稼働率でグループ化されたIoTデバイスの相互監視による異常検知システム

高橋 建一[1,a)]　串田 高幸[1]

概要：通信技術の発達により IoT を多様に扱う現代において設置した機器が正常に動作しないことや，異常に気付くのが遅れてデータを取得できていないことが問題の背景である．本論文では稼働率によってグループ化されたノード群の相互監視による異常検知システムの提案をする．多数に設置し稼働する IoT から異常が無いかグループ内で相互監視をおこなうこと，また，稼働率によってグルーピングを行うことにより正常な相互監視ができるかどうかの検証が目的である．異常を検知する方法として IoT ノードの応答確認ができる ping コマンドを採用する．また，ping コマンドで送受信されるデータは ICMP と呼ばれるエラー通知や制御メッセージとする．グルーピングされたノード間でお互いに ping コマンドの ICMP 送受信を行い反応が返ってくるかどうかで異常を判断する手法を用いる．実際にプログラムを動かしその結果を本論文で紹介する．

1. はじめに

1.1 背景

現代社会において IoT というのは社会基盤や生活基盤に広く浸透している．IoT の活用現場として医療，物流，製造業，農業がある．医療では緊急事態の初期段階において情報収集という点，物流では IoT を利用することにより同時スケジューリングという点，製造業では伝統的な製造システムから最新のデジタル化されたものに移行すると同時に IoT の採用により産業用ビッグデータの形成という点，農業では IoT の導入によりモニタリング，収穫の予想という点で役に立っている [1][2][3][4]．いずれにしても通信技術の発達により IoT の活躍する環境が広くなりつつあることがいえる．

1.2 課題

医療，物流，製造業，農業分野で活躍する IoT だが共通して抱える問題がある．それは，使用している IoT 機器が突然異常や故障を起こすことである．本論文においての異常とは IoT 機器が電源が通っていないまたは，故障により ping コマンドによって送られる ICMP に返答できない状態のことを指す．IoT が異常を起こすことで引き起こされる問題が 2 つある．1 つ目は機能の停止である．IoT が故障するということは機能が停止することである．これはデータ収集の役目を果たすことができないということである．2 つ目は機会の損失である．IoT が故障していることで本来収集できたはずのデータが収集できなかったということである．2 つの問題から IoT の異常や故障は IoT を使用者にとっては避けたい問題であると言える．しかし，IoT の異常や故障というのは IoT を使用する上では避けられない問題 [8] であり，抱える課題とも言える．

本論文では IoT の課題である異常や故障に対して異常検知というアプローチで解決を図る．既存の検知手法，監視手法について調査し，新規に提案する検知手法との比較や相違点を挙げて優位性があるかどうかの提案を行うものとする．このレポートの 2 章は関連研究と本論文との比較や違いを挙げる．3 章では提案内容についての説明をする．4 章では実装方法について説明をする．5 章では本論文での取り組みであらわとなった課題について議論する．6 章では本論文のまとめと次回の展望について記述する．

2. 関連研究

本論文で述べる研究内容は既存研究においてどのような位置づけにあるのか，また，どのような優位性があるのか関連研究を交えてこの章では述べる．

既存の IoT の監視に関する研究の一つにトロイの木馬から IoT を保護する相互監査フレームワークという研究がある [5]．これはトロイの木馬と呼ばれる IoT に壊滅的な障害を引き起こすプログラムから防御するために IoT の通信を監視する軽量のフレームワークについて研究したものなっ

[1] 東京工科大学コンピュータサイエンス学部
〒 192–0982 東京都八王子市片倉町 1404-1
[a)] C0117183

ている．本研究と比較すると違いがある．それは，監視方法の違いである．先ほど挙げた研究ではトロイの木馬と呼ばれる外部からの悪意，脅威から防御するための監視である．それとは別に本レポートの研究は外部からの脅威ではなく IoT 本体が ping コマンドによって送られた ICMP を返答できない異常や故障が起きてないかを監視する位置づけである．また，それに付随してネットワークに異常がないかの監視も含まれる．その為，外部のネットワークを監視するのではなく，ICMP を用いて IoT を監視するという点では優位性があるといえる．

グルーピングに関する既存の研究の一つに IoT ネットワークを効率的に高速にする研究がある [6]．この研究は IoT の数が増え続けることによる遅延とパケット衝突率の増加の問題に対して，IEEE 802.11ah ベースの IoT ネットワークでチャネルを高速かつ効率的にアクセスするためのデバイスグループ化スキームの研究である．しかし，グルーピングで効率的に高速化においては稼働率は触れられていない．効率的という観点において稼働率のグルーピングについて本研究では取り組む．

相互監視に関する既存の研究の一つに動的相互情報類似性分析を用いた研究がある [7]．この研究は相互情報ベースの類似性を用いてシステムステータスの転送先として考えられる遷移を識別する研究である．本研究との違いとしては監視方法である．本論文の相互監視の方法は ping コマンドを用いた ICMP の送受信の応答反応により監視を行う．対して，動的相互類似性分析を用いた研究では統計が対応する条件を満たさない場合に故障と判断される．本研究と比較して相互監視の手法に違いがある．それは，ping コマンドによる ICMP の送受信による応答があるかないかどうかと統計を用いた条件の設定の違いである．このことから，ping コマンドを用いた手法では複雑な構造が無く ICMP による応答判断のみのためシステムにかかる負荷が少ないという点で優位性があると言える．

3. 提案

本論文で提案するのは稼働率によってグルーピングされた IoT 機器の相互監視である．ここでいう稼働率とは，設置した IoT ノードが算出した個々の稼働率である．また，ここでいうグルーピングの定義とは，個々に分かれている IoT ノードが 2 台によるペアリングまたは，3 台以上によるグルーピングのことを指す．設置した IoT 機器には故障の可能性，不具合の可能性が潜んでいる．いつ壊れるかわからない IoT 機器に対しては不具合を検知するシステムが必要である．提案内容としては IoT の個々の稼働率を調べグルーピングを行う．稼働率でグルーピングを行う理由としては，稼働率の高い機器と低い機器を組み合わせることで総合的に見てそのグループをより長く稼働させるためである．グルーピングを終えた後はグループ内の機器同士で相互監視を行う．この際にリーダーノードはグループ内で一番稼働率が高いものとする．相互監視では ICMP をお互いに送受信し，生存確認を行う．ここでいう生存確認とはリーダーノードが他ノードに ping コマンドで ICMP を送信し，その反応が返ってくることである．一通りの生存確認を終えたらリーダーノードがグループに繋がっているサーバに報告を行う図の 1 には 2 台ずつペアリングした場合の構成図を，図の 2 には 3 台ずつグルーピングした場合の構成図を示す．

図の 1 は 2 台ずつペアリングした場合の構成図である．上位サーバとはグループ 1 とグループ 2 からの報告を受けるサーバであり，収集した稼働率のデータからランキング化しグルーピングを行う．管理者 PC とはこのシステムの使用者であり，上位サーバからデータをもらい受けることで全体の把握ができる．グループ 1 内にはノード 1 とノード 2 が所属しており，ノード 1 とノード 2 はお互いに ping コマンドを用いて ICMP を送りあい応答確認を行っている．同様にグループ 2 内のノード 3 とノード 4 も ICMP を送りあい応答確認を行っている．上位サーバへの報告役としてグループ 1 からはノード 1 がリーダーとして上位サーバに報告を行っている．同様にグループ 2 からはノード 3 がリーダーとして上位サーバに報告を行っている．

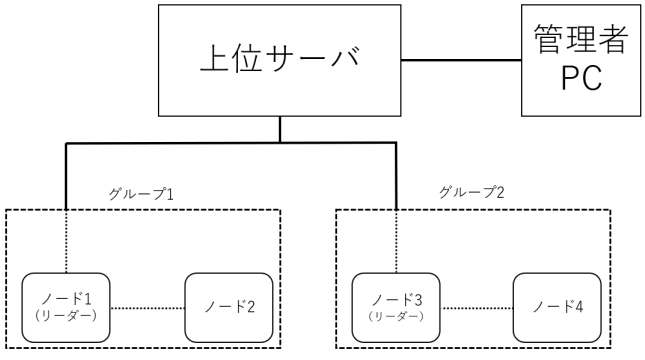

図 1　2 台の場合の構成図

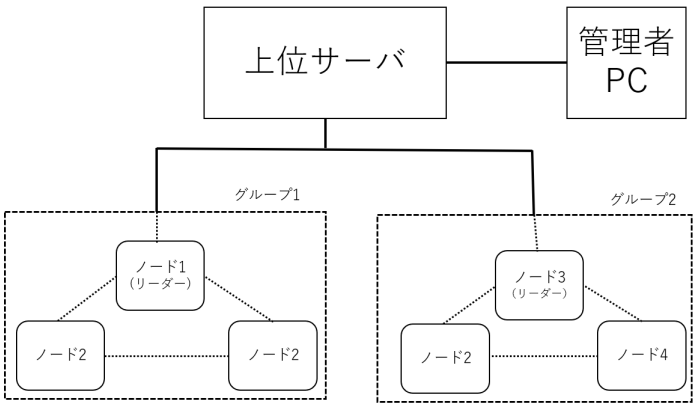

図 2　2 台の場合の構成図

図の 2 は 3 台ずつグルーピングした場合の構成図である．上位サーバと管理者 PC の構成は変わらないが，グルーピングした中身のノードの数が違うことが図の 1 と図の 2 の比較でわかる．図の 2 のグループ 1 の場合どのように相互監視をするのか説明をする．グループ 1 内のノード 1 がノード 2 とノード 3 に ICMP を送り，その応答確認を行っている．同様にノード 2 はノード 1 とノード 3 に ICMP を送り，その応答確認を行っている．同様にノード 3 はノード 1 とノード 2 に ICMP を送り，その応答確認を行っている．また，グループ 2 でも同様の ICMP の送受信を行っているため説明を割愛する．

この提案の利点として単純なコストダウンがある．従来の監視方法ならば個々の IoT はそれぞれサーバに 1 対 1 で繋がっている．それにより，多く設置する Iot においては 1 つ増えるたびに比例して設置コストと維持コストが増えることとなる．また，従来の相互監視においては個々のノードが複数繋がって相互監視を行っている状況である．従来の相互監視と比較して本論文の提案はグループ毎にノードを割り当てているため個々のノードに比べて管理がしやすく，整理しやすいという利点がある．

本論文で提案するグルーピングによる相互監視を用いれば，1 対グループの関係になるために，1 対 1 と多対多に比べて繋ぐ本数が減るためコストダウンに繋がる．

本論文においてのユースケースとして工場内で稼働している機会などに取り付ける IoT 機器を想定している．日中もしくは一日中稼働し続けている工場内の機器に取り付けられた IoT には工場内の温度，湿度，電気，人的ミスといった IoT が停止してしまう要因が数多く潜んでいる．このような状況において IoT を監視するシステムは必須である．そのようなユースケースにおいて少しでも生存率を高める為に稼働率をグルーピングに相互監視を使う手法は有効的であると考えられる．

3.1 グルーピング方法

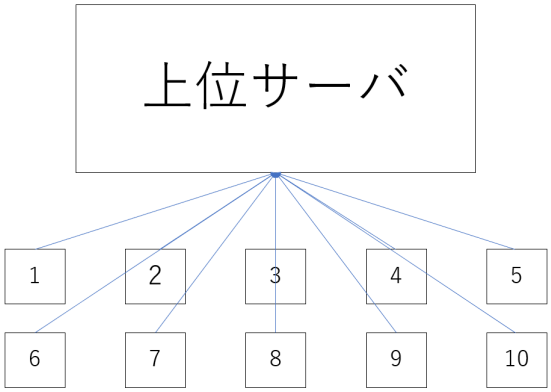

図 3 ステップ 1

この章では稼働率でどのようにグルーピングするのか記述する．IoT ノードは複数ある前提だが，今回の説明では１０台と仮定する．

ステップ 1 として最初にグルーピングに必要なデータである，稼働率を収集する必要がある．（ここいつごろからいつごろまで計測するのか，また，どれくらい動いてから計測するのかのせる）個々のノードが稼働率を計測し，そのデータを上位のサーバに送信する．この場合の上位サーバとは個々のノードから送られて稼働率のデータを束ねるサーバのことである．また，束ねたデータをランキング形式に判断しグルーピングの判断を行うのも上位サーバの役割である．図の 3 にそのイメージ図を載せる．図の 3 では 10 個あるノードの個々から上位サーバに向けて稼働率のデータを送っている図となる．

ステップ 2 では送られてきた稼働率のデータをサーバが集約し，ランキングを作成する．今回の説明の場合だとランキングは１０位まで作られる．個々のノードの稼働率からランキングを作成すればステップ 2 は完了である．

ステップ 3 では，作成されたランキングをもとにグルーピングを行う．今回は１０台の IoT ノードのケースであるので，２台ずつ組み合わせるペアリングを行う．ペアリングの方法として，１位のノードと１０位のノードをペアリングする．同様に 2 位と 9 位，3 位と 8 位，4 位と 7 位，5 位と 6 位，という組み合わせとする．なぜ 1 位と１０位のような組み合わせかその説明をする．１位～５位のノードを上位ノードとした場合，６位～１０位のノードが下位ノードと今回は仮定する．上位ノードと下位ノードがペアリングすることにより，停止しやすい下位ノードを上位ノードが監視することが可能になる．また，相互監視という名目上，グループ内のノードがすべて全滅してしまってはサーバに報告することができなくなってしまうため，グループ全体が停止しないという意味を込めて稼働率の高いノードを混ぜる必要がある．以上がステップ 3 である．

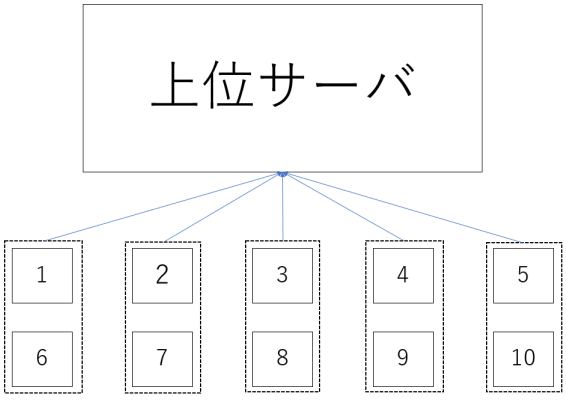

図 4 ステップ 4

ステップ 4 ではサーバ内で決めたグルーピングを今度は IoT のグルーピングに反映させる．以上がグルーピング方法である．グルーピングの方法をまとめたを図を図の 3 に

示す．

3.2 相互監視方法

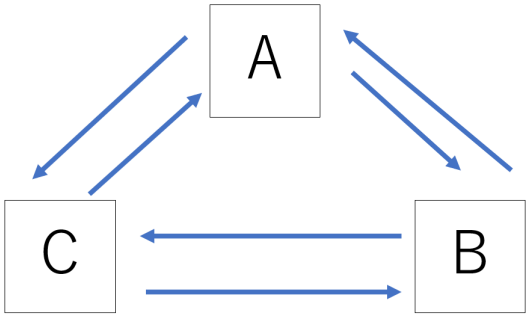

図 5 3 台ノードがある場合の相互監視図

この章ではグルーピングされた後，どのようにグループ内で相互監視を行うのかの方法を記述する．グループ内のノード数は今回の仮定では 2 個である．つまり，この 2 個のノードがお互いに相互監視を行うということである．また，サーバに報告する役目を持つをノードは上位ノードの中でも稼働率が高いものをグループ内のリーダーノードとする．なぜ上位ノードがリーダーノードになるのか説明をする．サーバに報告をするという役割を持つため停止のリスクが少ないノードであるほうが，サーバへの報告が滞りなくできるからである．また，相互監視といっても異常があった場合の確認役として停止リスクの少ない稼働率の高いノードが必要になってくるからだ．

相互監視の方法は ping コマンドによって ICMP をお互いに送りあい，その反応が返ってくるかどうかで生存確認を行っている．なぜ数ある相互監視，生存確認の方法がある中 ping コマンドの ICMP 送受信を今回の実装で使うのか説明する．ping コマンドを使う理由は 2 つある．1 つ目は ping コマンドを用いて相互監視を行うからである．新たにプログラムを作り ICMP の送受信のプログラムを新規に作るよりも既存のコマンドを流用するのが確実な応答確認ができると考えたため採用した．2 つ目は相互監視をわかりやすくするためである．複雑な条件を設けて異常を検知する手法に比べて，ICMP の送受信の反応による異常検知はシステムに対する負荷が軽く，一目瞭然で生存確認ができる．

今回の提案の説明ではグループ内には 2 つのノードがあるという設定だが，もし 3 つあった場合の相互監視について説明する．2 つノードがある場合と同様に稼働率が高いものをリーダーノードとする．リーダーノードを個体名 A とした場合，ほかの 2 つは B,C とする．A が B と C に ICMP を送り異常検知を行う．同様に B が A と C に，C が A と B に ICMP を送りあう．つまり，1 つのノードは ICMP を送る相手は 2 つあるということである．これは，グループ内のノードが 3 つ以上ある場合の方法である．同様の図を図の 5 に示す．図の 5 では ICMP の送信先，受信先の方向として青い矢印を用いている．

3.3 データの保持

ping コマンドにより ICMP のデータやり取りで出力されたデータの取り扱いに関してこの章で説明する．相互監視によって発生したデータはグループ内のリーダーノードが集める．集めたデータを次は上位サーバに送信する．データを送信された上位サーバはそのデータを記録しサーバ内に保持するという流れである．

4. 実装と評価

4.1 実装

本レポートの ping のプログラムの部分を紹介する．プログラムに書いた部分を図の 2 に示す．

図の 6 に書かれているシェルスクリプトのコードの働きは特定の IP アドレスに ping を送り帰ってくれば「success」と表示される．また，ping を送ったのにも関わらず返事が返ってこない場合は「failure」と表示される．今回送っている IP アドレス先は Google の IP アドレスである．今回のプログラムで Google の IP アドレスを送っている理由としてプログラムが動作するかを確認する為に最適という理由から用いている．

また，このプログラムのが動作した際の図を図の 3 に示す．図の 3 によると，Ubuntu 内のシェルスクリプトで実行する ping 送受信プログラムが起動すればインターネットを通して google の IP アドレスに送られ返ってくるという一連の流れがこのプログラムによって起こすことが可能である．

```
#!/bin/sh

target=”xxx.xxx.xx.xx”
aaa=$(ping -c 1 ”$target” | grep ”received” | awk ’print $4’)
if [ $aaa -eq 1 ] then
   echo ”success”
else
   echo ”failure”
fi
```

図 6 プログラム内容

4.2 実験環境

今回の使用する道具は ubuntu, シェルスクリプト, PC である．これらを用いて ping の送受信のプログラムを作る．ubuntu を用いるのは Linux 系の OS の中で広く普及されておりまた，現在においてもアップデートが行われておりサポートが充実しているという理由から用いている．

シェルスクリプトを用いる理由としては今回の ping の送受信を行う手法として ubuntu で利用することが可能である ping コマンドを用いるからである．また，ping コマンド自体を自動的に数回送る方法としてシェルスクリプトが代表的だったことから今回は用いている．

4.3 評価

図の 7 にプログラムの実行結果を示す．本レポートでの議論では今回作った ICMP 送受信プログラムについて評価を行う．今回の評価は相互監視に使うプログラムとしては必要十分の効果があると考えられる．図の 4 からわかるように ping コマンドを用いて ICMP を対象の IP アドレスに送った際に ubntu がインターネットに繋がっている状態とインターネットを切った状態でプログラムを実行し，ICMP を送った．ネットワークにつないだ状態で送った結果，「succes」と表示された．また，ネットワークを切った状態でプログラムを実行した結果「failure」と表示された．この結果から，ping を利用した生存確認能力はあると判断できる．

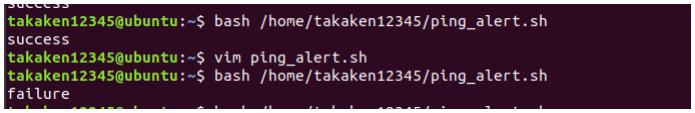

図 7 プログラムの実行結果

今後の展望としては相互監視の要であるノード同士がお互いに ICMP を送るという環境を目指す．相互監視に必要なものは，複数のノード同士がお互いに監視しあうことである．そのためには，本レポートで紹介した ICMP 送受信プログラムが必要である．それに付随して何回 ping コマンドを起動するか，また，どのように複数のノードの IP アドレスを識別するのか今後の課題である．

5. 議論

本研究で取り組むべき議論は 2 つある．1 つ目は一度決めたグルーピングの再グルーピングである．一度目に計測した稼働率によるグルーピングが永遠に変化せずに稼働率が一定であり続けることは不可能である．稼働率の変化に合わせて期間を定めて提案で述べた方法で再びグルーピングをする必要性が本研究で判明した．　2 つ目は相互監視についてである．本論文での提案では ping コマンドを用いた ICMP エコー要求とエコー応答のやり取りによって異常が無いか確認を行っている．もし仮に IoT ノード自体には何も異常が無く正常に稼働しているのにもかかわらず ICMP の要求が通らない場合にどのように判断するかが議論によって問題として浮き上がった．ネットワークがダウンしたことにより通信が通らず ICMP が通らないというケースもあるためこの問題に対してどのようにアプローチをかけるのか重要である．

6. おわりに

今回の提案内容は稼働率によってグルーピングされた IoT デバイスの相互監視の提案である．本レポートで取り扱った内容としては ping 送受信プログラムの作成である．作成したプログラムを実際に動かした評価としては相互監視に必要な生存確認を行うことができると確認できた．今後の取り組みとしてはグループ内のノード同士で ping コマンドを用いて ICMP を送り検証を行うことである．また，グループ内のノードの IP アドレスを識別するという課題が今後の取り組みとしてある．

参考文献

[1] Ali, H.S., Arun, K.S., Sandeep, P., et al.:Green media-aware medical IoT system,Springer Science+Business Media, LLC, part of Springer Nature (2018)
[2] Yingfeng, Z., Zhengang, G., Jingxiang, L., Ying, L.:A Framework for Smart Production-Logistics Systems Based on CPS and Industrial IoT, Proc. IEEE Transactions on Industrial Informatics(2018)
[3] D, Mourtzis., E, Vlachou., N, Milas.:Industrial Big Data as a Result of IoT Adoption in Manufacturing, Laboratory for Manufacturing Systems and Automation (LMS), University of Patras, Rion Patras 26500, Greece(2016)
[4] Meonghun, L., Jeonghwan, H., Hyun, Y.: Agricultural Production System Based on IoT, Proc. IEEE 16th International Conference on Computational Science and Engineering(2013)
[5] Chen, L., Patrick, C., Chengmo, Y.:A mutual auditing framework to protect IoT against hardware Trojans,Proc. 21st Asia and South Pacific Design Automation Conference (ASP-DAC)(2016)
[6] Sabin, B., Shree, K.S., Xianbin, W.:Device Grouping for Fast and Efficient Channel Access in IEEE 802.11ah Based IoT Networks, Proc. IEEE International Conference on Communications Workshops (ICC Workshops)(2018)
[7] Yuchen, H., Zhiqiang, G., Zhihuan, S.:Adaptive monitoring for transition process using dynamic mutual information similarity analysis, Proc. Chinese Control and Decision Conference (CCDC)(2016)
[8] Boris, O., Felipe, N.:Robust Gossiping for Distributed Average Consensus in IoT Environments, Proc. IEEE Access (Volume:7)(2018)