



**Vorlesung Rechnernetze**

**Laborübung**

**Einstieg in die Socketprogrammierung**

**Prof. Dr. Dirk Staehle**

Die Abgabe erfolgt durch Hochladen der bearbeiteten Word-Datei in Moodle.

**Bearbeitung in Zweier-Teams**

**Team-Mitglied 1: Alex Schapelt**

**Team-Mitglied 2: Josef Müller**

**Team-Mitglied 3: Walter Vötsch Cortés**

# Einleitung

# Vorbereitung

# Mail

## SMTP über telnet

1. Betrachten Sie die Aufzeichnung in WireShark. Was fällt Ihnen auf.

Ganz normale TCP Verbindung, E-Mails und Authentifizierungen laufen über SMTP (Kommando-Übertragungen)

1. Schreiben Sie noch eine Mail an ihren Email-Account. Verwenden Sie willkürliche Email-Adressen für MAIL-FROM sowie für das „from:“-Feld in der Mail. Siehe auch Wikipedia Beispiel. Lesen Sie die Mail in ihrem Postfach. Was fällt Ihnen auf?

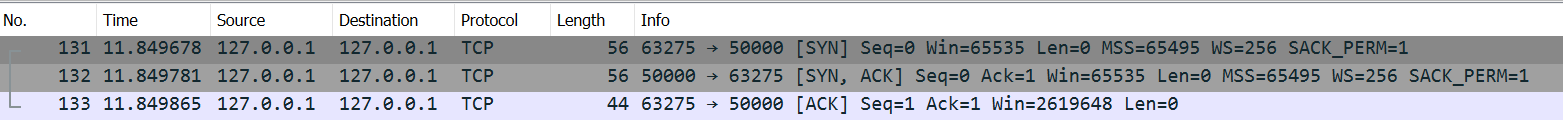
Ganz „normale“ E-Mail; Sender-Adresse, Betreff usw. kann jedoch weggelassen werden  
  
Hinweis: Die Erkenntnisse, die Sie hier erlangen dienen LEDIGLICH dazu, Sie darauf hinzuweisen, wie einfach es ist, Fake-Mails zu versenden. Bitte schreiben Sie keine Fake-Mails, auch nicht zum Spaß.

## SMTP in Python

# Rechen-Server

## Lokale Kommunikation

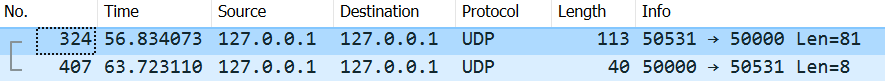
1. für jedes gesendete Paket bestimmen, welcher Befehl in welchem Skript (Client/Server) dafür verantwortlich ist, dass das Paket gesendet wird  
   connect vom Client nach accept vom Server ist verantwortlich für:



send im Client nach recv im Server (bei TCP verbindung) ist verantwortlich für:



Sendto im Client nach recvfrom im Server (bei UDP) verbindung und umgekehrt ist verantwortlich für:



1. für jeden blockierenden Befehl bestimmen, die Ankunft welches Pakets dafür verantwortlich ist, dass die Ausführung des Befehls vervollständigt wird

accept wird vervollständigt durch:



recv wird vervollständigt durch:



recfrom wird vervollständigt von:



## Netzwerk-Kommunikation

1. Wie können Sie im Client Python-Skript die IP-Adresse und Port-Nummer des verwendeten lokalen Sockets bestimmen (im Sinne von herausfinden)?  
   🡺 (ip, port) = socket.getsockname()
2. Wann (in welcher Code-Zeile) und woher erhält ein Client seine IP-Adresse und Port-Nummer?  
   🡺 socket.bind bzw. implizit durch socket.connect()
3. Wie können Sie im Client-Skript die IP-Adresse und Port-Nummer des Sockets setzen?  
   🡺 socket.bind
4. Warum müssen Sie Timeouts verwenden und wie funktioniert try … except? Mit welchem Befehl können Sie einen gemeinsamen Timeout für alle Sockets setzen?  
   🡺 socket.setdefaulttimeout()
5. Finden Sie experimentell heraus, ob Sie einen Server betreiben können, der ECHO-Anfragen auf dem gleichen Port für UDP und TCP beantwortet?

🡺 nicht hinbekommen

# Port Scan

## Beschreibung

In diesem Versuch führen wir einen Port-Scan durch, um herauszufinden, welche Ports auf einem Server geöffnet sind. Wir scannen zum einen die standardisierten Ports von 1-50 nach offenen TCP Ports. Wenn wir einen offenen Port finden, versuchen wir, eine Nachricht an diesen Port zu schicken. Weiterhin vermuten wir, dass auf dem Server ECHO-Dienste für die Übertragung mit TCP und UDP laufen.

Ist ein TCP-Port auf einem Server geöffnet, dann wird ein Verbindungsaufbau auf diesem Port akzeptiert. Ist der TCP-Port nicht offen, so antwortet der Server entweder nicht (Windows Fehler-Code 10060) oder mit einem RST+ACK, in dem der Verbindungsaufbau zurückgewiesen wird (Windows Fehler-Code 10061).

Ist ein UDP Port auf einem Server geöffnet, so antwortet der Server entweder mit einer Nachricht oder überhaupt nicht. Die Reaktion hängt sowohl vom empfangenden Dienst als auch von der Nachricht selbst ab. Ist der UDP Port nicht geöffnet, so antwortet der Server entweder nicht oder mit einem ICMP Paket vom Typ 3, mit dem er mitteilt, dass das Ziel nicht erreichbar ist (Windows Fehler-Code 10054).

## Versuch

### TCP Port Scanner

### UDP Port Scanner

## Fragen

1. Geben Sie die Liste der offenen TCP und UDP Ports an.
2. geöffnete TCP Ports:
3. 7, 9, 13, 17, 19
4. geöffnete UDP Ports:
5. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50
6. Wählen Sie für TCP und UDP jeweils einen offenen und einen geschlossenen Port und erklären Sie die entsprechende Paketsequenz, die Sie in WireShark aufgezeichnet haben.
7. Auf Port 7 des Servers läuft ein ECHO-Dienst. Testen Sie ihr Client-Script mit dem ECHO-Server. Versuchen Sie das TCP und das UDP Script.