Vežba 9 – Slanje paketa upotrebom Libpcap (Raspbian) –WinPcap (Windows) programske biblioteke

1. Uvod

WinPcap/libpcap omogućava da se preko mrežne kartice pošalju sirovi paketi koje je korisnik kreirao u svojoj aplikaciji. Zbog toga što se paket direktno prosleđuje mrežnoj kartici (izbegavanjem bilo kakve dodatne enkapsulacije) sama aplikacija se mora postarati za kreiranje svih potrebnih zaglavlja. Na primer, ako bismo želeli poslati UDP datagram 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 mreže izuzev fizičkog sloja i kontrolne sume okvira (FCS) na nivou veze, koju računa sama mrežna kartica.

WinPcap/libpcap korisniku pruža dve mogućnosti slanja paketa u mrežu. To su slanje pojedinačnog paketa i slanje reda (niza) paketa.

2. Slanje pojedinačnog paketa

Posle otvaranja adaptera, poziva se funkcija *pcap_sendpacket()* da bi se poslao „ručno pravljeni“ paket. Ova funkcija ima tri parametra:

1. Adapter koji će poslati paket
2. Bafer koji sadrži podatke koji će se slati
3. Dužina bafera.

Treba napomenuti da se paket šalje na mrežu onakav kakvog ga korisnik pripremi. Nikakvih dodatnih manipulacija od strane biblioteke ili mreže nema. Ovo znači da korisnička aplikacija mora sama da kreira i doda ispravna zaglavlja svih protokola kako bi se poslali, odnosno na prijemnoj strani primili smisleni podaci.

```
int pcap_sendpacket (pcap_t* device_handle, unsigned char * packet_data, int
packet_size);
```

Funkcija:	Opis:
pcap_sendpacket	Funkcija za slanje sirovih podataka (paketa pravljenih od strane korisnika).
Parametri:	**Opis:**
pcap_t* device_handle	Specificira adapter sa koga će se podaci slati.
unsigned char * packet_data	Bafer koji sadrži podatke koji će se slati (uključujući i različita zaglavlja protokola). Unutar zaglavlja na nivou linka ne treba uključiti FCS polje, jer se ono izračunava i dodaje od strane drajvera mrežnog adaptera.
int packet_size	Veličina bafera packet_data, odnosno veličina paketa koji se šalje.
Povratna vrednost:	**Opis:**
int	Ako se uspešno pošalje paket vraća 0. Ukoliko se desi greška, vraća -1.

Primer:

```
/*-------------------------------------------------------------------------*/
int main()
{
	pcap_t* device_handle;
	char error_buffer [PCAP_ERRBUF_SIZE];
	unsigned char packet[100];
	int i;

	// Open the device for sending
	if ((device_handle = pcap_open_live( device->name, // name of the device
				100, // portion of the packet
				1, // promiscuous mode
				2000, // read timeout
				error_buffer // error message
	   )) == NULL)
	{
		printf("\nUnable to open the adapter. %s is not supported by WinPcap\n",
			device->name);
		pcap_freealldevs(devices);
		return -1;
	}

	// Supposing to be on Ethernet, set MAC destination address to 1:1:1:1:1:1
	packet[0]=1;
	packet[1]=1;
	packet[2]=1;
	packet[3]=1;
	packet[4]=1;
	packet[5]=1;

	// Set MAC source address to 2:2:2:2:2:2
	packet[6]=2;
	packet[7]=2;
	packet[8]=2;
	packet[9]=2;
	packet[10]=2;
	packet[11]=2;
```

```
        // Fill the rest of the packet
        for(i=12;i<100;i++)
        {
                packet[i]=(unsigned char)i;
        }

        //Send down the packet
        if (pcap_sendpacket(device_handle, packet, 100) != 0)
        {
                printf("\n Error sending the packet: %s\n", pcap_geterr(device_handle));
                return -1;
        }
}
 /*-------------------------------------------------------------------------*/
```

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.

3. Slanje niza paketa – Podržano samo u WinPcap-u!

Dok funkcija *pcap_sendpacket()* omogućava jednostavno i brzo slanje pojedinačnog paketa, ona ne podržava red za slanje koji nudi unapređen i optimizovan način da se pošalje skup paketa. Red za slanje je kontejner za promenljiv broj paketa koji će se poslati kroz mrežu. Red za slanje je određen svojom veličinom koja predstavlja maksimalnu količinu bajta koju on može da primi.

Postupak za kreiranje i slanje niza paketa, sastoji se od sledećih koraka:

1. Kreiranje reda za slanje
2. Dodavanje paketa u kreirani red za slanje
3. Isporuka reda za slanje
4. Brisanje reda.

3.1 Kreiranje reda za slanje - Podržano samo u WinPcap-u!

Red za slanje se kreira pozivom *pcap_sendqueue_alloc()* funkcije, pri čemu se navodi veličina novog reda za slanje.

```
pcap_send_queue* pcap_sendqueue_alloc (unsigned int queue_size);
```

Funkcija:	Opis:
pcap_sendqueue_alloc	Funkcija za alociranje reda paketa za slanje.
Parametri:	**Opis:**
unsigned int queue_size	Veličina reda u bajtima. Određuje maksimalnu količinu podataka koje će red sadržati.
Povratna vrednost:	**Opis:**
pcap_send_queue*	Vraća alocirani red za slanje u okviru strukture pcap_send_queue.

Struktura *pcap_send_queue* služi za smeštanje reda sirovih paketa koji će se poslati na mrežu pomoću *pcap_sendqueue_transmit()*. Sledi opis polja strukture *pcap_send_queue*.

```
struct pcap_send_queue
{
	unsigned int maxlen;
	unsigned int len;
	char* buffer;
};
```

Sledi opis polja strukture *pcap_send_queue*.

Polje:	Značenje polja:
unsigned int maxlen	Maksimalna veličina reda u bajtima. Predstavlja veličinu polja buffer.
unsigned int len	Tekuća veličina reda u bajtima.
char* buffer	Bafer koji sadrži pakete koji će se poslati.

3.2 Dodavanje paketa u red za slanje - Podržano samo u WinPcap-u!

Nakon kreiranja reda, funkcija *pcap_sendqueue_queue()* se koristi da bi dodali paket u red za slanje. Ova funkcija uzima generičko zaglavlje (*pcap_pkthdr* struktura sa vremenskom oznakom (timestamp) i dužinom paketa) i bafer u kome su podaci paketa. Ovi parametri su isti kao parametri koje nam vrate funkcije *pcap_next_ex()* i *packet_handler()*. Iz tog razloga, stavljanje u red paketa (koji je uhvaćen sa mreže ili pročitan iz fajla) se svodi na prosleđivanje ovih parametara funkciji *pcap_sendqueue_queue()*.

```
int pcap_sendqueue_queue (pcap_send_queue* queue, const struct pcap_pkthdr*
packet_header, const unsigned char *packet_data);
```

Funkcija:	Opis:
pcap_sendqueue_queue	Funkcija za dodavanje paketa u red za slanje.
Parametri:	Opis:
pcap_send_queue* queue	Pokazivač na kraj reda za slanje gde će se dodati novi paket.
const struct pcap_pkthdr* packet_header	Pokazivač na pcap_pkthdr strukturu (generičko zaglavlje) koja sadrži vremensku oznaku i dužinu paketa.
const unsigned char *packet_data	Pokazivač na bafer koji sadrži podatke paketa.
Povratna vrednost:	Opis:
int	Ako se uspešno izvrši vraća 0. U suprotnom, vraća -1.

3.3 Isporuka reda za slanje - Podržano samo u WinPcap-u!

Za isporuku reda za slanje koristi se funkcija *pcap_sendqueue_transmit()*. Ova funkcija je znatno efikasnija nego niz poziva funkcije *pcap_sendpacket()*, zato što je u prvom slučaju red

za slanje baferovan na nivou kernela što drastično smanjuje broj konteksnih skokova (context switches). Ako se treći parametar *sync* postavi na nenultu vrednost onda će slanje paketa biti sinhronizovano, odnosno zavisiće od vremenskih oznaka na svakom paketu. Takvo slanje je izuzetno zahtevno po pitanju procesorskog vremena, ali rezultuje u visokoj preciznosti prenosa paketa.

```
unsigned int pcap_sendqueue_transmit (pcap_t* device_handle,
pcap_send_queue* queue, int sync);
```

Funkcija:	Opis:
pcap_sendqueue_transmit	Funkcija za slanje reda sirovih paketa na mrežu.
Parametri:	Opis:
pcap_t* device_handle	Pokazivač na adapter sa koga će se paketi slati.
pcap_send_queue* queue	Pokazivač na *pcap_send_queue* strukturu koja sadrži pakete za slanje.
int sync	Određuje da li operacija slanja mora biti sinhronizovana. Ako ima nenultu vrednost, paketi se šalju u skladu sa svojim vremenskim oznakama (timestamp). Ako ima nultu vrednost, paketi se šalju što pre je moguće.
Povratna vrednost:	Opis:
unsigned int	Povratna vrednost označava količinu bajta koji su poslati. Ako je ova brojka manja od veličine reda, onda je došlo do greške prilikom slanja.

Ukoliko se paketi za slanje reda paketa uzimaju iz ranije snimljenog fajla, prvo je potrebno otvoriti fajl sa *pcap_open_offline()*, zatim smestiti paket po paket iz fajla u prethodno alocirani red za slanje. Na kraju se vrši slanje reda.

Napomena: potrebno je proveriti protokol na sloju za pristup mreži za pakete iz fajla sa protokolom za pristup mreži za adapter koji će da šalje te pakete. Vrlo je važno da oni budu isti, jer u suprotnom slanje takvih paketa ne bi imalo smisla.

Nakon slanja celog reda, možemo obrisati red i osloboditi alociranu memoriju.

3.4 Brisanje reda - Podržano samo u WinPcap-u!

Kada nam red za slanje više ne treba možemo ga obrisati pomoću *pcap_sendqueue_destroy()* koja oslobađa sve bafere pridružene datom redu za slanje.

```
void pcap_sendqueue_destroy (pcap_send_queue* queue);
```

Funkcija:	Opis:
pcap_sendqueue_destroy	Funkcija za brisanje reda za slanje i oslobađanje svih bafera koji su mu pridruženi.
Parametri:	Opis:
pcap_send_queue* queue	Red za slanje koji se briše
Povratna vrednost:	Opis:
void	Nema povratne vrednosti.

Na kraju programa potrebno je pozvati funkciju za zatvaranje adaptera preko koga su se podaci slali. Tako se obezbeđuje garancija da će svi paketi zaista biti poslati sa datog adaptera.

```
void pcap_close (pcap_t* device_handle);
```

Funkcija:	Opis:
pcap_close	Funkcija za zatvaranje adaptera (ili fajla) i oslobađanje svi resursa koji su mu pridruženi.
Parametri:	**Opis:**
pcap_t* device_handle	Deskriptor adaptera (ili falja) koji treba zatvoriti.
Povratna vrednost:	**Opis:**
void	Nema povratne vrednosti.

Primer slanja reda paketa iz *.pcap* fajla preko izabranog mrežnog adaptera:

```
/*-------------------------------------------------------------------------*/
int main()
{
	pcap_if_t* devices;
	pcap_if_t* device;
	pcap_t* device_handle_in, *device_handle_out;
	char error_buffer [PCAP_ERRBUF_SIZE];
	int i=0;
	int device_number;
	int sync=0;
	int sentBytes;

	//Retrieve the device list on the local machine
	if (pcap_findalldevs(&devices, error_buffer) == -1)
	{
		printf("Error in pcap_findalldevs: %s\n", error_buffer);
		return -1;
	}

	// Print the list
	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");
	}

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

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

	if(device_number < 1 || device_number > i)
	{
```

```
		printf("\nInterface number out of range.\n");
		return -1;
	}

	// Select the first device...
	device=devices;
	// ...and then jump to chosen devices
	for (i=0; i<device_number-1; i++)
	{
		device=device->next;
	}

	// Open the capture file
	if ((device_handle_in = pcap_open_offline("example.pcap", // File name
							error_buffer // Error buffer
	   )) == NULL)
	{
		printf("\n Unable to open the file %s.\n", "example.pcap");
		return -1;
	}

	// Open the output adapter
	if ((device_handle_out = pcap_open_live(device->name, 65536, 1, 1000,
		error_buffer)) == NULL)
	{
		printf("\n Unable to open adapter %s.\n", device->name);
		return -1;
	}

	// Check the link layer. We support only Ethernet for simplicity.
	if(pcap_datalink(device_handle_in) != DLT_EN10MB ||
		pcap_datalink(device_handle_out) != DLT_EN10MB)
	{
		printf("\nThis program works only on Ethernet networks.\n");
		return -1;
	}

	// Allocate a send queue
	pcap_send_queue* queue = pcap_sendqueue_alloc(1024*1024); // 1 MB

	// Fill the queue with the packets from the file
	pcap_loop(device_handle_in, 0, packet_handler, (unsigned char*)queue);

	// Transmit the queue
	// ...parameter "sync" tells if the timestamps must be respected
	// sync=1 (true) or sync=0 (false)
	if ((sentBytes = pcap_sendqueue_transmit(device_handle_out, queue, sync))
		< queue->len)
	{
		printf("An error occurred sending the packets: %s. Only %d bytes were
		sent\n", pcap_geterr(device_handle_out), sentBytes);
	}

	// Free the send queue
	pcap_sendqueue_destroy(queue);

	// !!! IMPORTANT: remember to close the output adapter
	// otherwise there will be no guarantee that all the packets will be sent!
	pcap_close(device_handle_in);
	pcap_close(device_handle_out);
	return 0;
}
```

```
// Callback function invoked by WinPcap for every incoming packet
void packet_handler(unsigned char* queue, const struct pcap_pkthdr* packet_header,
	const unsigned char* packet_data)
{
	// Add current packet in the queue
	if (pcap_sendqueue_queue((pcap_send_queue*)queue, packet_header, packet_data)
		== -1)
	{
		printf("Warning: packet buffer too small, not all the packets will be
		sent.\n");
	}
}
/*-------------------------------------------------------------------------------*/
```

ZADATAK

U prilogu materijala vežbe dat je jedan primer implementacije slanja niza poruka iz *pcap* datoteke koristeći red za slanje.

1. Koristeći aplikaciju Wireshark uhvatiti određen broj paketa u datoteku i sačuvati je pod nazivom *example.pcap*. Ovu datoteku potrebno je nadalje koristiti kao ulaznu datoteku aplikacije.
2. Omogućiti pojedinačno slanje svih ARP paketa koji se nalaze u datoteci. (**1 bod**)

   Specifično za WinPcap:
3. Omogućiti da se pored postojeće implementacije slanja reda za slanje UDP paketa i TCP paketi šalju koristeći zaseban red za slanje.
   - Veličinu TCP reda podesiti na 512 kB. TCP pakete slati u skladu sa svojim vremenskim oznakama (veličina UDP reda je podešena na 256 kB i UDP paketi se šalju što je brže moguće). (**2 boda**)
   - Prvo je potrebno poslati sve UDP pakete iz datoteke, pa tek onda sve TCP pakete.