モデル検査を用いたタグ VLAN の設定検査

櫻田英樹†

ネットワーク機器の設定は通常人手によって行われる．このため，通信の断絶や情報の漏洩などネットワークに深刻な障害をもたらす設定ミスが発生することがある．本稿ではネットワーク機器，特にタグ VLAN を用いる Ethernet 機器の設定の正しさをモデル検査法を用いて検査する方法を提案する．この方法により，ネットワークに要求される様々な性質を効率的に検査できるようなる．また，設定に誤りがある場合には，この方法により起こりうる可能性のある障害について調べることができる．これは設定ミスを修正する際に便利である．

Model Checking Configurations for Tag VLAN

Hideki Sakurada†

Network nodes such as switches, routers, and terminals are usually configured by network engineers' hands. Even experienced network engineers sometimes make mistakes that may cause serious troubles such as packet loss or leakage of credential information to unexpected third-party. In this paper, we propose a method for verifying configurations of nodes on Ethernet LAN where VLAN-tagging is used. The method is based on model-checking, one of the most widely used formal verification techniques. It enables us to efficiently verify various properties about networks. Moreover, it enables us to find possible troubles in case configuration mistakes exist. They are useful in fixing the configurations.

1. はじめに

ネットワーク機器，特に Ethernet 機器の低価格化により，安価に LAN を構築できるようになり，様々な場所で多くの人がネットワークを利用できるようになった．また同時に機器の高機能化も進み，設定次第で様々な機能を用いて便利なネットワークを構築できるようになった．しかし，このような機能は設定を誤るとネットワークに思わぬ障害をもたらすことがある．

Ethernet 機器の便利な機能の 1 つとして，タグ VLAN [1] と呼ばれる機能がある．これは物理的な LAN を複数の仮想的な LAN セグメント（VLAN）に分割し，VLAN 内で閉じた通信を可能にする機能であり，多くの Ethernet 機器に実装されている．各 VLAN に属するパケットは，パケットに付加されたタグと呼ばれる情報によって識別される．**図 1** にタグ VLAN を用いた簡単なネットワークを示す．このネットワークは VLAN A および VLAN B の 2 つの VLAN から構成される．VLAN A に属する端末 `Term1` からパケットが送信されると，LAN スイッチである `Switch1` がそれを受信し，`Switch2` に転送する．このとき `Switch1` はポート 2 に VLAN A の端末である `Term1` が接続されていることから，パケットに VLAN A を表すタグ `100` を付加する．逆に `Switch2` は VLAN A を表すタグが付加されたパケットを，VLAN A の端末である `Term3` が接続されているポート 2 から送信する．なお，この例ではパケットの転送先は 1 カ所のみであるが，実際には転送先が複数であるのが通常である．このようにして VLAN A に属する端末 `Term1` および `Term3` は VLAN の中で閉じた通信を行うことができる．VLAN B についても同様である．タグ VLAN は企業などでネットワークを部署ごとに論理的に分割するためにしばしば用いられる．

タグ VLAN は便利な反面，設定を誤ると障害の原因となることが多い．たとえば，`Switch2` の設定が誤っており VLAN A を表すタグのついたパケットがポート 2 ではなくポート 3 に転送されるとしよう（**図 2**）．この場合，`Term1` と `Term3` の通信ができなくなるという障害が起こる．さらに，VLAN A のパケットが VLAN B に属する `Term4` に漏洩することになり，機密情報を通信している場合などでは，セキュリティ上の大きな問題ともなりうる．特に，通信ができなくな

† 日本電信電話株式会社 NTT コミュニケーション科学基礎研究所
NTT Communication Science Laboratories, NTT Corporation

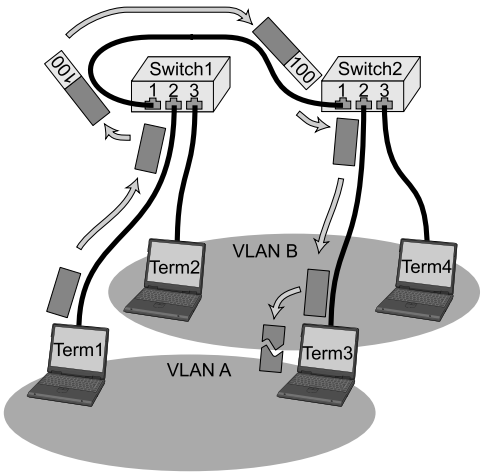

図 1 2 つのタグ VLAN から構成される LAN
Fig. 1 A LAN consists of two tag VLANs.

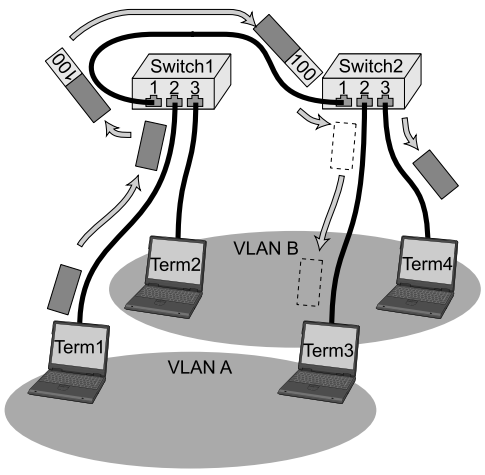

図 2 タグ VLAN の設定ミス
Fig. 2 An error in configurations of tag VLAN.

る障害はユーザがすぐに気付くことが多いが，パケットの漏洩はパケットが端末で廃棄されるだけであるため発見が遅れる恐れがある．

タグ VLAN では機器の間でタグの利用の方法を統一する必要がある．たとえば `Switch1` と `Switch2` では，VLAN A を表すタグとして同じものを用いなければならない．このため，タグ VLAN の設定の正しさを調べるためには，各ネットワーク機器の設定を独立に検査するのではなく，各機器の設定からタグの利用方法が全体として整合することを確認しなければならない．いいかえれば，機器の設定という局所的情報から，パケットの到達性や漏洩などネットワークの大域的性質を確認する必要がある．小規模なネットワークでは人手で確認することも可能であるが，ネットワークの規模が大きくなるにつれ困難になると同時に，確認誤りが生じやすくなる．特に最近では，階層的な VLAN を構成するために 1 つのパケットに複数のタグを同時に付加できる機能が利用されることもあり，その場合にはさらに困難さが増すと考えられる．このように，人手による設定の確認は，規模や正確さにおいて限界がある．

ネットワークを新たに構築したり，ネットワーク機器の設定を変更したりする際には，パケットを実際に送ってテストを行うことが多い．特に，通信ができなくなるような障害はテストによって比較的容易に発見できる．しかし，ネットワーク機器のすべてのポートと，利用される可能性があるすべてのパケットの組合せを網羅することは現実的には不可能であるため，テストに漏れが生じる恐れがある．特にパケットの漏洩を調べる場合には，ネットワーク機器のすべてのポートについて，パケットをダンプする装置または PC を接続し，パケットが到達するか否かを確認しなければならない．また，IP（インターネット・プロトコル）を用いたネットワークでは，ping や traceroute といった検査用のパケットを送信することにより，2 点間の到達性や中継経路を調べることが容易であるのに対し，特に Ethernet の場合にはそのような仕組みがないため，テストがより困難である．このように，設定の正しさをテストによって確認することは網羅性において限界がある．

テストなどによって誤りが存在することが分かった場合でも，その原因がどこにあるかを特定することは，誤りの発見と同様に困難であることがある．たとえばパケットの漏洩が存在することが分かった場合，途中の経路でのパケットの転送やタグの書き換えなどの誤りを特定するためには，パケットがどのように転送され，タグがどう書き換えられるかを確認する必要がある．これは従来は機器の設定から人手で確認を行うか，各中継機器の間にパケットをダンプする機器を狭んで通信をモニタする必要があった．このため，設定の人手での検査やテストと同様の問題があった．

本稿では，パケットの到達性や漏洩など，ネットワークに期待される性質が与えられたとき，この性質を満たすようネットワーク機器が設定されているか否かを，モデル検査法[2] に基づき計算機を用いて検査する方法を提案する．提案方法により，ネットワークの様々な性質を網羅的かつ高速に検査することが可能になる．モデル検査を実行するソフトウェアとして NuSMV[3] を用いる．NuSMV は CMU で開発されたモデル検査ソフトウェア SMV[4] を拡張したものであり，CTL

(計算木論理)[5)~7)]，および LTL（線型時相論理）[8)] の論理式で与えられた仕様を記号モデル検査[4),9),10)] により高速に検査できる．本方法ではまず，ネットワーク機器の配線の情報と設定の情報から計算機上にネットワークのモデルを構成する（2 章）．次に，要求仕様，すなわちネットワークに期待される検査すべき性質を CTL の論理式として記述する（3 章）．CTL を用いることにより，2 点間の到達性以外にも様々な性質を記述・検証することが可能になり，ネットワークの利用目的に応じた個々のネットワークに特有の性質を調べることができる．そしてネットワークのモデルと要求仕様の論理式をモデル検査ソフトウェアに入力し，検査を行う（4 章）．モデル検査では性質が満たされない場合に反例を求めることができる場合がある．これを用いて設定誤りによって生じる障害の具体例を求めることができ，設定誤りの箇所の特定などに利用することができる．さらに，検査に必要な時間の目安を得るために，人工的に生成したネットワークのデータに対して検査を行い，時間を計測し，現実的な時間内に検査を行うことができることを示した（5 章）．本方法は広い意味ではシミュレーションによる検査であり，一般にそのようなシミュレーションを網羅的に行おうとすると多くの計算が必要になるが，効率的なモデル検査アルゴリズムおよびその実装である NuSMV を利用することにより，網羅的な検査を高速に行うことができる．

提案方法は，ネットワークの配線と設定の情報をもとに機器の設定の誤りを発見するものであるため，ネットワーク機器の故障や，ケーブルの断線・配線ミスのような物理的原因による障害を発見することはできない．この意味で，テストによる障害発見と補完的関係にあるといえる．また，障害の原因が物理的なものである場合でも，本方法を用いて原因が機器の設定にあるのか否かという切り分けに利用できる．

本研究と関連の深い研究として，Guttman らによるファイアウォールのパケットフィルタの設定の正しさを検査する方法[11)] や，IPSec ゲートウェイの暗号化設定の正しさを検査する方法[12)] があるが，タグ VLAN の設定を検査を行ったのは本研究が最初である．これらの研究との比較は 6 章で再度行う．

2. ネットワークのモデル

タグ VLAN を用いた Ethernet のモデルを構成する方法について述べる．まず構成しようとするモデルについて述べ，次にこれを自動的に構成する方法について述べる．モデルを構成するには，状態の表現方法，初期状態，および状態遷移規則を与える必要がある．

タグ VLAN の設定がパケットの転送経路にどう反映されるかを調べるのが目的であるため，パケットが状態を遷移するモデルを考える．パケットの状態を次の 4 つの変数を用いて表す．

- パケットが存在するノード（スイッチや端末）を表す変数 `node`
- パケットが存在するノード上のポートを表す変数 `port`
- パケットに付加されているタグを表す変数 `tag`
- パケットの移動状態を表す変数 `phase`

パケットにタグが付加されていない場合は変数 `tag` の値は `null` であるとする．変数 `phase` は `incoming`, `outgoing` および `discarded` のいずれかの値を持ち，それぞれパケットがノードに受信された状態，ノードから送信された状態，および破棄された状態を表す．

モデルの初期状態は，パケットが最初に送信されたときの状態を表し，どの端末から送信されたパケットの性質を検査するかによって決まる．初期状態はたとえば次のように，

```
node = Term1 &
port = 1 &
tag = null &
phase = outgoing
```

変数の値を指定することにより与える．これは，ノード `Term1` のポート `1` からタグのないパケットが送信された状態を表す．また，単に

```
node = Term1 &
phase = outoging
```

のように一部の変数のみの値を指定してもよい．この場合は，初期状態の集合は `Term1` から送信された任意のパケットの状態からなる．さらに，初期状態をまったく指定しない場合は，任意の状態が初期状態になりうる．初期状態の指定は，どのような性質を検査するかにも依存するため，要求仕様記述（次章）と合わせて考える必要がある．

パケットはネットワークの物理的トポロジ，すなわちケーブルの配線と，ノードの VLAN 設定に従って状態を遷移する．図 1 のネットワークにおける遷移の一例を**図 3** に示す．左下のパケットはトポロジに従って端末 `Term1` のポート `1` からスイッチ `Switch1` のポート `2` に移動するため，変数 `node` および `port` がそれぞれ `Term1` および `1` である状態からそれぞれ `Switch1` および `2` である状態へ遷移する．このときタグは変化しないので，変数 `tag` の値は `null` のま

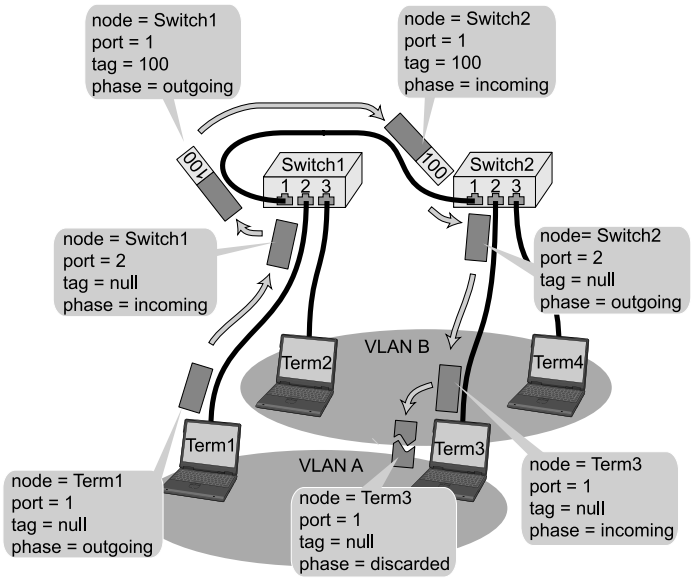

図 3 状態遷移列の例
Fig. 3 A series of state transitions.

までである．変数 phase の値は，パケットが送信されようとしている状態を表す outgoing から受信された状態を表す incoming へ変化する．次に，パケットはスイッチ Switch1 の VLAN 設定に従って，ポート 2 からポート 1 へスイッチされ，VLAN A に対応するタグ 100 を付加される．このとき変数 node は変化せず，変数 phase は incoming から outgoing へ変化する．以上のように，パケットは変数 phase の値が outgoing の状態にあるときトポロジに従って遷移し，incoming の状態にあるのときノードの VLAN 設定に従って遷移するものとする．また，変数 phase の値が discarded のとき，パケットは状態を遷移しないものとする．たとえば，端末 Term3 によって受信されたパケットは他の端末に転送されないため破棄されて discarded 状態に遷移し，その後は状態を遷移しない．この例ではパケットは端末 Term3 にしか到達しないが，一般に Ethernet のようなブロードキャスト型のネットワークではパケットは途中で複製されて複数の端末に転送される．たとえば，Switch2 がパケットを Term4 にも転送する場合が考えられる．この場合は，Switch2 が受信したパケットが，node = port2 である状態にも node = port3 である状態にも遷移可能であるとする．すなわち，パケットの複製をモデルの非決定性によって表現する．

本稿で利用するモデル検査ソフトウェアではモデルを論理式によって定義される関係を用いて表現する．図 1 のネットワークのモデルは**図 4** のように表現される．ここで，case 文

```
case
 cond_1 : exp_1 ;
 cond_2 : exp_2 ;
  ⋮
 cond_n : exp_n ;
 esac;
```

の真偽は，条件 $cond_1, cond_2, \ldots, cond_n$ に真であるものが存在しない場合は真であるとする．存在する場合はその最初のもの $cond_i$ に対応する exp_i の真偽をこの case 文の真偽とする．遷移後の状態における各変数 node，port，tag および phase の値を next(node)，next(port)，next(tag) および next(phase) で表し，可能な遷移は論理式を真とするような各変数の値によって表される．たとえば，図 3 の左下の状態では，phase = outgoing であるので，3–17 行目の case 文が評価され，node = Term1 かつ port = 1 であるので，12 行目の条件が最初に真となり 13 行目が評価され，図 3 における次状態については確かに真となることが分かる．パケットは通常複数の転送先，すなわち複数のポートに転送される．この場合，exp_i に相当する箇所にいずれの転送先についても真となり，それ以外で偽となる論理式を書けばよい．たとえば，Switch1 のポート 1 でタグ 100 を持つパケットが受信されたとき，これがポート 2 だけでなくポート 4 にも転送される場合には，図の 23 行目の next(port) = 2 を論

```
case
 phase = outgoing:
  case
   node = Switch1 & port = 1:
    next(node) = Switch2 & next(port) = 1 & next(tag) = tag & next(phase) = incoming;
   node = Switch1 & port = 2:
    next(node) = Term1 & next(port) = 1 & next(tag) = tag & next(phase) = incoming;
    :
   node = Switch2 & port = 1:
    next(node) = Switch1 & next(port) = 1 & next(tag) = tag & next(phase) = incoming;
    :
   node = Term1  & port = 1:
    next(node) = Switch1 & next(port) = 2 & next(tag) = tag & next(phase) = incoming;
    :
   TRUE:
    next(node) = node & next(port) = port & next(tag) = tag & next(phase) = discarded;
  esac;
 phase = incoming:
  case
   node = Switch1:
    case
     port = 1 & tag = 100:
      next(node) = node & next(port) = 2 & next(tag) = null & next(phase) = outgoing;
     port = 2 & tag = null:
      next(node) = node & next(port) = 1 & next(tag) = 100 & next(phase) = outgoing;
     port = 1 & tag = 200:
      next(node) = node & next(port) = 3 & next(tag) = null & next(phase) = outgoing;
     port = 3 & tag = null:
      next(node) = node & next(port) = 1 & next(tag) = 200 & next(phase) = outgoing;
     TRUE:
      next(node) = node & next(port) = port & next(tag) = tag & next(phase) = discarded;
    esac;
    :
   TRUE:
    next(node) = node & next(port) = port & next(tag) = tag & next(phase) = discarded;
  esac;
 phase = discarded:
  next(node) = node & next(port) = port & next(tag) = tag & next(phase) = phase;
esac
```

図 4 状態遷移規則の論理式
Fig. 4 Logical formula for a set of transition rules.

理和 (`next(port) = 2 | next(port) = 4`) で置き換えればよい.

次に，ネットワークのモデルを自動的に構成する方法について述べる．モデルを構成するために必要な情報は，ネットワークの物理的トポロジと，各機器のタグ VLAN の設定である.

ネットワークの物理的トポロジとはすなわち，どのネットワーク機器のどのポートが他のどのネットワーク機器のどのポートと接続されているか，という情報である．図 1 のネットワークの場合のトポロジは**表 1** のようなデータである．たとえば最初の行は，ノード `Switch1` のポート 1 はノード `Switch2` のポート 1 とケーブルで接続されていることを表す．本稿ではトポロジのデータの取得方法について特に述べないが，既

表 1 トポロジのデータ
Table 1 Data for the topology of a network.

ノード	ポート	ノード	ポート
Switch1	1	Switch2	1
Switch1	2	Term1	1
Switch1	3	Term2	1
Switch2	2	Term3	1
Switch2	3	Term4	1

存のネットワーク管理ツールのデータや，ネットワーク機器が持っている MAC アドレスとポートの対応データベース（いわゆる FDB）からある程度自動的に作成することができると考えられる．また，すでに発生している障害の原因を切り分けるためなどに本方法を用いる場合は，障害などにより稼働中のシステムのトポロジを自動的に作成することが困難な場合がある．この場合は，トポロジのデータをネットワークの設計仕様書などから作成することも考えられる．この場合，本方法による検査で障害が検出されないならば，原因はネットワーク機器の設定ではなく，設計仕様書のトポロジと実際のトポロジとの不整合（配線ミス，断線など）などであることが分かる．

トポロジのデータは，状態遷移モデルのトポロジに従う遷移規則に変換される．たとえば表 1 のデータは図 4 の 2–17 行目に変換される．パケットはケーブルを双方向に移動するため，トポロジのデータの各エントリはモデルの 2 つの遷移規則に対応する．たとえば表 1 の最初のエントリは，図 4 において `Switch1` から `Switch2` への移動を表す 4–5 行目の遷移と，逆方向の移動を表す 9–10 行目の遷移に変換される．この変換はテキスト処理によって実現可能である．

各機器のタグ VLAN の設定は，機器の設定ファイルの一部である．機器の設定ファイルは通常 TFTP などのプロトコルで取り出せることが多い．また，`telnet` コマンド，シリアルコンソール，web ブラウザなどから管理用のインタフェースにログインして表示することもできるため，いずれにせよ自動的に取得することができる．タグ VLAN の設定データの例を**図 5** および**図 6** に示す．これらはそれぞれエクストリームネットワークス社の製品とエスエムシーネットワークス社の製品の例で，両方とも図 1 のネットワークにおける Switch1 の動作をさせるための設定である．

図 5 ではまず VLAN A および B を宣言し（1–2 行目），次にそのそれぞれについて（4–6 行目および 8–9 行目）VLAN 識別用のタグ（それぞれ `100` および `200`）および各 VLAN に属するポートが設定されている．各ポートの設定に `tagged` または `untagged` の指定がある．`tagged` の指定があるポートでは，VLAN のパケットは，タグのついた状態で送受信される．`untagged` の指定があるポートでは，VLAN のパケットは，タグのない状態で送受信される．タグのない状態は，本稿のモデルではタグ `null` が付加された状態に相当する．

図 6 の例でも同様に VLAN を宣言（1–3 行目）し，今度はポート（interface）ごとにポートがどの VLAN に属するかを設定する．図 5 および図 6 のいずれの設定においても，ポートとタグの組が与えられたとき，それがどの VLAN に属するか，あるいはまったく属さないかを判断することができる．ネットワーク機器は，この情報をもとにパケットの転送を行っている．たとえば，ポート 2 からタグのないパケットを受信した場合には，このパケットが VLAN A に属することが分かり，同じ VLAN に属するすべてのポートとタグの組を転送先として選ぶ．この場合はポート `1` とタグ `100` の組のみであるので，タグを `100` に書き換えてポート `1` から送信する．

```
1  create vlan "VLAN-A"
2  create vlan "VLAN-B"
3
4  configure vlan "VLAN-A" tag 100
5  configure vlan "VLAN-A" add port 1 tagged
6  configure vlan "VLAN-A" add port 2 untagged
7
8  configure vlan "VLAN-B" tag 200
9  configure vlan "VLAN-B" add port 3 untagged
10 configure vlan "VLAN-B" add port 3 untagged
```

図 5 エクストリームネットワークス社の製品の VLAN 設定例
Fig. 5 A VLAN configuration on a LAN switch product by Extreme Networks Corp.

```
1  vlan database
2    vlan 100 name VLAN-A
3    vlan 200 name VLAN-B
4
5  interface ethernet 1/1
6  switchport allowed vlan add 100,200 tagged
7
8  interface ethernet 1/2
9  switchport allowed vlan add 100 untagged
10 switchport native vlan 100
11
12 interface ethernet 1/3
13 switchport allowed vlan add 200 untagged
14 switchport native vlan 200
```

図 6 エスエムシーネットワーク社の製品の VLAN 設定例
Fig. 6 A VLAN configuration on a LAN switch product by SMC Networks Corp.

タグ VLAN の設定データは，状態遷移モデルの VLAN 設定に従う遷移の規則に変換される．たとえば図 5 あるいは図 6 のデータは，図 4 では VLAN 設定に従う遷移を記述した部分である 18–36 行目のうち，`Switch1` の設定に関する部分である 20–32 行目に変換される．各遷移規則は，ネットワーク機器が行う動作をそのまま記述すればよい．たとえば，VLAN A に関しては 22–23 行目および 24–25 行目において 2 つの遷移規則が記述されている．前者はポート `1` からタグ `100` が付加されたパケットを同じ VLAN に属するポート 2 からタグなしで転送する動作である．後者は

その逆の転送である．このような VLAN 設定のモデルへの変換もテキスト処理によって実現可能である．

なお，ケーブルの接続されていないポートからパケットが送信された場合や，パケットが他のポートに転送されない場合は，パケットは破棄される．これは，破棄状態（phase = discarded）への遷移によって表現される．図 4 の 15–16 行目および 30–31 行目などがこれに相当する．37–38 行目は破棄状態のパケットが同じ状態にとどまり続けることを表している．これは，モデル検査ソフトウェアではすべての状態について次状態が存在しなければならないという前提条件があるためである．

本稿では，1 つのパケットにはたかだか 1 つのタグが付加されるタグ VLAN を考えたが，タグをタグの配列に置き換えたモデルを構成することも容易である．この拡張により，1 つのパケットにある上限以下の複数個のタグが同時に付加されるような，拡張されたタグ VLAN にも本方法は応用可能となる．

3. 要求仕様記述

本提案方法では，ネットワークの要求仕様，すなわちネットワークに期待される検査すべき性質を，CTL の論理式を用いて記述する．CTL は状態遷移の分岐構造の性質を記述できる．このため，提案方法のモデル化においては，パケットの中継ノードにおける複製による分岐構造の性質を記述できることになる．

定義 1（CTL 論理式） CTL の論理式を次の文法により定義する．

$P ::= ap$
| ! P | P & P | P \| P | P -> P
| AX P | AF P | AG P
| EX P | EF P | EG P
| A[P U P]
| E[P U P]

論理式の形式的意味の定義は文献 2) に譲り，本稿では以降の議論に必要となる CTL の論理式の直感的意味について述べる．CTL の論理式の真偽はモデルの状態に対し評価される．ap は原子論理式であり，変数 node，port，tag および phase を左辺とし，これらの変数のとりうる値を右辺とする等式である．! P，P & P，P | P および P -> P はそれぞれ否定，論理積，論理和，および含意である．AX P，AF P，および AG P の意味はそれぞれ，「任意の次状態において P が成り立つ」，「与えられた状態からの任意の状態遷移列において，将来 P が成り立つ」，および「与えられた状態からの任意の状態遷移列において，つねに P が成り立つ」である．EX P，EF P，および EG P の意味はそれぞれ，「P が成り立つ次状態が存在する」，「与えられた状態からの状態遷移列で，将来 P が成り立つものが存在する」，および「与えられた状態からの状態遷移列で，つねに P が成り立つものが存在する」である．A[P U Q] の意味は，「与えられた状態からの任意の状態遷移列において，将来 Q が成り立ち，Q が成り立つまでは P が成り立つ」である．E[P U Q] の意味は，「与えられた状態からの状態遷移列で，将来 Q が成り立ち，Q が成り立つまでは P が成り立つものが存在する」である．

モデル検査はモデルの初期状態において論理式が成り立つか否かを判定する手続きである．このため本方法ではネットワークの性質をモデルの初期状態において成り立つ CTL の論理式を与えることにより記述する．また，検査において注目するパケットを指定するために，論理式だけでなく初期状態も検査する性質に応じて決める場合がある．

例 1 端末 Term1 から送信したパケットが端末 Term3 に到達するという，通信が可能であることを意味する性質を考える．この場合，まず，パケットが最初に端末 Term1 上に存在することをモデルの初期状態として，

```
node = Term1
```

のように指定しておき，CTL の論理式として

```
EF (node = Term3)
```

を記述する．

これは，Term1 のポート 1 で送信フェーズにあるパケットを初期状態とし，この状態から始まる状態遷移列で将来ノードが Term3 になるものが存在するということになる．ここで指定した初期状態には，タグ，ポート番号，フェーズが異なる複数の状態が含まれる．このため，CTL の論理式はこれらの状態すべてについて性質が成り立つという主張と解釈される．

本方法のモデル化では，パケットの複製をモデルの非決定性によって表現するため，複製されたすべてのパケットを表す AF でなく，複製されたパケットの 1 つを表す EF を用いる．別のモデル化として，モデルの状態が複製されたパケットの集合を表すようにモデル化する方法も考えられる．この場合は，AF を用いて，どの遷移列においても将来必ずパケットの集合に端末 Term3 のパケットが含まれる状態に遷移する，という記述のしかたが可能である．ただし，このモデル化の方法では，特に Ethernet のようにパケットの複製が頻繁なネットワークではモデルの状態数が多くなり，効率的でない場合がある．

例 2 端末 Term1 から送信したパケットが他の VLAN に属する端末 Term3 に到達しない，という性質は，前の例の CTL の論理式の否定

```
! EF (node = Term3)
```

によって記述される．これはパケットの漏洩が起こらないというセキュリティ上重要な性質を表す論理式である．

例 3 パケットがループしない，という性質は次の論理式で記述される．

```
AF (phase = discarded)
```

これは，パケットは任意の状態遷移列について，将来必ず廃棄フェーズに遷移するということを意味する．本方法ではパケットの廃棄は廃棄状態への遷移として明示的にモデルの中で表現されるため，このように記述できる．ここでは，初期状態を指定していない．すなわち，任意の状態を初期状態として，上の論理式が成り立つという性質を表している．Ethernet のようなブロードキャスト型のネットワークの場合，パケットのループが発生するとネットワークが輻輳し通信速度の低下や通信断絶を招く．このため，パケットのループの検出は運用上重要な性質である．

例 4 ネットワーク上に，機密情報を保管しているノード credential およびこのノードへのアクセスを監視するノード audit が存在するとする．このとき，credential に届くパケットは必ず audit により監視される，すなわち audit に届く，という性質は以下の論理式で記述される．

```
EF (node = credential)
-> EF (node = audit)
```

このように，EF などの時相演算子を複数組み合わせてネットワークに関する様々な性質を CTL で記述できる．

本提案方法ではネットワークの要求仕様を CTL の論理式を用いて記述したが，CTL のかわりに他の言語を用いて記述することも考えられる．この場合，言語の記述能力と記述した性質のモデル検査の効率について考慮する必要がある．一般に両者はトレードオフの関係にある．すなわち，記述能力が高い言語ほどモデル検査の効率は悪くなる．また，両者を考慮してある性質に関しては CTL を用い他の性質に関しては他の言語で記述するという利用法も考えられる．

記述能力については，通信が可能であること（例 2），パケットが漏洩しないこと（例 2）およびループがないこと（例 3）に関しては CTL で記述できると同時に，システム検証の際に CTL と同様によく用いられる LTL でも記述できる．しかし，例 4 のようなパケットの複製による分岐構造に関する性質を表現することは，提案方法で用いているモデル上ではできない．逆に，パケットの転送経路の 1 つを線として見るような性質については CTL のかわりに LTL を用いなければならない場合があると考えられる．

モデル検査の効率については，CTL も LTL もともに記号モデル検査により高速に検査できることが知られている．特に CTL のモデル検査の計算量はモデルと論理式の大きさについて線型時間であることが知られている[13]．

4. 検　　査

ネットワークの性質を検査するためには，ネットワークのモデルの論理式（2 章）および検査する性質を表す（3 章）を入力としてモデル検査ソフトウェアを実行すればよい．

モデル検査ソフトウェアは，モデルの初期状態における CTL の論理式の真偽を判定し，その結果を出力する．たとえば図 1 のネットワークのモデル（図 4）に対して，3 章の例 2 の論理式を入力して検査を行うと，この性質は真（true）であるので，出力は次のようになる．

```
-- specification
   !EF node = Term4
   is true
```

「パケットが到達しない」というような性質を検査し，この性質が偽である場合には，モデル検査ソフトウェアは反例（Counterexample）として，パケットが到達するような状態遷移の列を出力する．例 2 の性質を同じモデルについて検査すると，出力は**図 7** のようになる．ここでは，ノード Term1 のポート 1 からタグのないパケットが送信されようとしている初期状態（State: 1.1）から，変数 node，port および phase が変化して次状態（State: 1.2）に遷移する．以降遷移の繰返しによりパケットが Term3 に到達するまでのすべての遷移が記述されており，経路およびタグの書き換えがどのように行われたか分かる．

パケットの漏洩やそれに類似の問題が発生した場合には，原因の特定のために，パケットがどのような経路を通り漏洩したのか知る必要がある．反例の出力はまさにパケットの経路を出力するものであり，障害の原因の特定に役立つ．また，モデル検査ソフトウェアには与えられた初期状態から可能な状態遷移を対話的に選択してシミュレーションを行う機能もあり，原因の特定などに役立てることが可能である．

```
-- specification
   !EF node = Term3
   is false
-- as demonstrated by the following
   execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
-> State: 1.1 <-
  node = Term1
  port = 1
  tag = null
  phase = outgoing
-> State: 1.2 <-
  node = Switch1
  port = 2
  phase = incoming
-> State: 1.3 <-
  port = 1
  tag = 100
  phase = outgoing
-> State: 1.4 <-
  node = Switch2
  phase = incoming
-> State: 1.5 <-
  port = 2
  tag = null
  phase = outgoing
-> State: 1.6 <-
  node = Term3
  port = 1
  phase = incoming
```

図 7 反例
Fig. 7 A counterexample.

5. 性能評価

前述したように CTL のモデル検査では線型時間のアルゴリズムが知られており，高速に検査を実行することができる．本章では，検査に必要な時間の定量的な目安を得るため，人工的にランダムに構成したネットワークのデータについて検査時間を測定する．

検査を行うネットワークはフルメッシュ型でランダムに構成したものである．すなわち，すべてのノードのポートがランダムに選ばれたいずれかのノードのいずれかのポートとケーブルで接続されているとする．さらに，パケットのスイッチングとタグの書き換えもランダムに行われるとする．すなわち，任意のポートで受信した任意のパケットについて，ランダムに選ばれたポートへランダムに選ばれたタグを付けて転送する．

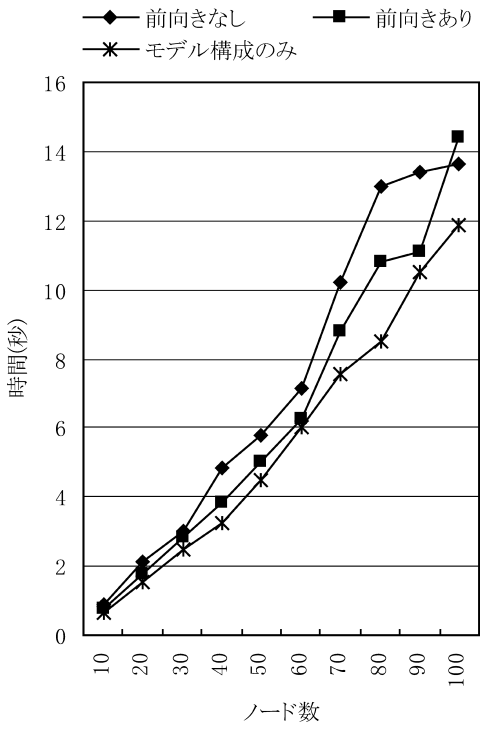

図 8 到達不能性の検査時間
Fig. 8 Time consumed by verification of unreachability.

このネットワーク上で，あらかじめ決められた 2 つのポート間でパケットが到達しないという性質および，ループが存在しないという性質を検査する．以下では 1 ノードあたりのポート数を 24, 使用するタグの数を 20 にした場合の検査時間を示す．モデル検査ソフトウェアとして NuSMV のバージョン 2.2.2 を用いた．ハードウェアとして，インテル社の Xeon プロセッサの 3.20 GHz およびメモリを 4 GB 搭載した計算機を用いた．

あらかじめ決められた 2 つのポート間でパケットが到達しないという性質（到達不能性）を検査するために初期状態として

```
node = 0 & port = 0
```

を指定し，要求仕様として

```
! EF (node = 1 & port = 0)
```

を指定する．これは例 2 と類似の性質である．検査時間を**図 8** に示す．前向き探索を行わずに検査した場合，前向き探索を行い検査した場合，および，検査を行わずその前処理であるモデルの内部表現の構成のみ

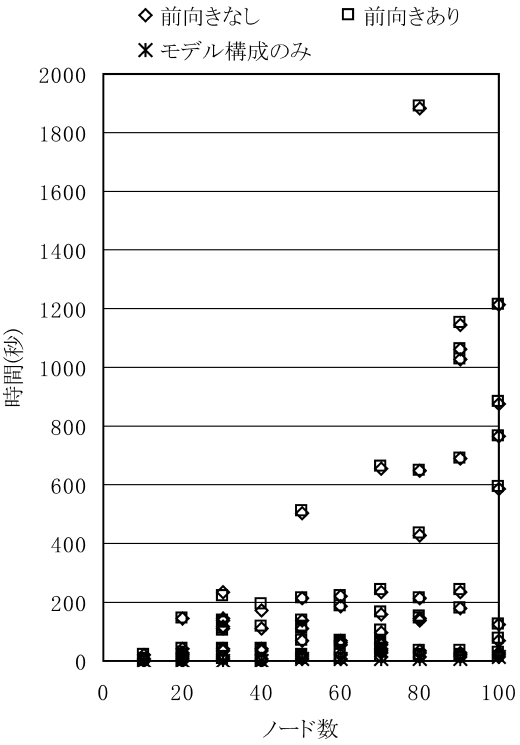

図 9 無ループ性の検査時間
Fig. 9 Time consumed by verification of loop-freeness.

を行った場合の検査時間を示した．前向き探索とは，あらかじめモデルの状態を初期状態から到達可能なものに制限する処理で，`NuSMV` で`-f` オプションを使用する場合に相当する．モデルの状態数に対して，初期状態から到達可能な状態の数が小さい場合には，前向き探索を行うと検査時間を短くすることができる．図に示した結果においても，その効果が現れている．いずれの場合も，検査は十数秒で完了している．

ループが存在しないという性質（無ループ性）を検査するために，要求仕様として

```
AF (phase = discarded)
```

を指定した．また，ネットワーク上で作られるすべてのパケットについてループが起こらないことを示すために，すべての状態が初期状態になりうるとする．このため，初期状態を特に指定せずに検査する．これは例 3 と同じ性質である．評価に用いたネットワークはフルメッシュの構成であるため，この性質は成り立たず，反例が出力される．検査時間を**図 9** に示す．ネットワークの構成によって検査時間にばらつきがあるため，異なる 10 個のネットワークについて時間を測定し，その値をプロットした．この場合では，前向き探索を行う場合と行わない場合では検査時間がほとんど同じである．これは，すべての状態が初期状態になりうるため，前向き探索によって状態を減らすことができないためであると考えられる．検査には最大で約 30 分ほどの時間を要するが，現実的に利用可能な範囲であると考えられる．また，実際に利用するネットワークを構成する際には，ツリー状を基本としてネットワークを構成することが多い．すなわち，測定に用いたネットワークは通常用いられるネットワークよりも検査にはるかに多くの時間が必要な例である．ツリー状のネットワークで行った同様の実験では，ノード数が約 2,800 台の場合でも約 15 分で計算が完了する．これは約 470 台の 8 ポートスイッチを 4 段のツリー状に接続し，その下に残りのノードを端末として接続したネットワークである．

6. おわりに

本稿では，ネットワーク機器の設定の正しさ，特に Ethernet LAN におけるタグ VLAN の設定の正しさをモデル検査を用いて検査する方法を提案し，モデルの構成方法，検査すべき性質の記述方法，検査の実行および性能評価について述べた．これにより，提案方法を用いてネットワークの様々な性質を記述・検証可能であること，検査結果に含まれる反例を原因の特定に利用できること，および，検査を現実的に利用可能な時間で行えることを示した．

前述（1 章）したように，本研究と関連する研究として，Guttman らによるものがある．彼らはファイアウォールのパケットフィルタの設定および IPsec ゲートウェイの暗号化の設定の正しさを検査する方法を提案した．問題設定として，ファイアウォールあるいは IPsec ゲートウェイがネットワーク上に複数存在し，そのそれぞれの設定の組合せにより要求仕様（セキュリティ）が実現される場合を考えている．これは，複数の Ethernet 機器の VLAN 設定の組合せによりパケットの適切な転送が実現されるという本研究の問題設定に近い．対象とするネットワークが異なること以外の相違点として，彼らが設定の検査だけでなく，設定の自動生成を行うアルゴリズムも同時に提案したことと，本研究ではモデル検査を用いたことがあげられる．設定の自動生成については，タグ VLAN の場合にもある程度可能であると考えられるが，実際のネットワークでは，トラフィックの集中の回避などを考慮するため，これを表現できるようモデルを拡張する必要がある．LAN に求められる性質は，LAN の利用目的・利用形態によって様々であるため，ファイアウォールや IPsec ゲートウェイの場合よりも幅広い性質が検査できる必要がある．本研究ではこれを，論理式によって

幅広い性質を検査できるモデル検査の利点を活かして実現している．また，本研究を応用し，彼らが対象としたネットワークについてもモデル検査により検証することが可能であると考えている．

一般にモデル検査を用いてシステムを検査する場合，モデルや検査する論理式の記述に試行錯誤が必要である[14),15)]．本研究の方法は検査の対象が VLAN を用いたネットワークであるため，モデルを自動的に構成することができ，モデルの構成に必要な情報もある程度自動的に収集できる（2 章）．検査する論理式については，個別のネットワークの独自の性質を記述する場合には論理式に関する知識がある程度要求され，試行錯誤が必要である．しかし，パケットの到達性や漏洩などのどのネットワークでも共通して求められるような性質について論理式のひな形を与えることで，ある程度楽に記述することが可能である．

謝辞 日頃よりご助言をいただいている NTT コミュニケーション科学基礎研究所の白柳潔グループリーダ（現東海大学教授），塚田恭章氏，真野健氏，河辺義信氏に感謝いたします．査読者の方々からは数多くの有益なコメントをいただきました．深く感謝いたします．

参 考 文 献

1) IEEE: IEEE Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks (2003). IEEE Std 802.1Q-2003.
2) Clarke, E.M., Grumberg, O. and Peled, D.: *Model Checking*, MIT Press (2000).
3) Cimatti, A., Clarke, E., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R. and Tacchella, A.: NuSMV Version 2: An OpenSource Tool for Symbolic Model Checking, *Proc. Computer Aided Verification, 14th International Conference, CAV 2002*, Copenhagen, Denmark, July 27-31, 2002, Lecture Notes in Computer Science, Vol.2404, pp.359–364, Springer (2002).
4) McMillan, K.L.: *Symbolic Model Checking: An Approach to the State Explosion Problem*, Kluwer Academic Press (1993).
5) Ben-Ari, M., Pnueli, A. and Manna, Z.: The Temporal Logic of Branching Time, *Acta Informatica*, Vol.20, pp.207–226 (1983).
6) Clarke, E.M. and Emerson, E.A.: Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic, *Logic of Programs*, Kozen, D. (Ed.), Lecture Notes in Computer Science, Vol.131, pp.52–71, Springer (1981).
7) Emerson, E.A. and Clarke, E.M.: Characterizing Correctness Properties of Parallel Programs Using Fixpoints, *ICALP*, de Bakker, J.W. and van Leeuwen, J. (Eds.), Lecture Notes in Computer Science, Vol.85, pp.169–181, Springer (1980).
8) Pnueli, A.: A temporal logic of concurrent programs, *Theoretical Computer Science*, Vol.13, pp.45–60 (1981).
9) Burch, J.R., Clarke, E.M., McMillan, K.L., Dill, D.L. and Hwang, L.J.: Symbolic Model Checking: 10^{20} States and Beyond, *Inf. Comput.*, Vol.98, No.2, pp.142–170 (1992).
10) Clarke, E.M., Grumberg, O. and Hamaguchi, K.: Another Look at LTL Model Checking., *Formal Methods in System Design*, Vol.10, No.1, pp.47–71 (1997).
11) Guttman, J.D.: Filtering Postures: Local Enforcement for Global Policies, *Proc. 1997 IEEE Symposium on Security and Privacy*, pp.120–129, IEEE Computer Society (1997).
12) Guttman, J.D., Herzog, A.L. and Thayer, F.J.: Authentication and Confidentiality via IPSEC, *Proc. Computer Security — ESORICS 2000, 6th European Symposium on Research in Computer Security, Toulouse, France, October 4-6, 2000*, Lecture Notes in Computer Science, Vol.1895, pp.255–272, Springer (2000).
13) Clarke, E.M., Emerson, E.A. and Sistla, A.P.: Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications., *ACM Trans. Programming Languages and Systems*, Vol.8, No.2, pp.244–263 (1986).
14) 中島 震，玉井哲雄：EJB コンポーネントアーキテクチャの SPIN による振舞い解析，コンピュータソフトウェア，Vol.19, No.2, pp.2–18 (2002).
15) 水口大知，渡邊 宏：組み込みソフトウェア開発におけるモデル検査の適用事例，コンピュータソフトウェア，Vol.22, No.1, pp.77–90 (2005).

(平成 17 年 10 月 28 日受付)
(平成 18 年 4 月 4 日採録)

櫻田 英樹（正会員）
昭和 49 年生．平成 11 年京都大学大学院工学研究科情報工学専攻修士課程修了．同年日本電信電話（株）入社．通信システムの形式的検証技術の研究に従事．