NICドライバ改変によるパケットキャプチャ方法

2016/5/31

藤田将輝

1　はじめに

開発支援環境でパケットを擬似するにあたって，NICがメモリに配置したパケットをキャプチャし，その構成を確認した．本資料では，NICドライバを改変することによってパケットをキャプチャし，パケットの内容を確認する方法について示す．

2　目的

開発支援環境はNICハードウェアのエミュレートを行う．NICハードウェアの機能の1つにパケットをメモリに配置する機能がある．この機能を開発支援OSが再現する際，パケットを作成する必要がある．パケットを作成する機能を実現するにあたって，実際のNICがメモリに配置するパケットの構成を確認した．具体的には，NICドライバを改変し，NICドライバがメモリからパケットを取得するタイミングであらかじめ確保した配列にパケットを保存し，任意のタイミングでその内容を確認する機構を実装し，確認した．

3　実装環境

パケットキャプチャ機構の実装環境を表1に示す．

表1　実験環境

項目名	環境
OS	Fedora14 x86_64(Linux 3.0.8)
NICドライバ	RTL8169
ソースコード	r8169.c

4　パケットキャプチャ機構の処理流れ

NICドライバを改変し，あらかじめ確保した配列に最新10個のパケットを格納するものとする．NICがパケットを受信してからパケットをキャプチャするまでの処理流れについて図1に示し，以下で説明する．

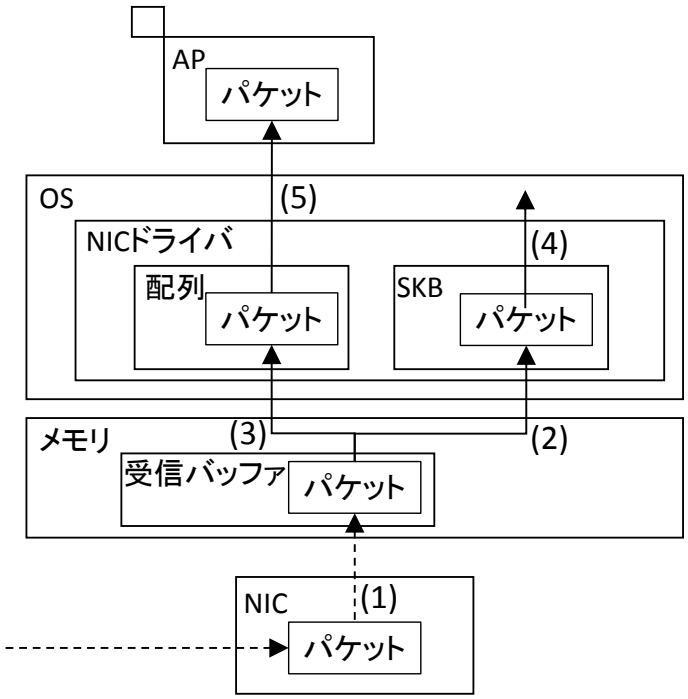

図 1　パケットキャプチャの処理流れ

(1) NIC がパケットを受信し，パケットを受信バッファに格納する．
(2) NIC ドライバが動作し，受信バッファに格納されているパケットをソケットバッファに複写する．
(3) ソケットバッファに格納した直後にカーネルに確保した配列にもパケットを複写する．
(4) ソケットバッファは上位層に転送される．
(5) カーネルに確保した配列に格納されたパケットはシステムコールにより，ユーザ空間に複写され，内容を確認できる．

5　実装

4 章で示したパケットキャプチャを実装するにあたり，以下の機能を実装した．ソースコードは GitHub における, fujita-m/Linux-3.0.8 の packet_capture ブランチの 17326f3bbe35ad1e6a72286637dd2df90d19f975 (コミット ID) にある．なお，システムコールの追加方法については「< new 249-06 > Linux カーネルへのシステムコール実装手順書」を参考にすると良い．

(1) カーネルに配列を確保
NIC が受信したパケットを保存しておくため，カーネルに配列を確保する．今回は，受信した最新の 10 個のパケットを確保することとする．また，1 つのパケットサイズの上限は MTU である 1500B としている．この機能は arch/x86/kernel/sys_packet_capture.c に記述している．
(2) 確保した配列にパケットを格納

NIC ドライバがパケットをソケットバッファに複写するタイミングで同じパケットを確保した配列に格納する．この機能は drivers/net/r8169.c に記述している．

(3) システムコールによりパケットをユーザ空間に複写
配列へのポインタを受け取り，保存しているパケットを複写するシステムコールを実装する．これにより，受信した最新 10 個のパケットをユーザ空間で確認できる．この機能は arch/x86/kernel/sys_packet_capture.c に記述している．

6 動作

実装したシステムコールを呼び出すアプリケーションを作成し，実装したパケットキャプチャ機能を動作させた際，以下の結果が得られた．なお，キャプチャしたパケットの先頭から 34B までを表示している．

```
------------------------------------------------------------------
1  ----start----
2  20cf303e497a8851fb677c8608004500005c7f7c40008006c17fac1530e0ac153095
3  ----start----
4  20cf303e497a8851fb677c86080045000028 7f7940008006c1b6ac1530e0ac153095
5  ----start----
6  ffffffffffff480fcf3844a008060001080006040001480fcf3844a0ac1530c30000
7  ----start----
8  20cf303e497a8851fb677c8608004500005c7f7a40008006c181ac1530e0ac153095
9  ----start----
10 ffffffffffff8851fb6ece1f080600010800060400018851fb6ece1fac1530100000
11 ----start----
12 20cf303e497a8851fb677c860800450000287f7b40008006c1b4ac1530e0ac153095
13 ----start----
14 ffffffffffff8851fb6ecdd1080600010800060400018851fb6ecdd1ac15304c0000
15 ----start----
16 ffffffffffff8851fb677c78080600010800060400018851fb677c78ac1530e70000
17 ----start----
18 ffffffffffff74d435140592080600010800060400 0174d435140592962e01050000
19 ----start----
20 ffffffffffff8851fb6ecdd308004500004e29e30000801187eaac1530a7ac15ffff
------------------------------------------------------------------
```

上記の結果を得た際，SSH を用いて実験用計算機と文書作成用計算機とで通信を行なっていた．パケットの Ether ヘッダには，宛先 MAC アドレスと差出元 MAC アドレスが含まれており，キャプチャし

たパケットの Ether ヘッダには実験用計算機と文書作成用計算機の MAC アドレスが含まれていた (2,4,8,12 行目の先頭から 12B)．また，IP ヘッダには宛先 IP アドレスと差出元 IP アドレスが含まれており，キャプチャしたパケットの IP ヘッダには実験用計算機と文書用計算機に割り当てた IP アドレスが含まれていた (2,4,8,12 行目の末尾から 8B)．これらにより，正しくパケットをキャプチャできていると言える．

7 おわりに

本資料では，NIC を改変することにより，パケットをキャプチャする方法について示した．これを用いることで，パケットの構成を知ることができる．