量と最悪データロス時間のそれぞれについて結果の悪化率を評価する．結果の悪化率算出手順を以下に示す．

(1) 発電上限値が 10 [W]，50 [W]，100 [W] の場合において，それぞれ $T_{\mathrm{cyclebase}}$ を 800 [sec] から 2000 [sec] まで変化させて 7 日間のシミュレーションを 10000 回実行し，データ取得量と最悪データロス時間を計測する．
(2) 発電上限値が 10 [W]，$T_{\mathrm{cyclebase}}$ を 800 [sec] から 2000 [sec] まで変化させた結果の中で，データ取得量の最大値と最悪データロス時間の最小値をそれぞれ最も良い結果とする．
(3) データ取得量と最悪データロス時間の両方について，最も良い結果とそれ以外の結果の差分を算出する．差分を最も良い結果で割った値を結果の悪化率とする．
(4) 発電上限値が 50 [W]，100 [W] の場合も同様に算出する．

データ取得量と最悪データロス時間のどちらにおいても，悪化率が小さいほど高い性能であることを意味する．

図 10 に，$T_{\mathrm{cyclebase}}$ の値を 800 [sec] から 2000 [sec] の範囲で変化させた場合のデータ取得量と最悪データロス時間の悪化率を示す．α の値は全て 0.01 である．横軸が $T_{\mathrm{cyclebase}}$，縦軸が結果の悪化率である．図 10 から，以下の 2 つのことがわかる．

1 つ目は，$T_{\mathrm{cyclebase}}$ が大きくなるにつれてデータ取得量の悪化率が低下していることである．例えば発電上限値が 50 [W] の場合，$T_{\mathrm{cyclebase}}$ が 1000 [sec] のときデータ取得量の悪化率は約 39%であるが，$T_{\mathrm{cyclebase}}$ が 1800 [sec] のときは悪化率は約 3%である．これは，$T_{\mathrm{cyclebase}}$ が大きくなるにつれて 1 サイクルの時間が増加するためだと考えられる．DC-LQ では $T_{\mathrm{cyclebase}}$ を用いてスリープ時間を決定するため，α の値に関わらず $T_{\mathrm{cyclebase}}$ が大きくなるとスリープ時間は増加する．スリープ時間が増加すると，同じデューティ比でも 1 サイクルの時間が増加する．例えばデューティ比 0.5 において，スリープ時間が 100 [sec] の場合 1 サイクルの時間は 200 [sec] だが，スリープ時間スリープ時間が 500 [sec] の場合 1 サイクルの時間は 1000 [sec] となる．1 サイクルの時間が増加すると多くのセンシング時間を確保できるため，データ取得量が増加する．

2 つ目は，$T_{\mathrm{cyclebase}}$ が 1300 [sec] の場合，最悪データロス時間の悪化率が最も低くなっていることである．$T_{\mathrm{cyclebase}}$ が 1300 [sec] 未満の場合，データ取得量における結果の悪化率が高いため，センサノードが起動するだけで 1 サイクルの時間が終了していると考えられる．センサノードが起動してもデータを取得しなければ，最悪データロス時間は増加する．一方，$T_{\mathrm{cyclebase}}$ が 1400 [sec] の場合，データ取得量における結果の悪化率は低いため，1 サイクルの時間が増加していると考えられる．つまり 1 サイクルあたりで消費する電力が増加するため，スリープ時間が増加し，最悪データロス時間が増加する．

6. 関連研究

本研究は，構造ヘルスモニタリングと環境発電を用いたセンサネットワークシステムの研究に関連する．

構造ヘルスモニタリングは，建築構造物にセンサを設置してその建築構造物の性能を診断する技術である．建築構造分野においては，構造ヘルスモニタリングの理論面での研究は既に完成の域にあり [3–7]，現在では，センサネットワークを構造ヘルスモニタリングに活用する研究 [8–10] や，サービスの提供 [11] が進められている．本研究でも，センサネットワークを用いて実際の建築構造物の崩壊現象をモニタリングすることで実データを元にした建築構造解析技術への貢献を目指している．

環境発電を用いたセンサネットワークシステムを利用する研究としては，太陽光発電を用いた動物の生息環境モニタリング [12–14] が挙げられる．これらは動物の生息状況という時間変化が比較的早い現象を対象としている．それに対して本研究では，軍艦島での建築構造物の劣化という時間変化が緩やかな現象を対象としてエネルギーの利用を効率化することを目的としている．

環境発電を用いたセンサネットワークシステムにおけるエネルギー利用の効率化を目的とした研究として，ハードウェアの設計によってエネルギー利用を効率化する研究 [15,16]，センサノードのデューティサイクリングの研究 [2,15,17] が挙げられる．本研究ではデューティサイクリングに着目する．

文献 [15,17] では，発電量の予測を元にデューティ比を決定する手法が提案されている．例えば，文献 [15] では，発電量の予測値と実際の発電量を用いて年単位の長期的な稼働を視野に入れてデューティ比を決定している．しかしながら，消費電力や発電量を正確に予測するためには，事前に使用するセンサノードの消費電力やシステムの発電量を知る必要がある．それに対して本研究では，バッテリレベルのみを用いてデューティ比を決定している．

デューティサイクリングを利用したものとして，B-MAC [18]，A-MAC [19]，X-MAC [20]，ContikiMAC [21] などの通信プロトコルの研究が挙げられる．例えば，B-MAC では，一定間隔のスリープとウェイクを繰り返してウェイクした際にプリアンブルを受信したら完全にウェイクしてデータを受信する仕組みを導入することで低消費電力な通信を実現している．これらの研究は基本的には通信に要するエネルギーの最小化を目的としている．それに対して，本研究では，獲得できるエネルギー量の変動に対してデータ取得量の最大化と最悪データロス時間の最小化を目的としている．

7. おわりに

本稿では，軍艦島モニタリングにおけるデータ中心型タスクスケジューリング方式として，BAAD scheduling を提案した．BAAD scheduling では，DC-LQ によってスリープ時間の変動を抑えるようにセンサノードの起動時間とスリープ時間を決定する．また，BLT Allocation によってバッテリ残量に応じて実行するタスクを決定する．性能評価から，BAAD scheduling が既存方式と比較して，データ取得量を最大にしながら，最悪データロス時間の短縮を実現していることがわかった．現在，軍艦島における実証を進めている．

謝辞

本研究は科学研究費補助金（26289194，代表: 濱本卓司）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長崎市世界遺産推進室，日本航空電子の富岡昭浩氏に感謝致します．

参考文献

[1] Battleship Island monitoring, http://sarulab.inf.shizuoka.ac.jp/battleship/.
[2] Vigorito, C. M., Ganesan, D. and Barto, A. G.: Adaptive Control of Duty Cycling in Energy-Harvesting Wireless Sensor Networks, *SECON'07*, pp. 21–30 (2007).
[3] Natke, H. G., Tomlimson, G. R. and Yao, J. T.: *Safety Evaluation Based on Identification Approaches*, Vieweg (1993).
[4] Natke, H. G. and Cempel, C.: *Model-Aided Diagnosis of Mechanical Systems*, Springer-Verlag (1997).
[5] Haldar, A.: *Health Assessment of Engineered Structures*, World Scientific (2013).
[6] 濱本卓司：建築物の耐震性能評価のためのモニタリング技術，計測自動制御学会，計測と制御， Vol. 46, No. 8, pp. 605–611 (2007).
[7] 濱本卓司：建築物のヘルスモニタリング，基礎工，特集「基礎工におけるモニタリングとその活用」， Vol. 43, No. 11 (2015).
[8] Xu, N., Rangwata, S., Chintalapudi, K. K., Ganesan, D., Broad, A., Govindan, R. and Estrin, D.: A Wireless Sensor Network for Structural Monitoring, *SenSys'04*, pp. 13–24 (2004).
[9] 長井望，三田彰，矢向高弘，佐藤忠信：構造ヘルスモニタリング用ワイヤレスセンサに関する研究，日本地震工学会論文集， Vol. 3, No. 4, pp. 1–13 (2003).
[10] Kurata, N., Suzuki, M., Saruwatari, S. and Morikawa, H.: Actual Application of Ubiquitous Structural Monitoring System using Wireless Sensor Networks, *14WCEE*, pp. 1–9 (2008).
[11] NTT データ：Xrosscloud 橋梁監視ソリューション (BRIMOS), http://www.nttdata.com/jp/ja/lineup/brimos/.
[12] Zhang, P., Sadler, C. M., Lyon, S. A. and Martonosi, M.: Hardware Design Experiences in ZebraNet, *SenSys'04*, pp. 227–238 (2004).
[13] Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R. and Anderson, J.: Wireless Sensor Networks for Habitat Monitoring, *WSNA'02*, pp. 88–97 (2002).
[14] Szewczyk, R., Osterweil, E., Polastre, J., Hamilton, M., Mainwaring, A. and Estrin, D.: Habitat monitoring with sensor networks, *Communications of the ACM*, Vol. 47, No. 6, pp. 34–40 (2004).
[15] Buchli, B., Sutton, F., Beutel, J. and Thiele, L.: Dynamic Power Management for Long-Term Energy Neutral Operation of Solar Energy Harvesting Systems, *SenSys'14*, pp. 31–45 (2014).
[16] Alippi, C. and Galperti, C.: An Adaptive System for Optimal Solar Energy Harvesting in Wireless Sensor Network Nodes, *IEEE Transactions on Circuits and Systems*, Vol. 55, No. 6, pp. 1742–1750 (2008).
[17] Kansal, A., Hsu, J., Zahedi, S. and Srivastava, M. B.: Power Management in Energy Harvesting Sensor Networks, *TECS*, Vol. 6, No. 4 (2007).
[18] Polastre, J., Hill, J. and Culler, D.: Versatile Low Power Media Access for Wireless Sensor Networks, *SenSys'04*, pp. 95–107 (2004).
[19] Dutta, P., Haggerty, S. D., Chen, Y., Liang, C. J. M. and Terzis, A.: Design and Evaluation of a Versatile and Efficient Receiver-initiated Link Layer for Low-power Wireless, *SenSys'10*, pp. 1–14 (2010).
[20] Buettner, M., Yee, G. V., Anderson, E. and Han, R.: X-MAC: A Short Preamble MAC Protocol for Duty-Cycled Wireless Sensor Networks, *SenSys'06*, pp. 307–320 (2006).
[21] Dunkels, A.: The ContikiMAC Radio Duty Cycling Protocol, Technical Report T2011:13, SICS (2011).