

## Übung 1.1

Im Quelltext 1.4 haben Sie bereits kennengelernt, wie man mithilfe des `dpkt`-Moduls ein Array in dem Format [Zeitstempel, Paket] erstellt und durch dieses Array iterieren kann. Erstellen Sie ein Skript, das die Anzahl aller Pakete in einer `pcap`-Datei zählt und diese auf der Konsole ausgibt.

Um alle Pakete zu zählen wird ein counter in den Programmcode 1.4. gefügt und hochgezählt wenn ein Paket untersucht wird.

```
def printPcap(pcap):  
    counter = 0  
    for (ts, buf) in pcap:  
        try:  
            eth = dpkt.ethernet.Ethernet(buf)  
            counter += 1
```



Anschließend wird dieser am Ende des Skripts ausgegeben.

```
        src = socket.inet_ntoa(ip.src)  
        dst = socket.inet_ntoa(ip.dst)  
        print '[+] Src: ' + src + ' --> Dst: ' + dst  
    except:  
        pass  
print "Die Mitschnitt beinhaltet %s Pakete" % counter
```

Kommentierter Quelltext:

```
1 # Untersuchung der Kommunikation von digitalen Paketen  
2 import dpkt  
3 import socket  
4  
5 # untersucht eine uebergebene *.pcap-Datei nach IP-Adressen und zaehlt die Pakete  
6 # @param pcap: pcap-Datei  
7 def printPcap(pcap):  
8     counter = 0  
9     for (ts, buf) in pcap:  
10         try:  
11             eth = dpkt.ethernet.Ethernet(buf)  
12             counter += 1  
13             ip = eth.data  
14             src = socket.inet_ntoa(ip.src)  
15             dst = socket.inet_ntoa(ip.dst)  
16             print '[+] Src: ' + src + ' --> Dst: ' + dst  
17         except:  
18             pass  
19     print "Die Mitschnitt beinhaltet %s Pakete." % counter  
20  
21 # oeffnet pcap-Datei und initialisiert Objekt  
22 def main():  
23     f = open('capture.pcap')  
24     # pcap wird initialisiert  
25     pcap = dpkt.pcap.Reader(f)  
26     printPcap(pcap)  
27  
28 if __name__ == '__main__':  
29     main()
```

Konsolenausgabe:

```
Console    
<terminated> quelltext_1_4.py [/usr/bin/python]  
[+] Src: 192.168.179.21 --> Dst: 109.193.192.139  
[+] Src: 192.168.179.21 --> Dst: 109.193.192.139  
[+] Src: 192.168.179.21 --> Dst: 109.193.192.139  
[+] Src: 192.168.179.21 --> Dst: 109.193.192.139  
[+] Src: 157.56.192.147 --> Dst: 192.168.179.21  
Die Mitschnitt beinhaltet 11703 Pakete.
```

## Übung 1.2

Anschließend soll die Übung 1.1 erweitert werden. Neben der Anzahl aller Pakete sollen zudem folgende Kennwerte der übergebenen *pcap*-Datei ausgegeben werden:

- Anzahl der IP-Pakete
- Anzahl der TCP-Pakete • Anzahl der UDP-Pakete

Ob es sich bei dem Arrayelement um ein Paket der IP-Schicht oder der darunterliegenden TCP- und UDP-Schicht handelt, können Sie mittels `eth.type==dpkt.ethernet.ETH_TYPE_IP` überprüfen. `eth` ist ein Objekt der Klasse `dpkt` (siehe Quelltext 1.4). Anschließend können Sie mittels `eth.data.p==dpkt.ip.IP_PROTO_TCP` bzw. `eth.data.p==dpkt.ip.IP_PROTO_UDP` die Art des Pakets genauer bestimmen.

Es werden drei weitere Zähler in den Programmcode eingefügt: `counter_ip`, `counter_tcp`, `counter_udp`.

```
def printPcap(pcap):  
    counter = 0  
    counter_ip = 0  
    counter_tcp = 0  
    counter_udp = 0  
    for (timestamp, buf) in  
        pcap:  
        try:
```

Alle Pakete werden zuerst auf IP-Pakete geprüft. Bei einem Treffer wird `counter_ip` hochgezählt.

```
        counter_ip += 1  
        if eth.type == dpkt.ethernet.ETH_TYPE_IP:  
            counter_ip += 1  
            if eth.data.p == dpkt.ip.IP_PROTO_TCP:
```

Danach werden die IP-Pakete auf die Protokolle TCP und UDP geprüft und ggf. die counter hochgezählt.

```
            if eth.type == dpkt.ethernet.ETH_TYPE_IP:  
                counter_ip += 1  
                if eth.data.p == dpkt.ip.IP_PROTO_TCP:  
                    counter_tcp += 1  
                elif eth.data.p == dpkt.ip.IP_PROTO_UDP:  
                    counter_udp += 1
```

Anschließend werden die Anzahl der jeweiligen Pakete auf der Konsole ausgegeben.

```
        except:  
            pass  
    print('Die Datei hat %s Pakete.' % counter)  
    print('Die Datei hat %s IP-Pakete.' % counter_ip)  
    print('Die Datei hat %s TCP-Pakete.' % counter_tcp)  
    print('Die Datei hat %s UDP-Pakete.' % counter_udp)
```

## Kommentierter Quelltext:

```
1 # Untersuchung der Kommunikation von digitalen Paketen
2 import dpkt
3 import socket
4
5 # untersucht eine uebergebene *.pcap-Datei nach IP-Adressen und zaehlt die Pakete
6 # gepar nach pcap-Datei
7 def printPcap(pcap):
8     counter = 0
9     counter_ip = 0
10    counter_tcp = 0
11    counter_udp = 0
12    for (timestamp, buf) in pcap:
13        try:
14            eth = dpkt.ethernet.Ethernet(buf)
15            counter += 1
16            # untersuchen auf IP-Pakete
17            if eth.type == dpkt.ethernet.ETH_TYPE_IP:
18                counter_ip += 1
19            # untersuchen auf TCP
20            if eth.data.p == dpkt.ip.IP_PROTO_TCP:
21                counter_tcp += 1
22            # untersuchen auf UDP
23            elif eth.data.p == dpkt.ip.IP_PROTO_UDP:
24                counter_udp += 1
25            ip = eth.data
26            src = socket.inet_ntoa(ip.src)
27            dst = socket.inet_ntoa(ip.dst)
28            print '[+] Src: %s --> Dst: %s' % (src, dst)
29        except:
30            pass
31    print 'Die Datei hat %s Pakete.' % counter
32    print 'Die Datei hat %s IP-Pakete.' % counter_ip
33    print 'Die Datei hat %s TCP-Pakete.' % counter_tcp
34    print 'Die Datei hat %s UDP-Pakete.' % counter_udp
35
36 # öffnet pcap-Datei und initialisiert Objekt
37 def main():
38     f = open('capture.pcap')
39     # pcap wird initialisiert
40     pcap = dpkt.pcap.Reader(f)
41     printPcap(pcap)
42
43 if __name__ == '__main__':
44     main()
```

## Konsolenausgabe:

```
Console
<terminated> uebung1_1.py [/usr/bin/python]
[+] Src: 109.193.192.139 --> Dst: 192.168.179.21
[+] Src: 109.193.192.139 --> Dst: 192.168.179.21
[+] Src: 109.193.192.139 --> Dst: 192.168.179.21
[+] Src: 192.168.179.21 --> Dst: 109.193.192.139
[+] Src: 192.168.179.21 --> Dst: 109.193.192.139
[+] Src: 192.168.179.21 --> Dst: 109.193.192.139
[+] Src: 192.168.179.21 --> Dst: 109.193.192.139
[+] Src: 192.168.179.21 --> Dst: 109.193.192.139
[+] Src: 192.168.179.21 --> Dst: 109.193.192.139
[+] Src: 157.56.192.147 --> Dst: 192.168.179.21
Die Datei hat 11703 Pakete.
Die Datei hat 7548 IP-Pakete.
Die Datei hat 7185 TCP-Pakete.
Die Datei hat 347 UDP-Pakete.
```

## Übung 1.3

Erstellen Sie ein Python-Skript, das den Kommunikationsverlauf der Computer in einem Netzwerksegment aus einer *pcap*-Datei ausliest. Es sollen somit alle Pakete mit der Quell-IP-Adresse 192.168.179.x ausgegeben werden. Zudem sollen der 'user-agent' und der 'host' des HTTP-Headers ausgegeben werden. Als Gerüst können Sie erneut den Quelltext 1.4 verwenden. Eine mögliche Ausgabe könnte wie folgt aussehen: DropboxDesktopClient/2.6.2 (Windows; 7; i32; de) notify4.dropbox.com

```
192.168.179.21
-----
MSDW
watson.microsoft.com
192.168.179.23
-----
DropboxDesktopClient/2.6.2 (Windows; 7; i32; de)
notify3.dropbox.com
192.168.179.23
-----
...
```

Zuerst wird die zu untersuchende IP 192.168.179. an die printPcap-Methode übergeben.

```
def main():
    f = open('capture.pcap')
    pcap = dpkt.pcap.Reader(f)
    printPcap(pcap, "192.168.179.")
```

An die Methode ipInfo() werden die aus der pcap-Datei ausgelesene IP-Adresse sowie die gesuchte IP-Adresse übergeben.

```
ipInfo(src, target_ip, tcp)¶  
ipInfo(dst, target_ip, tcp)¶
```

Dort werden beide IP's verglichen und bei einer Übereinstimmung, alle geforderten Daten ausgegeben.

```
def ipInfo(ip, ip_pattern, protocol):¶  
    ip_ar = ip.split('.')¶  
    ip_ar[3] = ''¶  
    if ip_ar == ip_pattern:¶  
        if protocol.dport == 80 and len(protocol.data) > 0:¶  
            http = dpkt.http.Request(protocol.data)¶  
            print http.headers['user-agent']¶  
            print http.headers['host']¶  
            print ip¶  
            print '-----'¶
```

Kommentierter Quelltext:

```
1 # Untersuchung von Kommunikation, sowie HTTP-Informationen¶  
2 import dpkt¶  
3 import socket¶  
4 ¶  
5 # untersucht eine uebergebene *.pcap-Datei nach IP-Adressen¶  
6 # @param pcap: pcap-Datei¶  
7 # @param target_ip: gesuchte IP-Adresse¶  
8 def printPcap(pcap, target_ip):¶  
9     for(timestamp, buf) in pcap:¶  
10         try:¶  
11             eth = dpkt.ethernet.Ethernet(buf)¶  
12             ip = eth.data¶  
13             tcp = ip.data¶  
14             src = socket.inet_ntoa(ip.src)¶  
15             dst = socket.inet_ntoa(ip.dst)¶  
16             ¶  
17             ¶  
18             ipInfo(src, target_ip, tcp)¶  
19             ipInfo(dst, target_ip, tcp)¶  
20         except:¶  
21             pass¶  
22     ¶  
23 ¶  
24 # Vergleich der IP-Adressen, sowie Ausgabe der HTTP-Informationen¶  
25 # @param ip: IP-Adresse aus Datei¶  
26 # @param target_ip: gesuchte IP-Adresse¶  
27 # @param tcp: TCP¶  
28 def ipInfo(ip, target_ip, tcp):¶  
29     ip_pattern = target_ip.split('.')¶  
30     ip_ar = ip.split('.')¶  
31     ip_ar[3] = ''¶  
32     if ip_ar == ip_pattern:¶  
33         if tcp.dport == 80 and len(tcp.data) > 0:¶  
34             http = dpkt.http.Request(tcp.data)¶  
35             ¶  
36             print http.headers['user-agent']¶  
37             print http.headers['host']¶  
38             print ip¶  
39             print '-----'¶  
40     ¶  
41 # öffnet pcap-Datei und initialisiert Objekt¶  
42 # ueber gibt pcap-Datei und gesuchte IP-Adresse¶  
43 def main():¶  
44     f = open('capture.pcap')¶  
45     pcap = dpkt.pcap.Reader(f)¶  
46     printPcap(pcap, "192.168.179.2")¶  
47     ¶  
48 if __name__ == '__main__':¶  
49     main()
```

Konsolenausgabe:

```
Console  [X]  
<terminated> uebung1_3.py [usr/bin/python]  
DropboxDesktopClient/2.6.2 (Windows; 7; i32; de)  
notify4.dropbox.com  
192.168.179.21  
-----  
MSDW  
watson.microsoft.com  
192.168.179.23  
-----  
DropboxDesktopClient/2.6.2 (Windows; 7; i32; de)  
notify3.dropbox.com  
192.168.179.23  
-----  
MSDW  
watson.microsoft.com  
192.168.179.23  
-----  
DropboxDesktopClient/2.6.2 (Windows; 7; i32; de)
```

Anhang:

- uebung1\_1.py
- uebung1\_2.py
- uebung1\_3.py