

# Working with raw format files

Victor Moreno

February 12, 2014

## 1 Objective

This document describes the structure of the "raw" format files generated by the usage of the HPCAP2 driver as explained in [1].

## 2 File data structures

A raw file is composed by a set of consecutive packets. Each packet is preceded by its corresponding header which contains information related to the packet just as shown in Fig.1:

**Seconds** 4 bytes containing the seconds field of the packet timestamp.

**Nanoseconds** 4 bytes containing the nanoseconds field of the packet timestamp.

**Caplen** 2 bytes containing the amount of bytes of the packet included in the file.

**Len** 2 bytes containing the real size of the packet.

The end of the file is denoted by the appearance of a pseudo packet showing the amount of padding bytes added at the end of the file (in order to generate files of the same size). The padding pseudo-packet has a similar header than any other packet in the file with the difference that both the "Seconds" and the "Nanoseconds" fields are set to zeros. Once the padding pseudo-packet has been located, the padding size can be read from any of the "Len" or "Caplen" fields. Note that the padding length could be zero.

## 3 Example code

The next pages show an example code for a programs that reads a raw file and generates a new pcap file with the contents of the first one.

```

#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <sched.h>
#include <unistd.h>
#include <assert.h>
#include <errno.h>
#include <sys/time.h>
#include <pcap/pcap.h>

#include "../include/hpcap.h"
#include "raw2.h"

int main(int argc, char **argv)
{
    FILE* fraw;
    pcap_t* pcap_open=NULL;
    pcap_dumper_t* pcapure=NULL;
    u_char buf[4096];
    struct pcap_pkthdr h;
    u_int32_t secs,nsecs;
    u_int16_t len;
    u_int16_t caplen;
    int i=0,j=0,k=0,ret=0;
    char filename[100];
    u_int64_t filesize=0;

    if( argc != 3 )
    {
        printf("Uso: %s <fichero_RAW_de_entrada> <fichero_PCAP_de_salida>\n", argv[0]);
        exit(-1);
    }

    fraw=fopen(argv[1],"r");
    if( !fraw )
    {
        perror("fopen");
        exit(-1);
    }

    //abrir fichero de salida
    #ifdef DUMP_PCAP
        sprintf(filename,"%s_%d.pcap",argv[2],j);
        pcap_open=pcap_open_dead(DLT_EN10MB,CAPLEN);
        pcapure=pcap_dump_open(pcap_open,filename);
        if( !pcapure )
        {
            perror("Error in pcap_dump_open");
            exit(-1);
        }
    #endif

```

```

#endif

while(1)
{
    #ifdef PKT_LIMIT
    i=0;
    while( i<PKT_LIMIT )
    #endif
    {
        /* Lectura de info asociada a cada paquete */
        if( fread(&secs,1,sizeof(u_int32_t),fraw)!=sizeof(u_int32_t) )
        {
            printf("Segundos\n");
            break;
        }
        if( fread(&nsecs,1,sizeof(u_int32_t),fraw)!=sizeof(u_int32_t) )
        {
            printf("Nanosegundos\n");
            break;
        }
        if( nsecs >= NSECS_PER_SEC )
        {
            printf("Wrong NS value (file=%d,pkt=%d)\n",j,i);
            //break;
        }
        if( (secs==0) && (nsecs==0) )
        {
            fread(&caplen,1,sizeof(u_int16_t),fraw);
            fread(&len,1,sizeof(u_int16_t),fraw);
            if( len != caplen )
                printf("Wrong padding format [len=%d,caplen=%d]\n", len, caplen);
            else
                printf("Padding de %d bytes\n", caplen);
            break;
        }

        if( fread(&caplen,1,sizeof(u_int16_t),fraw)!=sizeof(u_int16_t) )
        {
            printf("Caplen\n");
            break;
        }
        if( fread(&len,1,sizeof(u_int16_t),fraw)!=sizeof(u_int16_t) )
        {
            printf("Longitud\n");
            break;
        }

        /* Escritura de cabecera */
        h.ts.tv_sec=secs;
        h.ts.tv_usec=nsecs/1000;
    }
}

```

```

h.caplen=caplen;
h.len=len;

#ifdef DEBUG
printf("[%09ld.%09ld] %u bytes (cap %d), %lu, %d,%d\n", secs, nsecs, len, caplen, filesize);
#endif
if( caplen > MAX_PACKET_LEN )
{
    break;
}
else
{
    /* Lectura del paquete */
    if( caplen > 0 )
    {
        memset(buf,0,MAX_PACKET_LEN);
        ret = fread(buf,1,caplen,fraw);
        if( ret != caplen )
        {
            printf("Lectura del paquete\n");
            break;
        }
#ifdef DEBUG2
        for(k=0;k<caplen;k+=8)
        {
            printf( "\t%02x %02x %02x %02x\t%02x %02x %02x %02x\n",
                buf[k], buf[k+1], buf[k+2], buf[k+3],
                buf[k+4], buf[k+5], buf[k+6], buf[k+7]);
        }
#endif
    }
}

/* Escribir a fichero */
#ifdef DUMP_PCAP
pcap_dump( (u_char*)pcapture, &h, buf);
#endif
i++;
filesize += sizeof(u_int32_t)*3+len;
}

#ifdef PKT_LIMIT
printf("%d paquetes leidos\n",i);
if(i<PKT_LIMIT)
    break;
j++;
#ifdef DUMP_PCAP
pcap_dump_close(pcapture);
//abrir nuevo fichero de salida
sprintf(filename,"%s_%d.pcap",argv[2],j);
pcap_open=pcap_open_dead(DLT_EN10MB,CAPLEN);

```

```

        pcapure=pcap_dump_open(pcap_open,filename);
        if( !pcapture)
        {
            perror("Error in pcap_dump_open");
            exit(-1);
        }
        #endif
    #endif
}
printf("out\n");

#ifdef PKT_LIMIT
#ifdef DUMP_PCAP
    pcap_dump_close(pcapture);
#endif
    j++;
    printf("%d paquetes leidos\n",i);
#endif

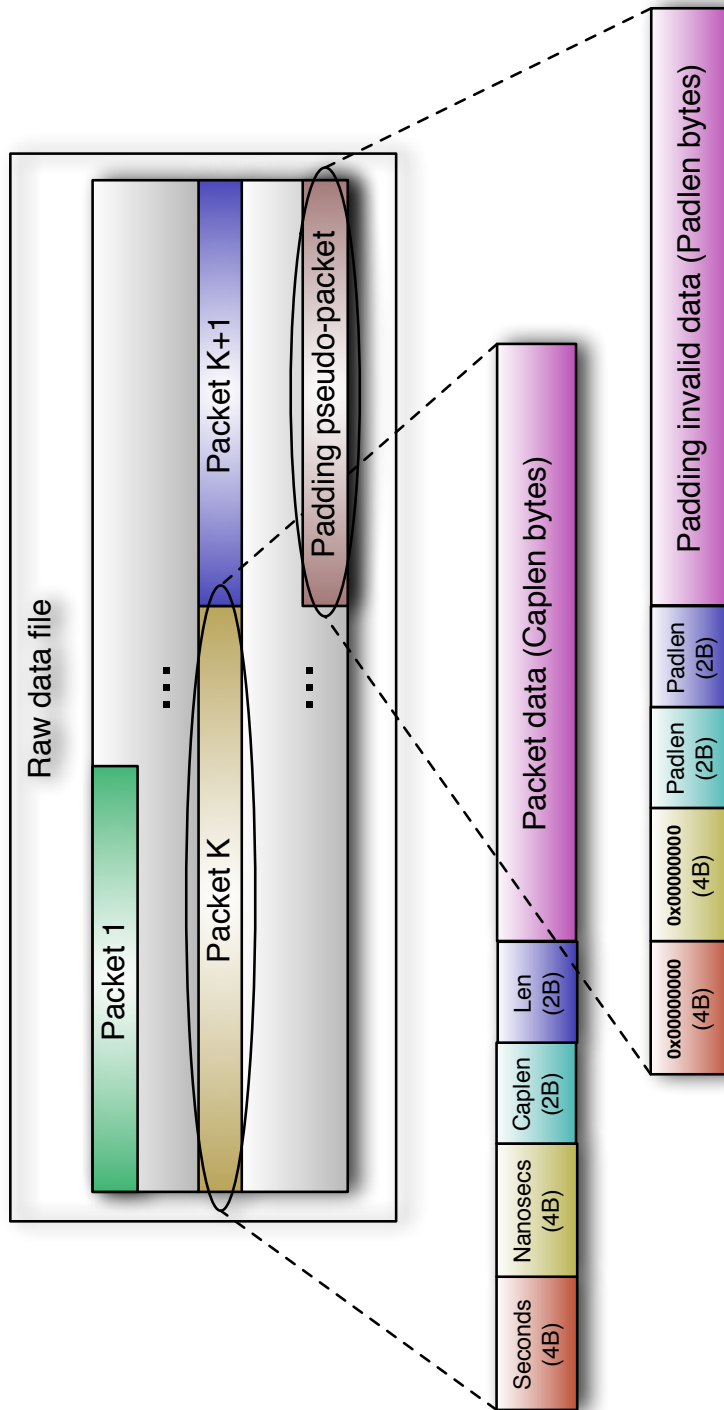
printf("%d ficheros generados\n",j);
fclose(fraw);

return 0;
}

```

## References

- [1] V. Moreno. Development and evaluation of a low-cost scalable architecture for network traffic capture and storage for 10gbps networks. Master's thesis, Escuela Politecnica Superior UAM, 2012.



6  
Figure 1: Raw file format