BSM 458-AĞ PROGRAMLAMA

Hafta2: PCAP (Packet Capture)

Dr. Öğr. Üyesi Musa BALTA
Bilgisayar Mühendisliği Bölümü
Bilgisayar ve Bilişim Bilimleri Fakültesi

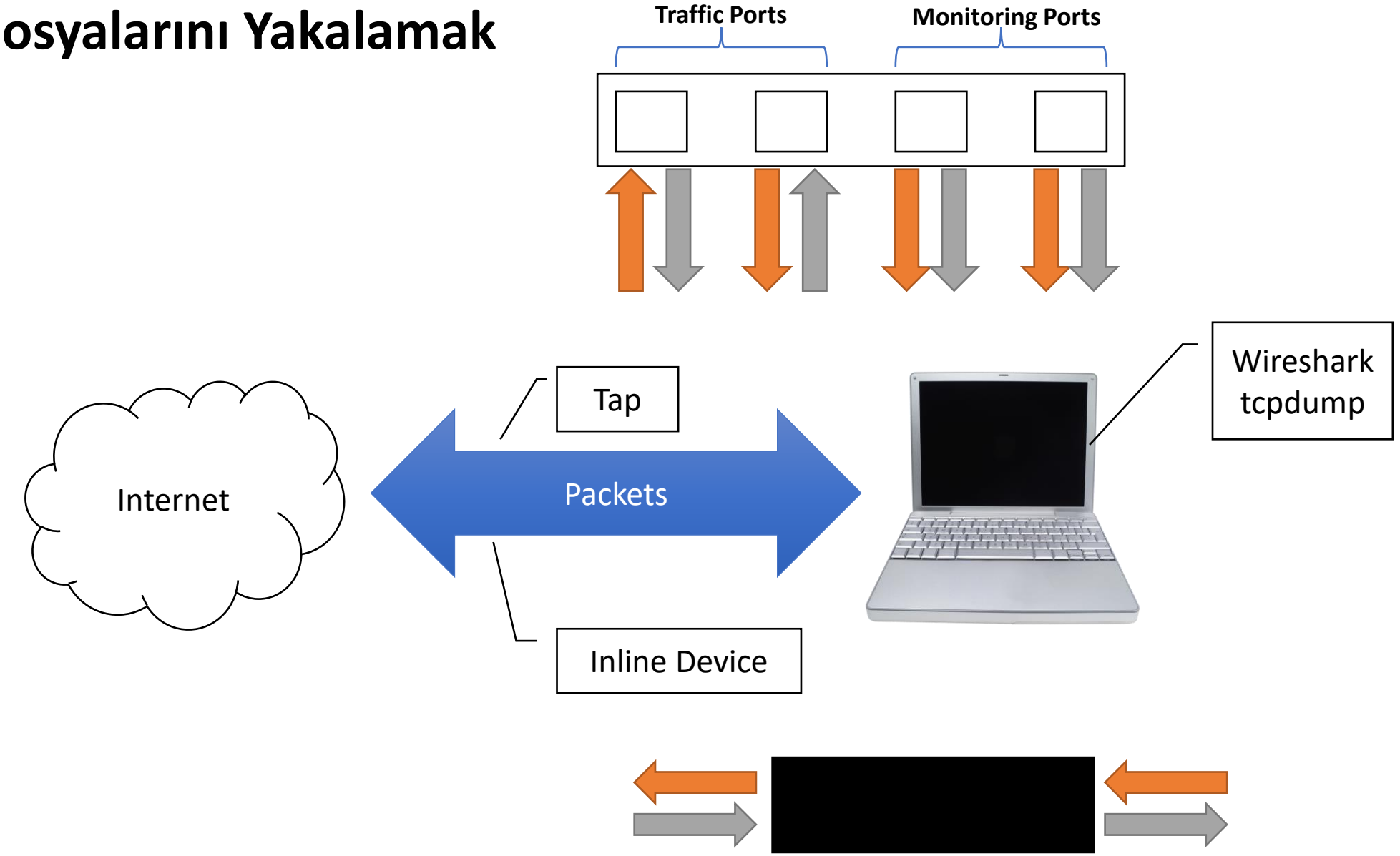
Haftalık İçerik

- PCAP Kavramı
- Libpcap Kavramı
- Ağ Dinleme Araçları
- Libpcap Syntax Yapısı

PCAP Kavramı

- PCAP == **P**acket **Cap**ture
- Ağ aktivitelerin tüm kaydı
  - Katman 2 – 7
- En yaygın kullanım şekli: `libpcap`
  - Açık-kaynak
  - *nix ve Windows sistemlere uyumlu
  - C kütüphanesi ve birçok dile uyumluluk
  - Others proprietary formats not covered
- Kullanım Alanlar;
  - Ağ trafiğini dinlemek
  - Bant genişliği kullanımı kontrol etme
  - Pasif DNS çözümlemesi
  - Saldırı girişimlerini görüntüleme
  - Araştırma çalışmaları için testler

PCAP Dosyalarını Yakalamak



Ham PCAP Depolama

$$1\text{gbps} \times 3600 \times 24 = 86400 \text{ gigabits}$$

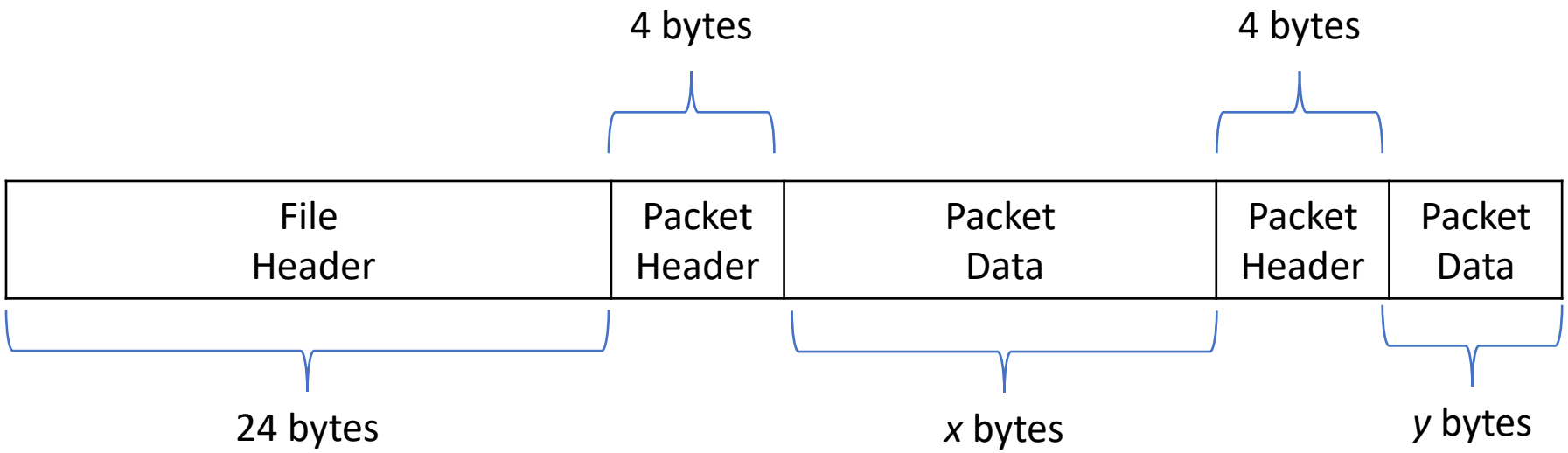
$$86400 \div 8 = 10800 \text{ gigabytes}$$

$$1\text{gbps} \div (64\,\text{bytes} \times 8\,\text{bits}) =$$

$$10^9\,\text{bits/s} \div 512\,\text{bits} = 1953125\text{pps}$$

$$N\,\text{pps} \times H_{\text{PCAP}} = NH\,\text{Bps}$$

libpcap Format



libpcap Overhead

Avg. Packet Size (bytes)	Packets Per Second	Overhead (MB/s)	Overhead (GB / day)
64	1,953,125	7.45	628.64
1500	83,333	0.32	26.82
7981	15,662	0.06	5.04
9000	13,888	0.05	4.47

Ağ Dinleme Araçları

- Ağ üzerindeki trafiği sezebilen, çözebilen ve değiştirebilen donanım ve yazılım araçlarının bir kombinasyonudur
  - Pasif izleme (sezme)
  - Aktif (atak yapma)
- Hem ticari hem de ücretsiz sürümleri bulunmaktadır
- Genelde yazılım tabanlıdır
- Sniffer olarak da bilinirler
  - Ağ üzerindeki veriyi pasif olarak izleyen bir programdır
  - Makineniz üzerinde çalışan uygulamalar ve protokoller tarafından gönderilen ya da alınan paketlerin bir kopyasını alır
- Yaygın kullanılan ağ protokolü analiz programları
  - Wireshark , Ethereal , Windump, Ve diğerleri
- Sniffer programlarını efektif bir şekilde kullanmak için iyi bir ağ bilgisine sahip olmak gerekmektedir.

Ağ Dinleme Araçları (devam)

- Sistem yöneticileri
  - Sistem problemlerini ve performansını anlama
  - Saldırıları tespit etme
  - Uygulama operasyonlarını test etme
- Saldırganlar (Kötü niyetli kişiler)
  - Protokoller üzerinden pasif olarak veri toplar
    - FTP, POP3, IMAP, SMATP, rlogin, HTTP, vb.
    - VoIP data
  - Hedef ağın trafiğini keşfeder
  - Trafik desenini keşfeder
  - Ağ içerisine dahil olur (backdoor teknikleri ile)

Ağ Dinleme Araçları (Wireshark)

- Gerald Combs tarafından Ethereal ismi ile başlatılan bir projedir
- İlk versiyonu 1998 yılında yayınlanmıştır, Wireshark ismi haziran 2006'da verilmiştir
- Paket sniffer uygulamasıdır, dolayısıyla bir ağın haritasını çıkartmak için kullanılamaz
- Fonksiyonelliği tcpdump'a çok benzerdir ve diğer bir çok sniffer ile de uyumludur
- Birçok bilgiyi sıralayan ve filtreleyen özelliklere, komut satırına ve grafik arabirimine sahiptir
- 750'nin üzerinde protokol destekler ve bu protokollerin yapısını gösterir
- Kapsüllemeli bir yapıda görünüm sunar ve anlamlarını yorumlar
- Sadece pcap tarafından desteklenen ağlar üzerinde veri yakalama yapabilir
- Açık kaynak kodludur ve farklı işletim sistemleri üzerinde çalışabilir
- İnternet ortamında bir çok online kaynak mevcuttur
- Pasif bir izleme aracıdır, dolayısıyla ağ verisi üretmez

Ağ Dinleme Araçları (Wireshark-WinPcap&libpcap)

Wireshark – Paketleri Snif etmek için uygulama

WinPcap – Paket yakalamak için açık kaynak kütüphanesi

Operating System – Windows & Unix/Linux

Network Card Drivers – Ethernet/WiFi Kartı

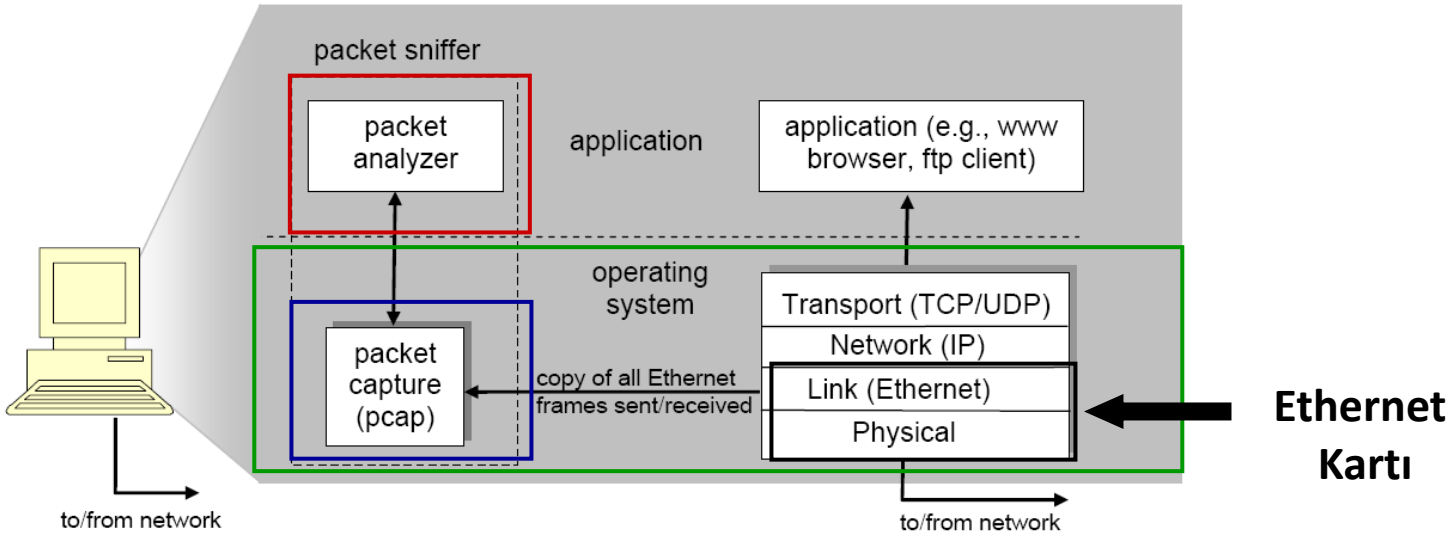
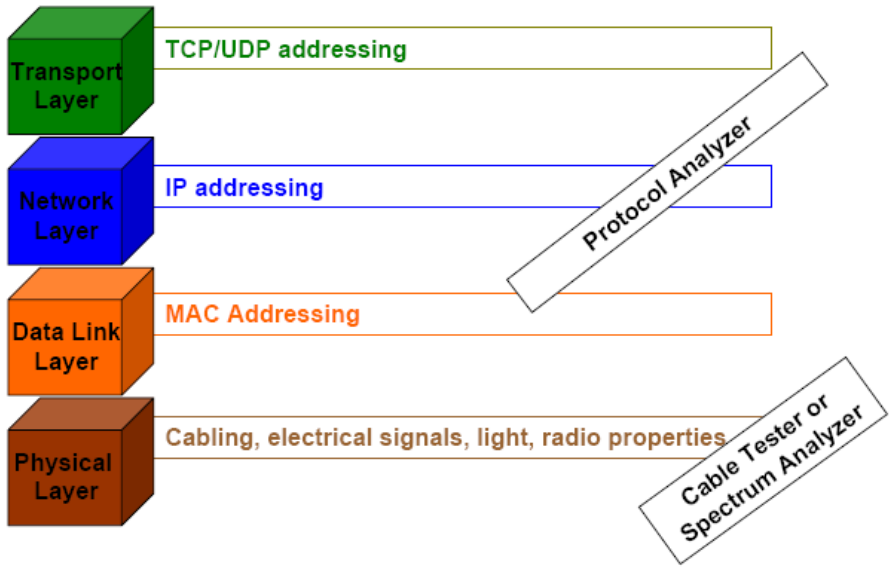
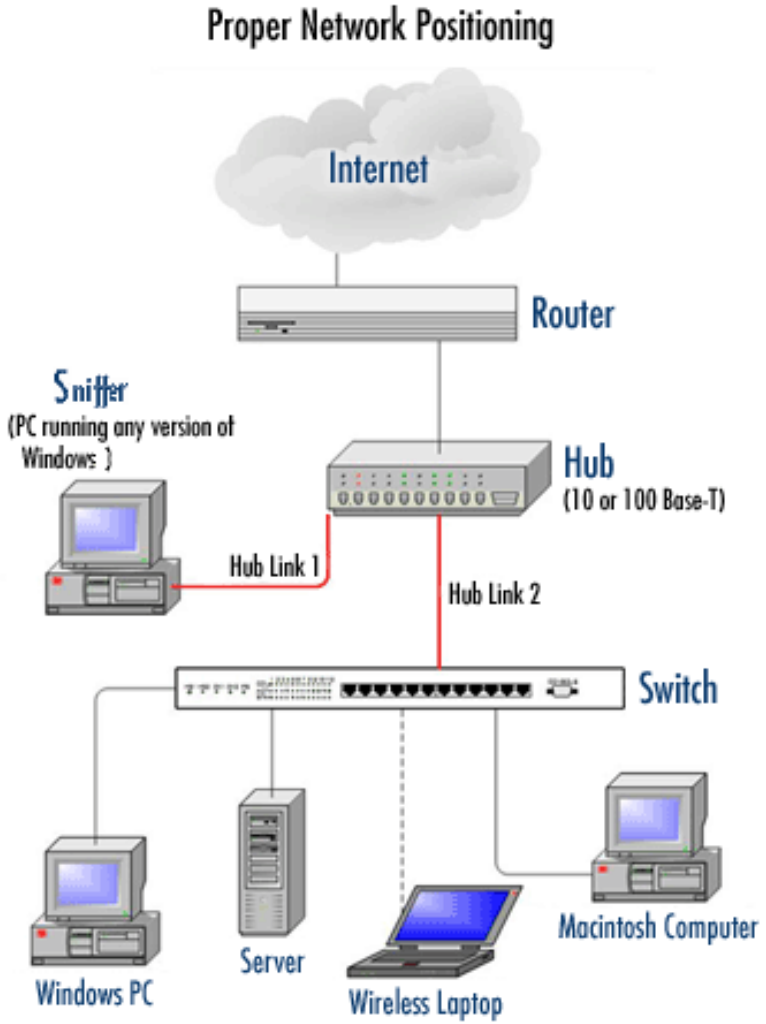


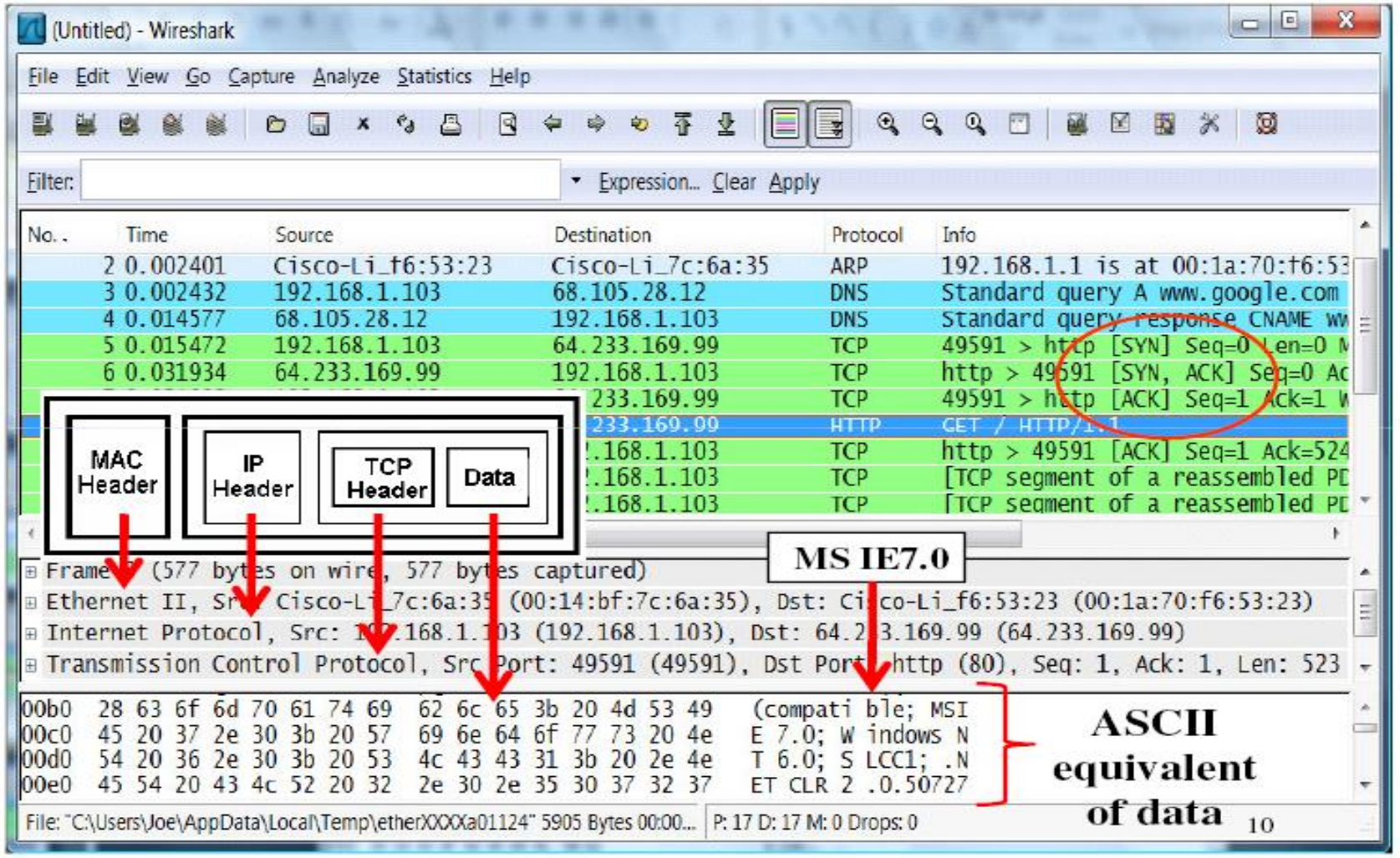
Figure 1: Packet sniffer structure

Ağ Dinleme Araçları (Sniffer ve Topolojideki Yeri)





Ağ Dinleme Araçları (Wireshark Kullanıcı Arayüzü)



Libpcap-Cihaz Ayarlama

```c
#include <stdio.h>
    #include <pcap.h>

    int main(int argc, char *argv[])
    {
        char *dev = argv[1];

        printf("Device: %s\n", dev);
        return(0);
    }
```

```c
#include <stdio.h>
    #include <pcap.h>

    int main(int argc, char *argv[])
    {
        char *dev, errbuf[PCAP_ERRBUF_SIZE];

        dev = pcap_lookupdev(errbuf);
        if (dev == NULL) {
            fprintf(stderr, "Couldn't find default device: %s\n", errbuf);
            return(2);
        }
        printf("Device: %s\n", dev);
        return(0);
    }
```

Libpcap-Cihaz Koklaması

```
pcap_t *pcap_open_live(char *device, int snaplen, int promisc, int to_ms,
char *ebuf)
```

```
#include <pcap.h>
    ...
    pcap_t *handle;

    handle = pcap_open_live(dev, BUFSIZ, 1, 1000, errbuf);
    if (handle == NULL) {
        fprintf(stderr, "Couldn't open device %s: %s\n", dev, errbuf);
        return(2);
    }
```

Libpcap-Paket Filtreleme (Örnek Kod)

```
#include <pcap.h>
...
pcap_t *handle;         /* Session handle */
char dev[] = "rl0";         /* Device to sniff on */
char errbuf[PCAP_ERRBUF_SIZE]; /* Error string */
struct bpf_program fp;      /* The compiled filter expression */
char filter_exp[] = "port 23"; /* The filter expression */
bpf_u_int32 mask;       /* The netmask of our sniffing device */
bpf_u_int32 net;        /* The IP of our sniffing device */

if (pcap_lookupnet(dev, &net, &mask, errbuf) == -1) {
    fprintf(stderr, "Can't get netmask for device %s\n", dev);
    net = 0;
    mask = 0;
}
handle = pcap_open_live(dev, BUFSIZ, 1, 1000, errbuf);
if (handle == NULL) {
    fprintf(stderr, "Couldn't open device %s: %s\n", dev, errbuf);
    return(2);
}
if (pcap_compile(handle, &fp, filter_exp, 0, net) == -1) {
    fprintf(stderr, "Couldn't parse filter %s: %s\n", filter_exp, pcap_geterr(handle));
    return(2);
}
if (pcap_setfilter(handle, &fp) == -1) {
    fprintf(stderr, "Couldn't install filter %s: %s\n", filter_exp, pcap_geterr(handle));
    return(2);
}
```