SDN-enabled MANET におけるサイバー攻撃対処に関する研究

李　珠熙 [1,*]　中村　康弘 [2]

概要：現在、様々な災難状況や戦術的な作戦で特別なインフラが要らないモバイルアドホックネットワーク(MANET)の需要が増加されている．そのため MANET は従来からが活発に研究されて、多少良い性能になっている．ところが、インフラが無くても動くように設計された MANET の独立的な特徴とオーバーヘッドによる効率の問題で柔軟なコントロールができない問題がある．それを解決するために、現在 SDN を用いて MANET を精巧にコントロールするための様々な研究が行われている.SDN enabled MANET に関する既存の研究は方法論的なアイデア提示とその可能性の予想がほとんどであり、具体的に適用方法について説明していない．本研究ではサイバー攻撃対処の観点を顧慮して、SDN enabled MANET の適用による得られる重要なメリットとして提案する．

キーワード：ソフトウェア定義ネットワーク， Mobile Ad Hoc Network, サイバー攻撃

The Study about SDN Enabled MANET to Cope with Cyber Attack

Juhee Lee[1,*]　Yasuhiro Nakamura[2]

Abstract: The human cannot predict the place where disasters and modern tactical situations will occur, so the Mobile Ad Hoc Network(MANET) has been demanded. MANET has been actively studied and has become somewhat better but it cannot satisfy the need of today yet. Initially, MANET designed to work in a wireless environment, without special infrastructure. But this made MANET too independent, so it couldn't be controlled in an entire aspect. When we try to control MANET not only by the count of hop or distance but also in a detailed way, it starts to issue an overhead problem, or it cannot cope with the situation. This prevented MANET from reaching flexible control. Therefore, SDN(Software Defined Network) enabled MANET research is now underway to control MANET in a flexible way. However, most of traditional studies purpose SDN enabled MANET in a methodological way, and estimate their potential for utilization. This paper will purpose the way how to apply SDN enabled MANET to cope with cyber attack. This will show significant aspect of the advantage of applying SDN to MANET.

Keywords: Software Defined Network, Mobile Ad hoc Network, Cyber Attack

1. はじめに

様々な災難や大型事故､戦術作戦の共通点は何が､いつ、どこで起きるのかが予想できないということである．任意の場所で発生する事件に常時対応するために数十年前から Mobile Ad hoc Network（以下 MANET）に関する様々な研究が行われている．活発に行われた研究のおかげで、MANET は Proactive、Reactive、そして Hybrid 方式で分かれており、必要な状況によってかなり良い性能で動作している．しかし、従来の MANET 技術では柔軟な要求に対応できない問題がある．また、過去と比べて現代の災難や作戦は、ある組織の力量を投入することで終わらず、他の組織との協力が必要な時もある．そのため、更に複雑な機能への要求が増えている．

その対策の一つとして、Software Defined Network（以下 SDN）と MANET を組み合わせる方法が提案されてある．実は、本来の考え方として MANET と SDN は相反する概念である．MANET はネットワークを構成しているそれぞれのノードがどちらからの指示が無くても相互通信を通じてネットワーク全体の通信ができる事である．逆に SDN はコントローラという概念が必ず必要である．SDN に含まれている全てのノードはあるコントローラに繋がっている事を前提にしてから動作も動作の説明もできる．こういった特徴の事で従来の SDN の活用に関する研究は主に有線環境であった．

本研究では SDN と MANET の組み合わせる方法を簡単に説明して、その後、ある不正のノードから大容量のパケットを一瞬送る Denial of Service 攻撃を模擬する.その後、既存 MANET での、特に Optimized Link State Routing Protocol（以下 OLSR）に置いての影響と、SDN enabled MANET での影響を確認する．その後、同じ攻撃に対する二つの対策の違いを説明しながら、将来、より強健な MANET へ向かう一つの対策として SDN の適用をていじする．

1 防衛大学校情報工学科
Computer Science, National Defense Academy
2 防衛大学校理工学研究科
Graduate School of Science and Engineering, National Defense Academy.
* em57054@nda.ac.jp

2. MANET 及び SDN enabled に関する既存研究

これまでに SDN enabled MANET に関する様々な既存研究においては、SDN enabled MANET をコントローラの位置によってどういうアーキテクチャーになるかに関する研究と、あるアーキテクチャーの前提で実際にルーティングのプロセスを記述して研究などがある．今回は SDN や MANET それぞれの概念及び、普段提案している SDN enabled MANET のアーキテクチャーについての説明はほぼ省略する．代わりに SDN があってから行われるルーティングプロセスについて詳しく説明しながら、そのプロセスが今後サイバー攻撃の対策まで繋がる流れについて説明する．

これからの説明は後で比較の対象になる MANET の OLSR ついて、また、本来 SDN の OpenFlow Network のルーティングプロセスについて、そしてそれをモチーフにした SDN enabled MANET のルーティングプロセスについて説明する．

(1) OLSR のルーティングプロセス

OLSR は[1]RFC3626 に書いてある Proactive 方式のルーティングプロトコルである．Proactive 方式の特徴は、Reactive 方式と相反する概念である．Reactive 方式はコネクション・リクエストがある場合、目的地までの経路を On-demand で設定する．ところが、Proactive 方式は周期的に移動ノード間の経路を設定する事である．

Proactive 方式で常に経路情報をアップデートするためには、経路変化による Flooding を通して最新経路情報をブロードキャストする．そのメッセージの伝達過程で際発生する宛先重複のオーバーヘッドは帯域幅の浪費と繋がる問題であった．OLSR の核心は「Multipoint Relays」（以下 MPR）を選んで、最小の伝送行為だけで、すべての経路情報が伝達されるようにしたのである．MPR の選択によって、制御メッセージを最小化して、オーバーヘッドを減らし、帯域幅の節約ができるようにしたのが OLSR である．

(2) OpenFlow Network でのルーティングプロセス

SDN は大きく三つの要素で区分されます．データの伝送を担当する Data Plane 領域、OS の機能を担当する Control Plane 領域、ネットワークの知能化機能を担当する Application 領域である[2]．OpenFlow は Control Plane と Data Plane 間で使われる代表的なプロトコルである．

OpenFlow Network では OpenFlow Message 遣り取りしながら Topology を Discovery する．全体のトポロジーを認識した後、OpenFlow Network 内部通信としては、Shortest Path Fisrt(SPF)アルゴリズムを用いて最適な経路を計算する．Spanning Tree Protocol(STP)の原因で障害が発生しても、コントローラの制御によって、迅速な復旧が出来る．

(3) SDN enabled MANET のルーティングプロセス

SDN enabled MANET のモデルとしては何らかのモデルが提案されている．「3」によると、コントローラの配置により、Centralized, Hierarchical, Federated などの Approach またはモデルがある．ここで実現する SDN enabled MANET モデル[4]、[5]は、上記の区分を適用すると、Centralized モデルであり、コントローラの機能を MANET ネットワーク中の一つのノードに付与する．そして、全てのノードがコントローラへの経路を持つためにメッセージを遣り取り．そして、すべてのノードが SDN Controller まで届ける状態になる事でコントローラによる制御が可能である．そして 1 ホップではコントローラに届かない場合、マルチホップを用いて到達できる．その後かなネットワーク内部の通信はダイクストラ・アルゴリズムによって最適経路を決める．この方式でネットワークを構成すれば、既存の MANET のように、特別なインフラとして事前に設置する必要はない．ところが、SDN のコントローラは持つことが出来るので、SDN と MANET のメリットを活用する事が出来るような予想が出来る．

3. 実験手順

3.1 ネットワークの構成

この論文での実験は NS-3 シミュレータで行われる．NS-3 はルーティング及び Ad Hoc 研究で活発に使われているネットワークシミュレータである．これから試してみるプロトコルは OLSR と SDN enabled MANET で、ノードの数や配置、ワイヤレス通信の設定などの前提は両方同じで、ルーティングプロトコルの適用だけ異なる事でふたつを比較する．全体のトポロジーは NetAnim という NS-3 の XML 可視化ツールを用いて図 1 に描いてある．

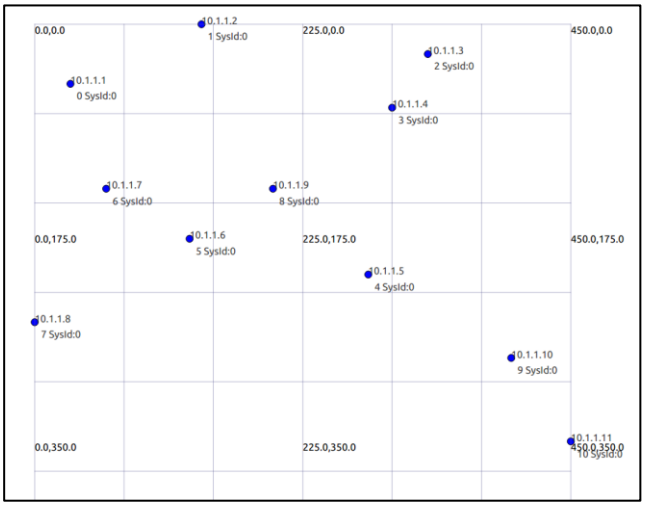

図 1　NetAnim から観察したネットワーク構成

(1) ノード生成

11 個のノードを生成する．ノードの配置はプロトコルに関わらず同一にする．その間の距離は無線通信の到達距離を顧慮する．伝送距離のデフォルト値で設定されている到達距離は約 150m である．つまり、全てのノードが密集し

てしまうと、ルーティングのプロセスを通らなくても無線通信で隣のノードと通信が出来るようになるので、あまり密集しないようにした.

(2) **通信の設定**

各ノード間の無線通信の設定は IEEE 80211g である. 遅延モデルとしては Constant Speed Propagation Delay Model を使用. コントローラとホストの通信は今回は OpenFlow ではなく、UDP をしようした.

3.2 SD-MANET 実現について

本実験では Simple Controller と Normal Controller の二つ Normal Controller は[7]を改善した物であって、Simple の場合はルールを手動で、Normal Controller の場合は自動でせっていする. 最終には自動的にルールを追加する Normal Controller を SDN enabled MANET に適用する事を目標にしているが、今回では本論文が提案している論理の根拠として提示できる最初の段階からスタートしたところで、Simple Controller ルールの手動追加・削除まで実現した.

3.3 DoS 攻撃の模擬

DoS 攻撃は[8]を参考して攻撃を行われる. ここでは 1024Byte のパケットを 1 秒ごとに 60 パケットを送ることからどんどん遅延を発生するモデルである. 他の DoS 攻撃としても Low Rate TCP DoS Attack モデル、MANET 向けの Worm Hole Attack, Black Hole Attack, Jellyfish Attack などのモデルがあるが、今回はその他の攻撃はこりょしていない.

3.4 攻撃対応

MANET での不正ノード発生、または外部からの攻撃については様々な研究結果が既にある. しかし、その攻撃対象ネットワークが MANET なのか SDN enabled MANET かによって、対応方法がかわる. ここではそれについて説明する.

MANET での攻撃対応策

MANET で起きる DoS 攻撃のような不正ノード発生に対する研究はかなり多くの研究がある. MANET の場合、それに対応するメカニズムが基本は入ってない. そういう対応が出来ない場合、不正ノードは自由に攻撃を続けられるし、そして他のノードは、経路選択アルゴリズム経路の混雑に関するパラメータが無い限り、いくら時間かかっても気づかない問題がある. 図 2 に MANET の攻撃対応について示してある.

上記の問題をかしこいアイデアで解決しようとしても、次の問題が発生する. 実はこの論文で模擬した DoS 攻撃の検知は技術的に難しくない. しかし、良いアイデアで研究が行われても既存のルータなら、つまり、SDN をサポートしていない装備なら、適用のためには既存ベンダーからの協力が必要である. ベンダーからの動きが無いと、そのアイデアの反映するのは難しい問題があった. それで、必ず不正ノードを防ぐ機能が必要なところでは、その機能だけ改善するのは大変難し事であり、その一つの機能だけを追加するためにはほぼ新し機器を買うしか仕方が無い. そして、その機能を使おうとしている全てののーどがそのきのうに対応しなければならないので、全てのノードを新たにするしかない.

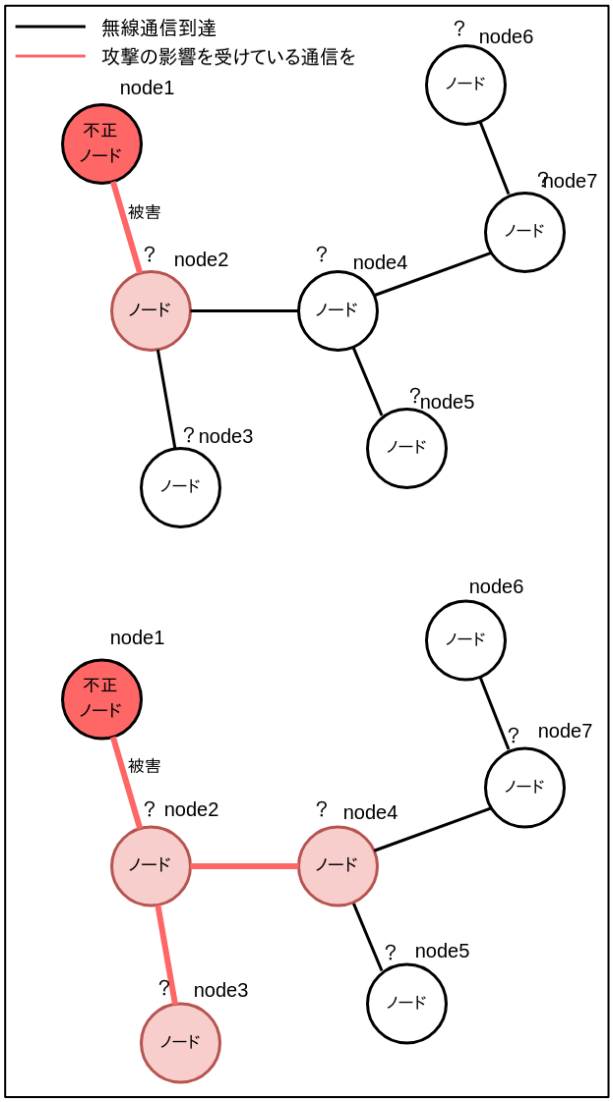

図 2 従来 MANET モデルでの攻撃対応

SDN enabled MANET での対策

SDN での DoS 攻撃検知、及び攻撃防止についても MANET のように既に多様な研究が行われている. まだ、ちょうど SDN enabled MANET 向けの DoS 攻撃防止策は無さそう. SDN enabled MANET のは既存の MANET と比べて、コントローラだけに防御対策を入れる事で全体ネットワークの脆弱性が改善できるようになるのがここでの重要点である. 図 3 に示してあるが、あるノードの異常が検知できつと、次はコントローラのルールに反映することで対応できる部分がある.

SDN enabled モデルの良い点は、本研究のように、最低、手動でコントローラにルールを入れられることで多様な対応が出来ることである. そして、ますます発展する攻撃のパターンに対しても単純にアプリケーション

を入れ替える事で、また対応できるようになるのが重要な可能性であると思っている．

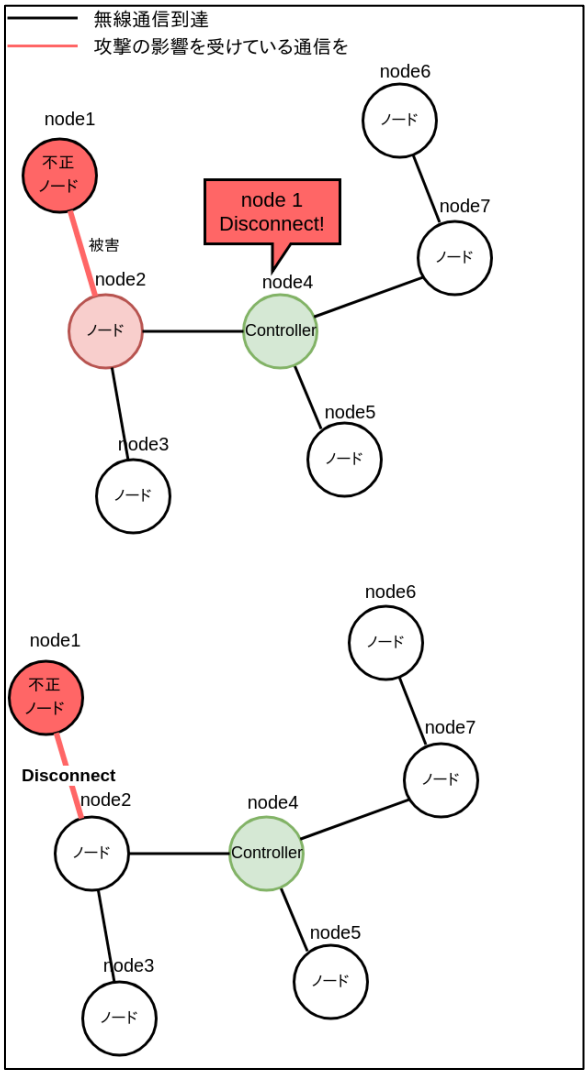

図 3　SDN enabled MANET モデルでの攻撃対応

4. 実験結果

4.1 ルーティング

これまでの結果としては、OLSR と SDN enabled MANET のルーティング・テーブルの生成まで出来てある．図 4 を見ると正常のように見えるが、何らかの問題で SDN コントローラのような役割まではできていない．

```
Node: 5, Time: +60.0s, Local time: +60.0s, Ipv4ListRouting table
  Priority: 0 Protocol: ns3::Ipv4StaticRouting
Node: 5, Time: +60.0s, Local time: +60.0s, Ipv4StaticRouting table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
127.0.0.0       0.0.0.0         255.0.0.0       U     0      -      -   0
10.1.1.0        0.0.0.0         255.255.255.0   U     0      -      -   1
10.1.1.7        10.1.1.7        255.255.255.255 UHS   0      -      -   1
10.1.1.1        10.1.1.1        255.255.255.255 UHS   0      -      -   1
10.1.1.8        10.1.1.8        255.255.255.255 UHS   0      -      -   1
10.1.1.6        10.1.1.6        255.255.255.255 UHS   0      -      -   1
10.1.1.9        10.1.1.9        255.255.255.255 UHS   0      -      -   1
10.1.1.8        10.1.1.8        255.255.255.255 UHS   0      -      -   1
```

図 4　Simple Controller から参考した Routing Table

```
Node: 5, Time: +60.0s, Local time: +60.0s, Ipv4ListRouting table
  Priority: 10 Protocol: ns3::olsr::RoutingProtocol
Node: 5, Time: +60.0s, Local time: +60.0s, OLSR Routing table
Destination                NextHop           Interface       Distance
10.1.1.1                   10.1.1.7          1               2
10.1.1.2                   10.1.1.9          1               2
10.1.1.3                   10.1.1.9          1               3
10.1.1.4                   10.1.1.9          1               2
10.1.1.5                   10.1.1.9          1               2
10.1.1.7                   10.1.1.7          1               1
10.1.1.8                   10.1.1.8          1               1
10.1.1.9                   10.1.1.9          1               1
10.1.1.10                  10.1.1.9          1               3
10.1.1.11                  10.1.1.9          1               4
 HNA Routing Table: empty
```

図 5　OLSR で決まった Routing Table

5. おわりに

本研究では SDN enabled MANET への DoS 攻撃模擬によって、既存の MANET と SDN enabled MANET の対応方式が違う事についてまとめた．この研究の完成度を高める同時に、次の段階としては、手動では無く、自動でルールの登録をする方法について研究しる計画である．

参考文献

[1]T. Clausen and P. Jacquet.「Optimized Link State Routing Protocol (OLSR). IETF RFC 3626」,2003 [online] in URL https://tools.ietf.org/html/rfc3626.

[2]서영석, 이미주,「오픈소스를 활용한 OpenFlow 이해하기: SDN 입문」, Youngjin.com, 2015, pp.49, 121-123.

[3]Jeferson Nobre, Denis Rosario, Cristiano Both, Eduardo Cerqueira, Mario Gerla, 「Towards Software-Defined Battlefield Networking」, IEEE Commucications Magazine, 2016, Vol.54, pp.152-157

[4]Junfeng Wang, Yiming Miao, Ping Zhou, M. Shamim Hossain, Sk Md Mizanur Rahman,「A software defined network routing in wireless multihop network」, Journal of Network and Computer Applications, 2017, Vol. 85, pp. 76-83

[5]Vinod K Mishra; Ayush Dusia, Adarsh Sethi,「Routing in Software-Defined Mobile Ad hoc Networks(SD-MANET)」, 2018, Army Research Laboratory

[6]Network Simulator,「NS-3」, https://www.nsnam.org/(2019/08/22)

[7]NS-3 module for Software Defined Mobile Adhoc Networks, https://github.com/NKSG/sd-manet(2019/08/22)

[8]NS-3 module for dos attack, https://github.com/thakurakhil/dos-attack-ns3(2019/08/22)