セキュリティの仕組み理解への演習の開発

代表研究者　　立岩　佑一郎　　名古屋工業大学大学院　工学研究科　助教
共同研究者　　長谷川　皓一　　名古屋大学　情報連携統括本部情報戦略室　助教

1 研究の背景

情報化社会の到来やIoTの時代と言われ，様々な物事がネットワークとつながっている．このような社会を支えているのはネットワークの構築やセキュリティの管理，通信アプリの開発といったスキルを持った人々であり，今後もこのような技術者を養成していく必要がある．

ネットワークの技術者となるには，基礎知識である通信の仕組みについて理解していなければならない．通信の仕組みとは，例えば通信の実施において，通信デバイスがどのような通信データを受け取ると，デバイス内でどのような動作行われるか．あるデバイスの動作が，デバイス内のどの設定値と，通信データのどのパラメータを参照し，どのような条件で起こるのかといったことなどである．

通信の仕組みを十分に理解しきれていない人々が技術者となってしまうと，通信データがうまく届かないネットワークを構築することがある.セキュリティの管理では,攻撃を防衛することができないことがある．通信アプリの作成の際は，バグを多く含むことがある．

通信の仕組みの従来の学習方法として，座学での学習が行われてきた．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通信データ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．しかし，この学習では理解が不十分な学習者がいる．

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通信データの内容と送信タイミングで，通信を実行する．

B) 学習者が,通信データ配送の様子,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．

このため，上記の問題点を解決するため，以下の特徴を持つシミュレータを提案する．

特徴1) ネットワーク構成，送信したい通信データの内容および送信タイミングを入力とする．

特徴 2) 通信データの送信処理，受信処理，通信データ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通信データ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)．

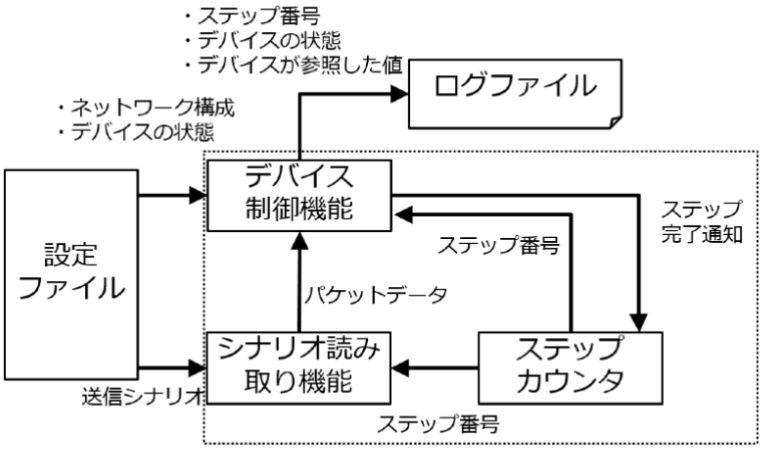
特徴3) ログファイルを学習者が編集可能で,編集後のログファイルからシミュレーションを再開できる．

特徴1によって，問題点(A)の学習を行うことができる．特徴2によって，問題点(B)の学習を行うことができる．特徴1,3によって，問題点(C)の学習を行うことができる．

2 シミュレータの実現法

2-1 概要

シミュレータの構成を下記の図に示す．シミュレータはデバイス制御機能，シナリオ読み取り機能，ステップカウンタから構成される．シミュレータの入力は設定ファイルで，出力はログファイルとなっている．



シミュレーション開始前の準備として，以下の動作を行う．

1. シミュレータに設定ファイルを与える．
2. デバイス制御機能が,ネットワーク構成情報N,デバイスの状態Stをもとにネットワークを構築する．
3. シナリオ読み取り機能が，ケーブル情報 c を読み込む．
4. ステップカウンタが，St をもとに現在のステップ番号を設定する．St が与えられていない場合は，現在のステップ番号を 0 とする．

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 のホスト数 x,スイッチ数 y,ケーブル数 z から，それぞれ x 個のホスト，y 個のスイッチ，2z 個のケーブルのインスタンスを作成する．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 個のインスタンス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，ホスト情報 H の情報をもとに，ホスト識別子 hid を設定する．
3. それぞれのスイッチに対し，スイッチ情報 S の情報をもとに，スイッチ識別子 sid を設定する．
4. ケーブル情報 C の情報をもとに，接続関係を設定する．接続先 id1 のデバイスに対し，接続ポート po1 という id のポートを作成する．また，接続先 id2 のデバイスに対し，接続ポート po2 という id のポートを作成する．
5. ケーブルのインスタンス 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 アドレス ip，MAC アドレス mac，ゲートウェイ gw，サブネットマスク sb を設定する．
8. St をもとに各デバイスの情報を入力する．St が指定されていない場合は何もしない．指定されている場合は，デバイス名と一致するデバイスを探し，デバイスの持つ各データを設定する．

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，シナリオ読み取り機能へ，現在のステップ番号 s を通知する．
2. シナリオ読み取り機能が，送信シナリオの中から，受け取った s と一致するデータを探す．あれば該当するデバイスへパケットデータを送信する．
3. ステップカウンタが，s をデバイス制御機能に通知する．
4. デバイス制御機能が，すべてのデバイスに s を通知し，通知を受け取った各デバイスは，1 ステップ分動作する．
5. 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．
6. 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．
7. 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，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．

2-2 設定ファイル

設定ファイルは，ネットワーク構成情報 N，送信シナリオ Sc，デバイスの状態 St からなる．N はホスト数 x，スイッチ数 y，ケーブル数 z とし，ホスト情報の集合 H，スイッチ情報の集合 S，ケーブル情報の集合 C としたとき，組(x,y,z,H,S,C)である．h∈H は，識別子 hid，IP アドレス ip,MAC アドレス mac,ゲートウェイ gw,サブネットマスク sb,による組（hid，ip，mac，gw，sb）である．s∈S は，識別子 sid である．c∈C は，デバイス id1 のポート po1 と，デバイス id2 のポート po2 がケーブルで接続されているとき，組(id1,po1,id2,po2)である．ここで，id1,id2 は hid または sid である．Sc は送信パケット情報の集合 Pa とシミュレーション終了ステップ番号 t としたときの組(Pa,t)である．pa∈Pa は送信するステップ番号を s，送信元ホスト ID を hid，プロトコル名を pr，フィールド値の系列を F=としたとき，の組(s,hid,pr,F)である．フィールド値の系列はあらかじめ決められたものとする．学習者に入力させないパラメータは，シミュレータ側で値を決定する．

2-3 ステップ実行

本研究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．シミュレーションをステップ単位で実行する理由は，学習させたい内容の単位で通信プログラムを分割することで，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．しかし実時間を用いてしまうと時刻が連続的なものとなるため，値を変更して再開するためのログを取る回数が現実的でなくなってしまう．デバイスごとにステップが実行されたときにログを取ることも考えられたが，デバイスの数が増えていくにつれてログの数が膨大になってしまう．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ステップ実行を行うようにした．このようにすることで，ログをとるタイミングを決定することができる．

本研究でのステップとは，学習者に理解させたい通信の動作の単位である．例えば学習させたい内容が，ホストに関して，パケットの受信と送信のみである場合は，パケットの受信で 1 ステップ，送信で 1 ステップといった割り振りとなる．しかし通信プログラムは，あるパケットを受信した場合に関数が呼び出され，関数内で別の関数が呼び出されることを繰り返し，結果的にパケットの処理がすべて終了してから関数を出てくる．そのため，パケットを受信して関数が呼び出されると，すべての処理が完了してしまうということになる．

そこで，通信をステップごとに行うために，通信プログラムをステップ単位に分割するという方法をとった．方法としては，関数を展開し，ステップ単位で関数を分割し，1 つ 1 つをモジュールとして扱う．モジュールには，それぞれモジュール間同士の関係性がある．例えばある関数を 3 つのモジュール A,B,C に分割した場合は,A が実行された後に B が実行される.B が実行された後は C が実行されるといった関係性がある．この関係性は，分割する前のプログラムの実行の流れと同じになっていなければならない．しかし，通信プログラムをステップ単位に分割したことにより，以下の問題が発生する．

1. 複数のステップにまたがったローカル変数について，ローカル変数が定義されているモジュールでしか使えなくなってしまう．
2. あるモジュール実行後に，次はどのモジュールが実行されるかがわからない．

この問題点を解決するために以下のような方法をとる．

1. 複数のステップにまたがる可能性がある変数を，デバイスに持たせることで，どのモジュールでも呼び出し可能とする．
2. モジュールに ID を割り振り，モジュールの実行結果によって，次回実行されるモジュールの ID を決定する．

この方法により，複数のモジュールにわたって使用される変数が使用可能となる．また，モジュールに ID を振ることで，モジュール間の関係性を定義するとともに，モジュールを単独で実行可能となる．

2-4 デバイス制御機能

デバイス制御機能は，シミュレータ内で作成するネットワークを動作させる機能である．このネットワークは，ホスト，スイッチ，ケーブルから構成される．ホスト，スイッチ，ケーブルはそれぞれプログラムのインスタンスである．設定ファイルの N から読み込んだホスト，スイッチ，ケーブルの個数分だけインスタンスが準備される.ケーブル情報には接続関係が含まれているので,デバイスとデバイスの接続も完了する．

St を読み込むことで各デバイスの状態を決定する．

1 ステップ内でのデバイスの動作は，ホスト，スイッチ，ケーブルの順に行われる．ホストの配列内のすべての要素で 1 ステップ実行し，スイッチの配列内のすべての要素で 1 ステップ実行し，ケーブルの配列内のすべての要素で 1 ステップ実行する．ここでこの順番で直列に実行する問題点が発生する．あるステップ番号で，ホストやスイッチがパケットを送信したとする．このとき，ケーブルが最後に実行されることが原因で，同じステップ番号で，ケーブルが受信したパケットに対してもう一度処理してしまう．結果として 1 ステップのうちに 2 度ステップを実行されてしまうパケットが存在してしまう．この問題点を解決するためにケーブルには reservbuf が準備してある．reservbuf のパケットはケーブル実行時に処理しない．すべてのデバイスの処理が完了した後で reservbuf からパケットを取り出すことで，2 重にステップ実行されることを防いでいる．ホストとスイッチはケーブルとしかつながらず，ホストとスイッチはケーブルより先に実行されるため，この問題は発生しない．

（1）ホスト

ホスト集合を Hst とする．hst∈Hst は，パケット送信部 sender，パケット受信部 receiver，ホストのポート port，シナリオ待機キューqueue1，受信処理待機キューqueue2 という処理機構からなる．

sender：パケットの送信処理を行う．queue1 からパケット情報を取り出し，pr,smac,dmac,sip,dip の値に応じてパケットを作成する．または receiver から応答のために受信パケットを受け取り，応答パケットを作成する．作成したパケットは，hst の port へ渡される．

receiver：パケットの受信処理を行う．queue2 からパケット情報を取り出し，パケットのデータに応じて処理を行う．応答が必要な場合は，パケットを sender に渡す．

port：port は，ケーブルとホストまたはスイッチとの接続点である．port はケーブルからパケットを受け取ると，queue2 に格納する．sender からパケットが入ってきたとき，接続するケーブルへパケットを送る．

queue1：シナリオ送信機能から受け取ったパケット作成用データを先入れ先出しで溜めておく場所である．

queue2：ポートから受信したパケットを先入れ先出しで溜めておく場所である．

デバイス制御機能がホストに対して 1 ステップで行う動作を以下に示す．動作 ID は，ホスト内でのパケット処理に対する動作に対応しており，例えば，「0=処理待ち」，「1=eth ヘッダの MAC アドレス確認」，「2=arp ヘッダの IP アドレス確認」などが割り当てられている．

手順 1）動作 ID を確認する．

手順 2）動作 ID の処理を実行する．

手順 3）動作 ID を手順 2 の結果にしたがって変更する．

[id=0] シナリオ待機キューを確認する．シナリオ待機キューに要素があれば，デキューする．プロトコル名，フィールド値の系列を入力とする．プロトコル名が icmp であれば id=100 とし，arp であれば id=200 とする．そのほかは id=-1 とする．シナリオ待機キューが空であった場合は，受信処理待機キューを確認する．受信処理待機キューに要素があれば，デキューし，eth ヘッダの送信元 MAC アドレスとホストの MAC アドレスを比較する．送信元 MAC アドレスがホストの MAC アドレスと一致する，または送信元 MAC アドレスがブロードキャストアドレスであれば，id=1 とする．送信元 MAC アドレスがホストの MAC アドレスと一致せず，ブロードキャストアドレスでもない場合は，id=-1 とする．

[id=1] パケットのヘッダを確認する．arp パケットであるならば id=2 とする．ip パケットであるならば，id=10 とする．

[id=2] arp パケットの送信元 IP アドレスとホストの IP アドレスを比較する．一致していた場合は id=3 とし，一致していなかった場合は id=-1 とする．

[id=3] arp パケットの送信元 IP アドレスと送信元 MAC アドレスの組を arp テーブルに登録する．すでに存在する場合はそのままで，すでに存在する IP アドレスに対し，別の MAC アドレスで登録する場合は，元の組を削除し，新しい組を登録する．この処理が終了後に，id=4 とする．

[id=4] arp パケットの operation 部分を確認する．request であるなら id=5 とし，reply であるなら id=-1 とする．

[id=5] arp-reply パケットを送信する．送信元を宛先にし，送信元にはホストのデータとしたパケットを作成し，送信する．送信後に id=200 とする．

[id=10] IP パケットに対する受信処理を行う．応答が必要な場合は応答パケットを送信し，id=0 とする．

[id=100] パケットの宛先 IP アドレスを確認する．ホストと同じセグメント宛てであればそのまま宛先 IP アドレスをパケットにセットし，別セグメント宛てであれば，ゲートウェイの IP アドレスをパケットにセットする．宛先 IP アドレスに対し，arp テーブルを検索する．テーブルに存在していなかった場合は id=101 とし，存在していた場合は id=110 とする．

[id=101] MAC アドレス解決を失敗したため，arp パケットを送信する．送信元 IP アドレス，送信元 MAC アドレス，宛先 IP アドレスをセットし，宛先 MAC アドレスはブロードキャストアドレスとする．operation を request として，パケットを作成する．id=200 とする．

[id=110] MAC アドレス解決が完了しているので，送信元 IP アドレス，送信元 MAC アドレス，宛先 IP アドレス，宛先 MAC アドレスをセットして，icmp-echo-request としたパケットを作成する．id=300 とする．

[id=200] 作成した arp パケットを送信する．id=0 とする．

[id=300] 作成した icmp パケットを送信する．id=0 とする．

[id=-1] パケットを破棄する．id=0 とする．

（2）スイッチ

スイッチ集合を Sw とする．sw∈Sw は，パケット送信部 sender，パケット受信部 receiver，スイッチのポート port，受信処理待機キューqueue という処理機構からなる．

[sender] パケットの送信処理を行う．receiver からパケットと送信先ポートを受け取り，指定されたポートへパケットを渡す．

[receiver] パケットの受信処理を行う．queue からパケットを取り出し，MAC アドレステーブルの更新や，送信先ポートの決定を行う．志う新先ポートへパケットを渡す．

[port] ケーブルとデバイスの接続点である．port はケーブルからパケットを受け取ると，queue に格納する．パケットとともにパケットの受信ポートも格納する．受信ポートは MAC アドレステーブルの更新と，ブロードキャスト時の送信しないポートを決定するときに使用する．sender からパケットを受け取ると，接続するケーブルへパケットを渡す．

[queue] ポートから受信したパケットを先入れ先出しで溜めておく場所である．

スイッチの動作 ID を下記のように定義する．

[id=0] 受信処理待機キューを確認する．受信処理待機キューに要素があれば，デキューし，id=1 とする．

[id=1] パケットの送信元 MAC アドレスを確認し，スイッチの MAC アドレステーブルに登録されていなければ，パケットを受信したポート番号と，送信元 MAC アドレスを組として登録する．id=2 とする．

[id=2] パケットの宛先 MAC アドレスを確認する．MAC アドレステーブルに登録されている MAC アドレスであれば，対応するポートを送信ポートとする．登録されていない，またはブロードキャストアドレスであれば，ブロードキャストとなる．id=2 とする．

[id=3] パケットの送信ポートに従って送信する．ユニキャストであった場合は，対応するポートに送信し，id=0 とする．ブロードキャストの場合は，パケットの受信ポート以外のすべてにパケットを送信する．1 ステップで送信できるパケットは 1 つであり，パケットの送信が完了した場合は id=0 とする．完了していない場合は id=3 のままとなる．

スイッチが 1 ステップで行う動作を以下に示す．

[1] スイッチの動作 ID を確認する．

[2] スイッチの動作 ID のモジュールを実行する．

[3] スイッチの動作 ID を変更する．

（3）ケーブル

ケーブルは，送信先のポート rcvport，ケーブル長 len，パケット位置 c，パケット格納配列 buf，パケット待機場所 reservbuf から構成される．ケーブルが 1 ステップで行う動作を以下に示す．

[1] buf の(c+1) mod len 番目のパケットを rcvport へ送信する．送信されたパケットは，ポートを経由し，デバイスの受信キューへ格納される．

[2] c を 1 増加する．ただし c=len-1 の場合，c=0 とする．
[3] reservbuf にパケットが入っていた場合，中身を取り出し，buf の c 番目にパケットを格納する．

2-5 シナリオ読み取り機能

送信シナリオ読み取り機能は，学習者が書いたシナリオに従って，各デバイスに命令を与える．送信するステップ番号を i，送信元ホスト ID を hid，プロトコル名を pr，フィールド値の系列を $F=\langle f_1, f_2, \ldots, f_n \rangle$ としたとき，組(i,hid,pr,F)を要素とする集合である．フィールド値の集合はプロトコルによってあらかじめ決めてある．送信元 IP アドレスを sip,送信元 MAC アドレスを smac,宛先 IP アドレスを dip,宛先 MAC アドレスを dmac としたとき，icmp は F=<sip,dip>，arp は F=<sip,smac,dip,dmac>となっている．送信シナリオ読み取り機能で行われる処理は以下のようになっている．

[1] ステップカウンタから現在のステップ番号を受け取る．
[2] 受け取ったステップ番号が，シナリオの終了ステップ番号と一致していたら，シミュレーションを終了する．それ以外は手順 3 へ進む．
[3] 送信シナリオの要素 s において，s.i=c となる s を求める．
[4] s があった場合には，s.hid と一致するデバイス名を持つホストを探す．
[5] 該当するホストの queue1 に対し，s.pr と s.F の値を入力したデータをエンキューし，手順 2 へ戻る．
[6] s.c=i となる s がなかった場合は手順 1 へ戻る．

2-6 ステップカウンタ

ステップカウンタは，現在のステップ番号を管理するとともに，シナリオ読み取り機能，デバイス制御機能にステップ番号を通知する．ステップカウンタの動作は以下のようになっている．

[1] シミュレータの入力となっている設定ファイルの St を確認する．
[2] St がない場合は現在のステップ番号の値を 0 とする．St にログファイルが指定されていた場合は，ログのステップ番号を現在のステップ番号とする．
[3] シナリオ読み取り機能に現在のステップ番号を通知する．
[4] シナリオ読み取り機能から完了通知を受け取ったら，デバイス制御機能に現在のステップ番号を通知する．
[5] デバイス制御機能から完了通知を受け取ったら，現在のステップ番号を 1 増加させ，手順 3 へ戻る．

2-7 レジューム機能

レジューム機能は，一度シミュレーションを行った内容に対して，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可能にする機能である．設定ファイルの N,Sc,St を入力とすることで，シミュレーションを行うことができ，St をログファイルから設定することで任意のステップ番号から再開可能となる．St は hdata の集合 HL,sdata の集合 SL,cdata の集合 CL,ステップ番号からなる．デバイス制御機能にてネットワーク構築後，St をもとに各デバイスの持つデータを与えることで，特定のネットワークの状態を再現することができる．さらに，ステップ番号をステップカウンタに与えることにより，ステップ番号も復元した状態から再開が可能となる．

St はログファイルを読み込むことで設定されるので，St に合う形式で記述されているファイルであれば，手動で作成したファイルでも St として読み込むことができる．そのため，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，シミュレーション結果の途中で，あるデバイスの動作の際に，ある値を変更したらどのような影響があるかという局所的な実行も可能となる．しかし，値を変更するにあたって，元の変数と同じ形式で値を入力する必要がある．例えばあるホストの IP アドレスの部分に IP アドレス以外の値を入れてしまうとシミュレータは動作しない．

2-8 ログファイル

ログファイルは，レジュームと可視化に必要なデータを収集したものである．ログファイルはステップごとに作成され，1 つのファイルに各ホスト，各スイッチ，各ケーブルのデータが保存されている．ホストは hdata，sender，receiver，port，queue1，queue2 が保存されている．スイッチは sdata，sender，receiver，port，queue が保存されている．ケーブルは cdata が保存されている．

ログファイルは，シミュレーションのステップ実行中に出力される．各デバイスの動作が完了した直後に，その時のステップ番号と，各デバイスのログが出力される．このログファイルは，各デバイスのインスタン

スが存在している状態でログファイルを与えると，各デバイスの設定が完了する必要がある．そのためログファイルは，シミュレータが読み込むことができる形式で保存されている．

3 プロトタイプシステム

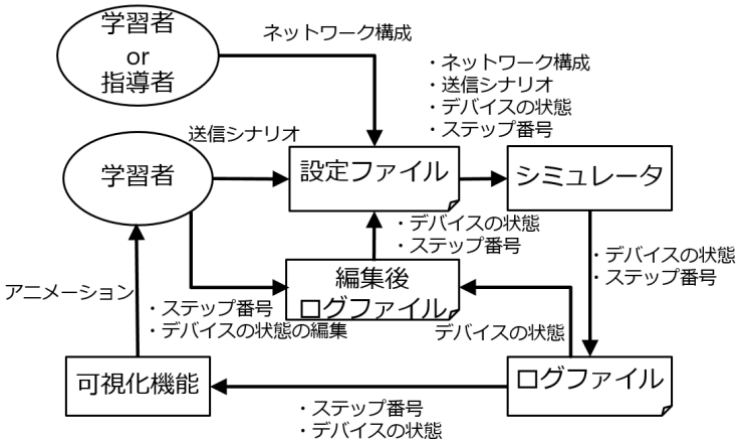
3-1 動作環境および実装概要

［動作環境ホスト 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 （文献 1）
［仮想環境 OS］　Ubuntu 12.04.2（文献 2）
［プログラミング言語］　c++
［開発環境］　Qt4（文献 3）

ホスト内の受信処理については文献[4]の付録であるソースコードを参考にして実装した．スイッチングハブについては文献[5]の uml_switch のソースコードを参考にして実装した．

3-2 システム構成

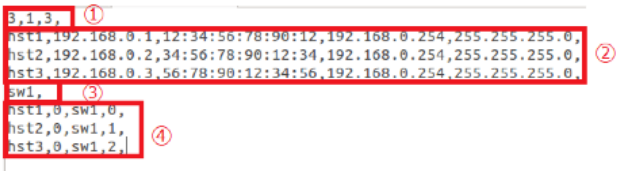
システム構成を下図に示す．



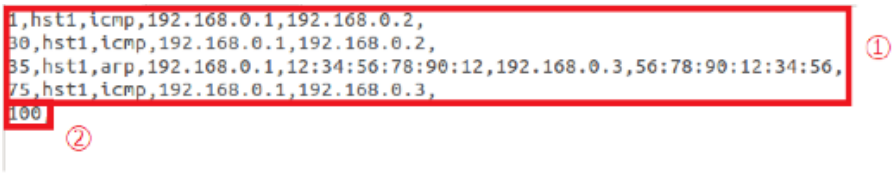
3-3 設定ファイル

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．

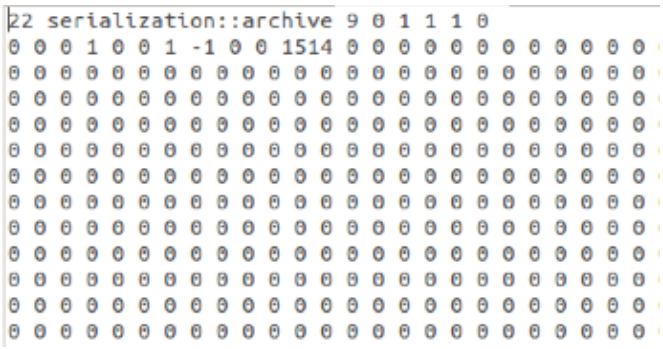
ネットワーク構成ファイルの例を下図に示す．1 の部分では，構築するネットワークで使用する各デバイスの数が記述されている．左から順にホストの数，スイッチの数，ケーブルの数となっている．2 の部分では，ホストの設定が記述されている．左から順にホスト ID，IP アドレス，Mac アドレス，デフォルトゲートウェイ，サブネットマスクとなっている．これはホストの数だけ設定しなければならない．3 の部分では，スイッチの設定が記述されている．スイッチ ID が記述されている．これはスイッチの数だけ設定しなければならない．4 の部分では，ケーブルの設定が記述されている．左から順に接続先 1 のデバイス ID，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．これらすべての情報がカンマ区切りで記述されているものがネットワーク構成ファイルである．



送信シナリオファイルの例を下図に示す．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．送信パケットの情報の数は任意である．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．これらすべての情報がカンマ区切りで記述されているものが送信シナリオファイルである．ただし，1，2 間は改行しなければならない．2 内の各送信パケットの情報も改行でパケットごとにまとめなければならない．



ログファイルの例を下図に示す．ログファイルは c++の boost:serialize を利用して作成している．各デバイスのログとして保存する必要があるメンバ変数を直列化して保存している．中身を直接読むのには適していないが，このファイルを読み込むのはシミュレータまたは可視化機能であり，どちらもログファイルを解釈することができるので問題ない．xml 形式に変換してログファイルを出力することも可能だが，タグによってログファイルのサイズが大きくなってしまい，編集する際に人が見ても行が多すぎてわかりづらいため，利用しなかった．

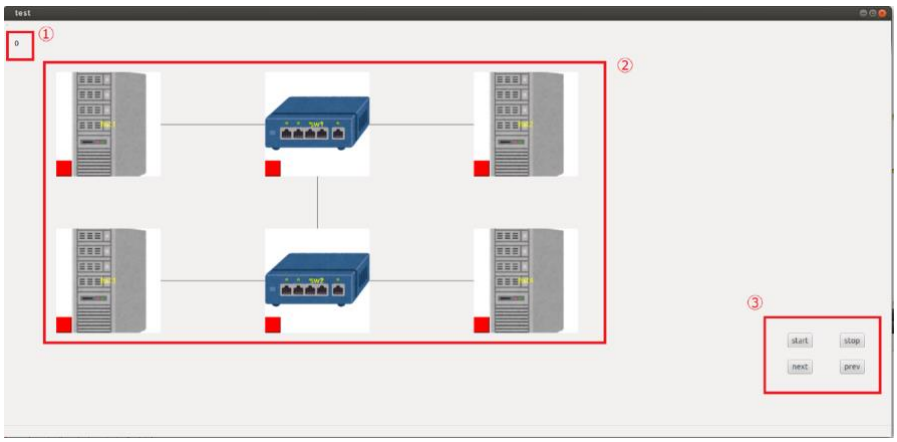


3-4 可視化機能

可視化機能は，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．本研究の目的としては，学習者に通信の仕組みを理解させることであるため，その目的を達成するための事柄を表示する．表示したい内容は以下の項目である．

- 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

(1) メインウィンドウ



1 に示されているデータは現在のステップ番号である．可視化システムを起動させた場合には 0 ステップ目が表示されるようになっている．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．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に使用するボタン群である．左上に「start」，右上に「stop」，左下に「next」，右下に「prev」というボタンが準備されている．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．

（2）パケット配送

デバイスの動作を可視化することも目的の一つであるため，デバイスがどのケーブルへパケットを送信したかを知ることも重要である．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．

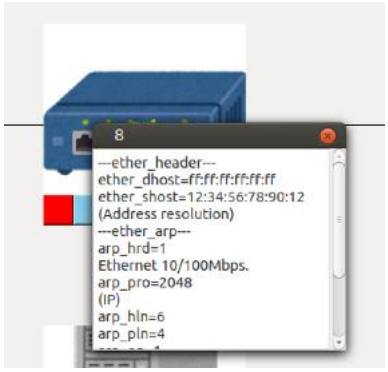
（3）デバイスアイコン

図ではスイッチを表現するボタンが表現されている．左下に設置されている正方形は，デバイスの動作説明ボタンである．詳細は図で説明する．デバイスの動作説明ボタンの隣にあるボタンは，処理中のパケットのボタンである．詳細はで説明する．さらにパケットを処理している最中に次のパケットを受信した場合，キューに格納される．この格納されたパケットはデバイスのボタンの上部に格納された順序とともに表現される．

（4）デバイス内の動作

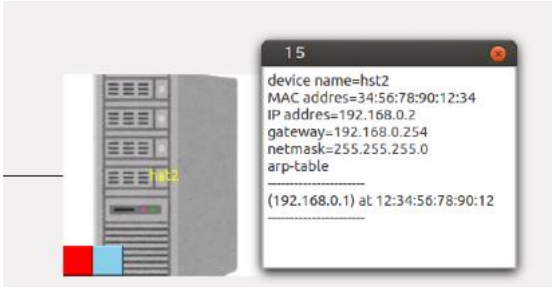
図はスイッチのデバイスの動作説明ボタンを押したときの結果である．ダイアログが出現し，タイトルはボタンが押された時のステップ番号である．ダイアログの内容は動作内容と，動作結果が示されている．今回の例では受信したパケットの宛先 MAC アドレスとスイッチの持っている MAC アドレステーブルを照合することによって，送信先ポートを決定するという動作である．動作結果は改行を挟んで下に書かれている．今回の例では送信先のポートがブロードキャスト(パケットの受信ポート以外)へ送信されるということが書かれており，理由としては，MAC アドレステーブルに照合しなかった，または宛先 MAC アドレスがブロードキャストアドレスであるという結果となっている．このどちらが原因であるかは処理中のパケットを確認することで知ることができる．

（5）デバイスで処理中のパケット



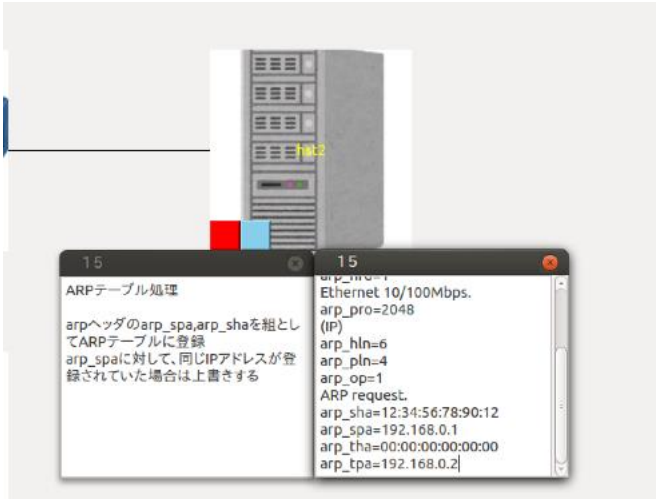
図はスイッチの処理中のパケットのボタンを押したときの結果である．ダイアログが出現し，タイトルはボタンが押された時のステップ番号である．ダイアログの内容はパケットの持つデータをフィールドごとに示したものとなっている．ヘッダごとにデータがまとめられており，各フィールドの名称と，その値が 1 行ごとに書かれている．ダイアログのサイズに収まりきらないデータはスクロールすることで確認することができる．図で示されていたブロードキャストの理由については，「etherheader」の「etherdhost」を確認することでブロードキャストアドレス宛てだったことが確認できる．

（6）デバイス情報

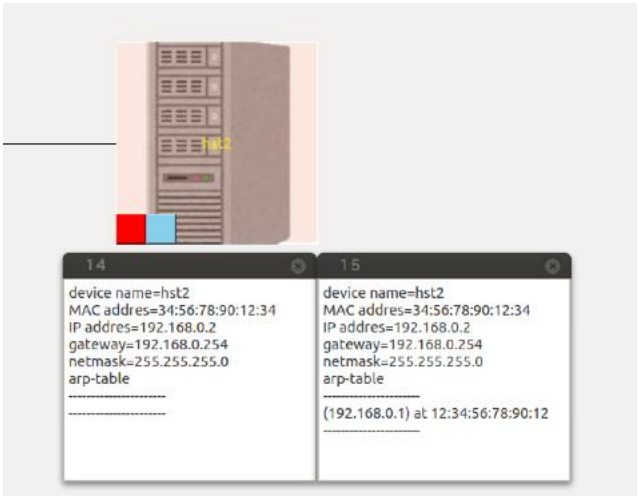


デバイスの処理内容や処理中のパケットのボタン以外の部分を押すと，デバイスの持つ情報を見ることができる．図では hst2 を押したときの結果となっている．ダイアログが出現し，タイトルはボタンが押された時のステップ番号である．ダイアログの内容はホストの持つデータを項目ごとに示したものとなっている．上

から順にデバイスの ID，MAC アドレス，IP アドレス，デフォルトゲートウェイ，サブネットマスク，ARP テーブル一覧である．



上図ではステップ番号 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．動作内容でパケットのフィールド値やデバイスのデータを使うことがあるため，その時は複数のダイアログを見比べながら動作を学習することが必要となる．そのため一度表示したダイアログは非表示ボタン(ダイアログ右上のボタン)を押さない限りは消えない．



上図ではステップ番号 14，15 についての hst2 のデバイスの情報を表示している．表示したダイアログは別のステップ番号となってもそのまま残り続ける．そのためステップ番号 14 終了時からステップ番号 15 終了時までにどのような変化が起こったかを簡単に確認することができる．今回の例では ARP テーブルに変化が生じており，図と合わせて考えると，ステップ番号 15 では，ARP テーブルに対する処理が行われており，「arpspa」,「arpsha」を組として登録することがわかる．実際の値は処理中のパケットから指定された値を見て確認することができる.そしてそれと同じ値がホストの ARP テーブルに登録されていることがわかる.また，動作の説明文に「arpspa に対して，同じ IP アドレスが登録されていた場合は上書きする」とあるが，ステップ番号 14 のホストの情報を確認すれば，今回の動作が上書きではなく，新しく登録したものだということもわかる．

4 評価実験

4-1 目的

本研究は学習用に使用す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うことが考えられる．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．

4-2 評価結果と考察

表 4.1 では，終了ステップ番号を固定し，送信シナリオ内の送信パケット数も 0 となった状態で，各デバイスの数の変化によって調べたものである．実行時間については，ホストについて考えると，ホストの数が 1 増えるたびに，実行時間が約 0.18 秒ずつ増加していることがわかる．この結果から，ホストの数に比例した時間がかかっていると考えられる．

スイッチについて考えると，スイッチの数が 1 増えるたびに，実行時間が約 0.05 秒ずつ増加していることがわかる．この結果から，スイッチの数に比例した時間がかかっていると考えられる．ホストの動作とスイッチの動作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，ホストの合計時間とスイッチの合計時間を足したものが実行時間になると考えられる．

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，ホストやスイッチに接続して，ホストやスイッチから実行時間を引く形で考える．ケーブルが 1 本の場合は 0.762-0.415=0.347 秒と考えられる．同様にケーブル 2 本の場合は，1.268-0.415-0.099=0.754 秒，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．このことからケーブルの数が 1 増えるたびに，実行時間が 0.35 秒ずつ増加していることが考えられる．

結果的に，各デバイスの数の増加に比例して，実行時間も増加していくという結果となった．CPU 使用率も似たような値となっているため，実行時間の信頼性もある．ただしデバイスを 1 つも設置しなかった場合は，処理が短すぎて CPU を使わなかった時間が半分を占めてしまった．しかしほかの実行時間に比べて小さな値であり，半分にした 0.014 秒をほかの結果から引いても，比例関係に問題はない．

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．今回使用している Ubuntu の環境では 4GB のメモリが割り当てられており，利用可能なメモリは 1.9GB あった．そのため，理論上数万個のデバイスを設置することも可能である．しかし，数万ものデバイスを設置してしまうと，実行時間が数十分となり，演習には現実的ではないため，一般的な PC ではメモリの問題はないと考えてよい．

1 ステップあたりのログサイズは，ホストについて考えると，ホストの数が 1 増えるたびに，ログサイズが約 12.8kB ずつ増加していることがわかる．この結果から，ホストの数に比例したサイズとなると考えられる．スイッチについて考えると，スイッチの数が 1 増えるたびに，ログサイズが約 3.5kB ずつ増加していることがわかる．この結果から，スイッチの数に比例したサイズとなると考えられる．ケーブルについて考えると，ケーブルが 1 本の場合は 49.9-25.6=24.3kB，2 本の場合は 77.8-25.6-3.5=48.7kB，3 本の場合は 115-38.4-3.5=73.1kB となり，約 24.4kB ずつ増加していることがわかる．この結果から，ケーブルの数に比例したサイズとなると考えられる．これらが実行されたステップ分出力されるため，デバイスの数を増やしすぎることや，終了ステップ番号を大きくしすぎることはできない．

表 4.2 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．結果はステップ数の増加に比例して実行時間も比例すると考えられる．これはシミュレーション中の時間の大部分がステップ実行に使われているためであり，シミュレーション準備段階のネットワークの構成や，シナリオの読み込みといった処理が無視できるほど大きいためと考えられる．この準備段階にかかる時間が，表のデバイスの数が 0 のときとほとんど同じであると考えられる．

表 4.3 では，設置するデバイスの数，終了ステップ番号を固定し，送信シナリオ(icmp パケット)の数にて評価を行った．送信パケットは同時刻に別のホストから arp パケットを送信した．結果としては，送信パケット数が増えれば 1 ステップ番号での最大ログサイズも大きくなるという結果となった．しかしこの結果は送信パケット数が増えれば，それに比例して大きくなるとは言えない．理由は，ログサイズが大きくなった原因がケーブルを流れるパケットにあると考えられるためである．そのため，送信シナリオが少なくても，ブロードキャストでコピーされるとログサイズが大きくなり，ユニキャストではログサイズがあまり大きくならない．したがって，現在ネットワークに流れているパケットが多ければ多いほど，ログのサイズが大きくなると考えられる．実際にパケットを送信し始めたタイミングではログサイズが少し増えただけだが，しばらくすると(スイッチでブロードキャストされたであろうタイミング)ログがさらに増加した．そのため，1 つのスイッチにたくさんのケーブルが接続されているとデータが大きくなりやすいと考えられる．ログの保

存方法として，データをシリアライズしているため，パケット内のデータもログのサイズに関連する．しかし，どれだけログのサイズが大きくなっても送信シナリオの送信パケット数が 0 の時と比べて 2 倍程度にしかならないと考えられる．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．実行時間は少し増加しているように考えられる．しかし，シナリオがあった場合となかった場合の違いは，各デバイスの実行するステップ ID であるため，シナリオの変更による実行時間の大きな違いが出なかったと考えられる．物理メモリの使用量は，シナリオがあった場合は増加したが，シナリオの数に合わせての変化は小さいという結果となった．

結果として，演習に現実的なレベルでのネットワークかどうかは，各デバイスの数，終了ステップ番号で決定すると考えられる．表 4.1，4.2，4.3 の結果と実行したいシミュレーションの入力を比較することで実行時間やログサイズを予測することができそうである．理論を学習したばかりの学習者向けで．デバイスの動作に焦点を当てたものであるため，デバイスの数をたくさん準備する必要はないと考えられる．そのため，今回のシミュレータを演習に使うことは可能であると考えられる．

表 4.1：デバイス数による評価

ホスト数	スイッチ数	ケーブル数	ログ (kB)	実行時間 (s)	RSS(kB)	CPU 使用率 (%)
0	0	0	0.03	0.028	9600	51
1	0	0	12.8	0.233	10176	86
2	0	0	25.6	0.415	10208	92
3	0	0	38.4	0.608	10256	88
0	1	0	3.5	0.099	10032	76
0	2	0	7	0.150	10048	81
0	3	0	10.5	0.205	10064	86
2	0	1	49.9	0.762	10352	93
2	1	2	77.8	1.268	10480	89
3	1	3	115	1.817	10544	89

表 4.2：ステップ数による評価

ホスト数	スイッチ数	ケーブル数	ステップ	実行時間 (s)	RSS(kB)	CPU 使用率 (%)
3	1	3	100	1.817	10544	89
3	1	3	200	3.724	10544	89
3	1	3	300	5.651	10560	89
3	1	3	1000	20.384	10560	89

表 4.3：送信シナリオによる評価（ホスト 3，スイッチ 1，ケーブル 3，ステップ数 100）

シナリオ数 (同時刻)	実行時間 (s)	RSS(kB)	CPU 使用率 (%)	最大ログサイズ (kB)
0	1.817	10544	89	115.0
1	1.844	10928	89	138.6
2	1.826	10928	89	146.8
3	1.856	10928	89	150.8

【参考文献】

[1] VMware Workstation Player:
https://www.vmware.com/jp/products/workstation-player.html. (2019/5/15 アクセス)
[2] Ubuntu: https://www.ubuntulinux.jp/. (2019/5/15 アクセス)

[3] Qt4: http://doc.qt.io/archives/qt-4.8/. (2019/5/15 アクセス)
[4] 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
[5] The User-mode Linux Kernel Home Page,
http://user-mode-linux.sourceforge.net/. (2019/5/15 アクセス)

〈発　表　資　料〉

題　　名	掲載誌・学会名等	発表年月
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	電子情報通信学会情報ネットワーク研究会	2018 年 11 月
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	情報処理学会 教育学習支援情報システム研究会	2019 年 3 月