パケットジェネレータの実装

2015/6/19

藤田将輝

1 はじめに

本資料では，デバッグ支援環境において，NIC ドライバに処理させるパケットを作成する機能を実装したことを示す．本デバッグ支援環境では NIC を用いずに任意のタイミングで割り込み処理を発生させるため，パケットを擬似する必要がある．このため，デバッグ支援 OS 上で動作し，パケットを作成するプログラム (以下，パケットジェネレータ) を実装した．プロトコルは UDP としている．現在は任意のメッセージを指定し，パケットを作成できる．このパケットを本環境により処理させたところ，デバッグ支援 OS 上で動作する UDP の受信プログラムでパケットを受信できることを確認した．

2 パケットジェネレータ

パケットジェネレータはデバッグ支援 OS 上で動作する AP として実装した．作成するパケットは UDP であり，Ether フレームを擬似している．パケットジェネレータを動作させるとパケットを作成し，作成したパケットとこのサイズを引数にデバッグ支援機構を呼び出す．また，パケットを作成する際，任意のメッセージを入力し，これをデバッグ対象 OS に送信できる．

3 パケット配送の処理流れ

3.1 概要

パケットジェネレータを使用し，デバッグ支援機構を呼び出すことで，デバッグ対象 OS へパケットを配送する．これを確認するため，デバッグ対象 OS 上で UDP の受信プログラムを動作させ，このプログラムが正常にパケットを受信し，指定したメッセージを出力するかどうかを実験した．以降でこの実験を行う際の環境の構成と，実験の流れについて説明する．

3.2 環境構成

本デバッグ環境は Mint を用いてデバッグ支援 OS とデバッグ対象 OS の 2 つの OS を動作させる．また，デバッグ支援 OS はデバッグ支援機構を保持し，デバッグ対象 OS は改変した NIC ドライバを保持している本実験を行う環境の構築手順について以下で説明する．

(1) デバッグ支援 OS からデバッグ対象 OS を起動する．
(2) デバッグ対象 OS でネットワークインタフェースを起動する．
(3) デバッグ対象 OS で UDP の受信プログラムを動作させる．

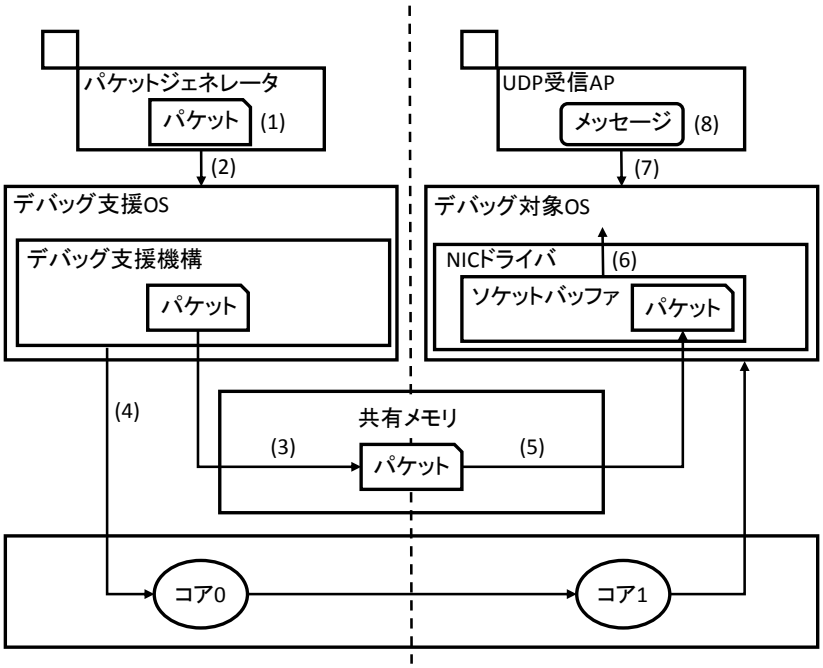

図 1　パケットの配送流れ

3.3　パケットの配送

構築した環境を用いて実験を行う．パケットジェネレータを用いて，パケットを作成し，これが正常に処理されることを実験する．パケットジェネレータを動作させてから，デバッグ対象 OS の画面上にメッセージが表示されるまでの流れを図 1 に示し，以下で説明する．

(1) デバッグ支援 OS でパケットジェネレータを動作させる．
(2) パケットジェネレータにより，パケットが作成され，デバッグ支援機構が呼び出される．
(3) デバッグ支援機構は作成されたパケットを共有メモリに配置する．
(4) デバッグ支援機構がデバッグ対象 OS に割り込みを発生させる．
(5) 割り込みハンドラが動作し，NIC ドライバはパケットを共有メモリからソケットバッファに格納する．
(6) NIC ドライバはソケットバッファを上位層に送信する．
(7) デバッグ対象 OS 上で動作する UDP の受信プログラムが UDP パケットを受け取る．
(8) デバッグ対象 OS 上で動作する UDP の受信プログラムがメッセージを画面に出力する．

4 実装

4.1 処理概要

パケットジェネレータは指定したメッセージに定義したヘッダを付与することで Ether フレームを擬似している．パケットジェネレータを動作させると，3 つのヘッダを定義し，これに適切な値を割り当てることで Ether フレームを作成している．以降で，定義したヘッダと，その内容について述べる．

4.2 定義したヘッダ

NIC ドライバが処理をするのは Ether フレームである．このため，Ether フレームを擬似する必要がある．現在は，正常に動作するパケットをキャプチャし，その内容から値を決定している．Ether フレームの擬似を行うため，以下の 3 つのヘッダを定義した．

(1) Ether ヘッダ
(2) IPv4 ヘッダ
(3) UDP ヘッダ

4.3 各ヘッダの情報

定義した各ヘッダにおけるメンバの役割について以下に示し，説明する．

(1) Ether ヘッダ
　（A）宛先 MAC アドレス (6byte)
　　宛先の MAC アドレスを表す．
　（B）送信元 MAC アドレス (6byte)
　　送信元の MAC アドレスを表す．
　（C）IP のバージョン (2byte)
　　次に続く IP ヘッダのバージョンを表す．IPv4 ヘッダならば 0x0800 である．
(2) IPv4 ヘッダ
　（A）バージョン (4bit)
　　IP プロトコルのバージョンを表す．IPv4 ならば 0x4 である．
　（B）ヘッダ長 (4bit)
　　データを除いたヘッダ部分のみのサイズを表す．4byte 単位で表しており，0x5 ならば 20byte となる．
　（C）サービスタイプ (1byte)
　　IP パケットの優先度などを表す．現在はほとんど使われておらず，意味を持っていないことが多い．

（D）データグラム長 (2byte)
IP パケット全体の長さを表す.
（E）ID(2byte)
フラグメンテーションが起きた際の識別に使用される．毎回ランダムな値が格納される.
（F）フラグ (3bit)
フラグメンテーションの際に使用される．パケットがまだ続くか否かを識別する.
（G）フラグメントオフセット (13bit)
フラグメントされたパケットが IP パケットのどの位置かを識別するために使用される.
（H）TTL(1byte)
パケットの寿命を表す.
（I）プロトコル番号 (1byte)
次に続くプロトコルの情報を表す.
（J）チェックサム (2byte)
IP ヘッダのチェックサムを表す．計算方法として 1 の補数演算を利用する.
（K）送信元 IP アドレス (4byte)
送信元の IP アドレスを表す.
（L）宛先 IP アドレス (4byte)
宛先の IP アドレスを表す.

(3) UDP ヘッダ
（A）送信元ポート (2byte)
送信元のポートを表す.
（B）宛先ポート (2byte)
宛先のポートを表す.
（C）サイズ (2byte)
UDP パケット全体の長さを表す.
（D）チェックサム (2byte)
UDP パケットのチェックサムを表す．計算方法は IP パケットと同様に 1 の補数演算を利用する.

5 パケットジェネレータの動作

パケットジェネレータを動作させると，パケットが作成され，デバッグ支援機構が呼び出される．パケットを作成する際のプログラムの動作について以下に示し，説明する.

(1) メッセージを指定し，プログラムを起動する.
(2) 各ヘッダを定義し，値を割り当てる．この際，割り当てる値は，キャプチャしたパケットを参考にしている.

(3) ヘッダの末尾に作成したメッセージを配置する．
(4) ヘッダ全体のサイズと作成したメッセージのサイズを足しあわせたものをパケットのサイズとする．
(5) パケットとパケットのサイズを引数にデバッグ支援機構を呼び出す．

6 課題

現在は，ヘッダの情報をキャプチャしたパケットから指定している．このため全ての情報が静的に割り当てられている．これらの情報の内，IP アドレス，ポート番号，サイズ，およびチェックサムはユーザの指定，またはユーザが指定したメッセージのようなデータから計算されるべきである．したがってこれらをユーザの入力によって変化させるよう改変する．

7 おわりに

本資料ではパケットジェネレータについて示した．動作について，デバッグ対象 OS 上で動作する UDP の受信プログラムが指定したメッセージを出力したことから，正常に処理されていることを確認した．また，今後の課題として，ユーザの入力によってヘッダの値を自動的に決定するよう，パケットジェネレータを改変する．