

通信の仕組みを理解するためのステップ実行および レジューム可能なネットワークシミュレータの実装

蔵永 武将¹ 立岩 佑一郎¹ 金 鎔煥¹ 片山 喜章¹ 長谷川 皓一²

概要: 様々なものやことがネットワークとつながり、通信技術はますます重要なものとなっている。規模も拡大しているため、新たな技術者の養成も必要である。技術者となるためにはネットワークの学習が必要であり、従来の学習方法は書籍での座学と、実機や仮想環境を使った演習であった。しかし、「通信の実施において、パケット送受信の際に、デバイスの動作がどのような条件で実現されるか、その動作がどのような設定値を参照しているか、などの通信の仕組み」を学習するにあたって問題がある。それは、学習者が想定するネットワークや設定で行うことができないことがある、デバイスの動作を1つ1つ確認することが困難であるなどの問題で、理解が不十分な学生が出てきてしまうことである。このため、ステップ実行およびレジューム可能なネットワークシミュレータを設計と実装を行った。ステップ実行により、デバイスの動作一つ一つを、順を追って確認することができる。レジューム機能により、デバイスやパケットのパラメータを変更して、任意の時点から再開することで、動作の違いを効率よく比較できる。評価として、シミュレータの特性評価を行い、学習に使用可能かどうかを検討し、初学者向けの学習であれば使用可能な性能であることがわかった。

1. はじめに

1.1 研究背景

情報化社会となっている今日では、ネットワークの技術を持った人々の養成が必要である。しかしネットワークの技術者となるためには、その基礎となっている通信についての知識が必要となる。通信がどのような仕組みで成立しているのかを知るということは、デバイスがパケットを送受信する際に、どのような動作が行われているか、その動作はデバイス内のどの設定値と、パケットのフィールド値を対象としたものかなどを知ることである。通信の仕組みを十分に理解できていない人々が技術者となってしまうと、表面上通信がうまくいくネットワークや通信アプリが完成する。しかし、通信の仕組みの理解が不十分なため、潜在的なバグを含むことがある。従来は通信の仕組みを学ぶために、講義での学習や、実機を使った学習が行われている。

1.2 問題点

従来の学習方法では、以下の学習を行うことができない。

- (A) 構築ネットワークで、学習者が送信したいパケットの内容と送信タイミングで、通信を実行する。
- (B) 学習者が、パケット配送の様子、受信後の機器の内部状態 (arp テーブル, MAC アドレステーブルなど) の変化を1つ1つ順に確認する。
- (C) 異なる入力での結果の違いを確認する。

1.3 ネットワークシミュレータの提案

問題点を解決するために、我々はこれまでに以下の特徴を持つネットワークシミュレータを提案してきた [1]。

特徴 1 ネットワーク構成、送信したいパケットの内容および送信タイミングを入力とすることから、問題点 (A) を解決する。

特徴 2 パケットの送信処理、受信処理、パケットの伝達をステップに分けて実行する。ステップ毎にシミュレータ内のホストの IP アドレスや送信中のパケットといった内部変数やデバイスの動作内容をファイル出力する (保存したものをログファイルと呼ぶ) ことから、問題点 (B) を解決する。

特徴 3 ログファイルを学習者が編集可能で、編集後のログファイルからシミュレーションを再開できることから、問題点 (C) を解決する。

本稿では、提案したシミュレータの設計に基づいて、プロ

¹ 名古屋工業大学
Nagoya Institute of Technology

² 名古屋大学
Nagoya University

トタイプとしてネットワークシミュレータを実装する。

2. 通信の仕組みを理解するための学習

通信の仕組みの理解を促進するには、学習者が想定したネットワークと送信パケットで、以下の項目を確認可能とすることが有効であると考えた。

- ネットワークのトポロジー
- 配送中のパケットの経路
- ホストやスイッチでのパケットに対する処理
- パケットの処理の際に参照したフィールド値、デバイスの設定値の関係性

学習者はネットワーク内でパケットがどこを通ったか、デバイス内でどのような処理がされたかを確認する。また、デバイスの内部処理は1つ1つ順に追って確認し、処理と参照した値との関係性を確認することも可能とすることで、細かな部分まで学習が可能になると考えられる。

3. 関連研究

本研究に関連する研究として、通信の仕組みの学習に関連するシステム、関連する学習方法、ネットワークシミュレータに関するシステムなどがある。

講義では、書籍や指導者が作った講義資料などを読み、ある通信を行う際にパケットはどのような構成となっているか、ホスト内でどのような処理が行われるかといった通信の仕組みを学ぶ。図や文での説明がされている。この学習方法では、問題点(A),(B),(C)を行うことができない。これは紙面の都合上発生する問題であり、ネットワークや送信パケット、動作などすべてを提示することが困難である。

実機では、学習者がコマンドを打つことでデバイスの設定や通信を行う。通信の結果はコンソールに出力される結果、ログや可視化ツールなど様々な方法で確認し、どのようなデータが送信されて処理されたかを学習する。この学習方法では、問題点(B),(C)を行うことができない。問題点(A)は仮想環境を使うなどして、自由にネットワークを構築できる。しかし、デバイスの動作を1つ1つ順に表示することは困難であり、特定のネットワークの状態から異なる入力で演習を行うことも困難である。

立岩らは、仮想的に構築したネットワークの動作を、TCP/IP理論と対応付けて可視化するシステムを開発し、学習効果の検証を行った[2]。LAN構築技術と、TCP/IP理論はどちらも重要であり、それぞれ単独で学習する手段は数多くある、しかしこの2つを結び付けて学習することは難しい。そこで仮想的なネットワークをUMLを用いて実現し、UMLのデバッグ機能とパケットダンプ機能を用いて得たデータを基に可視化を行った。この研究では問題点(B)を解決できる。しかし仮想環境を使っていることから、問題点(A),(C)が解決できない場合がある。(A)は

通信が人間の入力に比べて早すぎるため、任意のタイミングでコマンド入力をできない可能性がある。(C)は通信途中から異なる入力を実行したい場合に、同じ状況を作り出すことが難しいことがある。

荒井らは、実際のネットワークからデータを取得し、TCPのデータ構造、通信手順、制御方式などを可視化表示し、学習支援を行うシステムを開発した[3]。TCP/IPの従来の学習方法には問題点があった。座学ではさまざまな通信パターンの提示が難しく、イメージもわきづらい。ツールを使った学習は、ツールを使いこなすことが難しい、または学習用としてインタフェースが優れていないことがある。そこで実際に通信を行って発生したパケットをキャプチャし、そのパケットからどのような通信が行われたかを学習できるシステムとなっている。この研究では問題点(B)を解決可能である。しかし問題点(A)は学習用のTCP通信のデータを、実際の通信で発生したパケットを基にしているため、学習者が任意の通信を実行することが難しい。(C)は通信途中から異なる入力を実行したい場合に、同じ状況を作り出すことが難しいことから解決できないことがある。

鈴木は、通信の仕組みを理解するためのロールプレイ演習の開発と実践を行った[4]。ネットワーク機器を利用しない学習法で通信の仕組みを理解することを目標としている。演習内容は、学習者自身がルータ、スイッチ、PCといったネットワーク機器を演じ、データのやり取りを体験的に理解するというものである。この研究では問題点(A),(B)を解決可能である。しかし問題点(C)は実際に学習者が入力や出力、デバイスの動作を表現することから、1つの実行例を行うのに時間がかかってしまう。そのため異なる実行例を数多く行うことが困難である。

Muhammad Wannousらは仮想化および仮想ネットワークコンピューティングテクノロジーを実装した仮想ウェブベースの環境を開発した[5]。学習者は実際の状況の場合のように複数のデバイスで作業する練習をすることが出来る。コンピュータネットワークコースの一部として遠隔学習者のグループにケーススタディの練習を導入することによって評価実験が行われた。演習前後の参加者の答えをレベルテストと比較することによって、参加者の進歩を評価した。結果は、演習によってより良い結果となった。この研究では学習方法(A),(B),(C)ができない。コマンドなどで通信を行うが、通信の速さより人間の入力が遅いことから、任意の通信の途中でコマンドを打つなどできないことがある。そのため(A)ができない。(B)はデバイスの動作までトレースすることが困難であるためできない。(C)は送信元のIPアドレスを変更したパケットの送信などができない。ツールが存在する場合は可能であるが、準備が困難である場合がある。

精廬らは、教育向けネットワークシミュレータを開発し

た [6]。実際のネットワーク機器を用いてネットワーク構築を行うことが困難である場合があるため、ネットワークシミュレータを開発した。GUIを用いてネットワークのデバイスを配置、接続を行い、PCにはIPアドレスや、ゲートウェイといった設定を行うことで、論理的なネットワークを作成する。この研究では問題点 (A), (C) を解決可能である。しかし問題点 (B) はパケットの配送経路が提示されるが、デバイス内でどのような処理が行われた結果ということがわからないことから解決できない。

ネットワークシミュレータである OPNET[7] は入力されたネットワークポロジと送信シナリオに従ってシミュレーションを行うことができる。アプリケーション層から物理層までの振る舞いを確認することができ、シミュレーション結果をグラフィカルに提示する。また、ネットワークポロジや送信シナリオは GUI を用いて簡単に変更することができる。このシステムでは問題点 (A) を解決可能である。しかし問題点 (B) において、動作内容は示されているものの、パケットのどの値とデバイスのどの値が参照されたかは提示されていない。また、(C) はシミュレーション途中の状態から入力を変更して実行できないため、解決できない。

ネットワークシミュレータである ns2[8] はシステム利用者が構築したネットワークに対し、パケットを送受信することができる。パケット単位でシミュレーション結果を知ることができ、シミュレーション結果を可視化ツールで見ることが可能である。パケットの受信、キューに入る、キューから出る、破棄といったイベントから構成され、可視化用のログもこのイベントを基に作成される。このシステムでは問題点 (A) を解決可能である。しかし問題点 (B) はイベントがパケットの受信、キューに入る、キューから出る、破棄からなっているため、デバイス内でどのような処理がなされたかを見ることができない。(C) はシミュレーション途中の状態から入力を変更して実行できないため、できない。

ネットワークシミュレータである packet-tracer[9] は利用者が GUI を使用し、ネットワークを構築することが可能である。また、設定をコマンドによる入力、GUI を使った入力が可能で、任意のタイミングでパケットを送信することが可能である。パケットの配送経路、パケット内容を見ることが可能であり、受信、破棄を判別することもできる。その際、OSI 参照モデルのどのレイヤで何が行われたかということを見ることが可能である。このシステムでは問題点 (A) が解決可能で、(B) も一部解決可能である。(B) は、行われた動作を 1 つ 1 つ確認可能だが、動作の単位があらかじめ決まっているため、さらに細かい単位で見せることはできない。(C) はシミュレーション途中の状態から入力を変更して実行できないため、解決できない。

4. シミュレータの実現法

4.1 シミュレータ概要

シミュレータの構成を図 1 に示す。デバイス制御機能が、ネットワーク構成からネットワークを構築すること、シナリオ読み取り機能が、送信シナリオから送信パケットを決定することで、特徴 1 を実現する。デバイス制御機能で各デバイスにステップ実行を行うことで、特徴 2 を実現する。ログファイルをデバイス制御機能の入力とし、シミュレーションを再開可能とすることで、特徴 3 を実現する。シミュレータの動作を以下に示す。

- (1) シミュレータが設定ファイルを読み込む
- (2) 設定ファイルをもとに、ネットワークの構築、シナリオの読み込み、開始ステップ番号の決定を行う
- (3) ステップ実行を行う

4.2 設定ファイル

設定ファイルは、ネットワーク構成 N (表 1)、送信シナリオ Sc (表 4)、オブジェクトの状態 St からなる。ホストの状態 $hdata \in HD$ 、スイッチの状態 $sdata \in SD$ 、ケーブルの状態 $cdata \in CD$ により、 $St=(HD,SD,CD)$ となる。ここで本稿では、ホスト、スイッチをデバイスと呼び、デバイスとケーブルをオブジェクトと呼ぶ。

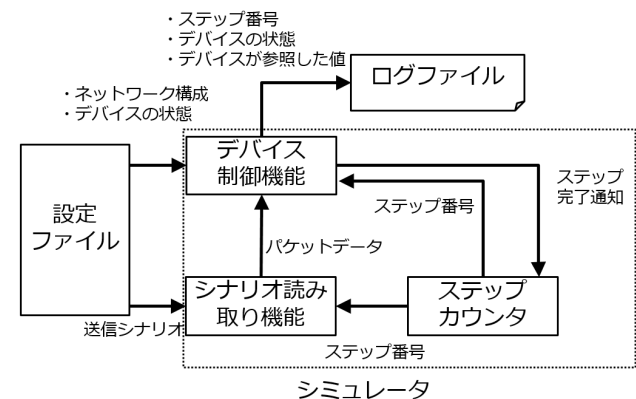


図 1 シミュレータ構成図

表 1 ネットワーク構成 $N=(H,S,C)$

要素名	概要
H	ホスト情報 h (表 2) の集合
S	スイッチ識別子の集合
C	ケーブル情報 c (表 3) の集合

表 2 ホスト情報 $h=(hid,ip,mac,gw,sb)$

要素名	概要
hid	ホストの識別子
ip	ホストの IP アドレス
mac	ホストの MAC アドレス
gw	ホストのデフォルトゲートウェイ
sb	ホストのサブネットマスク

表 3 ケーブル情報 $c=(id1,po1,id2,po2)$

要素名	概要
id1	接続先のデバイス識別子
po1	接続する id1 のデバイスのポート識別子
id2	接続先のデバイス識別子
po2	接続する id2 のデバイスのポート識別子

表 4 送信シナリオ $Sc=(Pa,t)$

要素名	概要
Pa	送信パケット情報 pa(表 5) の集合
t	シミュレーション終了ステップ番号

表 5 送信パケット情報 $pa=(s,hid,pr,F)$

要素名	概要
s	パケットを送信するステップ番号
hid	パケットの送信元ホストの識別子
pr	パケットのプロトコル
F	プロトコルに対応したパケットのフィールド値の系列

4.3 ステップ実行

提案シミュレータでは、ネットワークのシミュレーションをステップ単位で実行する。この理由は、学習させたい内容の単位で TCP/IP プロトコルスタックのプログラム(以降通信プログラムと呼ぶ)を分割することで、通信を細かく順に学習できるからである。また、シミュレーション結果から学習するとともに、途中で一部値を変更することによって、その先シミュレーション結果がどのように変わるかを学習可能とすることも目標である。そのためには、シミュレーション結果のどのタイミングからでも値を変更可能である必要があった。そこで、すべてのデバイスが1つのステップカウンタによって同時にステップ実行を行う設計となっている。

各デバイスがそれぞれステップ実行を行うには、各デバイスが通信プログラム内のある一部(ステップ)を単独で実行できなければならない。そこで、各デバイスをオブジェクト指向のクラスとして作成した。各クラスに通信プログラムを基に作成したステップ実行用の関数を持たせ、クラスごとにステップ関数を呼び出す設計とする。そのようにすることで、デバイスごとに別々のステップ実行が可能となる。しかし、今回ステップ実行を行うために使用する通信プログラムは C 言語で作成されているため、関数内の一部のみを実行可能とするために変更しなければならない。そのため、ステップ単位で実行するために以下の手順で通信プログラムを書き換えた。

手順 1 通信プログラム内の関数をすべて展開し、デバイスのメンバ関数化する

手順 2 ローカル変数をデバイスのメンバ変数とする

手順 3 ステップを管理する変数 (ID) を準備し、デバイスのメンバ変数化する

手順 4 ID を使った条件分岐により、通信プログラム内の

文をステップ単位に分割する

手順 5 各ステップに遷移先の ID を追加する

この手順で $ID=n$ のときにステップ n のみを実行する関数を作ることができる。遷移先の ID は元の通信プログラムに基づいて決定する。ステップ分割の例を図 2 に示す。

4.4 デバイス制御機能

デバイス制御機能は、ネットワークの構築と、ネットワークのシミュレーションをステップ実行で行う。デバイス制御機能がネットワークを構築する際の詳細手順を以下に示す。

- (1) N に基づいて、それぞれ $|H|$ 個のホスト、 $|S|$ 個のスイッチ、 $2 \times |C|$ 個のケーブルのオブジェクトを作成する。ケーブルに関しては、一方向のデータの流れとなるので、1本のケーブルにつき2個のオブジェクトを作成しなければならない。作成したデバイスは配列で管理される。
 - (2) それぞれのホストに対し、 H の情報をもとに、hid を設定する。
 - (3) それぞれのスイッチに対し、 S の情報をもとに、スイッチの識別子を設定する。
 - (4) C の情報をもとに、接続関係を設定する。
 - (5) ケーブルのオブジェクト $c1, c2$ を準備し、 $c1$ のケーブルの送信先ポートを $po1$ とし、 $c2$ のケーブルの送信先ポートを $po2$ とする。
 - (6) $po2$ の送信先ケーブルを $c1$ とし、 $po1$ の送信先ケーブルを $c2$ とする。
 - (7) ホストのポートに ip, mac, gw, sb を設定する。
 - (8) St をもとに各デバイスのメンバ変数を設定する。
- これらが完了した後、以下の手順に沿ってシミュレーションを行う。

- (1) ステップカウンタが、現在のステップ番号 s をデバイス制御機能に通知する。
- (2) デバイス制御機能が、ホスト、スイッチ、ケーブルの順に s を通知し、通知を受け取った各デバイスは、1

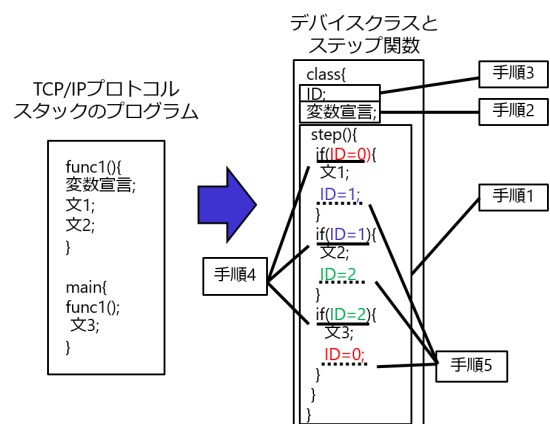


図 2 ステップ分割

ステップ分動作する。

- (3) デバイス制御機能が、各デバイスの状態を収集し、ログファイルへ出力する。
- (4) デバイス制御機能が、ステップカウンタへステップ完了通知を送る。
- (5) ステップカウンタは、現在のステップ番号を1増加する。ステップ番号が、終了ステップ番号を超えていなければ、手順1に戻る。終了ステップ番号を超えていた場合はシミュレーションを終了する。

4.4.1 待機バッファ

デバイス制御機能は、ホスト、スイッチ、ケーブルの順にステップ実行を行う。この順序により、ホストが送信したパケットをケーブルが受信し、同じステップでケーブルでも処理するという問題が発生する。そこで、デバイスがケーブルへパケットを送信する際に、パケットをケーブルの待機バッファに格納する。ケーブルのステップ実行時は待機バッファのパケットをステップ実行の対象から外す。ステップ実行完了後にデバイス制御機能は待機バッファからパケットを出し、ケーブルへ渡すことで、パケットの二重処理を防ぐ。

4.5 レジューム方法

一度シミュレーションを行った結果に対して、任意のステップ番号の時点から再度シミュレーションを実行することで可能としている。この方法を実現するために、すべてのデバイスが終了したときにログをとる。このログはステップ番号、シミュレータ内の各デバイスのメンバ変数からなる。このログを各デバイスのメンバ変数に与えることでログをとった時のデバイスと同じ状態にすることが可能となる。ログのステップ番号をステップカウンタの現在のステップ番号とすることで、ログをとったステップ番号からシミュレーションを再開することが可能となる。一度シミュレータによって出力されたログファイルの一部を編集し、再びシミュレーションを行うことも可能である。その結果、シミュレーションの途中で、あるデバイスの動作の際に、デバイスのメンバ変数を変更したらどのような影響があるかという検証も可能となる。

5. プロトタイプシステム

設計したシミュレータが正しく動作することの検証を目的として作成した。システム構成図を図3に示す。シミュレータは1ステップごとのログを出力し、ログ可視化ツールは各ステップでのパケット内容や配送の様子、デバイスの処理内容を確認できるという特徴を持つ。

5.1 シミュレータ

c++とQt4[10]を用いて実装した。ホスト内の受信処理については文献[11]の付録であるソースコードを参考に

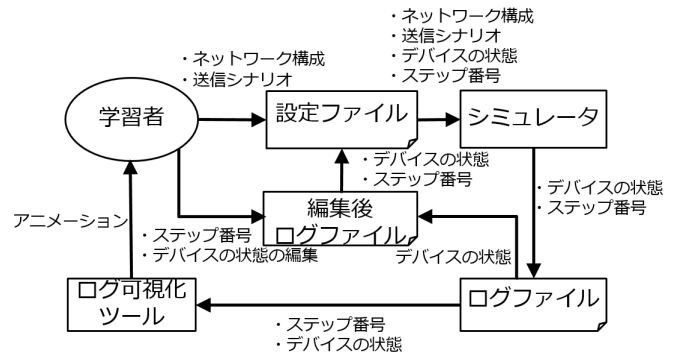


図3 システム構成図

して実装した。スイッチングハブについては文献[12]のumlswitchのソースコードを参考にして実装した。

5.2 設定ファイル

プロトタイプシステムでは、設定ファイルをネットワーク構成ファイル、送信シナリオファイル、ログファイルの3つのファイルに分割して管理する。

ネットワーク構成ファイルの例を図4に示す。ネットワーク構成ファイルを読み取ることにより、ネットワークを構成するように実装されている。1の部分では、構築するネットワークで使用する各デバイスの数が記述されている。左から順にホストの数、スイッチの数、ケーブルの数となっている。2の部分では、ホストの設定が記述されている。左から順にホストID、IPアドレス、Macアドレス、デフォルトゲートウェイ、サブネットマスクとなっている。3の部分では、スイッチの設定が記述されている。スイッチIDが記述されている。これはスイッチの数だけ設定しなければならない。4の部分では、ケーブルの設定が記述されている。左から順に接続先1のデバイスID、接続するポートID、接続先2のデバイスID、接続するポートとなっている。これはケーブルの数だけ設定しなければならない。

送信シナリオファイルの例を図5に示す。送信シナリオ

```

3,1,3, ①
hst1,192.168.0.1,12:34:56:78:90:12,192.168.0.254,255.255.255.0,
hst2,192.168.0.2,34:56:78:90:12:34,192.168.0.254,255.255.255.0, ②
hst3,192.168.0.3,56:78:90:12:34:56,192.168.0.254,255.255.255.0,
sw1, ③
hst1,0,sw1,0,
hst2,0,sw1,1,
hst3,0,sw1,2, ④

```

図4 ネットワーク構成ファイル

```

1,hst1,icmp,192.168.0.1,192.168.0.2, ①
30,hst1,icmp,192.168.0.1,192.168.0.2,
35,hst1,arp,192.168.0.1,12:34:56:78:90:12,192.168.0.3,56:78:90:12:34:56,
75,hst1,icmp,192.168.0.1,192.168.0.3, ②
100

```

図5 送信シナリオファイル

ファイルを読み取ることにより、送信パケットと終了ステップ番号を決定するように実装されている。1の部分では、送信パケットの情報が記述されている。左から順に送信タイミングとなるステップ番号、送信元ホストのID、パケットのプロトコル、プロトコルに応じたフィールド値の系列となっている。2の部分では、シミュレーションの終了ステップ番号が記述されている。

5.3 ネットワークとステップの例

図6のネットワークでホスト1からarpパケットを送信し、replyパケットをホスト1が処理し終わるまでを30ステップ、ホスト1がホスト2へicmpパケットを送信し、応答パケットの処理をし終わるまで21ステップかかるようなステップ分割となっている。ケーブルは各3ステップで配送できるようになっている。

5.4 ログ可視化ツール

ログ可視化ツールは、シミュレーション結果を学習者に見せるための機能であるため、2章で示した内容を表示する。ログ可視化ツールを図7に示す。1に示されているデータは現在のステップ番号である。2に示されているデータは作成したネットワークである。ホスト、スイッチ、ケーブルがどのように接続されているかを知ることができる。ホストとスイッチは画像が貼り付けられたボタンであり、ボタンの中心にはデバイス名が書かれている。押すと、デバイスの情報を見ることができる。3に示されているのは閲覧しているログのステップ番号を変更するためのボタン群である。ボタンの説明を以下で行う。

start 「start」ボタンを押すと、自動的に閲覧しているログのステップ番号が増加していく。0.3秒ごとに次のステップ番号へと変化していくため、パケット配送の流れを見るために役立つ。「start」ボタンが押されている間は「next」、「prev」のボタンを押すことができない。送信シナリオに記述した終了ステップ番号に到

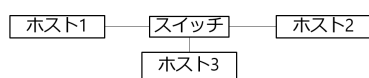


図6 ネットワークの例

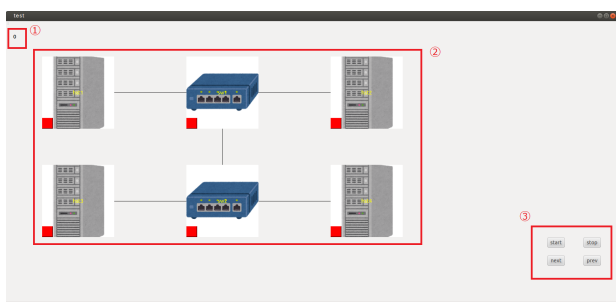


図7 ログ可視化ツール

達すると、自動的にステップ番号の増加を終了する。
stop 「stop」ボタンを押すと、ステップ番号が自動的に増加している状態を止めることができる。「start」ボタンが押されていない状態で押すと何も起こらない。

next 「next」ボタンを押すと、現在のステップ番号を1増加させる。このボタンはあるステップ番号でのデバイスの動作や、パケットの中身を確認するために役立つ。ほかのボタンを押さない限りは現在のステップ番号が変わることがないため、そのステップで発生した動作を分析することができる。現在のステップ番号が送信シナリオに記述した終了ステップ番号と同じ場合、「next」ボタンを押しても何も起こらない。

prev 「prev」ボタンを押すと、現在のステップ番号を1減少させる。このボタンも「next」ボタンと同じ用途である。現在のステップ番号が0であった場合、「prev」ボタンを押しても何も起こらない。

配送中のパケットについて図8に示す。デバイスからデバイスへとパケットが配送される様子は、図中の直線に隣接する正方形によって表現される。黒い直線がケーブルである。パケットはケーブル内を双方向に流れるため、左から右のデバイスに流れるパケットをケーブルの上、右から左のデバイスに流れるパケットをケーブルの下に表示する。図中のパケットはhst1からsw1へ配送されるパケットである。ケーブルが縦に配置されていることもあり、上から下に流れるパケットはケーブルの右、下から上へ流れるパケットはケーブルの左にパケットが表示される。ケーブル内では例えば3ステップかけて通過するのであれば、ケーブルの両端を含めて3段階で到達する。図中では3段階中の2段階目での図である。正方形を押すことで、流れているパケットの情報を見ることができる。

デバイスの処理と、処理の際に参照したフィールド値、デバイスの設定値の関係性の例を図9に示す。デバイスのボタンには1のボタンと2のボタンがある。1のボタンは現在のデバイスの処理を表示するボタン、2のボタンは現在処理中のパケットが存在している場合のみ出現し、押すと現在処理中のパケットを表示する。図9ではステップ番号15において、hst2のデバイスの動作ボタンによるダイアログと、処理中のパケットのダイアログを同時に開いている。処理のダイアログを見ると、arpテーブルに登録す

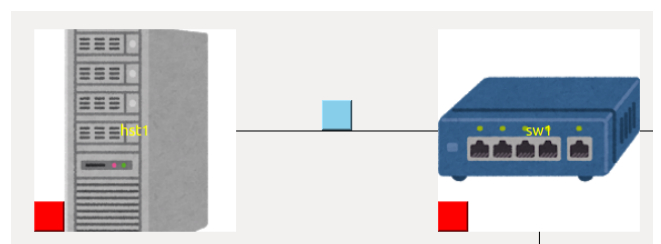


図8 配送中のパケット

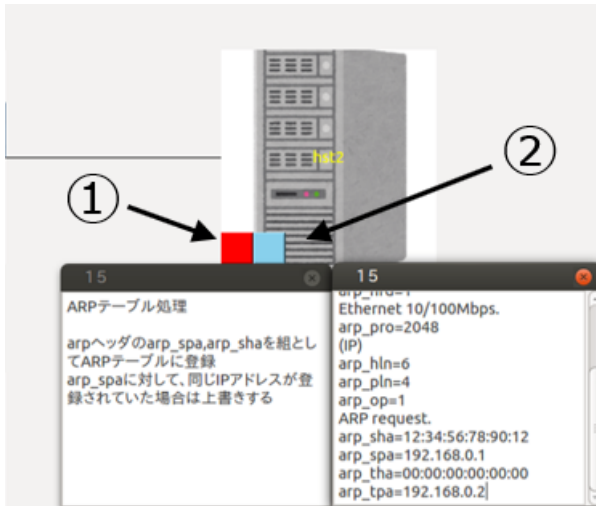


図 9 可視化システムの使用例

るため、arp_spa と arp_sha が参照されていることがわかる。そのため、16 ステップでは arp テーブルに arp_spa と arp_sha を組としたデータが追加される。

6. 提案シミュレータの評価

本章では提案シミュレータの性能が、学習に使うにあたって現実的かという評価をプロトタイプシステムを用いて行う。

6.1 評価実験の目的

提案シミュレータは通信の仕組みの理解を促進させることを目的としており、座学の後に提案シミュレータを使用することを想定している。したがって教育機関や学生が持っている PC で動作しなければならない。また、シミュレーションに多大な時間がかかってしまうことや、ログファイルのサイズが大きくなってしまふことが考えられる。そのためシミュレーションにかかる時間、シミュレーション時の物理メモリ使用量、シミュレーション時に出力されるログファイルの大きさが、学習を行うにあたって現実的かどうかを評価する。

6.2 実行環境

今回は以下の性能の実機を用いて評価を行なった。

使用 OS Windows10 64 ビット

メモリ (RAM) 8.00GB

プロセッサ Intel(R) Core(TM) i5-4690 CPU @ 3.50GHz 3.50GHz

仮想マシンソフトウェア VMware Workstation 12 Player[13]

仮想環境 OS Ubuntu 12.04.2[14] メモリ割り当て 4.00GB 仮想ディスク 20.00GB

windows 上で仮想マシンを動かしている。仮想 OS の ubuntu は linux ベースであり、ubuntu 上で提案シミュ

レータを実装し、評価を行った。

6.3 シミュレータの性能評価

評価はデバイス数に対する評価、ステップ数に対する評価、シナリオに対する評価を行った。結果を表 6,7,8 に示す。ログサイズは、1 ステップ分のものである。

6.4 考察

表 6 では、終了ステップ番号を 100 に固定し、送信シナリオ内の送信パケット数を 0 として、各デバイスに対して調べたものである。ケーブルは接続先が存在しないと設置することができないため、ケーブルの実行時間は、ホストやスイッチに接続して、ホストやスイッチの実行時間を引いて求める。例えばケーブル 3 本の場合は、 $1.817 - 0.608 - 0.099 = 1.110$ 秒となっている。ホスト、スイッチ、ケーブルのステップ実行は別々に行われているため、それぞれが複数存在していた場合は、ホストの実行時間、スイッチの実行時間、ケーブルの実行時間を足したものが実行時間になると考えられる。

使用した物理メモリは、デバイスが増えていくとともに増加した。しかしどのデバイスも 1 つ増えたところで数十 KB しか増加していないため、実行環境の 4GB と比較すると問題なく動作することが確認できた。

1 ステップあたりのログサイズも実行時間と同様にケーブルのみで測定できないため、ホストとケーブルのログ

表 6 デバイスの数に対する評価 (ステップ数 100, シナリオ空)

ホスト数	スイッチ数	ケーブル数	ログ (KB)	実行時間 (s)	RSS(MB)
0	0	0	0.03	0.028	9.6
1	0	0	12.8	0.233	10.2
2	0	0	25.6	0.415	10.2
3	0	0	38.4	0.608	10.3
0	1	0	3.5	0.099	10.0
0	2	0	7.0	0.150	10.0
0	3	0	10.5	0.205	10.1
2	0	1	49.9	0.762	10.3
2	1	2	77.8	1.268	10.5
3	1	3	115.0	1.817	10.5

表 7 ステップ数に対する評価 (ホスト 3, スイッチ 1, ケーブル 3, シナリオ空)

ステップ	実行時間 (s)	RSS(MB)
100	1.817	10.5
200	3.724	10.5
300	5.651	10.6
1000	20.384	10.6

表 8 送信パケットに対する評価 (ホスト 3, スイッチ 1, ケーブル 3, ステップ数 100)

送信パケット数	実行時間 (s)	RSS(MB)	最大ログサイズ (KB)
0	1.817	10.5	115.0
1	1.844	10.9	138.6
2	1.826	10.9	146.8
3	1.856	10.9	150.8

サイズを引いて求める。例えばケーブルが3本の場合は $115-38.4-3.5=73.1\text{KB}$ となる。

初学者向けの演習ではホスト数10、スイッチ数2で、ケーブルが11本のネットワークで行うことができる内容を考える。評価結果から、このネットワークにおいて、100ステップで実行すると、実行時間は5.75秒程度になると考えられる。これは演習の妨げになる長さではないと考えられる。使用物理メモリは、11~12MB程度と考えられるが、割り当てられた4.00GBと比べれば問題なく動作すると考えられる。1ステップ分のログサイズは、約400KBとすることが想定され、100ステップで約40MBと考えられる。割り当てられたハードディスク容量が20GBなので、複数保存することも可能なログサイズである。

表7では、設置するデバイスの数を固定し、送信シナリオの送信パケット数も0とした状態で、ステップ数だけを変更して実行時間を測定した。実行時間は、ステップ数の増加に比例して増加した。使用物理メモリはステップ数の増加でわずかに増加した。arpパケット、icmpパケットは30ステップあれば送信から受信が完了するため、1000ステップあれば十分にシナリオを設定することが可能であると考えられる。1000ステップでの実行時間は約1.9秒で、演習には影響がないと考えられる。使用物理メモリも4GBと比較して影響がないと考えられる。

表8では、設置するデバイスの数、終了ステップ番号を固定し、送信パケット (icmpパケット) の数にて評価を行った。結果としては、送信パケット数が増えれば1ステップでの最大ログサイズも大きくなるという結果となった。ログサイズが大きくなる原因は、ケーブルを流れるパケット、デバイスの受信バッファや処理中のパケットの中身と数にあると考えられる。例として、ケーブル内にarpパケットが1つ存在するとき、ログサイズが1.7KB大きくなった。これはパケットのデータがすべてログに保存されているためである。そのため、送信パケット数が少なくても、スイッチにおけるブロードキャストでパケットがコピーされるとログサイズが大きくなる。ログサイズの増え方は、デバイスやケーブルの持つパケットの数と1パケットあたりのログサイズの積だけ増える。仮にケーブルのすべてがarpパケットで埋め尽くされても、100KB程度の増加となる。ただし、ケーブルの配送にかかるステップ数が現在は3であるが、これが大きくなると、最大ログサイズは大きくなる可能性がある。しかし、20GBと比較すると小さいため、問題ないと考えられる。送信パケット数の増加により、実行時間は少し増加したが、演習に影響が出るほどではない。物理メモリの使用量は、送信パケットがあった場合は増加したが、変化は4GBと比較すると小さかったため、影響がないと考えられる。

7. おわりに

通信の仕組みを理解するためのネットワークシミュレータを実装した。デバイスの動作単位で通信プログラムを分割し、分割したプログラム単位でシミュレーションを実行可能という特徴がある。また、任意のタイミングから再開可能であるという特徴を持つ。演習に使用可能か評価を行ったところ、実行時間、使用物理メモリ、ログサイズから使用可能であることがわかった。

今後の課題としては、提案シミュレータを使った学習効果の測定が挙げられる。また、ルータやファイアウォールの導入、ステップ割り当てを指導者が設定できるようにすることが挙げられる。

謝辞 本研究の一部は電気通信普及財団の助成による。

参考文献

- [1] 蔵永武将, 立岩佑一郎, 金鎔煥, 片山喜章, 長谷川皓一: 通信の仕組みを理解するためのステップ実行およびレジューム可能なネットワークシミュレータの設計, 信学技報, vol. 118, no.304, IN2018-52, pp. 1-6, 2018.
- [2] 立岩佑一郎, 安田孝美, 横井茂樹: 仮想環境ソフトウェアに基づくLAN構築技能とTCP/IP理論の関連付け学習のためのネットワーク動作可視化システムの開発, 情報処理学会論文誌, Vol.48, No.4, pp.1684-1694, 2007.
- [3] 荒井正之, 田村尚也, 渡辺博芳, 小木曾千秋, 武井恵雄: TCP/IPプロトコル学習ツールの開発と評価, 情報処理学会論文誌, Vol.44, No.12, pp.3242-3251, 2003.
- [4] 鈴木大助: 通信の仕組みを理解するためのロールプレイ演習の開発と実践, 研究報告コンピュータと教育 (CE), Vol.2017-CE-140No.10, pp.1-7, 2017.
- [5] M.Wannous, H.Nakano: NVLab, a Networking Virtual Web-Based Laboratory that Implements Virtualization and Virtual Network Computing Technologies, IEEE Transactions on Learning Technologies, Volume:3, Issue:2, pp.129-138, 2010.
- [6] 精廬幹人, 木村昌史: 教育向けネットワークシミュレータの開発, 情報処理学会 65 回全国大会講演論文集, 2D-2, pp.273-274, 2003.
- [7] OPNET: 入手先< http://www.johokobo.co.jp/opnet_rd/opnet_rd_index.html> (2019.2.20).
- [8] The Network Simulator-ns2: 入手先< <http://www.isi.edu/nsnam/ns/>> (2019.2.20).
- [9] packet-tracer: 入手先< <https://www.netacad.com/ja/courses/packet-tracer>> (2019.2.20).
- [10] Qt4: 入手先< <http://doc.qt.io/archives/qt-4.8/>> (2019.2.20).
- [11] 小俣光之: ソースコードで体感するネットワークの仕組み, 技術評論社, 2018.
- [12] User-mode Linux Kernel: 入手先< <http://user-mode-linux.sourceforge.net/>> (2019.2.20).
- [13] VMware Workstation Player: 入手先< <https://www.vmware.com/jp/products/workstation-player.html>> (2019.2.20).
- [14] Ubuntu: 入手先< <https://www.ubuntu.jp/>> (2019.2.20).