# 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**：スマート工場内における IoT ノードの異常や通信の異常を背景としている．課題として IoT ノードの異常または通信の異常の検出に焦点を当てている．この課題に対してグループ内でリーダノードとノードが相互監視をするシステムを提案する．相互監視では ICMP を送受信する．ICMP の送受信の応答が返ってくるかどうかで異常を判断する．応答結果をリーダノードが集めサーバに送り異常が検出された場合に通知をする．応答結果から異常を判断が可能か評価を行う．評価から，100%と 0%の場合には正常と異常の判断が可能である．しかし，100%から 0%の間における異常の判断では現状の提案では対応できない．議論ではその 100%から 0%における判断方法として正常モデルを作成することで相互監視の結果と照らし合わせることで値が外れてないかを議論する．

## 1. はじめに

### 1.1 背景

現代社会において IoT というのは社会や生活基盤に広く浸透している．Ericsson 社のレポートによると IoT の接続数は年々増加しており 2015 年には 4 億台の規模だったのが，2019 年には 13 億台の市場規模となっている [1]．このまま市場規模が増加傾向で継続が続けば，2025 年には 50 億台の規模となることが予想される．世の中に広く浸透している IoT が活躍する現場の例の一つに製造業がある [2]．製造業では古くからある製造システムに IoT を組み込むことで稼働している機械から今まで取れていなかった温度データ，稼働状況がわかるデータといった産業用のデータの収集という点で活躍をしている．産業用のデータを収集することにより故障した際の原因究明や故障の予想などが可能になる．また他にも医療，物流，農業といった現場でも活躍をしている．医療では救急搬送された患者の初期分析という点で医学の知識がない人でも情報を得ることが可能という点である [3]．物流では IoT を利用した倉庫内の同時スケジューリングという点である [4]．農業では IoT を導入したことにより人の手で直接確認を行わずとも設置したセンサから遠隔で温度や湿度を確認しモニタリングすることで収穫の予想という点で役に立っている [5]．いずれにしても IoT の市場規模が増加したこと，通信技術の発達により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

市場規模が増加傾向にあり医療，物流，製造業，農業分野で活躍する IoT デバイスだが共通する問題がある．それは設置した IoT デバイスが知らぬ間に異常を起こすことである．その中で，本論文は http といったリクエストやレスポンスを返す通信データのやりとりを問題なく行っていた機器が突然通信不能になる状態を異常として捉えて検出することを課題とする．これにより，IoT デバイス本体または通信上の異常が起きた際に検出することが期待できる．

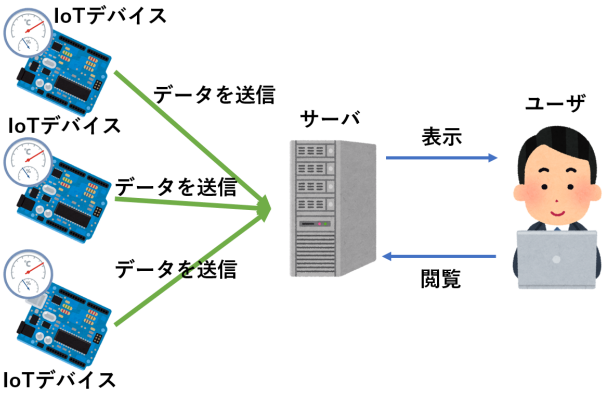


**図 1**　一般的な IoT デバイスの使用例

図 1 に一般的な IoT デバイスの使用例として図を載せる．図 1 の IoT デバイスにはデータを取得するためのセンサが取り付けられている．センサから取得したデータ（例：

[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

温度，湿度）を IoT デバイスはサーバに向けてデータを送信している．ここでいうデータ送信とは http 通信による post メソッドで送信するものと仮定する．データを受信したサーバはその情報をユーザが閲覧できる形で表示する．サーバが情報を閲覧可能にする形の例として web ページでの表示を例に挙げる．表示された情報をユーザは閲覧することでセンサが取得したデータを知ることができる．

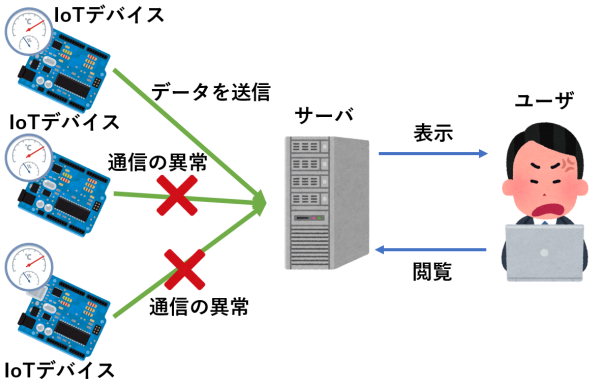


図 2　通信に異常が起きる例

図 1 の例を基に図 2 では本論文の課題を図を用いて説明する．図 2 は図 1 に異常が発生した場合の図である．本論文が課題として捉えている異常とは通信の異常の検出である．本論文における通信の異常の定義とは図 1 で例を挙げたような http 通信による通信に異常が起きることである．さらに，本論文における異常とは IoT デバイス付近にアルミや金属板のような通信に直接な障害をもたらすことができる物体が IoT デバイスとサーバ間に存在することで http 通信による通信の疎通ができず繋がることができないことを指す．つまり，本論文における課題とは IoT デバイスとサーバ間が http 通信によってデータの送受信を今まで問題なく動作していたものが，アルミや金属板によって IoT デバイスとサーバ間の通信が阻害され通信障害がおきることを異常として捉えて，その異常を検出することを課題とする．この課題が解決されない場合，ユーザはノードに異常があって閲覧ができないのか，サーバ側に異常があって閲覧できないのか，通信に異常があって閲覧できないのか原因の特定が難航する．この課題が解決されれば少なくともアルミや金属板といった障害物が原因での通信の異常を検出することができる．

### 1.3　各章の概要

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 章は関連研究では不十分な点を挙げる．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

相互監視に関する既存の研究に動的相互情報類似性分析を用いた研究がある [6]．この研究では相互情報を用いた類似性のシステムステータスに着目している．システムステータスの転送先について遷移を識別することをテーマとしている．この研究では DPLS モニタリングモデルアルゴリズムと組み合わせた新しい MI ベースの類似性，遷移の識別，モデリング，およびプロセスモニタリングの提案をすることにより課題である従来の MSPC アルゴリズムを使用してプロセス動作をモデル化および分析する相互監視を解決することができる．この課題解決結果に対して本論文の課題である通信の異常については課題解決が不十分であるといえる．なぜなら，通信の異常が起きた際にステータスの転送は不可になり検知ができないからである．

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 [7]．この研究は状況によって左右される IoT 機器に対して増え続けた場合による通信の遅延とパケットの衝突の増加に対してテーマを設定している．この研究の課題は超高密度の IoT ネットワークにおける高速なチャネルアクセス，信頼性の高い低遅延の通信リンクの提供の困難である．困難といえる理由は非効率的なチャネルアクセスメカニズム，リソースに制約のあるエッジデバイス，および利用可能な無線リソースの制限を理由に非常に困難であると分析されている．この課題の解決として IoT ネットワーク用のセクターベースのデバイスグループ化スキームを行うことで解決に至っている．しかし，この課題解決結果にたいして本論文の課題である通信の異常については解決が足りないといえる．なぜなら，IoT デバイス本体が異常を起こした場合に高速なチャネルアクセスは機能しないからである．

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 [8]．トロイの木馬からの攻撃による被害から防御するため IoT 機器の通信を監視するという研究である．この研究の課題はネットワークセキュリティである．既存の解決策は通信媒体を介してノードを操作することにより，外部からの攻撃を防御するために開発されている．つまり，ネットワーク内のデバイスは信頼できると見なされているためトロイの木馬のような攻撃に弱いという点で課題がある．この研究の解決策はノード間で信頼できるチャネルを成立することで解決を行っている．この結果に対して本論文の課題である通信の異常が起きた際に対応できないため不十分であるといえる．なぜなら，通信の障害が発生すればトロイの木馬からの攻撃以前に防ぐことも検知することもで

きないからである．

IoT の需要が広がりネットワークトラフィックが飛躍的に増加する状況の中，複雑になった IoT ネットワークの管理を隙を突いたサイバー攻撃を対象に異常検出の研究がある [9]．これは機械学習アルゴリズムを用いた IoT の異常検出である．課題としてはスマートシティにおける IoT デバイスのリソースや機能が少ないことによる IoT ネットワークへの攻撃の検出方法である．膨大なネットワークトラフィックの影に潜む致命的なネットワークへの攻撃を検出するのが課題である．解決方法としては IoT アプリケーション向けの IDS(Intrusion Detection System) を効率的に強化することで解決している．しかし，本論文の課題である通信の異常はネットワークトラフィックに潜む攻撃とは別の問題であり，その点に関して言えば不十分であるといえる．なぜなら，物理的に通信の障害が発生すればこのシステムは機能しないからである．

異常検知の分野で IoT 対応アプリケーションのセキュリティを強化するための多段階異常検出スキームという研究がある [10]．これは，従来の大規模データの異常を検出するために使用されている DBSCAN の問題点である最近傍探索とパラメータ選択によるパフォーマンス低下という課題がある．課題の解決策に対して提案では DBSCAN の問題点を修正することにより，異常検出の多段階モデルを提案している．しかし，この提案では大規模データの異常を検出すること可能だが，IoT 本体の異常または通信の異常の検知にはつながらないため不足しているといえる．なぜなら，IoT 本体の異常または通信の異常においては大規模データの中に隠れてしまうからである．

異常検知の分野でニューラルネットワークを用いた分散型異常検出というものがある [11]．これは IoT やワイヤレスネットワークといった限られたリソースの中で捉えどころのない異常に対してワイヤレスネットワークを課題として捉えている．課題に対しての解決はオートエンコーダニューラルネットワークを導入することにより課題解決を行っている．しかし，この課題では捉えどころのない通信の異常を課題にしているため本論文の課題である金属板やアルミといった物理的な障害の検出という点で不十分であるといえる．

別の研究では IoT 向けのゲーム理論を用いた異常検知の研究がある [12]. この研究の課題は DS による高い検出率と低い偽陽性の利点を組み合わせることが可能という事実によって，それを実際に IoT で運用させるとリソースが少ないことが原因で外部からの攻撃によって高い負荷がかかるという課題がある．課題解決として，ゲーム理論を用いることで攻撃が来ると予想されるときにアクティブ化することで高精度の検出とエネルギー消費のバランスが取れるという提案を行っている．しかし，通信の異常の検知という点では不十分であるといえる．なぜなら，外部からの攻撃を検知するには通信が正常でなければならないため，その通信が異常である検知ができないからである．

別の研究では産業用 IoT における時系列データを用いた詳細な異常検出がある [14]．これは産業用 IoT における異常を正確にタイムリーに検出することが必要になった課題から生まれた研究である．異常検出の手法は時系列データを感知するための通信効率の高いオンデバイス連合学習ベースの異常検出フレームワークを用いている．連合学習とはデータを集約せずに分散した状態で機械学習を行う方法であり，2017 年に Google 社が提唱した [15]．連合学習フレームワークを導入することで分散型エッジデバイスが異常検出モデルを共同で学習することが可能になる．これにより，一般化能力の向上と異常を正確に検出するための畳み込みニューラルネットワークモデルの提案が可能となる．しかし，この研究ではエッジデバイスによって収集されたデータの送信経路における通信の異常に関する対策がなされておらず対応できないといえる．なぜなら，エッジデバイスからの通信は途絶える可能性があるからである．

別の研究では IoT デバイスの供給電流に着目して監視する研究がある [16]．これは IoT デバイスの機能パラメータの 1 つである供給電流を介してセキュリティの指標とする研究である．課題としては IoT デバイスのセキュリティの異常を課題としている．解決方法としては間接的な方法で，具体的には IoT デバイスの異常を検出するための供給電流の監視を介してセキュリティの課題の解決を行う．デバイスには機能と操作上の特徴の点で制限があることを考慮すると，通常の操作からのかけ離れた操作，挙動は消費電力にも比例して同様の挙動がみられると予想される．そのため予期しないデバイス操作の監視することが課題に対する提案といえる．しかし，この提案では通信に異常に関しては不十分であるといえる．なぜなら，通信に異常があった際の IoT デバイスの挙動は同様にして供給電流に反映される．つまり，物理的に通信に異常があった場合にそれが IoT デバイスが攻撃されているのかどうか区別がつかないからである．

異常検出の分野に IoT を用いた行動モデリング侵入検知システムによる行動ベースの異常検出の研究がある [17]. この研究の課題は従来の侵入検知では本来検出する異常かどうかに関わらず不確実なものえであっても異常として報告されることが課題として捉えられている．解決ではシミュレーションサービスに IoT 対応のオブジェクトを使用することでシミュレートされた IoT 周辺の行動についてそれをベースとした異常の検出を行う研究である．シミュレーションでは行動モデリング侵入検知システム (BMIDS) によって監視される．BMIDS によってアルゴリズムを用いることで抽出された行動パターンが目的の行動と一致するかどうか，また目的の行動から逸脱するかを区別する．この研究では行動パターンが逸脱しているか，目的から逸

れてないかで異常かどうかを判断するためデータの通信に異常に関する課題が不足している．なぜなら，通信の異常は突発的に発生する場合があるため検知できないからである．

別の研究では IoT 通信における異常の検出と監視をテーマにした研究がある [18]．急速に成長する IoT の需要や規模を背景に大規模なデバイス数に対応したネットワーク自体の監視，早期検出が課題としてその解決法を提案している．提案手法としては複数の異種ネットワーク (SNMPv1, SNMPv2, SNMPv3) に対応するためにそれぞれを持つ異なるバージョンの SNMP を用いて生データを抽出する．抽出したデータを元に要約したデータ情報のネットワークマッピングを行い，CPU 処理使用率，メモリ使用量を照らし合わせ異常が無いかを判断する．この研究においては複数のデバイスから送られてくるデータが送信途中に通信に異常が起きること関しては触れられてないため不十分であるといえる．なぜなら，仮に通信に異常があることを検出できなければデータを送ることができないからである．

クラウド中心の IoT における異常検出をテーマにした研究がある [19]．IoT からクラウドへ読み込む際にセキュリティやプライバシーの問題が適切に処理されなければ攻撃の対象にさらされる可能性がある．そのセキュリティを背景に異常検出が必要だとこの研究では提唱している．主な課題として IoT クラウドに送り出される膨大な量のデータから外れ値を引き出すことである．この課題で解決できるのは送られてきたデータから異常を判断することである．データを送るために必要な通信経路上の異常に関しては触れられておらず仮に通信に異常が起きた際にそれが送信元であるデバイスの異常なのか，通信の異常なのか区別がつかない．

別の研究では IoT デバイス向けの超軽量ディープパケットの異常検出をテーマにした研究がある [20]．これは小型の組み込みデバイスの侵入検知システムの要件を満たすことができないことによる，脆弱性を狙われた攻撃を課題としている．課題に対して提案手法はリソースに制限のある IoT デバイスにおいて正常なペイロードと異常なペイロードを適切に区別が可能な超軽量のディープパケと異常検出アプローチである．この手法は効率的な特徴選択によってビットパターンのマッチングを使用し，ビット単位の AND 演算と条件付きカウンターインクリメントのみ必要とすることで軽量化を可能にしている．しかし，この提案には IoT デバイス間の通信の異常に対するアプローチが触れられておらず不十分だといえる．なぜなら，脆弱性を狙った攻撃とそうでない攻撃とで区別ができないからである．

異常検出の分野の研究の 1 つにディープオートエンコーダーを使用した IoT ボットネット攻撃のネットワークベースの異常検出の研究がある [21]．IoT デバイスの急増を背景に IoT を狙ったボットネット攻撃が課題だとこの研究は提唱している．その課題に対して提案手法はネットワークの動作を抽出し，ディープオートエンコーダーを用いることで攻撃された IoT デバイスから異常なネットワークトラフィックを検出することが可能になる．しかし，ネットワークトラフィック自体に異常が起きた際にこの提案手法では異常かどうかを判断することが難しい．なぜなら，通信に異常が発生した場合ディープオートエンコーダーのデータにも異常をきたすからである．

別の研究では IoT ネットワークにおける異常な活動検出のための 2 レベルハイブリッドモデルという研究がある [22]．この研究の課題は不規則なアクションによるネットワークへの攻撃，データの盗難を課題としている．課題に対して解決ではレベル 1 とレベル 2 を用意することで不規則なアクションに対して異常検知の範囲を高めることに成功している．レベル 1 ではフローベースを使用することによりネットワークトラフィックを正常または異常と分類する．レベル 2 ではレベル 1 で検出した異常を受け取りパケットの内容を詳細に調査し異常のカテゴリを決める．レベル 2 では再帰的特徴除去を利用して特徴を選択し，合成マイノリティオーバーサンプリング手法を使用しオーバーサンプリングした後に Nearest Neighbors を使用してデータセットをクリーニングする．この提案手法だとネットワークトラフィックの異常を判断することができるが，ネットワーク自体に異常が起きた際の検知が難しいといえる．なぜなら，1 章で説明した課題のような物理的な通信の障害である場合それが，ネットワークへの攻撃かどうか判断がつかないからである．

別の研究では低リソース IoT デバイス向けの正確なセキュリティの研究がある [23]．課題には IoT デバイスは信頼できないネットワークに接続する現状と処理する情報からセキュリティ保護をする必要があることである．提案手法は侵入検知システム (IDS) とシグネチャ検出の技術を組み合わせることで課題解決に繋がる．高い検出率を達成するために，学習アルゴリズムに依存してノードの通常動作をモデル化しこの動作とは違うパターンが検出されると別のモデルとしてモデル化される．異常検出をすべての IoT デバイスで同時にアクティブにすると高いエネルギー消費が予想されるめ，異常が発生するポイントのみに異常検出を当てる方法を提案として挙げている．しかし，IDS で検知した侵入データの受け渡しの際に通信に異常があった場合の検出が不十分だといえる．なぜなら，通信の異常が見られたらそもそものデータの受け渡しが十分にできないからである．

別の研究では統計的学習手法を活用しすることでデバイスの特徴を定めて偏差を異常として捉える研究がある [24]．背景には一般的な IoT デバイスは容量が制限されているため，処理能力が低くなり結果的に計算リソースが限られてくるため通常のコンピュータに適用されていたトラフィッ

クアナライザやウイルス対策ソフトウェアの手法を適用するのが困難であることが背景である．提案手法としては統計的学習を使用することで CPU 使用率，ディスク使用率をアプリケーションプログラムで取得することによりフレームワーク，プラットフォームやデバイスに依存しない異常検知を提案している．この研究では統計的学習法に必要なデータを API を用いて集めている．しかし，API によるデータ収集の際に通信に異常があった場合にそれを検出する方法がこの研究においては不十分だといえる．なぜなら，通信の異常によって API のデータ収集が十分に行うことができないためである．

異常検出の分野の 1 つに消費者向けの IoT デバイスの機械学習 DDoS 検出という研究がある [25]．これは消費者向けの IoT を対象にインターネットインフラストラクチャに対して分散型サービス拒否（DDoS）攻撃を受けている IoT の攻撃トラフィックを自動的に検出する提案である．この提案においては通信の異常は対象にしておらずあくまでトラフィック量に注目した DDoS 攻撃の検出である．しかし，通信の異常が起きた際にそれが DDoS 攻撃によるものなのか単に障害物による通信障害なのか区別が難しい．なぜなら，1 章で説明した通信の異常は物理的な通信の異常である金属板やアルミを考慮しており，DDoS 攻撃に対しては検知が可能でも物理的な通信の異常の検知は不十分だからである．

別の研究では IoT システムセキュリティのための脅威視覚化ツールの研究がある [26]. これは IoT のセキュリティにおいて VisIoT と呼ばれる新しいネットワーク用の脅威視覚化ツールの提案である．どのように脅威を視覚化するのかその判断は異常検出と同様に求められることであるといえる．この研究の提案内容はあくまで敵意のある攻撃の脅威視覚化でありそのため検出である．そのため，障害物による通信の障害は考慮されていない．なぜなら，通信経路上にある障害物には敵意は無いがその区別が無いからである．

異常検出の分野の研究の 1 つに IoT 環境向けのハイブリッド異常検出システム研究がある [27]. これは家庭における IoT デバイスの教則な成長を背景に攻撃対象としてさらされている IoT の異常を検出するという課題を掲げている．提案手法としてトラフィックの特徴を抽出しトラフィックに異常がないかチェックを行い検出するものである．この研究においてはネットワークトラフィックにおける異常を検出することができると言っている．しかし，障害物による通信の異常においてはネットワークトラフィックとは関係のない異常で効果を発揮するのは難しい．なぜなら，IoT が使われる環境によって物理的な通信の異常によってネットワークトラフィックが途切れる場合がある．通信の異常を検知をすることが必要であるからである．

異常検出の分野の研究は他に信頼性の低いデータに対するロバストな異常検出という研究がある [28]. この研究の課題は信頼性の低いデータも異常検出の学習として使用しなければいけないという課題がある．解決策として異常に対して信頼性の低いものを検出した場合により堅牢な異常検出を決定づけるために 2 層の学習フレームワークを用いてロバストな異常検出に変化させる方法を用いている．この研究において通信の異常に関しての対策が不十分であるといえる．なぜなら，物理的な通信障害の場合と単純に信頼性が低いデータでは区別がつかないためである．

別の研究では IoT における階層的異常ベースの侵入検知の研究がある [29]．これは深層学習を利用した IoT ゲートウェイにおける侵入検知をテーマにした研究である．この研究の課題は IoT，WSN とゲートウェイにおいて攻撃にさらされている現状からから攻撃による異常検知を課題としている．提案手法としては階層構造を用いることで悪意ある攻撃に対して検出することを手法としている．しかし，この提案手法では IoT と IoT ゲートウェイを繋ぐ通信経路の異常を検知することは難しい．なぜなら，悪意ある攻撃というのは故意に行うものであるが，1 章で説明されている金属板やアルミによる異常は故意には含まれない．よって，通信の異常を検知できなければ誤検知という結果をもたらす為，不十分だといえる．

異常検出の研究分野に産業用 IoT インターネットにおけるエッジデバイスの異常検出の研究がある [30]．この研究の課題はエッジデバイスにおける異常な行動パターンの検知である．解決ではエッジコンピューティングにおける時系列データの異常検出をするためにスクイーズド畳み込みオートエンコーダーを用いて解決を行う．しかし，この提案手法ではエッジデバイスから送る時系列データの受け渡しに必要な通信環境に異常があった場合について検知ができないといえる．なぜなら，エッジデバイスからデータを送るにはネットワークを用いているため，ネットワークの異常を検知できなければそれはエッジデバイスの異常な行動パターンとして誤検知されるからである．

## 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．本論文における稼働率というのは，設置した IoT ノードが算出した個々の稼働率である．また，稼働率の種類としては時間稼働率を求めておりその式は

$$\text{稼働率} = \text{総稼働時間}/(\text{総稼働時間}+\text{総故障時間})$$

である．本論文におけるグルーピングとは個々に分かれている IoT ノード 2 台以上によるグルーピングのことを指す．大前提として本論文の提案はすで稼働率によるグルーピングが行われた後の話である．つまり，本論文の提案の要は稼働率によって分けられ，グルーピングが行われた後

のグループ内の IoT デバイスのリーダノードと IoT デバイスのノードの相互監視が提案内容であることを留意する．

いつ壊れるかわからない IoT 機器には故障の可能性や不具合が潜んでいる．なぜ不具合を検知する必要があるのか，それは不具合を検知することで異常が起きていることを把握することができる．異常を検知することは根本的な原因の追究に繋がり修理，改善を行うことができる．本論文の提案内容は IoT デバイス間の通信の異常を検知することである．以下は便宜上 IoT デバイスをノードと表現する．

### 3.1 提案方式

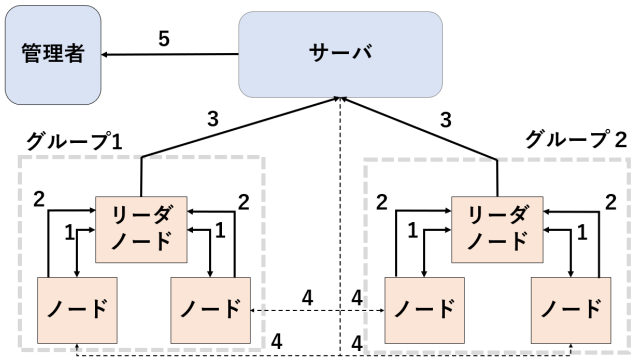


図 3　全体アーキテクチャ図

図 3 に稼働率によってグルーピングした後の全体のアーキテクチャ図を載せる．図 3 に出てくるノードとは IoT デバイスのことを指す．リーダノードも同様である．図 3 におけるサーバとはグループのリーダノードからの相互監視結果の受信の役割がある．また，サーバには受信の役割のほかに異常の判断と通知の役割を持っている．異常の判断ではリーダノードから送られてきたノードとリーダノード間の相互監視の結果を基に異常があるかどうかを判断する．通知の役割では異常があるかどうかの判断結果を通知用の web ページに通知する．図 3 の中にある管理者とはシステムの使用者のことを指しており，サーバと繋がることで全体の把握ができる．図 3 においてグループ 1 は 3 台のノードで構成されている．そのうちの 1 台はリーダノードの役割を持っており，これはノードと ICMP を送りあうことで相互監視をする役割を持っている．また，相互監視して集めた記録をサーバに送信する役割を持っている．ノードはリーダノードと相互監視を行うことで異常が無いか確認を行っている．同様にグループ 2 でも同じことが起きている．

図 3 においてその順番を説明する．まず図 3 の 1 の部分から相互監視が始まる．この 1 が最初となり，続けて 2，3，4，5 と順番で動作する．図 3 の 1 ではノードとリーダノード間の相互監視を行っている．相互監視では ICMP を用いる．ICMP とは通信プロトコルの一種であり，インターネット通信が正常にできるか確認するときに用いられる．本論文における ICMP の場合は通信に異常が無いか検知するために使われる．相互監視の説明には変数を用いて説明をする．変数 i を用いて相互監視は i 秒に i 回を 1 セットとする．変数 j を用いてセット数は j セットとする．相互監視をした結果はノードとリーダノードがそれぞれデータ保持する．次に図 3 の 2 ではノードが保持した相互監視の結果をリーダノードに向けて送信する．リーダノードは送信されたデータのために受信を行う．次に図 3 の 3 ではリーダノードがノードから受け取ったデータとノード間との相互監視で生成した自らのデータをひとまとめにしてサーバに向けてデータを送信する．次に図 3 の 4 ではサーバ側がリーダノードから受け取ったデータを基に異常の疑惑があると判断したときにノードに向けて ICMP を送り異常があるかどうか確認している．異常かどうかを判断する方法は 3 パターンによって決まる．1 つ目は監視結果から 100%疎通が確認できる場合は異常無しと判断する．2 つ目は監視結果から 0%の疎通の場合には異常と判断できる場合に異常有りと判断する．3 つ目は 0%から 100%未満の間の ICMP の疎通率の場合はサーバから対象ノードに向けて ICMP の送受信を試みる．この時の結果がリーダノードとノード間の疎通結果の成功値を超えるならば正常と判断する．疎通結果の成功値を下回るならば異常と判断する．成功値については議論の章で議論する．最後に図 3 の 5 ではノードで異常があるかどうかを確認した結果を通知する．ここでは異常が無い場合，ある場合を説明する．通信に異常が無い場合はサーバ側が通知用の web ページに異常が無いことを報告する．通信に異常がある場合はサーバ側が通知用の web ページに異常があることを通知する．通知には画面上に「異常有り」または「異常なし」と表示する．通知を受けた管理者はここでノードの通信に異常があることを察知することができる．

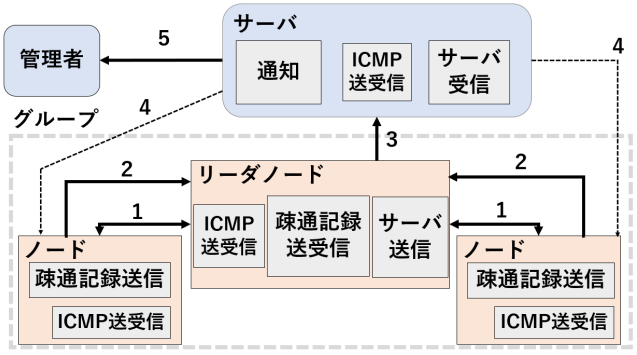


図 4　ソフトウェア構成図

図 4 では構成図においてどのようなソフトウェアやプログラムが動いているのかその説明をする．まず，図 4 のノード内では「ICMP 送受信」と「疎通記録送信」の 2 つのプログラムが入っている．これは他のノードにも同様に入っている．「ICMP 送受信」とはリーダノードに向けて

ICMP を送受信するプログラムであり，これによりリーダノードとの ICMP の送受信を行うことができる．「疎通記録送信」とはノード自身がリーダノードに向けてデータを送信するものである．送る方法としては http 通信による post メソッドによって送られる．

図 4 のリーダノードでは「ICMP 送受信」と「疎通記録送受信」と「サーバ報告」の 3 つのプログラムが入っている．「ICMP 送受信」とはノードに向けて ICMP を送受信するプログラムである．これにより，指定ノードの通信に異常が無いか相互監視が可能である．また，ノードの数に合わせて相互監視する指定ノードの IP アドレスの数も合わせる．これによりノード 1 台だけでなく複数のノードに向けて相互監視をすることが可能になる．「疎通記録送受信」とはリーダノード自身がノードから送信される相互監視の結果のデータを受信するためのプログラムである．これにより，ノードからリーダノード間の ICMP の応答結果をまとめたデータの送信，受信が可能になる．「サーバ報告」とはリーダノードからサーバへデータを送信するプログラムである．サーバへ報告する方法として http 通信の post メソッドを用いて報告する．

図 5 のサーバには「通知」,「ICMP 送受信」と「サーバ受信」の 3 つのプログラムがある．「通知」とはノードに異常があるかどうかサーバからノードに向けて確認の ICMP を送り返ってきた結果を通知用の web ページに表示する．もし異常が発見された場合にサーバは通知用の web ページに異常が発見されたというページを表示するものである．また，異常が検知されなかった場合にも同様に通知用の web ページに異常無しの旨の内容を通知する．これにより，管理者はノードに異常があるかどうかを知ることが可能になる．「ICMP 送受信」とはリーダノードからの報告でノードに異常があると知らされた場合に本当に異常が起きているのか，その確認のためにノードに向けて ICMP を送受信するためのプログラムである．これにより，本当にノードの通信に異常があるのチェックすることができる．異常が見られた場合は先ほどの「通知」に異常があるノードを通知し，異常が無ければ同様に異常無しの旨を通知する．「サーバ受信」とはリーダノードから送られる相互監視の結果をまとめたデータを受信するためのプログラムである．受信した後は「ICMP 送受信」が ICMP の疎通成功回数を計算し成功確率によって異常があるかどうかの確認をする．ノードに異常があると疑いがある場合は異常がないか確認をするために ICMP をノードに向けて送る．

### 3.2 ユースケースシナリオ

図 5 のユースケースシナリオ図を用いてユースケースシナリオを説明する．本論文におけるユースケースとして製造業の工場を想定している．またその工場は IoT によるスマート化が施されているものと捉える．つまりスマート工場である．その状況下で工場内で稼働している機械に取り付ける IoT 機器をユースケースとして想定している．日中もしくは一日中稼働している機械には工場に環境によっては温度，湿度，電気，人的ミスといった IoT が停止してしまうような要因が数多く潜んでいる．そのようなユースケースにおいて生存率を高めるために稼働率をグルーピングし相互監視を使う手法は有効的であると考えられる．図 5 に振られている番号は動作する順番である．

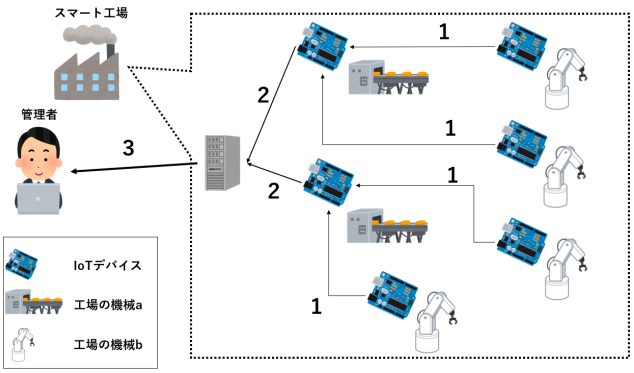


図 5　ユースケースシナリオ図

(1) 図 5 の 1 は稼働中の機械に取り付けられた IoT デバイスが稼働中の機械の動作データを取得し送信している．IoT デバイスにはもう一つ機能があり，それが本論文で提案する通信の異常の検知である．工場内というケースなので IoT デバイス自身に不具合が無くとも周りの環境に左右されて通信環境に問題がある可能性がある．このようなユースケース下で通信の異常を検知するのは有効的である．

(2) 図 5 の 2 でも同様にデータの送信を行っている．稼働中の機械の傍ら相互監視を行い，サーバに向けてデータを送っている．工場内なので同様に通信に異常を引き起こす可能性が潜んでいるので相互監視は有効的である．

(3) 図 5 の 3 では IoT デバイスから送られてきたデータをまとめて異常がないか判断し管理者に通知している．図 5 のユースケース化においてスマート工場内の通信の相互監視は通信の異常を引き起こす可能性がある工場内においては改めて有効的であるといえる．

### 3.3 相互監視方法

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その方法を説明する．説明ではグループ内のノード数は 2 個を仮定して説明する．相互監視する関係としてこの 2 個が相互監視を行うということになる．サーバに報告する役割を持つノード，つまり，リーダノードはこの場合稼働率が高いノードがその役割を引き受ける．この時なぜリーダノードが選ばれるかはサーバへの報告を滞り

なく報告する必要があるため稼働率が高いノードを選ぶことでサーバへの報告を滞りなく報告することができる．このことから停止リスクが少ない稼働率が高いノードが選ばれる．

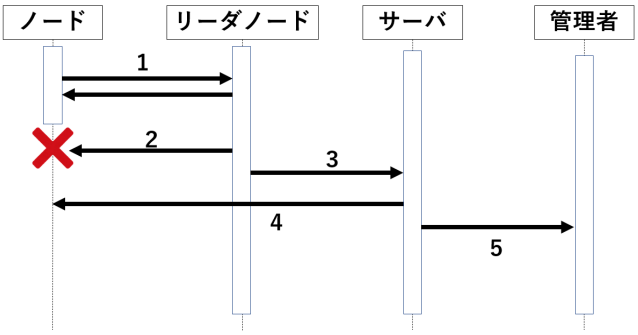


図 6　異常が起きた際の検知

相互監視の方法は ICMP を相互で送りあい，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．ICMP を用いた相互監視を用いた理由は既存のコマンドを用いた方法が確実に応答確認ができると考えたため採用する．また，複雑な条件を設けて異常を検知する方法と比較して ICMP の送受信を用いた方法は一目瞭然で生存確認ができるため用いている．

相互監視している際に異常が起きた場合の対応について図 6 を用いて説明を行う．

(1) 図 6 の 1 ではノードとリーダノード同士が相互監視をするための ICMP を送りあっている．i 秒 i 回を 1 セットとし，このセットを j 回行う．異常が無い場合，通常状態の場合はこのように ICMP を送りあい，異常がないことを確認している．

(2) 図 6 の 2 ではノードに異常がみられて ICMP が送れない状態となっている．この際，相互監視相手であるリーダノードが送った ICMP の反応が返ってこないことで ICMP の送信に対して返信が無いことが記録される．

(3) 図 6 の 3 では ICMP の反応が返ってこないことを記録したリーダノードはサーバに向けてリーダノード自身がまとめた ICMP の結果をまとめたデータを送信する．この時の送信はサーバにのみの通信である．

(4) 図 6 の 4 ではリーダノードの送信を受信し，異常が無いかを判断する．判断方法としては 10 セットの ICMP の応答結果を基に成功確率によって異常があるかどうかを判断する．この時点でノードに異常が起きているためその成功確率は 0%であるので異常ありと判断する．

(5) 図 6 の 5 ではサーバ側からノードの通信に異常が起きているのか確認する為にノードに向けて ICMP を送信する．しかし，この時点でノードは異常を起こしているので当然返答は返ってこない．その為，疎通成功率 0%であることが記録される．サーバはその後通知用の web ページにノードに異常があることを管理者に向けて通知する．

### 3.4　データの保持

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．相互監視によって集められたデータはノード，リーダノードそれぞれに貯められる．リーダノードは，グループ内の全ノードからデータを受け取りを確認したらサーバへデータを送る．サーバへデータを送った後はリーダノードは容量確保のために手元のデータを消去する．サーバはデータを受け取り一定期間保持した後，削除をする．ここでのデータのやりとりはテキストファイルで行うものとする．

## 4.　実装と実験環境

### 4.1　全体構成

図 7 では提案するシステムの全体構成を示している．前提としてグループ内はノードは 2 台，リーダノードは 1 台として説明をする．グループは複数あれどサーバは 1 台である．グループ内のノードは ESP32 を使う．同様にリーダノードも ESP32 を使う．サーバは研究室内のコンピュータを用いて VM 上にマシンを作成しサーバとして使う．図 7 においてノードでは「node_icmp.py」と「node_post.py」の 2 つのプログラムがある．詳しくは後述のソフトウェアで説明する．図 7 においてリーダノード内には「leader_icmp.py」，「leader_receive.py」，「leader_post.py」の 3 つのプログラムがある．詳しくは後述のソフトウェア設計の章で説明する．図 7 の VM server においては「server_receive.py」，「server_icmp.py」，「notice.py」の 3 つのプログラムがある．詳しくは同様に後述のソフトウェア設計の章で説明する．

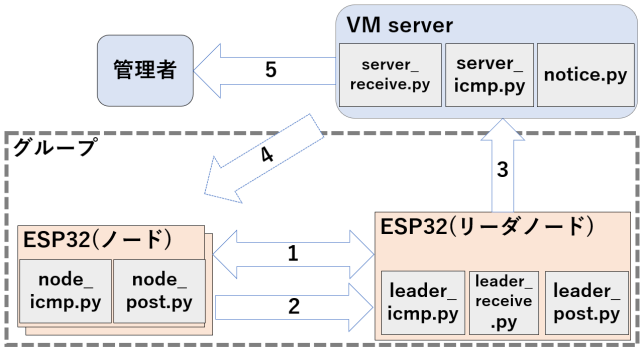


図 7　全体構成図

図 7 の 1 では相互監視を行っている．ノードとリーダノードにあるプログラムによってこの相互監視を行っている．図 7 の 2 では先ほどの相互監視によって生成した ICMP の応答結果のデータをノード側から送信している．

リーダノード側はそのデータを受信する．また，リーダノード側も自身で同様のデータを生成し持っている．図 7 の 3 ではリーダノードがノードから送られてきたデータと自身で生成したデータをまとめ，サーバへ送信している．図 7 の 4 ではリーダノードから送信されたデータをサーバが受信した後，通信に異常が疑わしいか判定をしている．通信に異常が起きているかもしれないとサーバ側が判断した際に図 7 の 4 のようにノードに向けて ICMP の送っている．通信に異常があれば返信が無いが，異常が無ければ返事は返ってくる．図 7 の 5 では 4 で行った確認の ICMP の応答結果を通知用の web ページに通知を行う．これにより管理者は通知内容を把握する．

### 4.2 ESP32(ノード) のソフトウェア設計

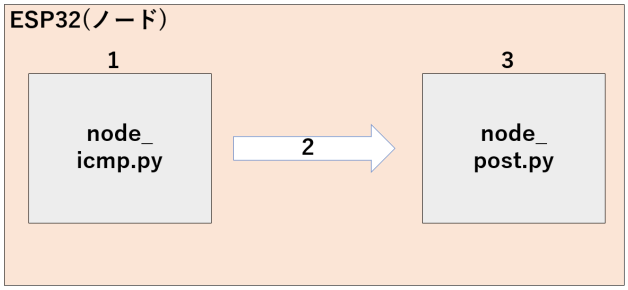


図 8 ESP32(ノード) のソフトウェア設計図

図 7 の ESP32(ノード) の説明を図 8 を用いて説明する．図 8 の nodo_icmp.py はリーダノードに向けて ICMP を送信するプログラムである．送信した後はその応答結果が返ってくる．応答結果をまとめるがまとめる内容は ICMP の疎通試行回数と成功回数を記録する．記録する形式はテキストファイルである．node_post.py はまとめた生成したテキストファイルをリーダノードに向けて送信するプログラムである．送信方法は http 通信による post メソッドを用いて送信する．

### 4.3 ESP32(リーダノード) のソフトウェア設計

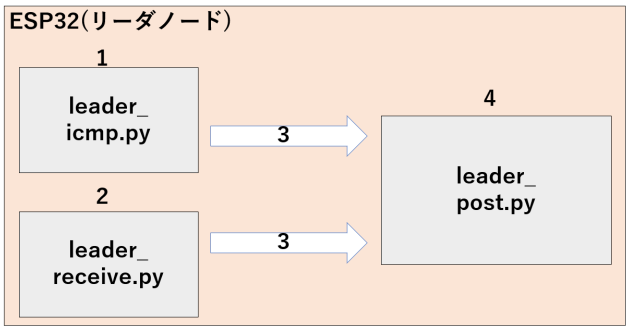


図 9 ESP32(リーダノード) のソフトウェア設計図

図 7 の ESP32(リーダノード) の説明を図 9 を用いて説明する．図 9 の leader_icmp.py はノードに向けて ICMP を送信するプログラムである．また，相互監視した結果を自身で生成する．形式はテキストファイルである．図 9 の leader_receive.py はノードから http 通信の post メソッドで送られてきたデータを受け取るための受信用プログラムである．図 9 の leader_post.py は自身で生成したデータと送られてきたデータをひとまとめにサーバへ送信するためのプログラムである．送信方法は http 通信による post メソッドである．

図 9 のソフトウェアの動作順を説明する．図 9 の 1 でノードと相互監視を行い，相互監視た結果をテキストファイル形式で保存する．図 9 の 2 ではノードから送られてきたデータを受信する．ここでいうデータとはノード側の相互監視の結果をテキストファイル形式に保存したテキストファイルである．図 9 の 3 でテキストファイルをサーバへ送信するための準備をしている．図 9 の 4 ではリーダノードが持っているテキストファイル形式の相互監視の結果をサーバへ送信する．

### 4.4 VM server のソフトウェア設計

図 7 の VM server の説明を図 10 を用いて説明する．図 10 の server_receive.py はリーダノードから送られてくる ICMP の応答結果をまとめたデータを受信するためのプログラムである．図 10 の server_icmp.py は送られてきたデータを基に ICMP による相互監視で行っている ICMP の疎通成功回数から成功値を算出する．閾値となる成功値以下ならば通信に異常がある疑いがあるので改めてノードに向けて ICMP を送信し確認するプログラムである．図 10 の notice.py は異常の確認で送った ICMP の応答結果を通知用の web ページに通知するプログラムである．

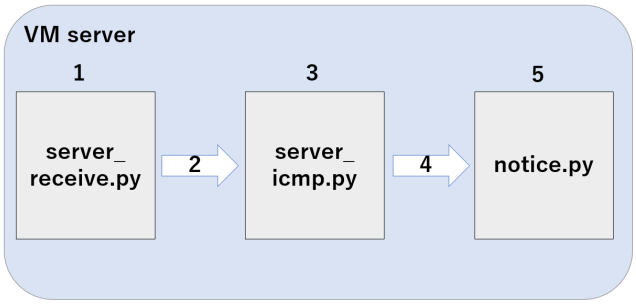


図 10 VM server のソフトウェア設計図

図 10 のソフトウェアの動作順を説明する．図 10 の 1 はリーダノードから送信されるデータを受信するためプログラムが動作する．図 10 の 2 は 1 で受け取ったデータを 3 に向けて流している．3 では受け取ったデータを基に異常があるかどうかを見極める．見極めた結果，異常があると判断した場合にサーバ側からノードに向けて本当に通信に異常があるのかどうかを確認する為にノードに向けて ICMP

を送信する．図 10 の 4 から 5 にかけては先ほどの確認で行った ICMP の応答結果を通知用の web ページに出力し通知する．

### 4.5 実装

ここでは図 11 を用いて説明する．図 11 は実装構成図である．図 11 のノードの実装では ESP32 を用いている．中身に関しては MicroPython を用いてプログラムを実装する．図 11 のリーダノードでは ESP32 を用いる．中身に関しては MicroPython を用いてプログラムを実装する．図 11 の VM server は研究室環境を用いて VM 上でサーバを立てる．サーバは Apache を用いる．中身には python を使って実装をする．

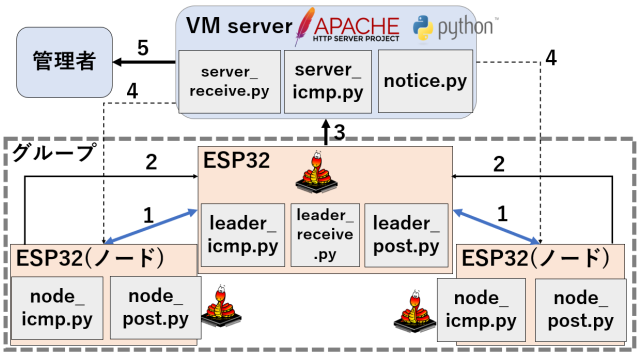


図 11　実装構成図

図 11 の動作順を説明する．図 11 の 1 ではノードとリーダノード間で相互監視を行う．ノード側では node_icmp.py を用いて相互監視のための ICMP を送受信している．リーダノード側は leader_icmp.py を用いて相互監視のための ICMP を送受信している．図 11 の 2 ではノードからリーダノードに向けてデータを送っている．ノード側が node_post.py を用いて http 通信による post メソッドを使用して送信リクエストを送る．リーダノード側は leader_receive.py を使用して送信リクエストを承諾し受信に入る．図 11 の 3 ではリーダノードは送られてきたデータと自身で生成したデータをひとまとめにし，サーバへ向けて leader_post.py を使用して送信リクエストを送る．サーバ側は server_receive.py によって送信リクエストを承諾し受信する．図 11 の 4 ではリーダノードから送られてきたデータから相互監視によって ICMP の疎通成功回数から成功確率を算出し通信に異常が無いかを確認する．通信の成功値が閾値以下だと判断した場合に通信の異常があると判定する．サーバ側は通信に異常があるノードに向けて ICMP を送り異常が無いかを確認する．この送られてきたデータから異常があるかどうかを判断，異常があるかどうかを確認の流れを server_icmp.py で行う．図 11 の 5 では先ほどの確認用に用いた ICMP の応答結果を通知用の web ページに掲示する．以上が実装構成図の流れである．

実験環境の説明を図 12 と図 13 の 2 つの図を交えて説明する．

図 12 は本論文で提案する手法の実験環境である．図 12 の 1 は ESP32 上で実装する MicroPython によってノードとリーダノード同士が ICMP を送受信する．図 12 で言うところの片方のノードの通信環境を悪くし通信に異常が発生しやすい状況を作る．ここでいる通信の異常とは ICMP の応答が返ってこないことを指す．また，通信に異常が発生しやすい状況を作るためにノードにはアルミホイルに包まれた状態で相互監視を行ってもらう．図 12 の 2 では相互監視によってノードが生成したデータをリーダノードへ送信している．図 12 の 3 では相互監視によって生成したデータ，送られてきたデータをまとめてサーバへ送信する．図 12 の 4 ではサーバは送られてきたデータを解析し異常がある疑いがあるノードに対して ICMP を使ってその確認をする．図 12 の 5 では異常がある疑いがあるノードの応答結果を通知用の web ページに出力する．この実験環境で評価できるものは誤検知率である．誤検知率に関しては後述の検証方法で説明する．

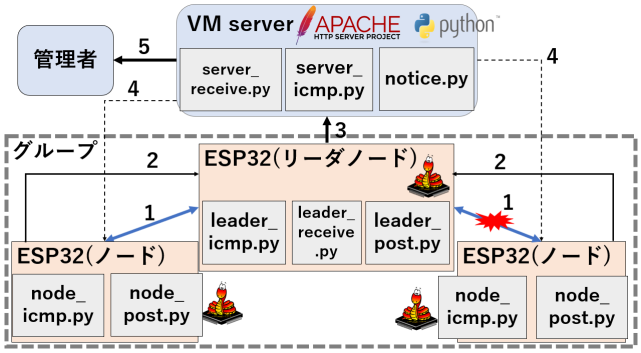


図 12　提案手法の実験環境

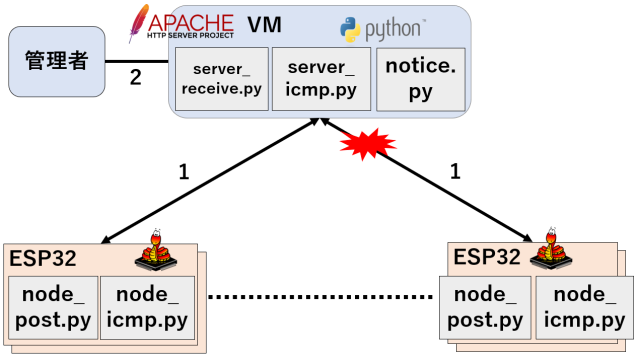


図 13　従来環境の実験環境

図 13 は従来の手法の実験環境である．図 13 のサーバと管理者の部分は図 12 の構成と変わらない．違う点はグループの存在である．従来の手法では個々のノードがサーバと繋がっていることを再現するために図 12 の環境からグループを消去する．図 13 の実験環境の動作について説

明する．図 13 の 1 では ESP32(以下ノードと表記する) と VM(以下サーバと表記する) がお互いに相互監視を行っている状況である．実験のため図 13 の環境かでは図 12 と同様に通信に異常が起こしやすいような状態を作る．通信に異常が起きやすい状態は先ほどの図 12 と同様にアルミホイルをノードに包んで実行する．図 13 の 2 ではノードとの通信に異常があると判断したサーバが同様にノードに向けて ICMP を実行しその応答結果を通知用の web ページに通知する．この実験環境でも同様に評価できるもの誤検知率である．同様に誤検知率に関しては後述の検証方法で説明する．

本論文の研究で実際に行った実験を図 14 に示す．リーダノードとノードがお互いに相互監視用の ICMP を送受信する．ICMP は i 回 i 秒を 1 セットとし，j 回行う．実験では i を 60，j を 10 とする．通信に異常が無いかを手動で判断をする．具体的にはリーダとノードとノード A が ICMP を送りあい，それとは別にリーダノードとノード B が ICMP を送りあう．通信に異常が無ければ「正常」と判断し，応答結果が返ってこない場合には「異常」と判断する．また，応答結果が返ってこない回数によって ICMP の送受信の成功確率を算出する．これによりサーバ側が判断するための ICMP の送受信の成功確率が有効であるかどうかを判明する．

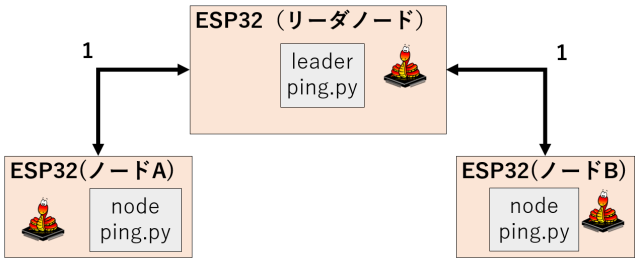


図 14　実際の実験

### 4.6　検証方法

課題は IoT ノード間の通信の異常を検出することである．その課題に対して本論文の提案は IoT ノード間を相互監視によって通信の異常を検出するということである．相互監視による異常検出において評価する点は誤検知率である．図 13 のような従来の異常検出はサーバとノードが直接つながり，ノードに異常があった場合はサーバがそれを検知し通知するという方法である．それに対して図 12 のような本論文の提案はノード間の相互監視によりまとめられたデータをリーダノードがサーバへ報告し異常を検知する手法である．従来の方法だと，サーバ側からの視点で見れば直接的な検知方法なのに対し，本論文の提案は間接的な検知である．直接的な検知と間接的な検知の 2 つが誤検知率に違いがないかで評価を決める．ここでいう誤検知率とはノードが正常に動いているにも関わらず誤って異常と判断することである．

仮予想として誤検知率が高いのは従来の方法である．従来の直接的なやり方ではノードの通信に異常が見られた場合にすぐに検知して通知するが，それが本当に通信の異常なのかどうか判断ができない．直接的なやり方は ICMP の応答結果が返ってこないと判断しただけである．対して，本論文の提案は間接的に検出する手法である．間接的な手法によりリーダノードから送られた異常という報告が本当なのか調べる為にノードに向けて ICMP の送受信を行うため，この時に異常か正常化のジャッジを行う．なので，直接的な方法と比較して時間がかかるかもしれないが，誤検知率が低くなることが予想できる．

## 5.　評価と分析

### 5.1　実験結果

次に図 14 で示した実際の実験の結果を図にして図 15, 図 16, 図 17, 図 18, 図 19 に示す．図 15 はリーダノードからノード A の ICMP 送受信結果を示す．図 15 からわかる通り，1 回目は疎通成功確率 98.3%である．それ以外の試行回数はすべて 100%の疎通成功確率となっており，10 セット分の全体成功確率は 99.8 %である．図 16 はリーダノードからノード B の ICMP 送受信結果を示す．図 16 かわわかる通り，10 セットすべての場合で 100%の成功確率示している．どちらの結果も障害物やアルミといった通信に障害を起こすものが無い状態で行った実験である．

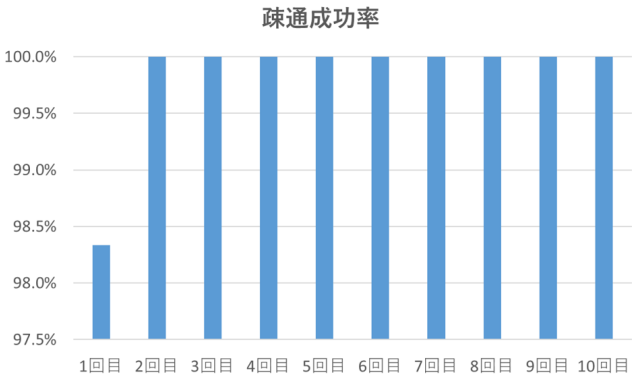


図 15　リーダノードからノード A の ICMP 送受信結果

次に図 17 はノード A からリーダノードの ICMP 送受信結果を示す．図 17 から 6 回目の疎通成功確率は 96.7%である．6 回目を除いて残りは成功確率は 100%である．10 セット分の成功確率は 99.7%である．図 18 はノード B からリーダノードの ICMP 送受信結果を示す．図 18 からわかるのは 6 回目の疎通成功確率が 98.3%である．それを除けば他のセットの成功確率は 100%である．また，10 セット分の疎通成功確率は 99.8%である．

最後にアルミホイルでノードを包んだ場合の ICMP の送

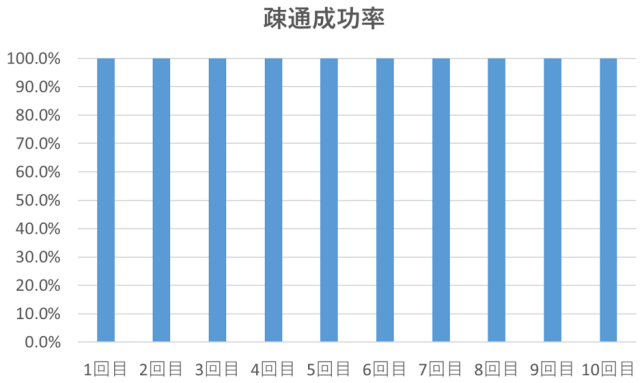


**図 16**　リーダノードからノード B の ICMP 送受信結果

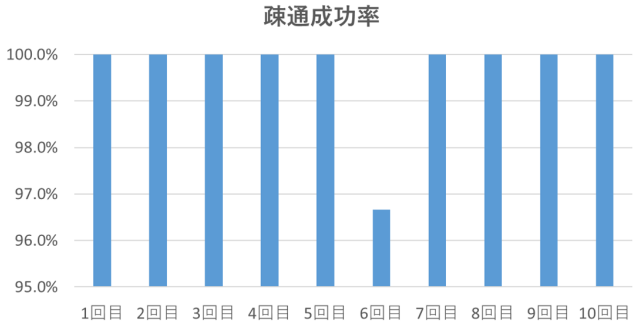


**図 17**　ノード A からリーダノードの ICMP 送受信結果

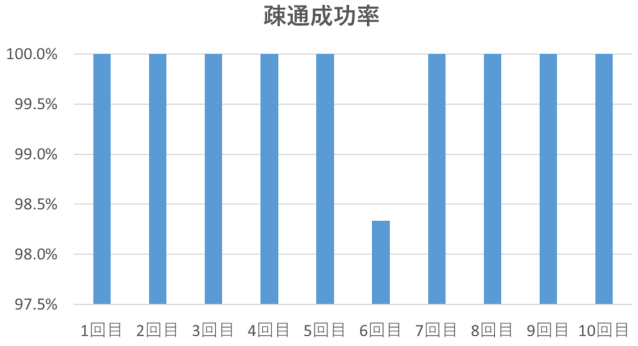


**図 18**　ード B からリーダノードの ICMP 送受信結果

受信結果を図 19 に示す．図からわかるのはアルミホイルで包んだことによりどのセットの疎通成功確率は 100%にはならないことである．最も成功確率が高いのは試行回数 4 回目の 88.3%である．最も成功確率が低いのは 8 回目の 11.7%である．以上の結果から次の分析の章で分析を行う．

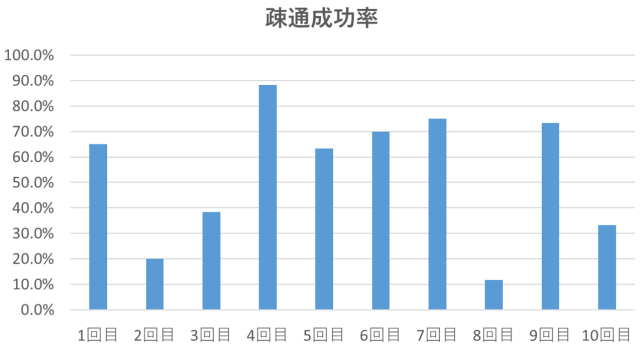


**図 19**　アルミホイルで包んだ場合の ICMP 送受信結果

### 5.2　分析

基礎実験の予想はノード間の通信は一定のものであると予想している．その理由は実際に稼働するノード間の距離は 1m 以内であること，その間に通信の障害となるものはないからである．しかし，結果は先ほどのグラフの通り一定のものとはならない．グラフでは ICMP の応答速度は一定はではなく上下を乱高下するように変動している．遅いときは 600ms を超えるが早い時だと 100ms を切る速さで応答している．なぜこのような結果になったのか分析をする．

応答速度が遅くなる原因の 1 つに電子レンジの電磁波干渉がある．これは電子レンジが発する 2.4GHz 帯の周波数がネットワーク帯に干渉し応答速度の低下に繋がる．しかし，実験では電子レンジの使用は無かったために，その原因は考えにくい．電子レンジを用いているのならグラフが乱高下する理由が見当たらない．なぜなら，電子レンジを用いたのならばその期間は一定的に応答速度が低下するからである．このことから障害物ではない別の原因が考えられる．

応答速度が遅くなる原因にルータがある．ノード間の通信というのは厳密にはノードとノードの間はルータを通して行っている．つまり，ルータ側が不調ならば，それが応答速度の低下に繋がってくる．現状ではグラフが乱高下する原因にルータが関わってくるのが少なくともあると考えられる．しかし，図??のようなサーバに向けて ICMP を送信した場合は安定した挙動であることがわかる．このことから ESP32 の CPU 処理能力が低いことによる処理の遅延が発生した可能性がある．その理由としては ESP32 同士の ICMP の送受信では極端に遅い時と速い時があり，これは ESP32 の CPU 処理が追いつかないときと追いつい

て処理している場合で分けれられると考えられる．条件を変えて ESP32 とサーバで ICMP を送信した場合サーバの CPU 処理能力が ESP32 を上回っているため極端な応答速度の遅さや速さの連続性が見られないからである．

次に図 15，図 16，図 17，図 18，図 19 の実験結果の分析，考察を行う．まずは実験の整理から行う．図 19 のアルミホイルを包んだ異常を再現する実験を除けば，すべての結果はほぼ 100%の ICMP の疎通成功確率を収めている．このことから，ESP32 同士の ICMP の送受信は通信の異常を引き起こすアルミや障害物を除けばほぼ通ることを示している．逆を示すなら ESP32 をアルミで包めば図 19 のような疎通成功確率を下げることができるといえる．このことからアルミは ESP32 同士の ICMP による送受信を妨害する働きを持っていることが実験結果からわかる．

誤検知率の点の評価は実装が開発途中のため，疎通成功確率の点で評価をする．ESP32 同士の相互監視による ICMP の送受信の精度は高いといえる．なぜなら，障害物が何もない状態での疎通成功確率はほぼ 100%なのに対して，アルミホイルを巻いた時の疎通成功確率は格段と減っていることが図 19 からわかる．しかし，この状態では 100%と 0%における異常判断は可能だが，0%¿監視結果¿100%における判定には適用できない．0%から 100%における異常判定について議論の章にて記述する．

## 6. 議論

どのようにグルーピングを行うのか議論する．IoT ノードは複数ある前提だが，説明では 6 台と仮定する．説明ではステップ数を使用して説明を行う．

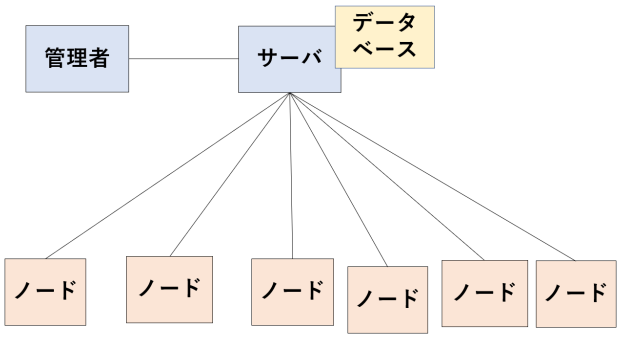


**図 20** ステップ 1

図 20 を用いてステップ 1 の説明を行う．ステップ 1 では最初にグルーピングに必要な稼働率を集める必要がある．個々のノードは図のように最初はバラバラでサーバと個別に繋がっている．サーバはこの時データベースを構築し，繋がったノードの IP アドレスや個体識別 ID などを割り振ったテーブルを作成する．これによりどのノードがどれなのか判別できる．ノードはグルーピングの指標となる稼働率を送信する．これをネームテーブルと名付ける．サーバはそれを受け取るとデータベース内のデータベース内のネームテーブルを用いて個体 ID と紐づけを行い，稼働率テーブルを作成する

図 21 を用いてステップ 2 の説明を行う．ステップ 2 ではサーバに送られてきた稼働率のデータを集約し，ランキングを作成する．ランキングの作成にあたってサーバ内で使われているデータベースを用いて送られてきた情報を集約し整理を行う．この時の情報とは稼働率のことを指す．この時にリーダノードとノードと別れて役職を決めて振り分ける．この時に決められた役職はテーブルによって記録される．役職テーブルと名付ける．ネームテーブルを用いて役職と個体 ID が紐づくようにする．

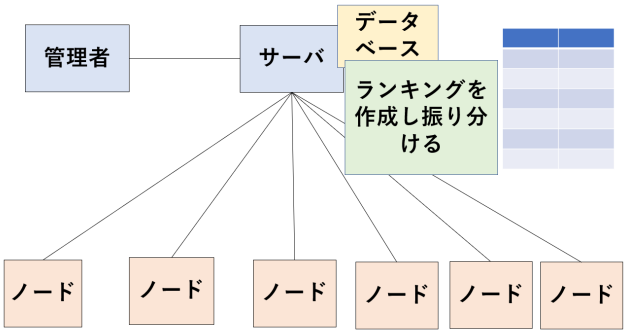


**図 21** ステップ 2

図 22 を用いてステップ 3 の説明を行う．ステップ 3 では，グルーピングを行う際にサーバ内で記録されたデータベースを元ランキングからグルーピングを行う．グルーピングの方法として 1 位のノードと 6 位と 5 位のノードをグルーピングする．同様に 2 位がリーダノードとなり 4 位と 3 位という組み合わせとする．グループの組み合わせが決まったところでその指示をサーバから個々のノードに伝えられる．そしてその結果を反映させたのが図である．反映させた後は提案内容でも述べたように相互監視を行い，互いに ICMP を送りあい応答確認を行う．

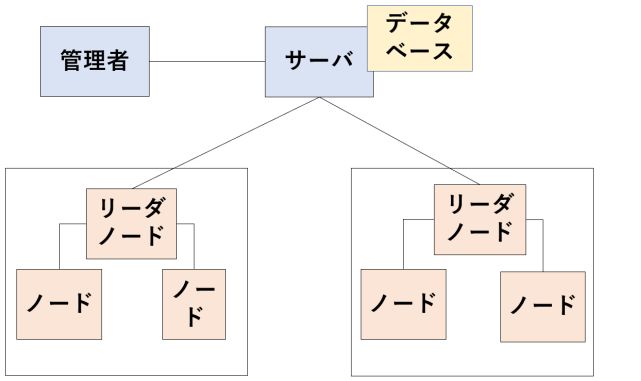


**図 22** ステップ 3

2 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．一度決めたグルーピングを継続して行うことは偏りが生まれてしまう可能性がある．偏りなくバランスをよくするために一定期間ごとに再グルーピングをする必要がある．各グループごとの稼働率に合わせて再びグルーピングをする必要があると提案から判明したのである．解決方法としては過去の疎通成功確率を基に疎通成功確率が高いグループと低いグループの差が 10%になる場合に再グルーピングを行う方法を提案する．これにより，疎通成功確率が恒常的に低いグループを解消して疎通成功確率が高いグループと組むことにより疎通成功確率の低さを解消することができる．

次にリーダノードが異常を起こした場合の議論を行う．本論文の提案の設計上，単一障害点となるのはリーダノードである．リーダノードが異常を起こした場合，ノードとの相互監視，サーバへのデータの送信を行うことができなくなる．その問題に対して解決方法はリーダノードを解消し，ノードすべてにリーダノードと同じを機能を持たせることである．グループ内のノードすべてにリーダノードと同じを機能を持たせ，ローテーションによってサーバへデータを送信する役割を変えることで単一障害点を無くすことに期待ができる．

提案で説明した成功値について議論する．成功値とは正常な状態のリーダノードとノードの正常な疎通結果のことを指す．成功値を基準に相互監視の異常の判定を行う．どのようにして成功値を定めるかについて議論する．正常な状態のリーダノードとノードの疎通結果を正常モデルとして定義づけをする．正常モデルの定義づけでは実際にリーダノードとノード間で ICMP の疎通を行うことで正規分布モデルを作成する．作成した正規分布モデルを基に相互監視の結果を異常か正常か判定をする．監視結果が 0%から 100%においての判断では正常モデルから値が外れているならば外れ値として異常と判断が可能になる．

## 7. おわりに

本論文の課題はノード間の通信の異常である．ノード本体に異常は無いがノード間の通信に異常があった場合にシステムの管理者はノードに異常があるのか通信に異常があるのか区別がつかない．このような課題に対して，本論文の提案は稼働率によってグルーピングされた後のグループ内の相互監視である．提案手法としては ICMP の送受信を用いることでリーダノードとノード間で通信の相互監視が可能になる．

本論文の研究の過程で ESP32 においての ICMP 送受信のプログラムを作成する．これにより ESP32 をノードと見立てることで，リーダノードとノード間で ICMP の送受信の相互監視を可能にする．作成したプログラムを用いて ICMP の疎通成功確率の実験を行う．疎通成功確率は 60 秒 60 回を 1 セットとして，10 セット行うことで約 10 分間の疎通成功確率を評価することができる．アルミホイルを巻いた状態と巻かない状態で実験を行った結果，巻かない状態と比較してアルミホイルを巻いた状態の疎通成功確率は大きく減ることが実験結果から分かる．このことから，10 セットを用いた疎通成功確率の評価はサーバ側の異常判断の手助けに繋がることが分析から分かる．本論文の研究によって通信の異常の検知を行うことができる．これにより，ノードの異常か通信の異常なのか区別がつかない問題に対して，区別をつくことができるという点で貢献ができる．

### 参考文献

[1] Hemmer, H., Ashraf, C. and Powell, A.: Ericsson Mobility Report, Technical report, Ericsson (2020).

[2] 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*, pp. 290-295 (2016)

[3] 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*, pp. 3045–3064 (2018)

[4] 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*, pp. 4019-4032 (2018)

[5] Meonghun, L., Jeonghwan, H., Hyun, Y.: Agricultural Production System Based on IoT, Proc. *IEEE 16th International Conference on Computational Science and Engineering*, pp. 833-837 (2013)

[6] Yuchen, H., Zhiqiang, G., Zhihuan, S.:Adaptive monitoring for transition process using dynamic mutual information similarity analysis, Proc. *Chinese Control and Decision Conference (CCDC)*, pp. 5832-5837 (2016)

[7] 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)

[8] 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)*, pp. 69-74 (2016)

[9] Ibrahim, A., Ali, Alqazzaz., Esam, A., et al.:AD-IoT: Anomaly Detection of IoT Cyberattacks in Smart City Using Machine Learning,Proc. *IEEE 9th Annual Computing and Communication Workshop and Conference(CCWC)*, pp. 305-310 (2019)

[10] Sahil, G., Kuljeet, K., Shalini, B., et al.:A multi-stage anomaly detection scheme for augmenting the security in IoT-enabled applications,Trans. *Future Generation Computer Systems Volume 104*, pp. 105-118 (2020)

[11] Tie, L., Sai, N.:Distributed Anomaly Detection Using Autoencoder Neural Networks in WSN for IoT,Proc. *IEEE International Conference on Communications (ICC)* (2018)

[12] Hichem, S., Sidi, M.S., Mohamad, A.:A lightweight anomaly detection technique for low-resource IoT devices: A game-theoretic methodology,Proc. *IEEE International Conference on Communications (ICC)* (2016)

[13] Marc-Oliver, P., Francois-Xavier, A.:All Eyes on You: Distributed Multi-Dimensional IoT Microservice Anomaly Detection,Proc. *14th International Conference on Network and Service Management (CNSM)*, pp. 72-80 (2018)

[14] Yi, L., Sahil, G., Jiangtian, N., Yang, Z, et al.:Deep Anomaly Detection for Time-series Data in Industrial IoT: A Communication-Efficient On-device Federated Learning Approach,Proc. *IEEE Internet of Things Journal (Early Access)*, pp. 1-11 (2020)

[15] Qiang, Y., Yang, L., Tianjian, C., Yongxin, T.:Federated Machine Learning: Concept and Applications,Proc. *ACM Transactions on Intelligent Systems and Technology* (2019)

[16] Dimitrios, M., Georgios, S., Athanasios, K., Dimitris, S., Joachim, L.:Anomaly detection in IoT devices via monitoring of supply current,Proc. *IEEE 8th International Conference on Consumer Electronics - Berlin (ICCE-Berlin)* (2018)

[17] Briana, A., LiEsa, B., Rahmira, R., Albert, E.:Behavioral Modeling Intrusion Detection System (BMIDS) Using Internet of Things (IoT) Behavior-Based Anomaly Detection via Immunity-Inspired Algorithms,Proc. *25th International Conference on Computer Communication and Networks (ICCCN)* (2016)

[18] Deris, S., Mohd, Y.I., Reza, F.M., Siti, N., Rahmat, B.:Anomaly detection and monitoring in Internet of Things communication,Proc. *8th International Conference on Information Technology and Electrical Engineering (ICITEE)* (2016)

[19] Ismail, B., Burak, Kantarci., Melike, E-K.:Anomaly detection and privacy preservation in cloud-centric Internet of Things,Proc. *IEEE International Conference on Communication Workshop (ICCW)*, pp. 2610-2615 (2015)

[20] Douglas, H.S., Kenneth, M.Z., Yu, C.:Ultra-lightweight deep packet anomaly detection for Internet of Things devices,Proc. *IEEE 34th International Performance Computing and Communications Conference (IPCCC)* (2015)

[21] Yair, M., Michael, B., Yael, M., Yisroel, M., Asaf, S., Dominik, B., Yuval, E.:N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders,Proc. *IEEE Pervasive Computing ( Volume: 17, Issue: 3, Jul.-Sep.)*, pp. 12-22 (2018)

[22] Imtiaz, U., Qusay, H.M.:A Two-Level Hybrid Model for Anomalous Activity Detection in IoT Networks,Proc. *16th IEEE Annual Consumer Communications Networking Conference (CCNC)* (2019)

[23] Hichem, S., Sidi, M.S., Tarik, T.:An Accurate Security Game for Low-Resource IoT Devices,Proc. *IEEE Transactions on Vehicular Technology ( Volume: 66, Issue: 10, Oct.)*, pp. 9381-9393 (2017)

[24] Fangyu, L., Aditya, S., Yang, S., Jin, Y., Xiang-Yang, L., Wenzhan, S.:System Statistics Learning-Based IoT Security: Feasibility and Suitability,Proc. *IEEE Internet of Things Journal ( Volume: 6, Issue: 4, Aug.)*, pp. 6396-6403 (2019)

[25] Rohan, D., Noah, A., Nick, F.:Machine Learning DDoS Detection for Consumer Internet of Things Devices,Proc. *IEEE Security and Privacy Workshops (SPW)*, pp. 29-35 (2018)

[26] Panagiotis, S., Eirini, K., Anastasios, A.E.:VisIoT: A threat visualisation tool for IoT systems security,Proc. *IEEE International Conference on Communication Workshop (ICCW)*, pp. 2633-2638 (2015)

[27] Parth, B., Anderson, M.:HADS: Hybrid Anomaly Detection System for IoT Environments,Proc. *International Conference on Internet of Things, Embedded Systems and Communications (IINTEC)*, pp. 191-196 (2018)

[28] Zilong, Z., Sophie, C., Robert, B., Bogdan, R., Sara, B., Sonia, B.M., Lydia, Y.C.:Robust Anomaly Detection on Unreliable Data,Proc. *49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 630-637 (2019)

[29] Aymen, Y., Takoua, A., Rabah, A.:Hierarchical anomaly based intrusion detection and localization in IoT,Proc. *15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pp. 108-113 (2019)

[30] Dohyung, K., Hyochang, Y., Minki, C., Sungzoon, C., Huijung, K., Minhee, K., Kyungwon, K., Eunseok, K.:Squeezed Convolutional Variational AutoEncoder for unsupervised anomaly detection in edge device industrial Internet of Things,Proc. *International Conference on Information and Computer Technologies (ICICT)*, pp. 67-71 (2018)

 