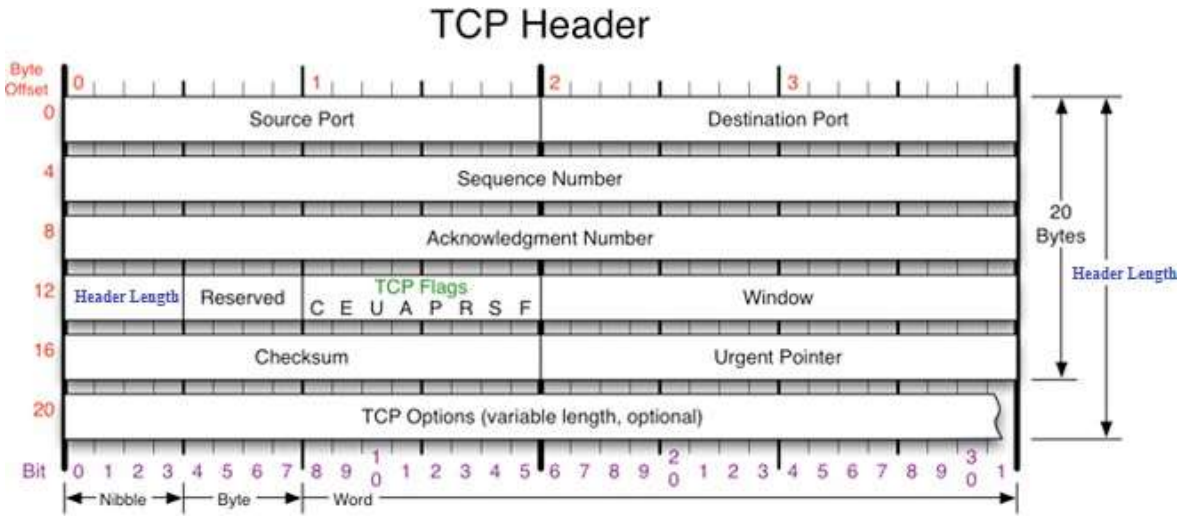
Vežba 11 – Interpretacija sadržaja paketa i rad sa datotekama (2. deo)

1. Zaglavlja protokola iz TCP/IP modela

1.1. Opis TCP zaglavlja

TCP segment se sastoji iz TCP zaglavlja i podataka koji su dobijeni od nekog aplikacionog protokola (npr. HTTP, SMTP ili FTP). Izgled TCP zaglavlja je dat na slici 1.



Slika 1. TCP zaglavlje

TCP zaglavlje se sastoji iz:

- ***Source port*** – port koji identifikuje aplikaciju na računaru pošiljaoca poruke.
- ***Destination port*** – port (prolaz) koji identifikuje aplikaciju na računaru primaoca poruke.
- ***Sequence number*** – Redni broj prvog okteta (bajta) podataka u segmentu (osim ukoliko je SYN postavljen). Kada je SYN prisutan, broj sekvence koji sledi je početni broj sekvence (ISN – Initial Sequence Number) i prvi oktet podataka ima vrednost ISN+1.
- ***Acknowledgment number*** – Redni broj narednog bajta koji jedna strana u ostvarenoj TCP vezi očekuje da primi od druge. Ovim mehanizmom se drugoj strani potvrđuje ispravan prijem svih prethodno poslatih bajtova do rednog broja ACK-1

- ***Header length*** – dužina zaglavlja TCP segmenta izražena u umnošcima 32-bitnih reči. Zbog polja "Options" dužina zaglavlja je promenljiva, pa nam ovaj podatak ukazuje kolika je stvrana dužina TCP zaglavlja.
- ***Reserved*** – rezervisana polja za buduću upotrebu (4 bita)
- *URG, ACK, PSH, RST, SYN, FIN* – kontrolni biti. (8 bita)
- ***Window*** – Broj okteta koje prijemna strana još može primiti. Ovo polje govori predajnoj strani da može slati segmente sve dok ukupni broj okteta koje treba poslati nije veći od broja okteta upisanih u polje "prozor". Kada je veličina prozora jednaka 0, predajna strana treba prekinuti slanje podataka dok ne dobije segment u kojem je veličina prozora veća od nule.
- ***Checksum*** – kontrolna suma za proveru bitskih grešaka.
- ***Urgent pointer*** – pokazivač prioriteta odnosno važnosti poruke koja se šalje. Ukazuje na broj sekvence okteta u kojem su hitni podaci. Može se interpretirati samo u segmentima za koje je URG upravljački bit postavljen.
- ***Options*** – opciona informacija.
- ***Data*** – Podaci koji se šalju (ako postoji opciona informacija, podaci počinju na 192. bitu, inače od 160. bita). Ovo su podaci dobijeni od protokola aplikacionog sloja.

Značenje određenih bita u polju kontrolnih bita:

- *URG* - polje urgentnog pokazivača je važeće.
- *ACK* - polje potvrde je važeće.
- *PSH* - ovaj segment zahteva operaciju potiskivanja ("push").
- *RST* - resetuj vezu.
- *SYN* - sinhronizuj brojeve sekvenci.
- *FIN* - pošiljalac je došao do kraja toka podataka.

U nastavku je dat izgled strukture koja omogućava pristup poljima zaglavlja TCP protokola. Pošto veličina opcionih informacija varira, TCP zaglavlje nema konstantnu veličinu. Iz tog razloga do mesta u memoriji gde su smešteni aplikativni podaci se može doći korišćenjem veličine TCP zaglavlja koja je zapisana u okviru polja `header_length`. Vrednost upisanu u polje `header_length` treba pomnožiti sa 4 da bi dobili dužinu TCP zaglavlja izraženu u bajtima.

```
// TCP header
typedef struct tcp_header {
   unsigned short src_port;            // Source port
   unsigned short dest_port;           // Destination port
   unsigned int sequence_num;          // Sequence Number
   unsigned int ack_num;               // Acknowledgement number
   unsigned char reserved :4;          // Reserved for future use (4 bits)
   unsigned char header_length :4;     // Header length (4 bits)
   unsigned char flags;                // Packet flags
   unsigned short windows_size;        // Window size
   unsigned short checksum;            // Header Checksum
   unsigned short urgent_pointer;      // Urgent pointer
   // + Option bytes
} tcp_header;
```

2. Rad sa datotekama - Snimanje paketa u datoteku

2.1. Otvaranje datoteke

Da bi sačuvali pakete koje hvatamo sa mreže, prvo je potrebno otvoriti datoteku u kojoj ćemo čuvati te pakete. Za tu namenu služi funkcija `pcap_dump_open()`. Ova funkcija osim što otvara fajl vrši i njegovo povezivanje sa adapterom sa koga se hvataju paketi. Naravno, neophodno je da pre poziva ove funkcije adapter bude otvoren za hvatanje paketa.

Format sačuvanih fajlova je libpcap. Ovaj format obezbeđuje čuvanje sadržaja uhvaćenih paketa u binarnom obliku. On se standardno koristi od strane mnogih mrežnih alata uključujući i WinDump, Ethereal i Snort.

```
pcap_dumper_t* pcap_dump_open (pcap_t* device_handle, const char*
      filename);
```

Funkcija:	Opis:
`pcap_dump_open`	Funkcija za otvaranje fajla u koji će se sačuvati uhvaćeni paketi
Parametri:	Opis:
`pcap_t* device_handle`	Deskriptor adaptera koji je otvoren za hvatanje paketa na mreži.
`const char* filename`	Ime fajla koji se otvara.
Povratna vrednost	Opis
`pcap_dumper_t*`	Ako se uspešno izvrši, vraća pokazivač na deskriptor fajla u koji se čuvaju paketi. Vraća NULL ako se desi greška. U tom slučaju može se pozvati `pcap_geterr()` radi ispisa greške.

2.2. Upis paketa u datoteku

Upis paketa u datoteku se vrši pomoću funkcije `pcap_dump()` koja se izvršava pri pozivu callback funkcije `pcap_handler()`. Parametri `pcap_dump()` funkcije u potpunosti odgovaraju parametrima `pcap_handler()` funkcije (preslikavanje je 1-1).

```
void pcap_dump(unsigned char* user, const struct pcap_pkthdr* packet_header,
     const unsigned char* packet_data);
```

Funkcija:	Opis:
`pcap_dump`	Funkcija za čuvanje paketa u fajl.
Parametri:	Opis:
`unsigned char* user`	Za ovaj parametar se prosleđuje parametar tipa `pcap_dumper_t` koji je vratila funkcija `pcap_dump_open`. Naravno, potrebno je da se prethodno kastuje u pokazivač tipa `(unsigned char*)`.
`const struct pcap_pkthdr* packet_header`	Pokazivač na generičko zaglavlje koje drajver za hvatanje paketa prikači na svaki uhvaćeni paket
`const unsigned char* packet_data`	Pokazivač na početak podataka u paketu, uključujući i zaglavlja protokola
Povratna vrednost	Opis
`void`	Nema povratne vrednosti

Da bi se ispravno snimili podaci i sam fajl zatvorio, potrebno je nakon snimanja u fajl zatvoriti adapter sa funkcijom `pcap_close()`.

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

Funkcija:	Opis:
`pcap_close`	Funkcija za zatvaranje fajlova koji su pridruženi deskriptoru adaptera `device_handle` i oslobađanje zauzetih memorijskih resursa.
Parametri:	Opis:
`pcap_t* device_handle`	Deskriptor adaptera sa koga se hvataju/čitaju paketi
Povratna vrednost	Opis
`void`	Nema povratne vrednosti

Primer:

```
int main()
{
   pcap_t* device_handle;

   ...

   // Open the dump file
   pcap_dumper_t* file_dumper = pcap_dump_open(device_handle, "example.pcap");

   if (file_dumper == NULL)
   {
      printf("\n Error opening output file\n");
      return -1;
   }

   ...

   // Start the capture
   pcap_loop(device_handle, 10, packet_handler, (unsigned char*) file_dumper);

   // Close the file associated with device_handle and deallocates resources
   pcap_close(device_handle);

   return 0;
}

// Callback function invoked by libpcap for every incoming packet
void packet_handler(unsigned char* fd, const struct pcap_pkthdr *
       packet_header, const unsigned char *packet_data)
{
    // Save the packet on the dump file
    pcap_dump(fd, packet_header, packet_data);
}
```

2.3. Čitanje paketa iz datoteke

Kada imamo sačuvane pakete unutar fajla, potrebno nam je i da znamo kako da iščitamo sadržaj *.pcap fajla. Za otvaranje fajla služi nam funkcija `pcap_open_offline()`, a zatim se sekvencijalno iščitavanje paketa vrši pomoću `pcap_loop()` ili `pcap_next_ex()`. Ovaj postupak za iščitavanje paketa iz fajla je veoma sličan hvatanju paketa sa mreže. Pre poziva funkcije za iščitavanje može se postaviti filter (pomoću funkcije `pcap_setfilter()`) kako bi se definisao željeni podskup mrežnog saobraćaja koji je potrebno čitatit iz fajla.

```
pcap_t* pcap_open_offline (const char* filename, char* error_buffer);
```

Funkcija:	Opis:
`pcap_open_offline`	Funkcija za otvaranje fajla radi čitanja paketa.
Parametri:	Opis:
`const char* filename`	Specificira ime fajla koji se otvara.
`char* error_buffer`	String koji sadrži tekst greške ukoliko se ona desi. Ima sadržaj samo ukoliko se funkcija ne izvrši uspešno (vrati NULL pokazivač)
Povratna vrednost	Opis
`pcap_t*`	Vraća deskriptor otvorenog fajla iz koga se čitaju paketi. Ukoliko se desi greška, vraća NULL.

.

Primer:

```
int main()
{
	pcap_t* device_handle;
	char error_buffer [PCAP_ERRBUF_SIZE];

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

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

	// Read and dispatch packets until EOF is reached
	pcap_loop(device_handle, 10, dispatcher_handler, NULL);

	// Close the file associated with device_handle and deallocates resources
	pcap_close(device_handle);
	return 0;
}


void dispatcher_handler(unsigned char* user, const struct pcap_pkthdr *
	packet_header, const unsigned char* packet_data)
{
	// Print packet timestamp and packet length
	printf("%ld:%ld (%ld)\n", packet_header->ts.tv_sec,
		packet_header->ts.tv_usec,
		packet_header->len);

	// Print the packet
	for (int i=0; (i < packet_ header->len); i++)
	{
		printf("%.2x ", packet_data[i]);
		if ( (i+1) % 16 == 0)
			printf("\n");
	}

	printf("\n\n");

}
```

Zadatak 1

1. Omogućiti odabir jedne od instaliranih mrežnih kartica. Ime odabrane mrežne kartice potrebno je ispisati na ekran.
2. Proveriti da li odabrana Ethernet mrežna kartica. Ukoliko nije potrebno je ugasiti program.
3. Implementirati hvatanje 10 paketa u normalnom režimu rada mrežne kartice.
4. Obezbediti da se korisničkoj aplikaciji prosleđuju samo paketi sa mrežne kartice koji zadovoljavaju sledeće uslove:
    - Paket je poslat sa fizičke adrese računara koji student koristi.
    - Na mrežnom nivou paket sadrži IPv4 protokol.
    - Za transport paketa se mogu koristiti UDP i TCP protokol.
5. Za svaki presreteni paket potrebno je ispisati na ekranu fizičku adresu primaoca i logičku adresu pošiljaoca.
6. Pomoću Wireshark aplikacije otkriti ime polja koje nosi informaciju da li paket sadrži *UDP* ili *TCP* protokol. Nakon toga, za svaki paket potrebno je ispisati da li je za transport korišćen *UDP* ili *TCP*.
7. Ispisati na ekranu port primaoca uhvaćenog paketa.
8. Omogućiti ispis aplikativnih podataka po bajtima korišćenjem heksadecimalnog zapisa.

Zadatak 2

Napisati WinPcap aplikaciju koja omogućava razvrstavanje prethodno uhvaćenih paketa po datotekama u zavisnosti od tipa protokola koji paketi koriste.

1. Učitatati presretene pakete koji se nalaze uskladišteni u datoteci *example.pcap* koja jedata u nastavnim materijalima.
2. Razvrstati pakete prema protokolu koji implementiraju (ARP, ICMP, UDP i TCP)
3. Razvrstane pakete potrebno je sačuvati u zasebne datoteke: *arp_packets.pcap*, *icmp_packets.pcap, udp_packets.pcap i tcp_packets.pcap*