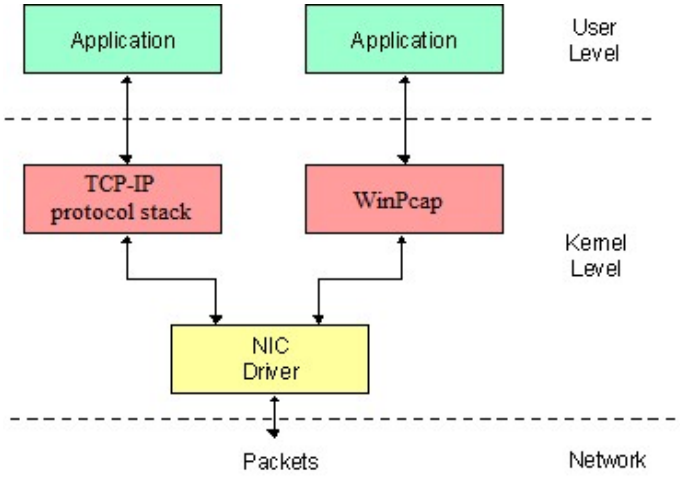
Vežba 8 – Uvod u WinPcap

1. WinPcap

WinPcap je programska biblioteka namenjena Windows operativnom sistemu koja omogućava hvatanje paketa (WinPcap = **Win**dows **P**acket **cap**ture). Hvatanje paketa predstavlja presretanje i analizu paketa tokom njihovog puta kroz mrežnu karticu. Ova biblioteka pruža direktan pristup mrežnoj kartici, zaobilazeći protokol stek i process enkapsulacije paketa, čime se omogućava pristup paketu na 2. nivou ISO-OSI modela (na nivou veze).



Slika 1- WinPcap drajver i biblioteka

WinPcap je *open source* programska biblioteka zasnovana na libcap C/C++ biblioteci. Potreba za bibliotekom je uočena kada je Microsoft onemogućio korišćenje programskih funkcija nižih nivoa potrebnih za upravljanje mrežnom fizičkom arhitekturom. Ovu biblioteku koriste brojni projekti otvorenog koda vezani za računarske mreže, a jedan od najpoznatijih i najčešće korišćenih je *Wireshark.*

Postoji javna rasprava o ispravnosti korišćenja WinPcap-a, jer korisnik ima mogućnost presresti sve poruke, a ne samo one koje su njemu namenjene i mogućnost slanja poruka čije informacije u zaglavljima ne moraju odgovarati stvarnosti, čime postoji mogućnost ugrožavanja bezbednosti mrežnog saobraćaja.

WinPcap može biti preuzet sa službene web stranice http://www.winpcap.org/, na kojoj se pored same instalacije nalazi i dokumentacija sa zanimljivim primerima. Takođe, ova biblioteka može biti instalirana u okviru instalacije Wireshark aplikacije.

Koje su mogućnosti 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

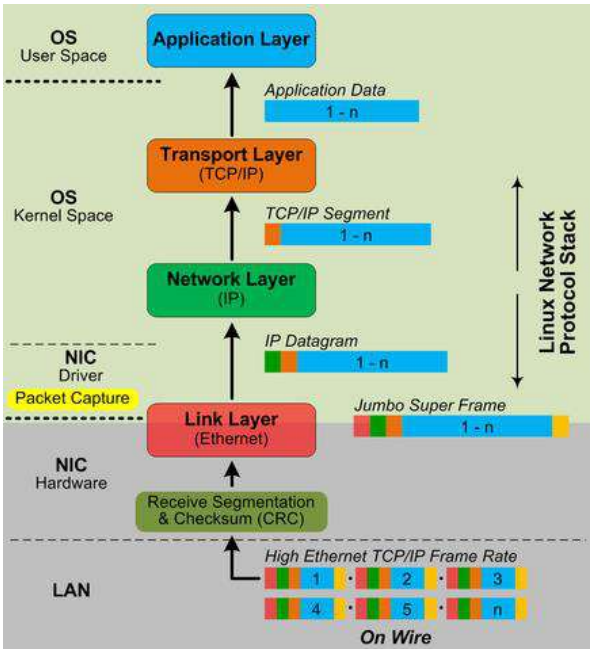
Izgradnja paketa

Slanje paketa

1.1 Hvatanje paketa

Put paketa od mrežne kartice kroz protokol stek do aplikacije

Svaki put kada mrežna kartica primi *Ethernet* okvir sa mreže, ona proverava da li fizička adresa odredišta paketa odgovara adresi koja je pridružena mrežnoj kartici. Ukoliko je to slučaj, mrežna kartica generiše prekid. Rukovalac prekidnih rutina mrežne kartice dodaje informacije o vremenu prijema paketa (engl. *Timestamp*) i kopira paket iz prihvatne memorijske zone kartice (engl. *Buffer*) u blok memorije rezidentnog prostora (engl. *Kernel space*). Tada, rukovalac utvrđuje koji je tip paketa dobijen posmatranjem "*ether-type"* polja (koje definiše tip zaglavlja koji sledi iza Ethernet zaglavlja) i prosleđuje isti odgovarajućem protokol-rukovaocu u protokol steku. U većini slučajeva okvir će sadržati IPv4 paket, tako da će IPv4 rukovalac paketima biti pozvan. Nakon što paket prođe proveru, IP zaglavlje se uklanja 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.

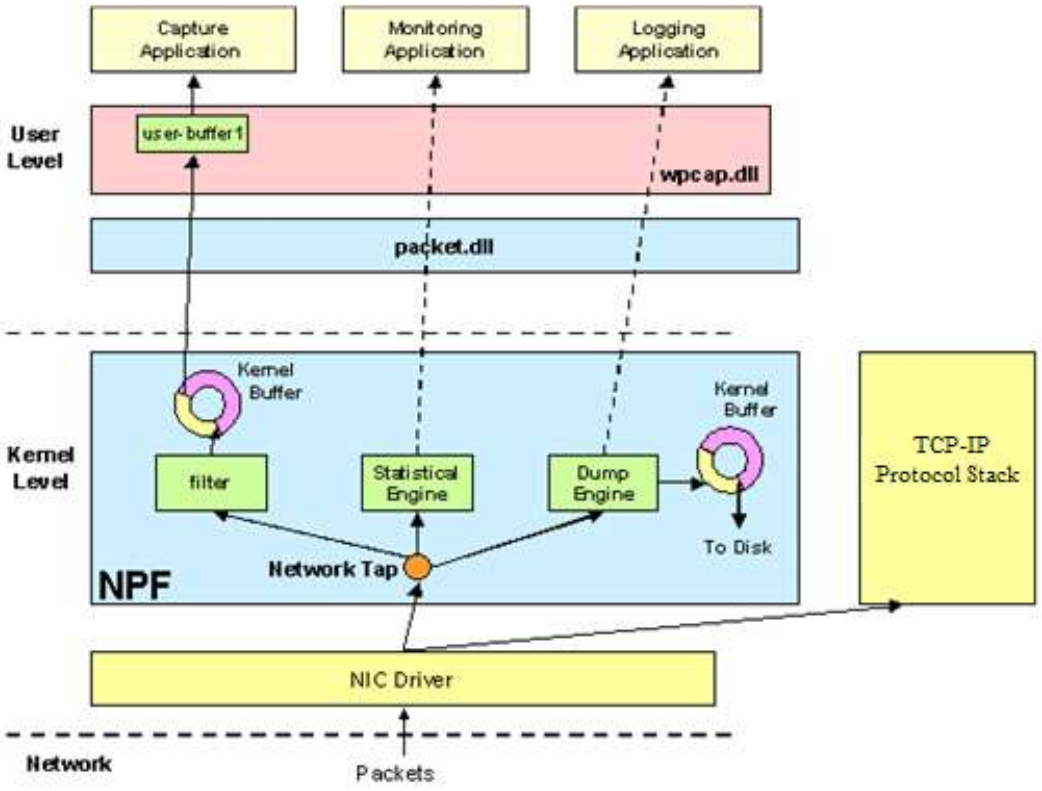


Slika 2 – TCP–IP protokol stek

Hvatanje paketa korišćenjem WinPcap biblioteke

Kada koristimo WinPcap, paketi prolaze kroz slično procesiranje, ali sa jednom razlikom: rukovalac mrežne kartice takođe šalje kopiju presretenog paketa delu kernela koji se naziva **Network Packet Filter** (NPF). WinPcap omogućava mrežnoj kartici da radi u slobodnom (engl. *Promiscuous*) režimu, tj. da prima sve pakete bez obzira na njihovu fizičku adresu odredišta.

Filter paketa odlučuje da li će uhvaćeni paket biti prosleđen aplikaciji, na osnovu izraza filtriranja koji mu je aplikacija prethodno zadala. Na primer, aplikacija od filtera može zatražiti da joj prosledi samo pakete koji koriste UDP transportni protokol. Uobičajno je da mnogo više paketa bude odbačeno nego prosleđeno aplikaciji. Korišćenje filtera direktno pozitivno utiče na performanse aplikacije.



Slika 3 – Network Packet Filter

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 u memoriji 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, ekstremno 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.

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

1.2. Izgradnja i slanje paketa

WinPcap omogućava da se preko mrežne kartice pošalju sirovi paketi koje je korisnik u svojoj aplikaciji kreirao. 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 `ioctlsocket` (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.

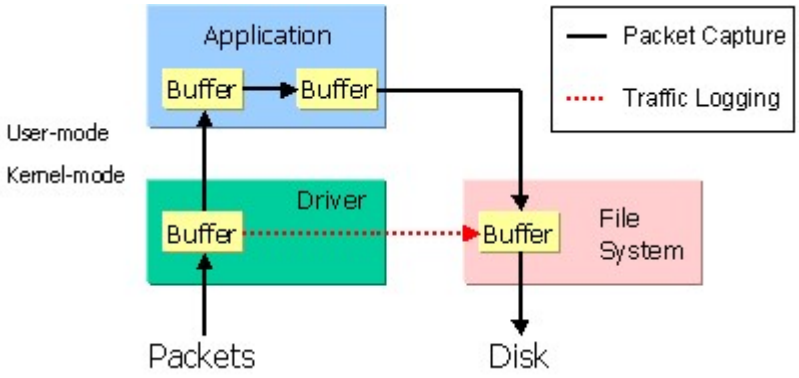
1.3 Praćenje statistike

WinPcap pruža mogućnost praćenja statistike saobraćaja na mreži korišćenjem 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.
koji su zadovoljili uslov i njihovoj ukupnoj veličini u bajtima. Period dobavljanja statistike kao I izrazi koji se koriste u filterima su konfigurabilni.

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.

1.4 Rad sa datotekama

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



Slika 4 – Zapisivanje uhvaćenih paketa na disk

Na slici 4 je (sa crnim strelicama) prikazan tradicionalni način koji zahteva korišćenje 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, korišćenjem kernela za istu namenu isti posao obavlja se korišćenjem dva bafera i samo jednog sistemskog poziva, čime se postižu znatno bolje performanse. Za čuvanje presretenih paketa koristi se `.pcap`ekstenzija, koju je moguće otvoriti pomoću WinPcap biblioteke ili aplikacije Wireshark.

2. Mrežni adapteri

2.1. Dobavljanje liste mrežnih adaptera

Na početku bilo koje aplikacije bazirane na WinPcap biblioteci potrebno je pribaviti listu dostupnih mrežnih adaptera. Za ovu svrhu koristi se funkcija `pcap_findalldevs`koja vraća listu `pcap_if_t`struktura, koje sadrže detaljne informacije o priključenim adapterima.

```
int pcap_findalldevs (pcap_if_t **devices, char *errorMsg);
```

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

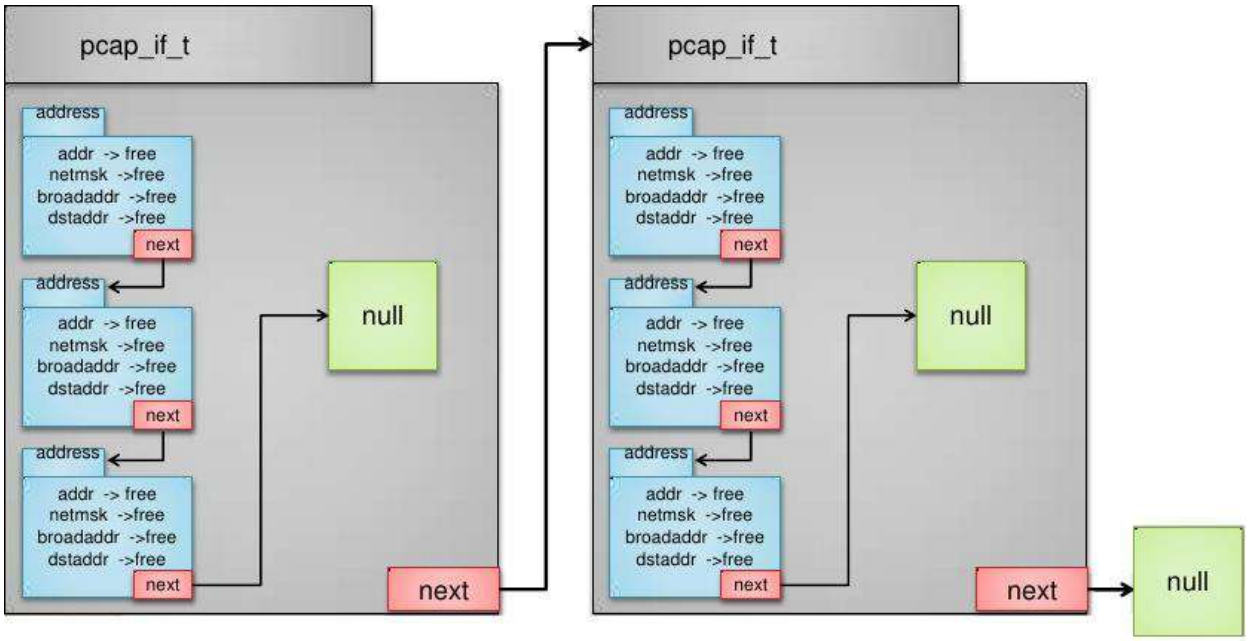
U nastavku su dati deklaracija i opis `pcap_if_t`strukture.

```
typedefstruct 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
`unsignedint 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 5 - Lista adaptera i njihovih adresa

```
struct pcap_addr {
      struct pcap_addr *next;
```

```
    struct sockaddr *addr;
    struct sockaddr *netmask;
    struct sockaddr *broadaddr;
    struct sockaddr *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 sockaddr *addr`	Pokazivač na adresu interfejsa
`struct sockaddr *netmask`	Pokazivač na subnet masku. *NULL vrednost označava da subnet maska nije konfigurisana.*
`struct sockaddr *broadaddr`	Pokazivač na broadcast adresu. *NULL vrednost označava da interfejs ne podržava broadcast.*
`struct sockaddr *dstaddr`	Pokazivač na odredišnu adresu. *NULL vrednost označava da interfejs nije point-to-point interfejs.*

2.2. Oslobađanje liste mrežnih adaptera

Nakon završetka rada sa listom svih mrežnih adaptera, potrebno je osloboditi ovu listu. Za tu namenu se na kraju svake 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`	`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;
}

// Pick one device from the list
int device_number;
printf("Enter the interface number (1-%d):",i);
scanf_s("%d", &device_number);

if (device_number < 1 || device_number > i)
{
	printf("\nInterface number out of range.\n");
	return NULL;
}

// Jump to the selected device
for (device=devices, i=0; i< device_number-1 ; device=device->next, i++);

// ...DO SOMETHING

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

Zadatak

U prilogu vežbe dat je program koji izlistava mrežne kartice koje se nalaze na računaru. Za svaku mrežnu karticu tipa IPv4 potrebno je ispisati logičku adresu, subnet masku i broadcast adresu. U slučaju da adresa pripada IPv6 protokolu, dovoljno je ispisati samo tip adrese.

U nastavku je dat primer kako bi trebalo da izgleda aplikacija.

```
============================ Interface ==================================

Name:           \Device\NPF_{5D0F12BE-E2F6-41AA-8041-1B248E0DA7A0}
Description:    Microsoft
Loopback:       no

ADDRESS
# Address Type:         IPv6

ADDRESS
# Address Type:         IPv4
# Address:              192.168.1.111
# Subnet mask:          255.255.255.0
# Broadcast Address:    255.255.255.255

============================ Interface ==================================

Name:           \Device\NPF_{832BA2CE-EB25-4ADC-986E-703E01554A67}
Description:    Microsoft
Loopback:       no

ADDRESS
# Address Type:         IPv6

ADDRESS
# Address Type:         IPv6
```

Slika 6 – Primer ispisa liste mrežnih kartica sa adresnim podacima