Vežba 4 – Libpcap (Raspbian) – WinPcap (Windows) programska biblioteka

1. Osnove

Libpcap je programska biblioteka namenjena pristupu mrežnim uslugama na niskom nivou (nivo linka) u realnom vremenu na Unix sistemima. Ona pruža interfejs visokog nivoa sistemima za analiziranje paketa, dobro je testirana i dokumentovana. Pomoću nje paketi se mogu sakupljati i skladištiti na disku ili se mogu direktno prosleđivati programima za analizu (tj. sondama).

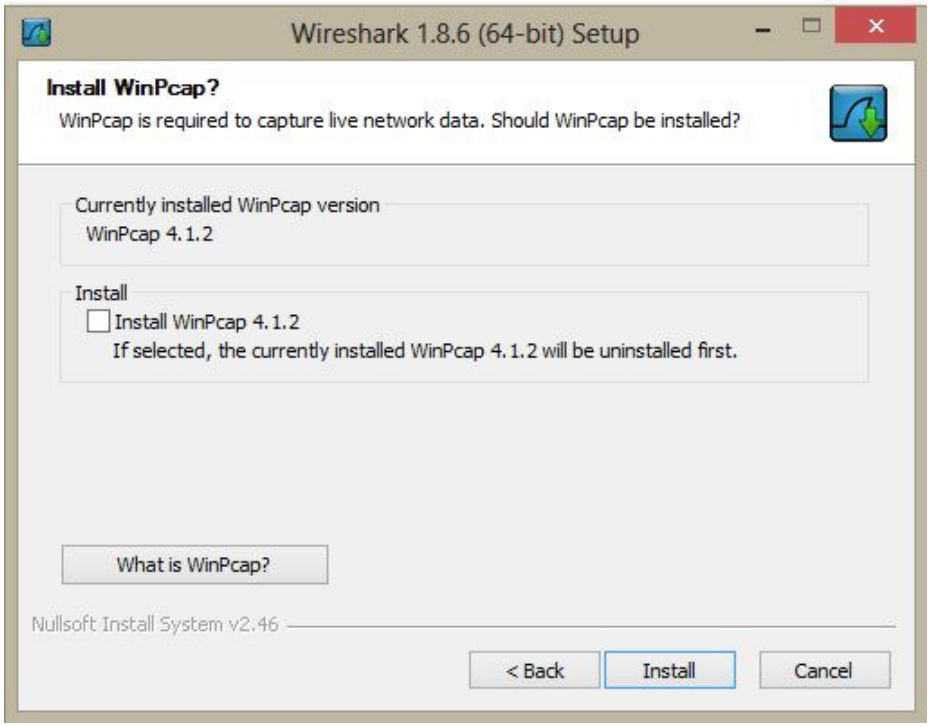
WinPcap je programski alat namenjen pristupu mrežnim uslugama na niskom nivou (nivo linka) u realnom vremenu namenjen Windows operativnim sistemima (Windows verzija *libpcap*-a). *WinPcap* je *open source* programska biblioteka zasnovana na *libpcap* C/C++ biblioteci. Potreba za bibliotekom je uočena kada je Microsoft onemogućio korištenje programskih funkcija nižih nivoa potrebnih za upravljanje mrežnom fizičkom arhitekturom.

Ove pakete koriste brojni projekti otvorenog koda vezani za računarske mreže, a jedan od najpoznatijih i najčešće korištenih je *Wireshark* (prethodno poznat pod nazivom Ethereal).

Libpcap instalacija (ukucati u terminal Raspbian-a): *sudo apt-get install libpcap-dev*. Više podataka o *libpcap* paketu može se naći na stranicama http://www.tcpdump.org/ ili iz terminala pomoću komande: *man pcap*.

Više podataka o *WinPcap* paketu može se naći na stranicama http://www.winpcap.org/. *WinPcap* može biti preuzet sa službene web stranice ili u okviru drugih aplikacija. Tipično, ovaj program može biti preuzet u okviru programskog paketa *Wireshark*. Kako bi sprečili instalaciju istog, potrebno je pratiti uputstvo prilikom instalacije (Slika 1).

Usled postojanja specifičnih razlika između *WinPcap* (Windows) i *libpcap* (Raspbian) API-a kao i funkcionalnosti, za pravilno korištenje neophodno je konsultovati dokumentaciju istih. U nastavku teksta će biti opisan *WinPcap* (Windows verzija *libpcap*-a).



Slika 1. Primer instalacionog prozora *WinPcap*-a (Wireshark)

WinPcap je programski alat koji dopušta mrežnoj infrastrukturnoj kartici da radi u hibridnom režimu rada. Takođe ovaj alat može biti nazvan "potencijalno neželjenom aplikacijom" koja bi trebala biti uklonjena sa sistema. *WinPcap* je alat o kome se dugo vodi rasprava na mnogim sigurnosnim forumima i, čini se, glavni razlog zašto ga nazivaju "virusom", su metode njegove distribucije. Kao i mnogo sličnih aplikacija, distribucija ovog programa zavisi od nekih besplatnih i javnih programa. Ova metoda je postala popularna za primoravanje samih korisnika da preuzmu aplikacije na svoje računare bez prikladnog pitanja za dozvolu. Osim toga, *WinPcap* ne samo da infiltrira računar – ovaj program može izazvati neprekidno preusmeravanje mrežnog saobraćaja u okviru web pretraživača, prikazivati neželjene pop-up reklame i pratiti svoje korisnike. Konačno, može izazvati i probleme pri pokušajima samog uklanjanja *WinPcap* sa sistema.

WinPcap se sastoji od rukovaoca (engl. *driver*), koji se oslanja i proširuje operativni sistem sa ciljem da obezbedi pristup mrežnim uslugama niskog nivoa, i biblioteke (engl. *library*) koja se koristi za pristup mrežnim slojevima niskog nivoa. Ova biblioteka takođe sadrži i Windows verziju poznatog *libpcap* Unix API-ja.

WinPcap programska biblioteka omogućava aplikacijama da uhvate i prenesu mrežne pakete zaobilazeći protokol stek, a ima i dodatne korisne funkcije, uključujući filtriranje paketa na kernel nivou kao i podršku za daljinsko hvatanje paketa i mrežnu statistiku.

2. Koje su mogućnosti Libpcap/WinPcap bibloteke?

Informacije o dostupnim mrežnim adapterima

Hvatanje paketa sa mrežne kartice

Analiza sadržaja pojedinačnih paketa

Praćenje statistike saobraćaja

Čitanje paketa iz datoteke i snimanje u datoteku

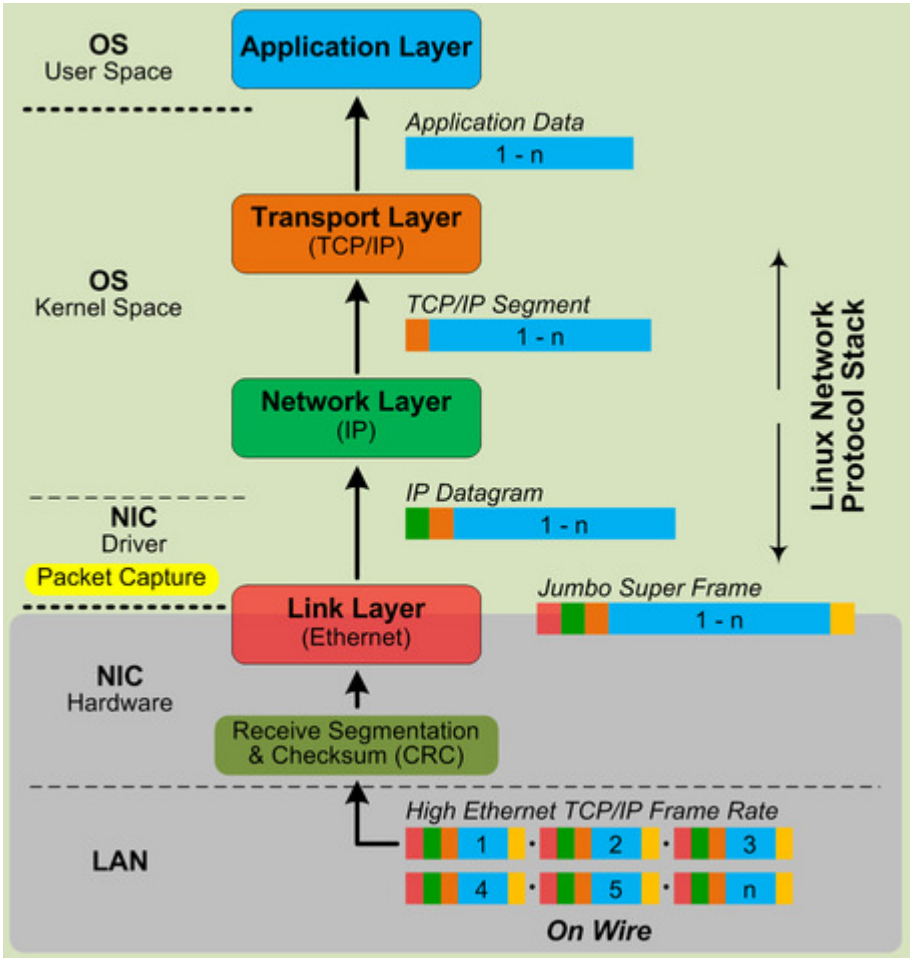
Izgradnja paketa

Slanje paketa

3. Hvatanje paketa

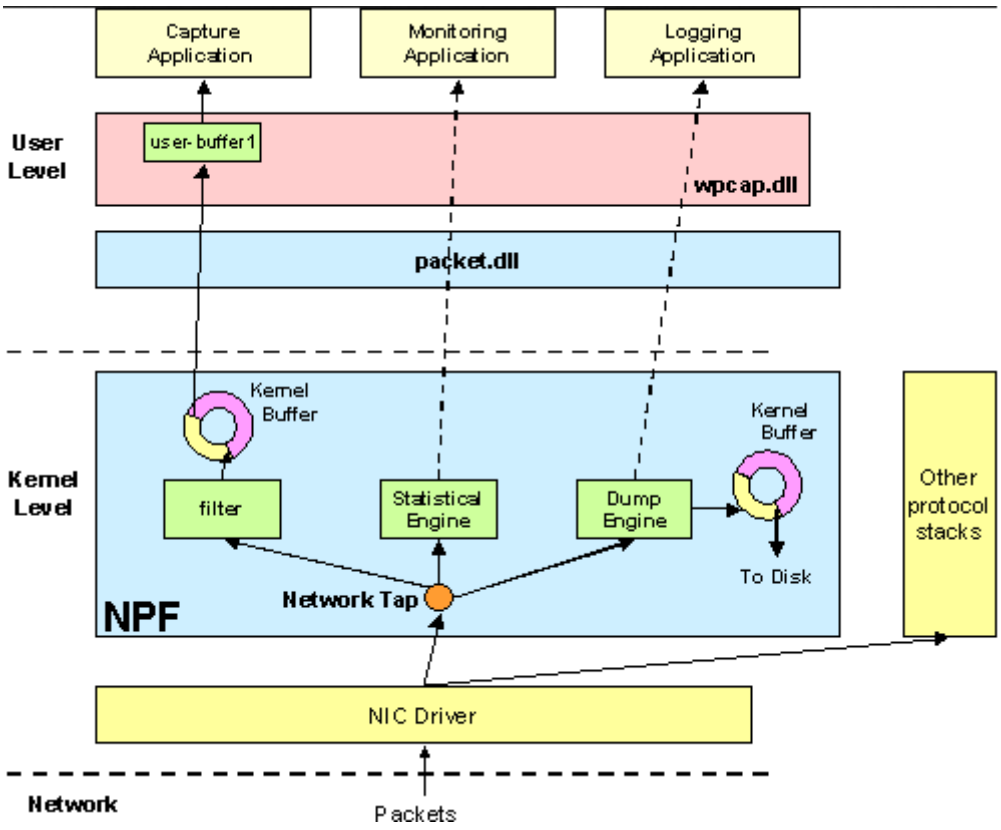
Hvatanje paketa (engl. *packet capture*) je akcija prikupljanja paketa tj. podataka tokom njihovog puta kroz mrežu. U nastavku ćemo opisati kako funkcioniše hvatanje paketa u *Ethernet*-zasnovanim mrežama.

Svaki put kada mrežna kartica primi *Ethernet* okvir, ona proverava da li odredišna MAC (ETH, PHY) adresa istog odgovara sopstvenoj MAC adresi. Ukoliko je to slučaj, ona generiše prekid. Rukovaoc prekidnih rutina mrežne kartice je zadužen za rukovanje ovim prekidima. Ovaj rukovaoc dodaje informacije o vremenu prijema na paket (engl. *timestamps*) i kopira paket iz prihvatne memorijske zone kartice (engl. *buffer*) u blok memorije rezidentnog prostora OS-a (engl. *kernel space*). Tada, rukovaoc utvrđuje koji je tip paketa dobijen posmatranjem "*ether-type*" polja *Ethernet* zaglavlja i prosleđuje isti odgovarajućem protokol-rukovaocu u protokol steku. U većini slučajeva okvir će sadržati IPv4 datagram, tako da će IPv4 rukovaoc paketima biti pozvan. Ovaj rukovaoc obavlja brojne provere sa ciljem da obezbedi, na primer, da paket nije primljen sa greškom i da je zapravo namenjen ovom odredišnom mrežnom entitetu. Ukoliko su svi testovi prošli, IP zaglavlja se uklanjaju i ostatak paketa se prosleđuje sledećem protokol-rukovaocu transportnog nivoa (TCP ili UDP). Ovaj proces se ponavlja sve dok podaci ne stignu do aplikativnog sloja.



Slika 2. TCP/IP protokol stek

Kada koristimo paket snifer, paketi prolaze kroz isti proces opisan gore, ali sa jednom razlikom: mrežni rukovaoc takođe šalje kopiju primljenog ili poslatog paketa delu kernela koji se naziva paket filter - **Network Packet Filter** (NPF). NPF je realizovan kao protokol drajver. Paket filtri su ono što čini hvatanje paketa mogućim. Oni propuštaju inicijalno sve pakete (engl. *by default*), ali kao što ćemo videti kasnije, oni obično nude i opcije naprednog filtriranja. Korištenje filtera direktno pozitivno utiče na performanse aplikacije. Hvatanje mrežnih paketa podrazumeva određene bezbednosne rizike, stoga većina sistema zahteva uvođenje administratorskih prava da bi se koristila ova mogućnost.

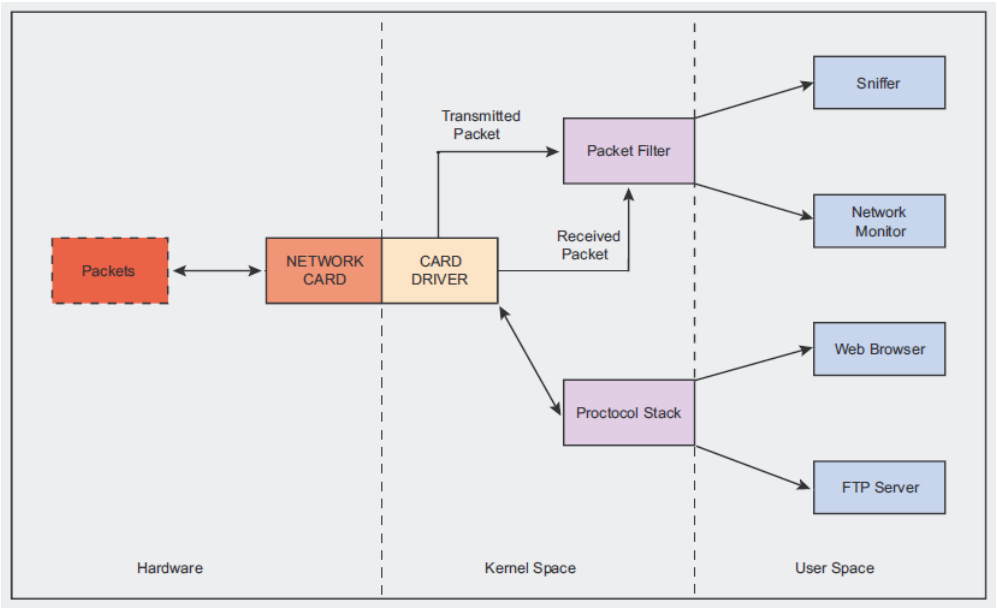


Slika 3. Network Packet Filter (NPF)

Mrežni paketi se smeštaju u kružni bafer zajedno sa zaglavljem u kome se čuvaju informacije o vremenu prispeća paketa i veličini paketa. Maksimalna veličina bafera je ograničena. Ukoliko se bafer prepuni, novopristigli paketi biće ignorisani, sve dok se u baferu ne napravi mesta za smeštanje novih paketa. Pristup paketima koji se nalaze u baferu je efikasno, tako da se uz pomoć jedne sistemske operacije čitanja može dobaviti grupa paketa iz bafera. Dakle, postoji direktna srazmera između veličine bafera i maksimalne količine podataka koja se može dobaviti jednim sistemskim pozivom. Pored toga, veoma je bitna činjenica kolika je minimalna količina podataka koja se može dobaviti iz bafera. Veća vrednost utiče da broj sistemskih poziva bude manji, što smanjuje zauzeće procesora, dok sa druge strane manja vrednost garantuje da će kernel kopirati poruke aplikaciji čim ona bude spremna da ih obradi što može biti jako bitno kod aplikacija koje rade u realnom vremenu.

Libpcap/WinPcap omogućava prijem paketa nezavisno od protokol steka, što znači da nije u mogućnosti da presretne pakete koji saobraćaju unutar same mašine. Da bi libpcap/WinPcap presreo neki paket, taj paket mora fizički proći kroz mrežnu karticu.

Slika 4 ilustruje prethodno opisan proces.



Slika 4. Elementi uključeni u proces hvatanja paketa

Libpcap programska biblioteka otvorenog koda je osmišljena tako da se koristi iz C/C++ izvornog koda. Međutim, upotrebom različitih programskih rešenja koja koriste prvobitni *libpcap* API (engl. *wrappers*) omogućeno je njeno korištenje iz programskih jezika kao što su Perl, Python, Java, C# ili Ruby.

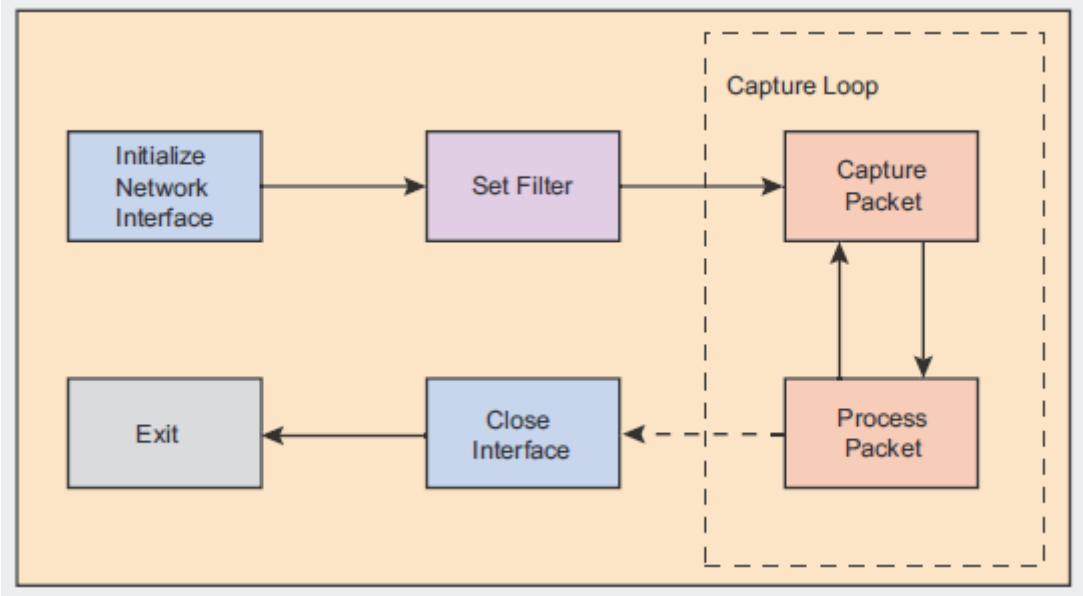
Ova biblioteka pruža API za hvatanje, analizu, kreiranje i slanje paketa na nivou veze.

Mogućnosti:

- Dobavljanje liste mrežnih adaptera
- Prihvatanje paketa sa mreže i slanje paketa
- Dobavljanje statistike
- Primena Berkley Packet filtera
- Čitanje paketa iz fajla i snimanje u fajl

Za izradu programa za analizu mrežnog saobraćaja neophodne su teorijske osnove vezane za poznavanje mehanizama funkcionisanja komunikacionih protokola, formata njihovih paketa, itd.

Slika 5 ilustruje tok programa tipične *pcap* aplikacije.



Slika 5. Tok programa tipične pcap aplikacije

4. Izgradnja i slanje paketa

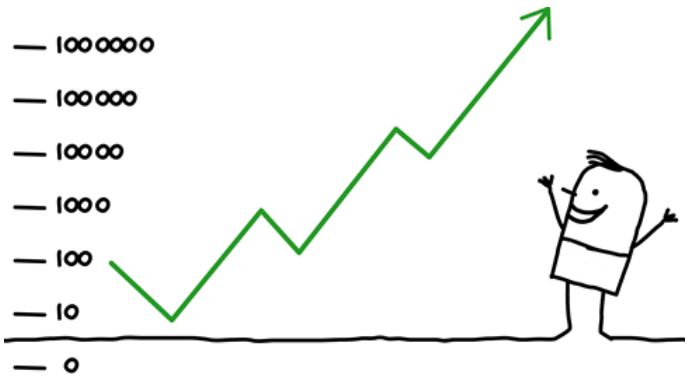
Libpcap/WinPcap omogućava da se preko mrežne kartice pošalju sirovi paketi koje je kreirao korisnik u svojoj aplikacij. Zbog toga što se paket direktno prosleđuje mrežnoj kartici (izbegavanjem bilo kakve dodatne enkapsulacije) aplikacija se mora postarati za kreiranje svih potrebnih zaglavlja. Na primer, ako bismo želeli poslati UDP datagram preko mreže koji smo sami kreirali, pored samih korisničkih podataka potrebno je popuniti UDP zaglavlje, IP zaglavlje i Ethernet zaglavlje. Time aplikacija vodi računa o svim slojevima OSI modela izuzev fizičkog sloja i kontrolne sume okvira (FCS) na nivou veze, koju računa sama mrežna kartica.

Brzina slanja paketa nije na zavidnom nivou, jer svako slanje zahteva novi sistemski poziv. Međutim, moguće je pomoću funkcije *ioctl* (sa kodom *pBIOCSWRITEREP*) promeniti standardno ponašanje slanja paketa i sa jednim sistemskim pozivom omogućiti da se više puta (npr. 1000 puta) pošalje isti paket. Ova mogućnost najčešće se koristi za testiranje performansi mreže.

5. Praćenje statistike

WinPcap pruža mogućnost praćenja statistike saobraćaja na mreži korištenjem programibilnog modula za monitoring na nivou kernela. Sakupljanje podataka i računanje statistike se procesira na nivou kernela, tako da aplikacija dobija rezultate statistike. Time se izbegava potreba za kopiranjem pojedinačnih paketa aplikaciji, što pozitivno utiče na opterećenje procesora i zauzeće memorije prilikom sakupljanja statistike.

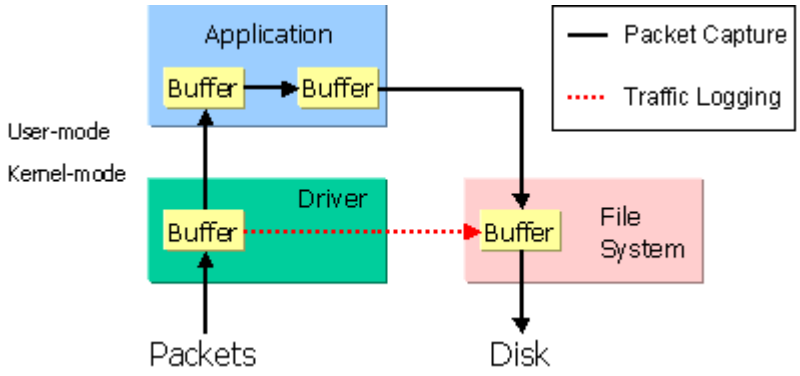
Za računanje statistke koriste se komponenta za filtriranje i brojači. Korisniku se pruža informacija o broju paketa koji su zadovoljili uslov i njihovoj ukupnoj veličini u bajtima. Period dobavljanja statistike kao i izrazi koji se koriste u filterima su konfigurabilni.



Slika 6. Praćenje statistike mrežnog saobraćaja

6. Rad sa datotekama

Libpcap/WinPcap omogućava da se mrežni podaci sačuvaju na disk direktno iz kernela.



Slika 7. Zapisivanje uhvaćenih mrežnih paketa na disk

Na slici 7 je (crnim strelicama) prikazan tradicionalni način koji zahteva korištenje 4 bafera (za hvatanje paketa u kernelu, za čuvanje uhvaćenih paketa u aplikaciji, za zapis podataka u datoteku (*stdio*) i jedan u file system-u) da bi se uhvaćeni paket zapisao u datoteku. Nasuprot tome, korištenjem kernela za istu namenu isti posao obavlja se korištenjem dva bafera i samo jednog sistemskog poziva, čime se postižu znatno bolje performanse. Za čuvanje presretenih paketa koristi se *.pcap* ekstenzija (u pitanju je binarna datoteka), koju je moguće otvoriti pomoću WinPcap biblioteke ili aplikacije Wireshark.

Dobavljanje liste mrežnih adaptera

Na početku bilo koje aplikacije zasnovane na libpcap/WinPcap biblioteci potrebno je pribaviti listu dostupnih mrežnih adaptera.

Dobavljanje liste priključenih mrežnih adaptera omogućeno je pomoću funkcija:

- ***int pcap_findalldevs (pcap_if_t **alldevsp, char *errbuf);***
  - ❖ Pravljenje liste mrežnih uređaja koji se mogu otvoriti pomoću funkcije ***pcap_open_live***. Vraća listu *pcap_if_t* struktura, koje sadrže detaljne informacije o priključenim adapterima.
- ***char* pcap_lookupdev (char *errbuf);***
  - ❖ Vraća prvi validni mrežni uređaj u sistemu.

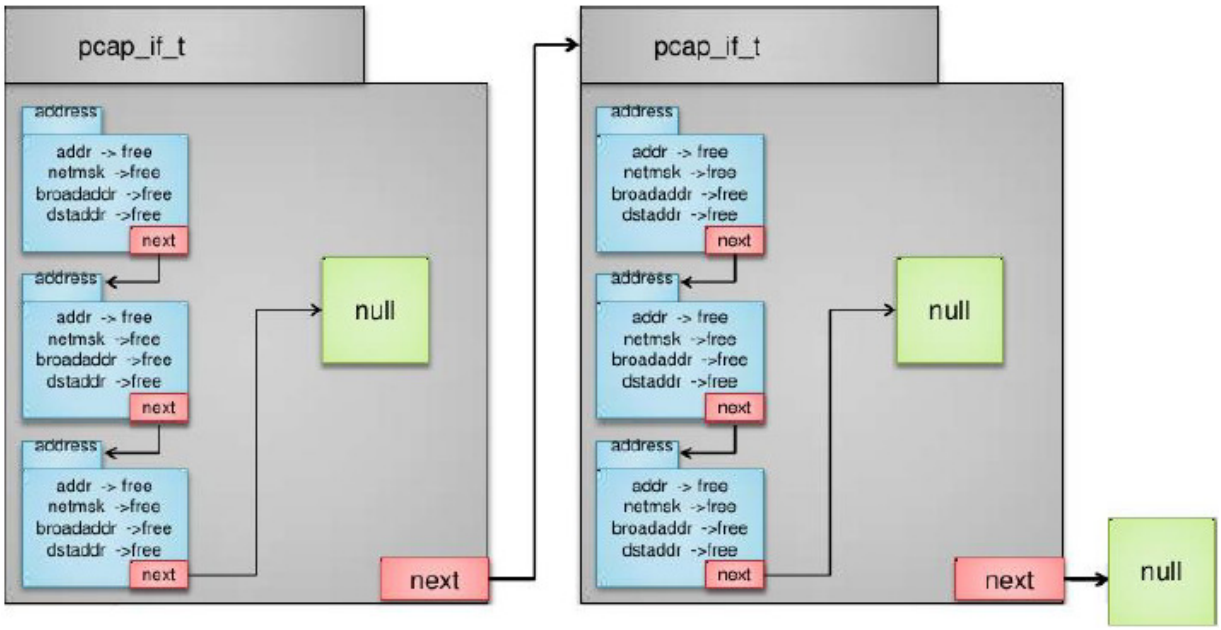
Funkcija:	Opis:
pcap_findalldevs	Dobavljanje liste priključenih mrežnih adaptera
Parametri:	Opis:
pcap_if_t **alldevsp	Lista mrežnih uređaja koji se mogu otvoriti pomoću funkcije *pcap_open_live*.
char *errbuf	Pokazivač na string gde će se nalaziti opis greške ukoliko se ona desi.
Povratna vrednost:	Opis:
int	Ako se funkcija uspešno izvrši vraća vrednost 0. U suprotnom, vraća se vrednost -1, a u parametru *errbuf* se nalazi poruka o grešci.

U nastavku su dati deklaracija i opis *pcap_if_t* strukture.

```
typedef struct pcap_if pcap_if_t;
struct pcap_if {
        struct pcap_if *next;
        char *name;
        char *description;
        struct pcap_addr *addresses;
        unsigned int flags;
};
```

Polje:	Značenje polja:
struct pcap_if *next	Pokazivač na sledeći element u listi. Ima vrednost NULL za zadnji element u listi.
char *name	Pokazivač na string koji sadrži ime adaptera koje se prosleđuje funkciji pcap_open_live().
char *description	Pokazivač na string koji sadrži čoveku razumljiv opis adaptera. Može da ima i vrednost NULL (tada nema opisa).
struct pcap_addr *addresses	Pokazivač na prvi element liste adresa interfejsa.
unsigned int flags	PCAP_IF_flago-ovi za interfejse. Trenutno, jedini mogući flag je PCAP_IF_LOOPBACK, koji se postavlja ukoliko je interfejs “loopback”.

Može se primetiti iz deklaracije pcap_if strukture da interfejs može sadržati jednu ili više adresnih struktura tipa pcap_addr. Svaka od struktura iz te liste sadrži informacije o adresi, subnet maski, broadcast adresi i odredišnoj adresi za tu adresu.



Slika 8. Lista adaptera i njihovih adresa

```
struct pcap_addr {
        struct pcap_addr *next;
        struct sockaddrr  *addr;
        struct sockaddrr  *netmask;
        struct sockaddrr  *broadaddr;
        struct sockaddrr  *dstaddr;
};
```

Polje:	Značenje polja:
struct pcap_addr *next	Pokazuje na sledeću adresu u listi i na taj način omogućava prolazak kroz sve adrese datog interfejsa. *NULL vrednost označava da nema više adresa u listi.*
struct sockaddrr *addr	Pokazivač na adresu interfejsa.
struct sockaddrr *netmask	Pokazivač na subnet masku. *NULL vrednost označava da subnet maska nije konfigurisana.*
struct sockaddrr *broadaddr	Pokazivač na broadcast adresu. *NULL vrednost označava da interfejs ne podržava broadcast.*
struct sockaddrr *dstaddr	Pokazivač na odredišnu adresu. *NULL vrednost označava da interfejs nije point-to-point interfejs.*

Oslobađanje liste mrežnih adaptera

Nakon završetka rada sa listom svih mrežnih adaptera, potrebno je osloboditi ovu listu. Za tu namenu se na kraju svake libpcap/WinPcap aplikacije poziva funkcija *pcap_freealldevs()*.

void pcap_freealldevs (pcap_if_t *devices);

Funkcija:	Opis:
pcap_freealldevs	Oslobađa listu mežnih adaptera koju je alocirala i popunila funkcija pcap_findalldevs().
Parametri:	**Opis:**
pcap_if_t *devices	Lista priključenih mrežnih adaptera.
Povratna vrednost:	**Opis:**
void	Nema povratne vrednosti.

Primer:

```
/*-------------------------------------------------------------------------------*/
pcap_if_t* devices; // List of network interfaces
pcap_if_t* device;  // Network interface
int i = 0;          // Interface counter
char errorMsg[PCAP_ERRBUF_SIZE+1]; // Buffer for errors

// Retrieve the device list of network intefaces
if(pcap_findalldevs(&devices, errorMsg) == -1)
{
	printf("Error in pcap_findalldevs: %s\n", errorMsg);
	return 1;
}

// Print the list of network interfaces
for(device=devices; device; device=device->next)
{
	printf("%d. %s", ++i, device->name);
	if (device->description)
		printf(" (%s)\n", device->description);
	else
		printf(" (No description available)\n");
}

if(i==0)
{
	printf("\nNo interfaces found! Make sure WinPcap is installed.\n");
	return -1;
}

// We don't need any more the device list. Free it
pcap_freealldevs(devices);
/*-------------------------------------------------------------------------------*/
```

ZADACI VEŽBE

1. Analiza i tumačenje sadržaja datog programskog koda *pcap* projekta koji izlistava mrežne adaptere koji se nalaze na računaru.

2. Prevođenje i pokretanje programskog koda *pcap* projekta na Raspberry Pi-u (Raspbian) i u Visual Studio 2010 okruženju (Windows).

3. Za svaki mrežni adapter tipa IPv4 potrebno je ispisati:

   - logičku adresu, (**0.5 bodova**)
   - subnet masku i (**0.5 bodova**)
   - broadcast adresu. (**0.5 bodova**)

4. Za svaki mrežni adapter u slučaju da adresa istog pripada IPv6 protokolu ispisati samo tip adrese (IPv6). (**0.5 bodova**)