OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

Fakultät Informatik und Mathematik

COMLAB
Datenkommunikation und Netzwerke

Prof. Dr. Waas

Praktikum zum Fach

**Kommunikationssysteme/
Rechnernetze**

Übung

LAYER- 2 TUTORIAL

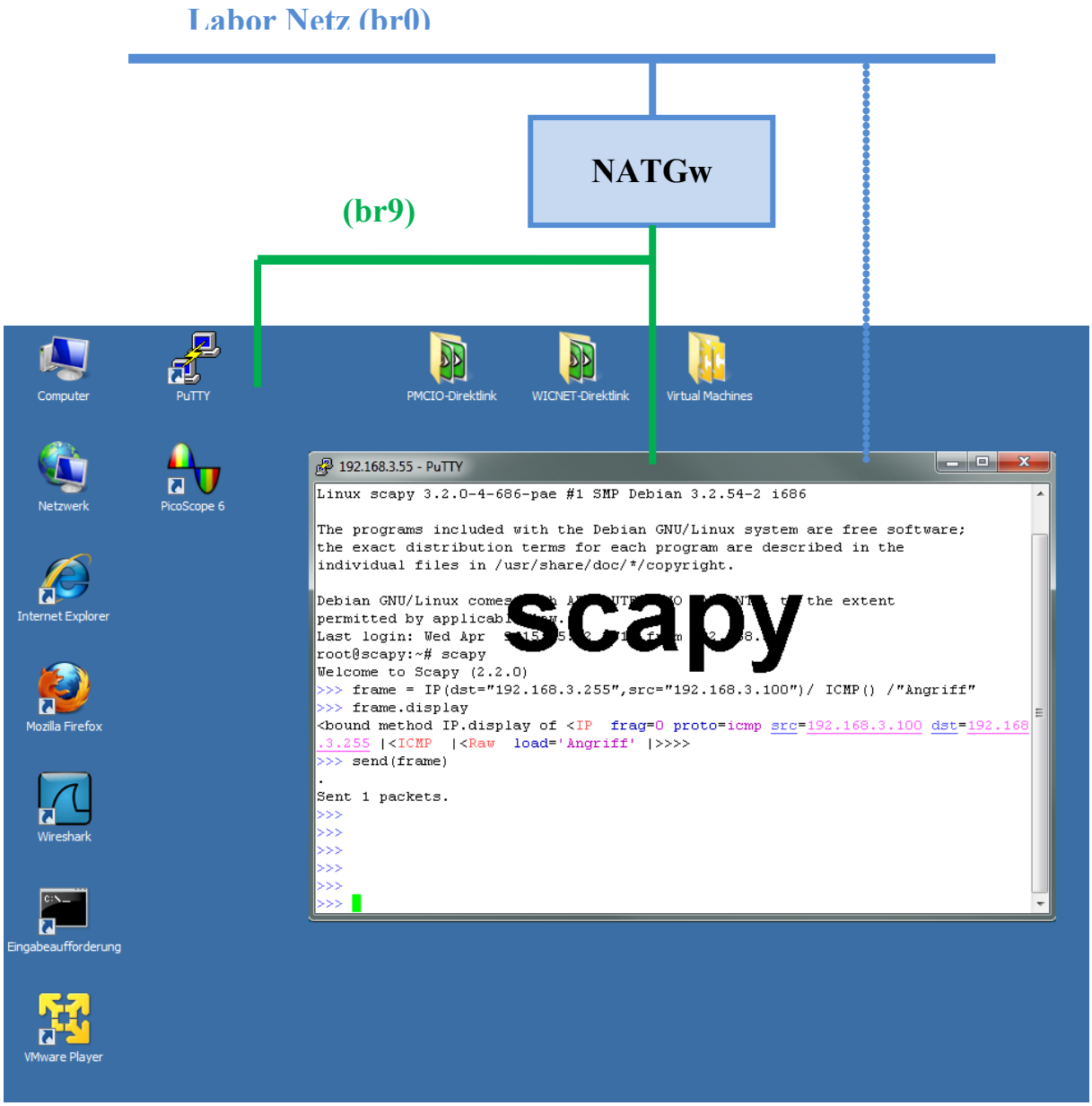
MIT SCAPY

Linux

(Version 18.04.2023 KVM)

1 EINFÜHRUNG

In der folgenden Übung sollen Sie mit den Werkzeugen Scapy Ethernet-Pakete erzeugen und mit Wireshark den korrekten Paketaufbau überprüfen. Am Anfang werden die Pakete über das interne virtuelle Netzwerk br9 gesendet. Später wird dann direkt auf das Labornetz über gegangen (blau gestrichelte Verbindung) und die Ethernet-Pakete können dann innerhalb des Labors gesendet und empfangen werden, d.h. von Labor-PC zu Labor-PC.

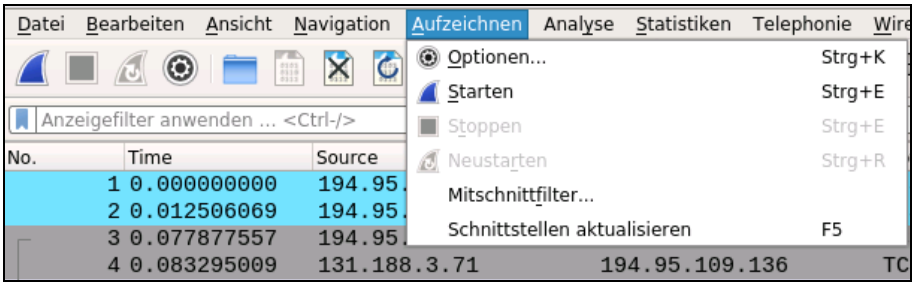


Scapy ist ein sehr umfangreiches System. Allein die Auflistung der möglichen Kommandowörter nimmt mehrere Seiten ein. Daher werden im Rahmen dieser Übung nur die Aspekte besprochen, die im direkten Zusammenhang mit Layer-2 stehen und in dieser Übung gebraucht werden.

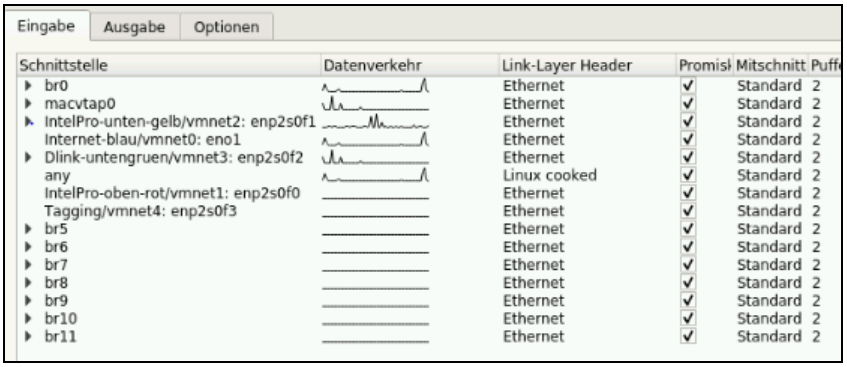
2 STARTEN DER ÜBUNGSUMGEBUNG

- Öffnen Sie den Ordner **Virtual Machines Links** auf dem Desktop.
- Starten Sie die virtuelle Maschine **NATGw** durch Doppelklick und minimisieren Sie ggf. das Fenster von NATGw in die Taskleiste. Auf dieser VM braucht man sich nicht einzuloggen.

- Starten Sie die virtuellen Maschine **scapy** durch Doppelklick.
- Starten Sie auf dem **Labor-PC** das Programm **WireShark.**
- Öffnen Sie das **Aufzeichnen Menü** und wählen Sie **Optionen...** aus.



- Wählen Sie das Interface **br9** aus und klicken Sie auf den **Start**-Button. Damit kann Wireshark die Ethernet-Pakete aufzeichnen, die Sie gleich erzeugen werden. Achten Sie darauf, dass Promiskuitiv (Promiscuous) eingeschaltet ist.



ZWEI METHODEN ZUR KOMMANDOEINGABE AN DER SCAPY KONSOLE

Methode 1:

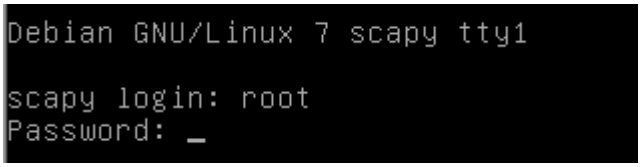
Sie können alle Kommandos direkt im **Fenster** des **virtual Viewers** eingeben. Die Schrift ist aber sehr klein. Wenn Sie das Fenster anklicken, verschwindet die Maus und Sie können nur noch Eingaben im Fenster machen. Cut&Paste ist nicht möglich. Durch Drücken von **<STRG>** und **<ALT>** wird die Maus wieder sichtbar (Release) und Sie haben wieder Zugriff zum Desktop.

Methode 2: **(empfohlen!)**

Verwenden Sie das **scapy Terminal 1**. Drücken Sie die Eingabe-Taste, bis das Login Prompt erscheint. **Hinweis: Cut & Paste** funktioniert im Terminal auch. Sollten Sie das Terminal versehentlich geschlossen haben, können Sie es über den gleichnamigen Shortcut im Verzeichnis **Virtual-Machine-Links** wieder öffnen.

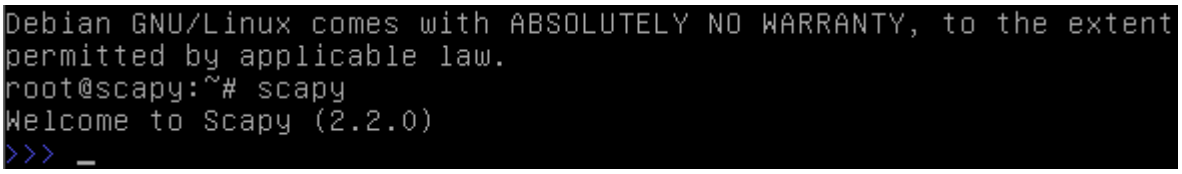
3 DAS ETHERNET PAKET

- Loggen Sie sich am **scapy Terminal 1** mit dem Benutzer **root** und Passwort **comlab** ein.



Anmerkung: Die Verwendung des Benutzers root ist nicht mehr üblich bzw. gewünscht (Sicherheitsgründe), aber eine Erleichterung.

- Geben Sie in der Kommandozeile **scapy** ein Es erscheint dann der Prompt „>>>“.



***WICHTIG:* Im virtual Viewer kommen Sie mit STRG ALT wieder auf den Desktop zurück. Beim Terminal 1 braucht man das nicht.**

3.1 EIN ETHERNER-II PAKET ERSTELLEN

- Führen Sie die folgenden Befehle auf Scapy durch und verstehen Sie deren Funktion und Bedeutung. Die Scapy-Kommandos sind **case-sensitive**. Beobachten Sie die Sendung des Pakets mit Wireshark.

Ein Ethernet-II-Paket besteht aus 6 Byte Zieladresse (.dst) und 6 Byte Absenderadresse (.src) und 2 Byte Typ Feld (.type). Im Typ Feld wird angegeben, welches Protokoll das Paket enthält. Es soll in der Übung der Type 0x8888 verwendet werden und eine Payload/ Nutzlast von 46 Bytes.

64 Bytes ist die minimale Paketlänge bei Ethernet. Die Maximale Paketlänge ist 1518 Bytes. Bei 18 Bytes Rahmen (incl. CRC-32) bleiben also 46 Byte minimale und 1500 Bytes maximale Payload.

Ziel Adr. (dst) 6 Byte	Absende Adr. (src) 6 Byte	Type 2 Byte	Payload / Nutzlast 46 bis 1500 Bytes	CRC-32 4 Byte
00:00:0c:29:53:f5	00:01:12:31:2f:7f	8888	„Hallo Welt…“	

- Der Befehl **ls(Ether)** zeigt die möglichen Parameter an, mit denen man in scapy ein Ethernet-Paket konstruieren kann.

```
>>> ls(Ether)
dst        : DestMACField         = (None)
src        : SourceMACField       = (None)
type       : XShortEnumField      = (0)
```

- Öffnen Sie mit einem Webbrowser folgende URL und prüfen Sie, ob er Ethernet-Type **0x8888**, der hier verwendet werden soll, schon anderweitig offiziell vergeben wurde.

http://www.iana.org/assignments/ieee-802-numbers/ieee-802-numbers.xhtml

- Erstellen Sie ein erstes Ethernet Paket mit den folgenden Befehlen:

```
>>> a=Ether()
>>> a.src="01:02:03:04:05:06"
>>> a.dst="01:02:03:04:05:06"
>>> a.type=0x8888
>>> data="Diese Payload ist 46 Buchstaben bzw Bytes lang"
```

- Sehen Sie sich das Paket in scapy nochmal an!

```
>>> a, data
(<Ether dst=01:02:03:04:05:06 src=01:02:03:04:05:06 type=0x8888 |>,
'Diese Payload ist 46 Buchstaben bzw Bytes lang')
```

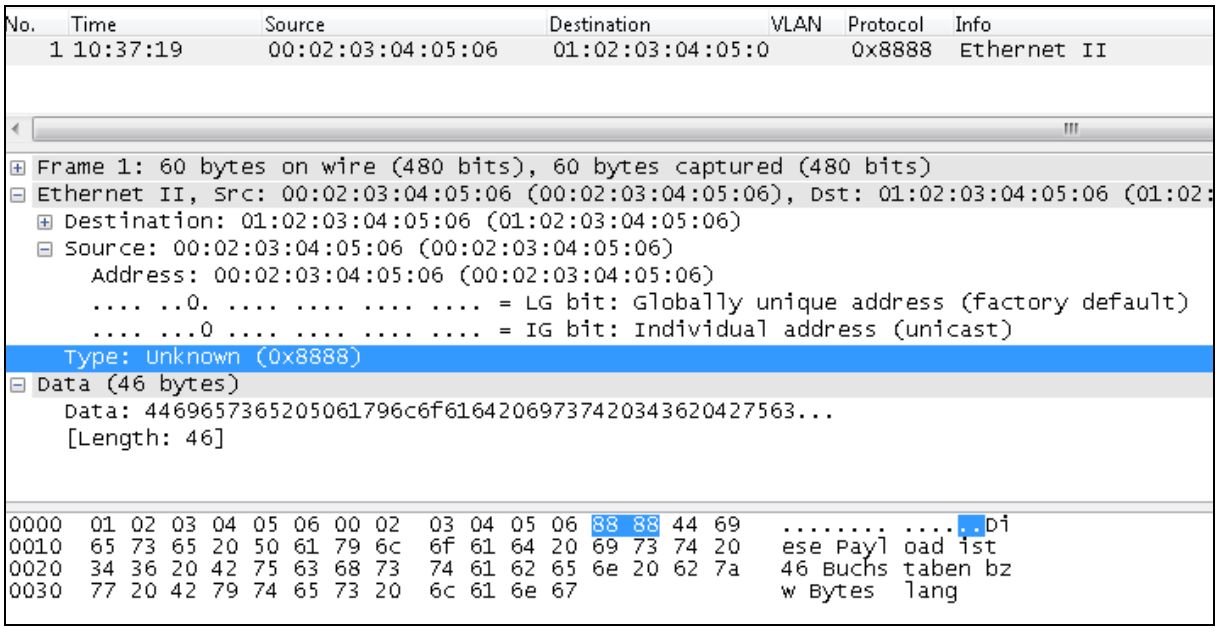
- Senden Sie das Paket und beobachten Sie auf Wireshark, ob dort etwas angezeigt wird.

```
>>> sendp(a/data)
```

<u>Ergebnis:</u> Das Paket wurde nicht gesendet, weil das IG-Bit=1 in der Absenderadresse verboten ist. Es ist nur bei Zieladressen zulässig. Das IG-Bit ist Bit 0 im ersten Byte der Absenderadresse.
IG-Bit einer Ethernet Adresse steht für Individual oder Group. IG=0 bedeutet, es ist eine ***Einzeladresse. IG=1 bedeutet, die Adresse ist eine Multicast- oder Broadcast-Adresse.***

- Ändern Sie die Absenderadresse, sodass kein IG-Bit=1 mehr verwendet wird.

```
>>> a.src="00:02:03:04:05:06"
>>> sendp(a/data)
```



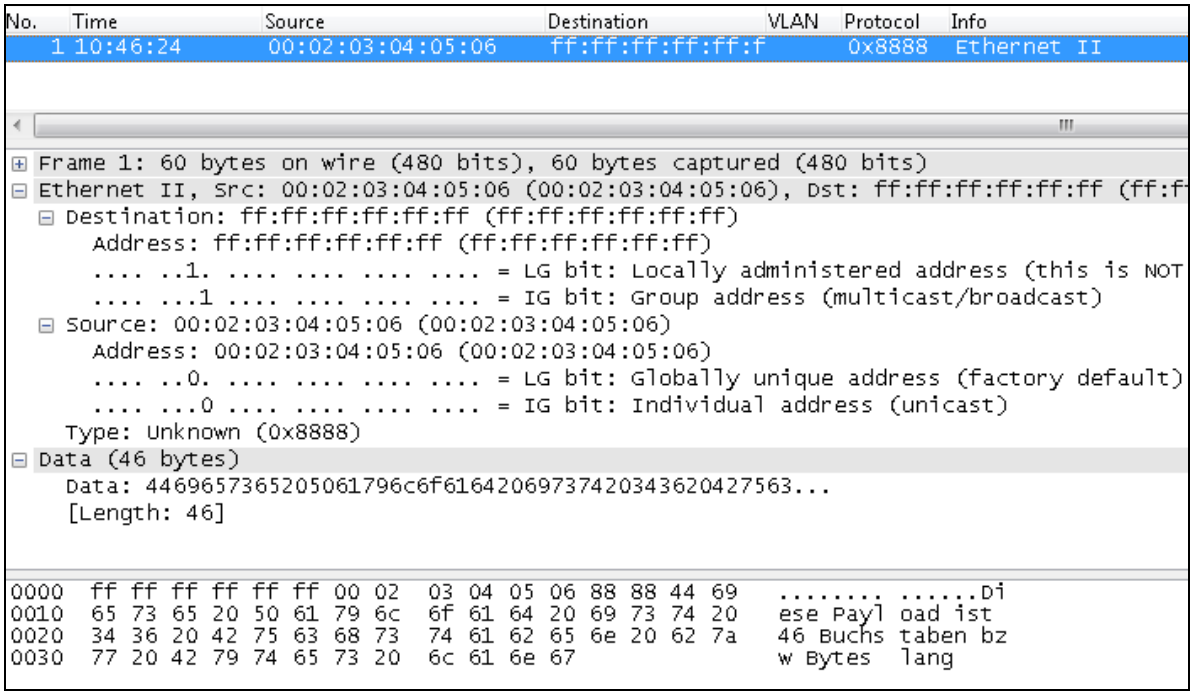
3.2 BROADCAST UND MULTICAST ADRESSEN

Eine Multicastadresse hat das IG-Bit gesetzt (Bit D0 im ersten Byte), d.h. das erste Byte ist ungerade. Eine Broadcast-Adresse ist eine Multicast-adresse, die alle Bits auf 1 hat. Bsp.:

Multicast Adresse	***01:00:3f:09:ab:21 oder 33:33:00:00:00:02***
Broadcast Adresse	***ff:ff:ff:ff:ff:ff***

- Senden Sie nun das obige Paket als Broadcast Paket, also mit der Broadcast-Adresse als Destination-Adresse.

```
>>> a.dst="ff:ff:ff:ff:ff:ff"
>>> sendp(a/data)
```

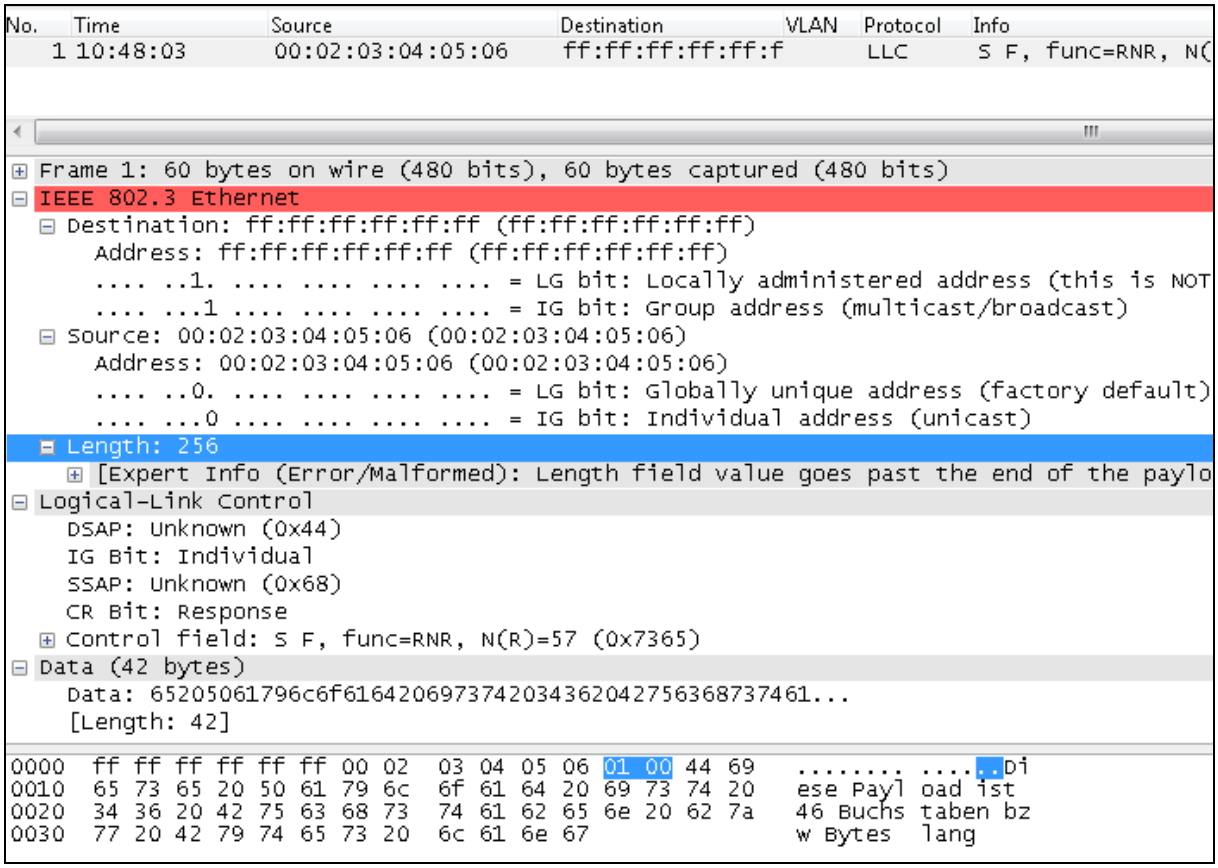


- Nun tauchen keine Fehler in Wireshark auf. Die Nutzlast wird ebenfalls korrekt mit 46 Bytes angegeben.

3.3 IEEE 802.2 LOGICAL LINK CONTROL PAKET

- Setzen Sie nun den Ethernet-Typ auf **0x100** und senden Sie das Paket erneut. Was zeigt der Wireshark nun an?

```
>>> a.type=0x100
>>> sendp(a/data)
```



Das Paket ist „**Malformed**“. Nun gibt es plötzlich ein Logical-Link Control (LLC) und die Länge des Pakets wird mit 256 angezeigt. Und von der Nutzlast sind nur noch 42 Bytes übrig. 4 Bytes wurden scheinbar als LLC interpretiert.

Der Paketaufbau von Ethernet-II unterscheidet sich vom Paketaufbau mit Logical Link Control (LLC) und ist in IEEE 802.2 spezifiziert. Das Typ-Feld wird zum Längen-Feld. Der LLC-Header wird zwischen Längen-Feld und Payload eingefügt (s. Bild).

Ziel Adr. (dst) 6 Byte	Absende Adr. (src) 6 Byte	Länge 2 Byte	LLC	Payload / Nutzlast 46 bis 1500 Bytes	CRC-32 4 Byte
00:00:0c:29:53:f5	00:01:12:31:2f:7f	115		„Hallo Welt…“	

DSAP 1 Byte	SSAP 1 Byte	Control 1 – 2 Bytes
55	55	34 44

- Ermitteln Sie durch **Try & Error**, ab welchem Wert im **Ethernet Typ-Feld** die Information nicht mehr als Protokoll, sondern als LLC interpretiert wird.
  Tip: Die max. Nutzlast von Ethernet Pakets liegt bei 1500 Bytes.

```
>>> a.type=0x600
>>> sendp(a/data)
                        ( Check Wireshark: Type oder LLC
>>> a.type=0x500
>>> sendp(a/data)
                        ( Check Wireshark: Type oder LLC
>>> usw.
```

Der genaue Aufbau von IEEE 802.2 LLC ist Stoff einer späteren Vorlesung. In diesem Tutorial soll ein LLC für SNAP (Sub Network Access Protocol) verwendet werden. SNAP ist ein Header, der für die Abbildung von Ethernet-II-Pakets in 802.2 verwendet wird. Die ursprüngliche Typ-Information wandert dabei in den SNAP-Header (s. Bild). Der SNAP-Header braucht also vorneweg ein LLC mit DSAP=SSAP=0xAA.

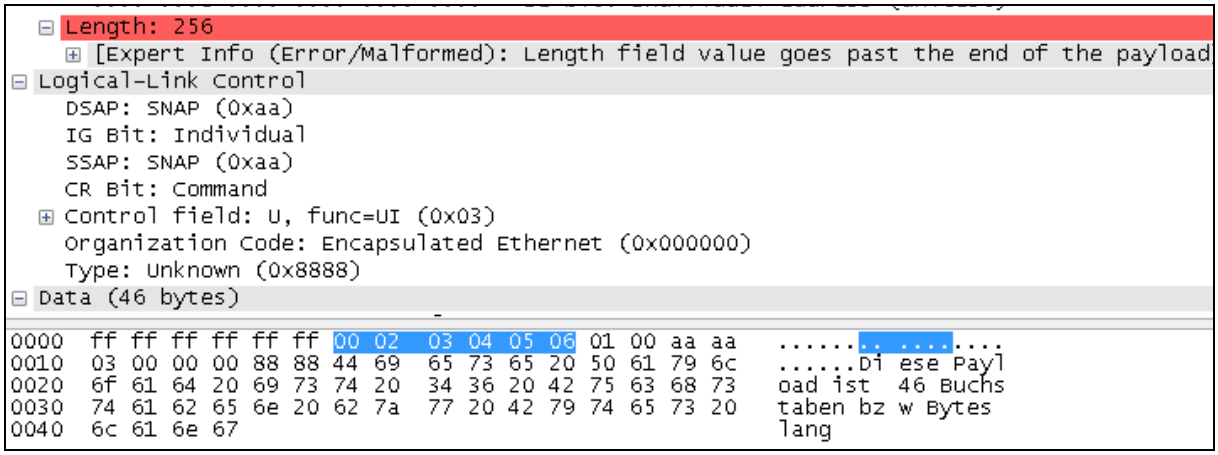
Ziel Adr. (dst) 6 Byte	Absende Adr. (src) 6 Byte	Länge 2 Byte	LLC	Payload / Nutzlast 46 bis 1500 Bytes	CRC-32 4 Byte
00:00:0c:29:53:f5	00:01:12:31:2f:7f	115		„Hallo Welt…“	

DSAP 1 Byte	SSAP 1 Byte	Control 1 Byte	SNAP 5 Bytes
AA	AA	03	00 00 00 88 88

Vendor Code 3 Byte	**Type 2 Byte**
00 00 00	8888

- Erzeugen Sie einen LLC-SNAP-Header für das Protokoll 0x8888 und senden Sie das Paket!

```
>>> a.type=0x100
>>> llc="AAAA030000008888".decode("hex")
>>> sendp(a/llc/data)
```



- Betrachten Sie die Darstellung in Wireshark (vgl. Bild oben). Der LLC-Header müsste jetzt in Ordnung sein, nur die Längeninformation des Pakets muss noch angepasst werden.

- Die Längenangabe des Pakets in **a.type** besteht aus 46 Bytes Nutzdaten und 8 Bytes LLC, macht zusammen 54 Bytes (= 0x36). Senden Sie ein Paket mit korrekter Länge, sodass im Wireshark kein Fehler mehr angezeigt wird.

```
>>> a.type=0x36
>>> sendp(a/llc/data
```

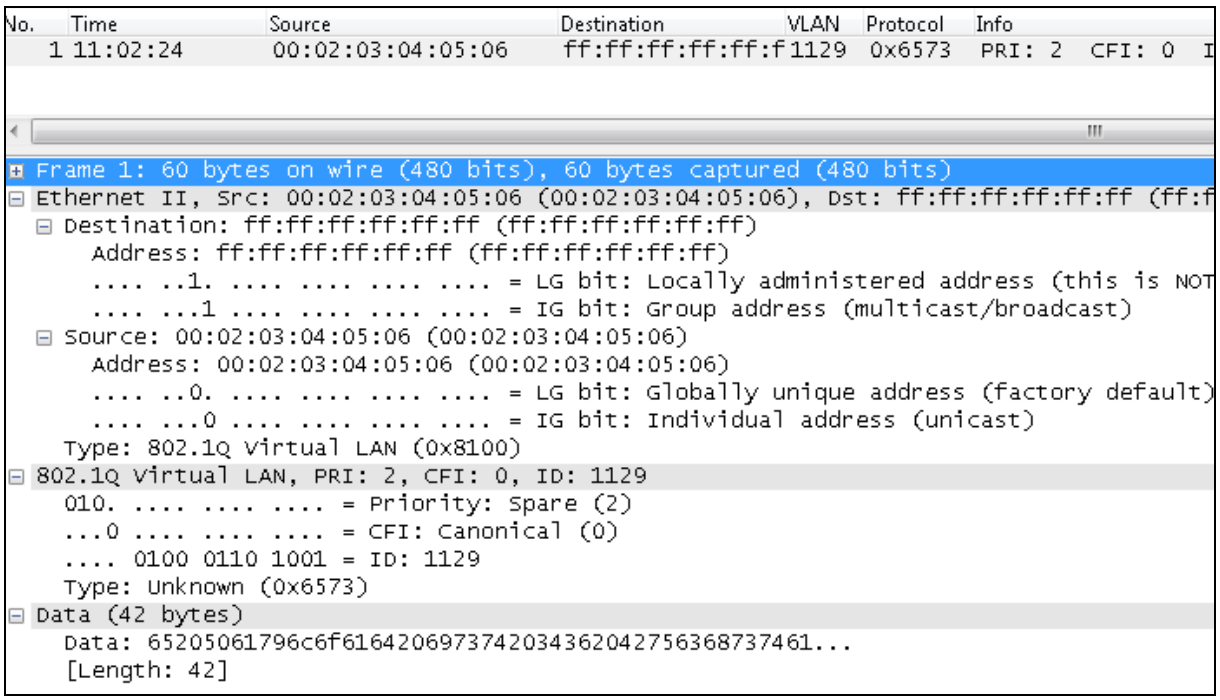
Kontrollfragen:

- ☑ In welchem Bereich wird das Typ-Feld als LLC interpretiert?
- ☑ Ab welchem Wert wird sie als Protokoll-Information interpretiert?
- ☑ Wie kann man einen Ethernet-II Paket von einem 802.2 LLC-Paket unterscheiden?
- ☑ Ist DSAP und SSAP bei LLC das Gegenstück zur Protokollinformation bei Ethernet-II?
- ☑ Wo steht beim SNAP-Header Paket die Protokoll-Information.
- ☑ Um wieviel Bytes ist die Nutzlast bei LLC verschoben gegenüber Ethernet-II.
- ☑ Wie lauten die minimale und maximale Paketlänge bei Ethernet-II und LLC.

3.4 VLAN TAG

- Setzen Sie den **Ethernet Typ** auf den Wert **0x8100** und senden Sie das Paket **ohne LLC.**

```
>>> a.type=0x8100
>>> sendp(a/data)
```



Nun wird im Wireshark ein **802.1Q Virtual LAN-Information** (=VLAN) angezeigt und ein Teil der Nutzlast als VLAN Information interpretiert. Die Nutzlast besteht nur noch aus den letzten 42 Bytes. Die VLAN-Information (genannt: VLAN Tag) scheint also 4 Byte lang zu sein. Der Wert 0x8100 im Type-Feld signalisiert, dass das Paket mit VLAN versehen ist.

Ein 802.1Q VLAN Tag besteht aus 4 Bytes. Da ein VLAN-Tag jederzeit, auch nachträglich, in ein Ethernet Paket eingefügt werden können muss, wurde die maximale Paketgröße von 1518 auf 1522 Bytes erweitert. Das Tag wird zwischen Absenderadresse und Typ/Längenfeld eingefügt.

Ziel Adr. (dst) 6 Byte	Absender Adr. (src) 6 Byte	802.1Q 4 Byte	Type 16 Bit	Payload / Nutzlast 46 bis 1500 Bytes	CRC-32 4 Byte
00:00:0c:29:53:f5	00:01:12:31:2f:7f		0x8888	„Hallo Welt…“	

ID	Priority 3 Bit	C 1	VLAN 12 BIt
8100	03	0	0x40

VLAN wird erst in späteren Vorlesungen behandelt. Einstweilen soll es genügen ein vorgegebenes VLAN Tag in das Paket einzubauen.

- Informieren Sie sich, welche Möglichkeiten scapy zur Erstellung des VLAN-Tags bietet.

```
>>> ls(Dot1Q)
prio       : BitField             = (0)
id         : BitField             = (0)
vlan       : BitField             = (1)
type       : XShortEnumField      = (0)
```

- Versuchen Sie nun eine gültige VLAN-Information in das Paket einzubauen und senden Sie es. Überprüfen Sie dann, ob der VLAN-Tag korrekt aufgebaut ist und keine Fehler angezeigt werden.

```
v=Dot1Q()
v.prio=3
v.vlan=0x40
v.type=0x8888
sendp(a/v/data)
```

```
    Type: 802.1Q Virtual LAN (0x8100)
⊟ 802.1Q Virtual LAN, PRI: 3, CFI: 0, ID: 64
    011. .... .... .... = Priority: Excellent Effort (3)
    ...0 .... .... .... = CFI: Canonical (0)
    .... 0000 0100 0000 = ID: 64
    Type: Unknown (0x8888)
⊟ Data (46 bytes)
    Data: 44696573652050617 96c6f6164206973742043620427563...
    [Length: 46]
```

- Überprüfen Sie, ob die Priorität, die VLAN-ID und der Type in Wireshark korrekt wiedergegeben werden und dass die Nutzlast 46 Bytes lang ist.

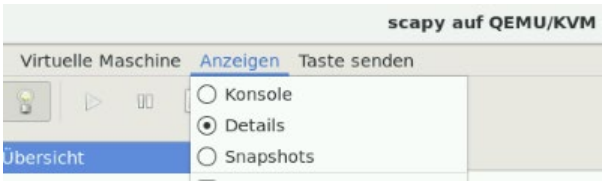
- Erzeugen und senden Sie nun ein Paket mit **VLAN-Tag** und **SNAP-Header** und überprüfen Sie, dass in Wireshark keine Fehler angezeigt werden.
  Anmerkung: **v.type** (im VLAN-Tag)) muss dabei die Länge der Payload bekommen (weil LLC folgt!). Der LLC mit SNAP und die Länge der Payload können Sie oben aus Kapitel 3.3 entnehmen. Dann mit **`sendp(a/v/llc/data)`** senden.

3.5 MELDUNGEN ÜBER DAS LAN SENDEN

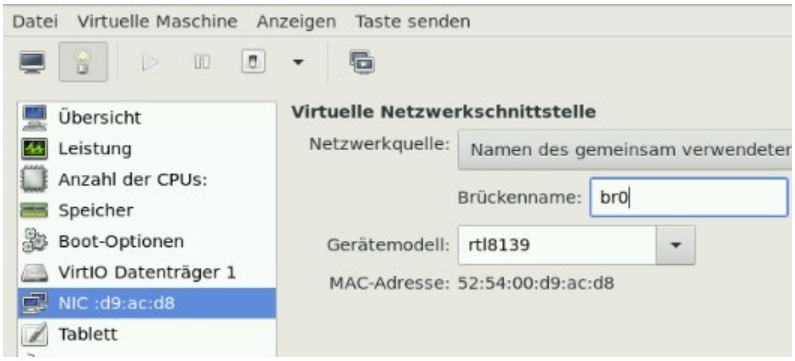
- Stellen Sie nun das LAN-Interface des virtuellen PC scapy um. Starten Sie dazu die **virtuelle Maschinenverwaltung (virtual Manager)** auf dem Desktop.

- Klicken Sie auf **scapy.**

- Klicken Sie im Menü Anzeigen auf die **Detail** Darstellung .



- Selektieren Sie **Network Adapter (NIC)** und ändern Sie den Brückennamen von br9 auf **br0**. Klicken Sie dann auf Anwenden.



- Stellen Sie die Anzeige auf **Konsole** zurück. Nun haben Sie wieder zwei Anzeigen, den Viewer und Terminal-1. Für die folgenden Schritte brauchen Sie beide Anzeigen.

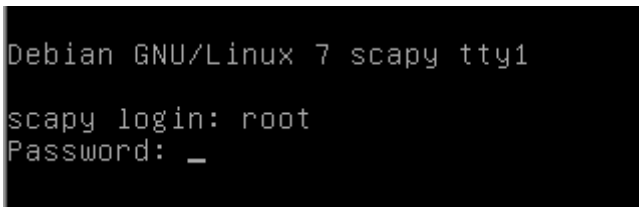
4 BROADCAST CHAT

Durch Verwendung der Broadcast-Adresse kann ein Rundspruch-Verfahren implementiert werden. Ein Ethernet-Paket, der an die Broadcast-Adresse gesendet wird, wird von allen Hosts im Netzwerk empfangen.

4.1 MELDUNGSEMPFÄNGER EINRICHTEN

Sie brauchen jetzt zwei Anzeigen für scapy um die folgenden Übungen durchzuführen, den scapy Viewer und das Terminal-1. Folgen Sie den Anweisungen, um die jeweils zusätzliche Konsole zu starten.

- **Falls noch nicht geschehen**, Loggen Sie sich auf beiden Anzeigen mit dem Benutzernamen **root** und dem Passwort **comlab** ein und geben Sie ggf. **scapy** ein, damit das Prompt „>>>“ erscheint.



- Geben Sie am **scapy Viewer** ein:

```
>>> sniff(filter="ether proto 0x8888", prn=lambda
x:x.sprintf("%Raw.load%"))
```

Die Funktion sniff läuft so lange, bis sie mit STRG-C abgebrochen wird. Soll sniff nur eine bestimmte Anzahl von Paketen empfangen, kann man die Option count=n. einbauen:
Beispiel: `sniff(filter……, count=1)`

4.2 MELDUNGEN TESTEN

- Geben Sie am **Terminal-1** ein:

```
>>>
sendp(Ether(dst="ff:ff:ff:ff:ff:ff",src="00:02:03:04:05:06",type=0x8888)
/"Hallo Welt ...")
```

Jedes Mal, wenn Sie eine Meldung senden, wird diese am anderen Display angezeigt. Die Gestaltung der „Nutzlast“ ist frei. Ist das Paket kleiner als 64 Bytes füllt der Netzwerkkarten-treiber automatisch mit sog. Pad Daten (0x00) auf.

4.3 VERWENDUNG VON MULTICAST ADRESSEN

Wenn ein Rundspruchverfahren nur einen bestimmten Benutzerkreis erreich soll, werden normalerweise Multicast-Adressen verwendet. Nur wer auf der entsprechenden Multicast-Adresse hört, kann die Daten empfangen. Unbeteiligte Hosts werden dadurch nicht belastet, im Gegensatz zur Verwendung der Broadcast Adresse.

- Erweitern Sie den Empfänger, dass er auch auf der Multicastadresse zuhört. Geben Sie am **scapy Viewer** ein:

```
>>> sniff(filter="ether dst 01:00:11:11:88:88",prn=lambda
x:x.sprintf("%Raw.load%"))
```

- senden Sie Ihre Pakete mit der Multicast Adresse 01:00:11:11:88:88 Geben Sie am **Terminal -1** ein:

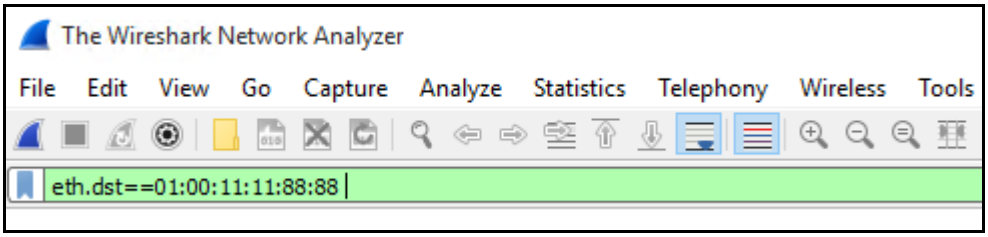
```
>>> sendp(Ether(dst="01:00:11:11:88:88",
src="00:02:03:04:05:06",type=0x8888)/"Hallo Welt ...")
```

Nun bekommen Sie nur die Meldungen, die an die Multicast-Adresse gesendet werden. Lassen Sie Sender und Empfänger eine Weile laufen.

- Falls noch nicht geschehen, stellen Sie im Wireshark das Mittschnitt-Interface auf das Interface mit dem Namen **Internet-blau** um und starten Sie Aufzeichnung neu.

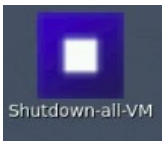
- Nun beobachten Sie die Pakete direkt auf dem Labornetz.

- Stellen Sie in Wireshark bei **Filter** folgendes ein**: eth.dst==01:00:11:11:88:88** und klicken Sie auf **Apply**.



  Die Zeile bei Filter müsste grün sein. Rot weist auf einen Fehler hin. Damit werden jetzt nur die Rundspruchmeldungen angezeigt. Alle anderen Pakets, die auch im Labornetz übertragen werden, werden unterdrückt.

- Beenden Sie nach einer Weile die Übung, indem Sie alle virtuellen Maschinen beenden. Klicken Sie dazu auf das Shortcut **Shutdown-all-VM** auf dem Desktop.



5 WEITERE ÜBUNGEN ZUR NACHARBEIT

5.1 Bitte interpretieren Sie **ALLE** Felder des folgenden Ethernet Pakets auf Layer 2 und ergänzen Sie die folgende Tabelle (dort wo Fragezeichen „?“stehen).

`Paket: |ff|ff|ff|ff|ff|ff|00|11|22|33|44|55|81|00|61|0e|88|88 |48|61|6c|6c|6f|20|57|65|6c|74|`

Feldname	**Inhalt**
Dest. Address	?
?	?
?	81 00 61 0e
?	88 88
Data	48 61 6c 6c 6f 20 57 65 6c 74

5.2 In der folgenden Tabelle sind die einzelnen Felder eines Ethernet Pakets angegeben. Bitte bauen Sie daraus einen Ethernet Paket zusammen. (Padding, Präambel, CRC werden dabei **nicht** berücksichtig).

Feldname	**Feldinhalt**
LLC	AA AA 03
Eth. Length	16 Byte (Angabe in Dezimal)
Eth. Address (Src. or Dest.)	80 00 02 af fe 01

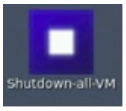
Payload Data	41 42 43 44 45 46 47 48
Eth. Address (Src. Or Dest.)	33 33 00 02 03 04
SNAP Header	00 00 00 88 88

5.3 Das folgende Bild zeigt Kommandos, mit denen man mittels Scapy einen Ethernet Paket aufgebaut hat.

```
a=Ether()
a.src="00:02:03:04:05:06"
a.dst="0C:00:08:04:05:06"
a.type=0x8888
data="Das ist die Payload"
v=Dot1Q()
v.prio=0
v.vlan=0x40
v.type=0x8888
sendp(a/v/data)
```

Geben Sie die einzelnen Bytes des gesendeten Ethernet-Pakets an, durch Komma getrennt. Die Payload (data) kann einfach als Text geschrieben werden. (Padding, Präambel, CRC können weggelassen werden).

6 ENDE DER ÜBUNG



- **Beenden Sie alle virtuellen Maschinen (Klick auf Shutdown-all-VM).**
- Beenden Sie alle Instanzen von Wireshark. Im Rahmen der Übung an Ihrem Arbeitsplatz erzielte Messergebnisse können Sie auf Ihren Memorystick zur späteren Nachbearbeitung abspeichern. Gewonnene sicherheitsrelevante Informationen insbesondere Passwörter, dürfen nicht weitergegeben oder unbefugt verwendet werden.
- Loggen Sie sich aus dem Labor-PC aus
- Lassen Sie den Labor-PC weiterlaufen. Es wird automatisch ausgeschaltet.

Bitte hinterlassen Sie Ihren Arbeitsplatz in ordentlichem Zustand!

Entsorgen Sie Mitgebrachtes selbst!

Schieben Sie den Stuhl an den Tisch!

ANHANG: ALLE KOMMANDOS DER ÜBUNG.

Leider werden im PDF-Dokument die Anführungszeichen mit falschem Code dargestellt. Falls man die Kommandos mit Cut$Paste in scapy einfügt gibt es Fehlermeldungen und man muss die Anführungszeichen ersetzen.

```
a=Ether()
a.src="01:02:03:04:05:06"
a.dst="01:02:03:04:05:06"
a.type=0x8888
data="Diese Payload ist 46 Buchstaben bzw Bytes lang"
a,data
sendp(a/data)

a.src="00:02:03:04:05:06"
sendp(a/data)

a.dst="ff:ff:ff:ff:ff:ff"
sendp(a/data)


a.type=0x100
sendp(a/data)

a.type=0x100
llc="AAAA030000008888".decode("hex")
sendp(a/llc/data)

a.type=0x36
sendp(a/llc/data)


a.type=0x8100
sendp(a/data)

v=Dot1Q()
v.prio=3
v.vlan=0x40
v.type=0x8888
sendp(a/v/data)


v.type=0x36
llc=(b'\xAA\xAA\x03\x00\x00\x00\x88\x88')
sendp(a/v/llc/data)
```