DAAS: WSN における分散型アプリケーション認証システムの提案と実装

DAAS: Proposal and Implementation of Distributed Application Authenticate System on WSN

川部勝也 †, 福田浩章 †
Katsuya Kawabe†, and Hiroaki Fukuda†
† 芝浦工業大学　工学部
† 芝浦工業大学　工学部
†Faculty of Engineering, Shibaura Institute Technology
†Faculty of Engineering, Shibaura Institute Technology

要旨

IoT の普及に伴い, 無線センサネットワーク (WSN) は広く利用され始めている. 従来の WSN は, ノードにデプロイしたアプリケーションのみをサポートするという方式を取っていた. しかし, この方式は新しいアプリケーションのためにノードを再利用できないという欠点があった. 近年の WSN の OS はその問題を解決するために, ノードの物理的なコンピュータリソースを仮想化することで複数の独立したユーザーによる使用を可能にしている. これらの OS やそれに付随するミドルウェアでは実行するプログラムをネットワーク経由で各ノードにデプロイすることで WSN の管理者が個々のノードに手作業でプログラムをインストールする必要がない. しかし, デプロイされたプログラムの安全は保証しないため, ノードが不正なプログラムを実行してしまう可能性がある. プログラムの認証にはホワイトリストの導入で, 確実に可能である. しかし, ホワイトリストを WSN の一つのノードで管理すると, アクセスが集中し, バッテリーが消費されてしまう. その結果, ノードのダウンによってそれ以降のプログラムの認証が不可能になってしまう.

そこで本研究では, 分散ハッシュテーブル (DHT: Distributed Hash Table) による, 実行可能なプログラムから算出したハッシュ値をホワイトリストとして管理することで, アクセスを分散しつつ, プログラムの認証を行う機構 (DAAS: Distributed Application Auhenticate System) の提案と実装を行う.

1. はじめに

無線センサネットワーク (WSN) は, センサを搭載した多数のノードによって構成されるネットワークである. 実用例として, 自然環境や構造物の監視, 対象物の検知と追跡等がある. 既に配置されている WSN を別の用途で使用する場合, プログラムをインストールし直すか, 新しいノードとプログラムを配置する必要があった. それを解決するために, OS はノードの物理的なコンピュータリソースを仮想化することで複数のプログラムの実行を実現し, WSN を独立した複数のユーザで共有することが可能となっている [1]. 加えて, OS やそれに付随するミドルウェア [3][2] ではネットワーク経由で実行するプログラムをデプロイすることで, 個々のノードに人間が手作業でプログラムをインストールする手間を省いている. しかし, これらの研究では実行するプログラムの安全は保証しておらず, 不正なプログラムを実行してしまう可能性がある.

そこで本研究では, WSN の管理者が実行可能と定義したプログラムから算出したハッシュ値をホワイトリストとして WSN で管理し, ノードがプログラムを実行する際に, ホワイトリストと照合するシステム (DAAS: Distributed Application Authenticate System) を提案, 実装する.

2.DHT によるホワイトリストの分散管理

一般的なホワイトリストによる認証はホワイトリストを一箇所で管理するが, WSN ではホワイトリストを管理するノードにアクセスが集中すると, そのノードのバッテリーだけが大きく消費されてしまう. その結果, そのノードがダウンした場合, それ以降の認証は行えなくなる. そのため, ホワイトリストを DHT で管理することでノードへのアクセスを分散し, バッテリーの節約とホワイトリストの効率的な管理を図る. ここでは, DAAS で用いる DHT と, それを構築するためのアルゴリズムについて述べる.

2.1.Chord

DHT は中央サーバが存在せず, 純粋な P2P 型のネットワーク型分散ファイルシステムを実現することができる技術である. 管理するデータの識別子はファイル名やファイル自体かハッシュ関数を用いて生成されるハッシュ値で表現される. DHT には複数のアルゴリズムが存在するが, 本研究では Chord[4]

を用いる．Chord は識別子が時計回りに増加していくリング状のネットワークを導入している．リングを時計回りで測定した長さが距離であり，この距離の定義から論理的なノード間の遠近が定まる．これによってノード総数 N に対して O(log N) での探索が可能となり，ホワイトリストを用いた認証において重要なデータが存在するか否かを確実に判定できる．しかし，Chord では，論理的に距離が近いノード同士の物理的距離が遠い可能性もあり，通信距離に制約のある WSN ではノード同士の配置によっては通信が行えない可能性もあることから以下に示すアルゴリズムを適用する．

2.2.CSN: Chord for Sensor Networks

CSN[5] は，図 1 のようにリング状ネットワークを階層化することで WSN に適応させたアルゴリズムである．最上位のリングの参加ノード数 N，最大階層数 M に対し，O(MlogN) での探索が可能である．しかし，これだけではホワイトリストの管理に CSN が最適かどうかを判断できない．よって，DAAS で Chord を WSN に適用する際，CSN が DAAS に適しているかを比較するため以下に示す CBT を提案する．

2.3.CBT: Chord for Binary Tree

CBT では図 2 に示す二分木のネットワークを構築し，二分木の末端ノードのみがホワイトリストを管理するノードとする．ノードの総数 N に対しての探索回数は木の階層数を M とした場合 O(MlogN) となる．

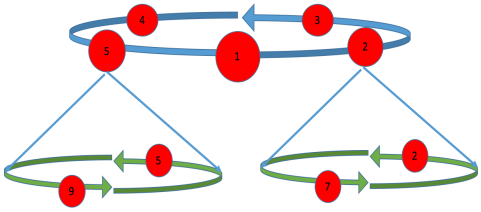

図 1: 階層化されたリング状ネットワーク

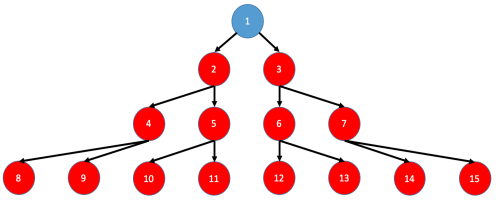

図 2: 二分木によるネットワーク

3. ネットワーク構築手順

本節では，CSN と CBT のネットワークを構築するための手順を述べる．CSN, CBT どちらのアルゴリズムでも WSN 全体でのクラスタヘッドの選出を予め行いネットワークを構築する．以下に CSN, CBT それぞれのネットワーク構築手順を示す．どちらもクラスタヘッドはノード N1 である．

3.1.CSN

図 3 に CSN でのネットワーク構築手順を示す．ネットワークの構築は図 3(i) のように N1 がリング構築メッセージをブロードキャストして開始する (1)．メッセージを受け取ったノードのうち，まだ他のノードにリンクされていないノードは ACK を返す (2)．N1 は受信した ACK の中で物理的に最も近いノード，N2 をリンク先に選び，その旨を N2 に対して通知する (3)．このメッセージには現在のリングに参加しているノード数，クラスタヘッドの ID の情報が含まれる．リンクを結ばれた N2 は同様に自身のリンク先を探索する．これを繰り返し，リングに参加するノード数が限界に到達したか，ブロードキャストに ACK が返ってこなかった時はリングの構築完了メッセージをクラスタヘッドに送信する．以上の手順を最下層のリングまで繰り返し行う．

3.2.CBT

図 4 に CBT でのネットワーク構築手順を示す．CSN と同様に N1 が構築メッセージをブロードキャストし，受け取ったノードのうち，まだ他のノードにリンクを結ばれていないノードが ACK を返す (1), (2)．N1 は受信した ACK の中で物理的に最も近いノードと二番目に近いノード，N2 と N3 に通知を送る．このメッセージには現在の木の階層数，自身が何番目のノードになるか，クラスタヘッドの ID の情報が含まれる．リンクを結ばれた N2, N3 は同様に自身のリンク先を探索し，ブロードキャストに ACK

が返ってこなかった場合は完了メッセージを自身にリンクしている親ノードに通知する．以上の手順をクラスタヘッドが二つの完了メッセージを受け取るまで繰り返す．

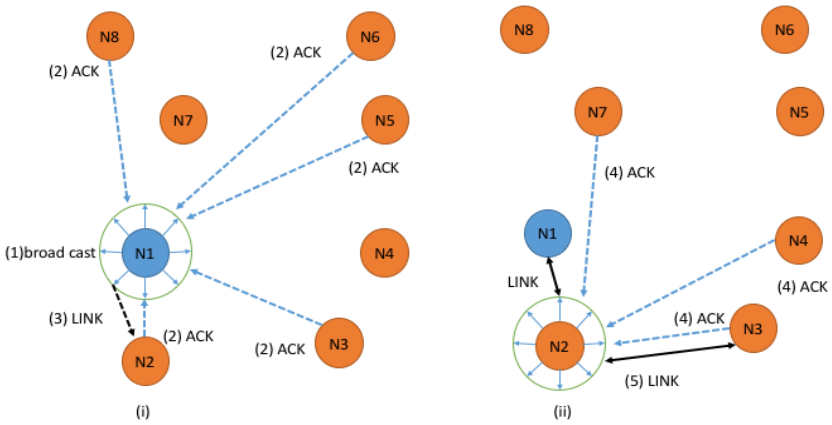

図 3: CSN: ネットワーク構築手順

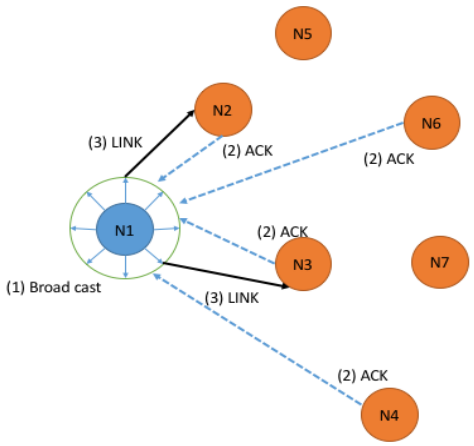

図 4: CBT: ネットワーク構築手順

4. ハッシュ値の割り当て

ネットワーク構築後, 各ノードに DHT のハッシュ値の範囲を割り当てる．ハッシュ値は SHA-1[6] を用いて算出する．CSN, CBT それぞれにおける手順を示す．説明のため, ハッシュ空間を 8bit として考える．

4.1.CSN

まず, 最上層では, 8bit のハッシュ空間を最上層参加ノード数で割り, ハッシュ空間を割り当てる．図 5 にその例を示す (N2 の場合, 担当範囲は 0 から 63 となる)．下層では, 一つ上の層の担当範囲をリング参加ノード数で割り, ハッシュ値を割り当てる．この時 N2 は最上層では 63 が割り当てられているが, 下層では N7 がその値を引き継いでいる．これは, どの層でも時計回りで検索を行うためである．

4.2.CBT

最初にクラスタヘッドである N1 は最大値 (ハッシュ空間が 8bit の場合は 255) を担当する．N1 は末端ノード総数を計算し, 一番左の末端ノード (N4) に伝達する．N4 は通知を受け取ると, 自身に割り当てるハッシュ値を計算する．図 6 にその例を示す (N4 の場合, 担当範囲は 0 から 51 となる)．次に, N4 は親に自身の担当するハッシュ値を通知し, 親は右側の子ノードへ通知を送る．子ノードが存在しない場合は

更に親に通知する. 通知を受け取った子ノードは末端ノードに到着するまで左側の子ノードに通知を送る. 次の末端ノードはその情報を用いて自身のハッシュ値を計算する. これを繰り返して, 左の末端ノードから順にハッシュ値を割り当てていく.

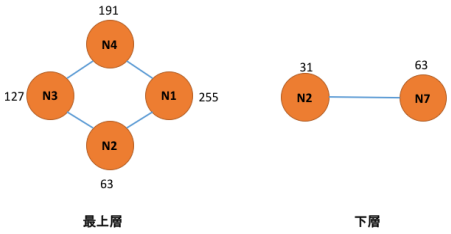

図 5: DHT 構築手順: CSN

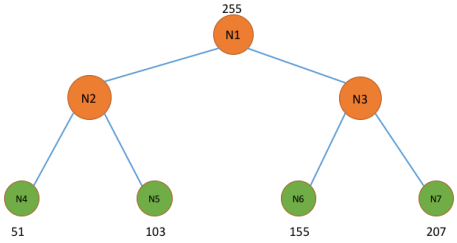

図 6: DHT 構築手順: CBT

4.3. 実装と評価

以上の手順を踏まえて, DAAS の実装を Contiki[7] を用いて行う. Contiki は OSS で開発されている WSN 用の OS である. 今回は評価のため, CSN, CBT のそれぞれのアルゴリズムでの実装を行う. DAAS でのキーの検索時間とキーの検索, 挿入, 更新, 削除によるネットワーク全体の平均バッテリーコストを評価し, ホワイトリストの管理を効率的に行えているかを検証する. 実験は Contiki に付属しているシミュレータ, Cooja を用いて行う.

参考文献

[1] Khan, Imran, et al. ”Wireless sensor network virtualization: A survey.” IEEE Communications Surveys and Tutorials 18.1 (2016): 553-576.

[2] Leontiadis, Ilias, et al. ”SenShare: transforming sensor networks into multi-application sensing infrastructures.” European Conference on Wireless Sensor Networks. Springer, Berlin, Heidelberg, 2012.

[3] Levis, Philip, et al. ”TinyOS: An operating system for sensor networks.” Ambient intelligence 35 (2005): 115-148.

[4] Stoica, Ion, et al. ”Chord: A scalable peer-to-peer lookup service for internet applications.” ACM SIGCOMM Computer Communication Review 31.4 (2001): 149-160.

[5] Ali, Muneeb, and Zartash Afzal Uzmi. ”CSN: A network protocol for serving dynamic queries in large-scale wireless sensor networks.” Communication Networks and Services Research, 2004. Proceedings. Second Annual Conference on. IEEE, 2004.

[6] Xiaoyun Wang, et al. Finding Collisions in the Full SHA-1, Crypto 2005

[7] Dunkels, Adam, Bjorn Gronvall, and Thiemo Voigt. ”Contiki-a lightweight and flexible operating system for tiny networked sensors.” Local Computer Networks, 2004. 29th Annual IEEE International Conference on. IEEE, 2004.