Hochschule Konstanz Technik, Wirtschaft und Gestaltung (HTWG)
Fakultät Informatik
Rechner- und Kommunikationsnetze
Prof. Dr. Dirk Staehle

Vorlesung Rechnernetze

Theorieübung Socketbefehle und Pakete

Prof. Dr. Dirk Staehle

Die Abgabe erfolgt durch Hochladen der Lösung in Moodle und exemplarisches Vorrechnen in der Laborübung.

Bearbeitung in Zweier-Teams

Team-Mitglied 1:

Team-Mitglied 2:

In dieser Aufgabe geht es darum, den Zusammenhang von Socket-Befehlen und übertragenen Pakete zu verstehen. Dazu ist in Abbildung 1 ein Python-Code dargestellt. Dieser Python Code wird auf einem Rechner zweimal mit unterschiedlichen Konfigurationen nacheinander ausgeführt. Bei der ersten Ausführung (A) ist My_IP=127.0.0.2 und Remote_IP=127.0.0.1. Bei der zweiten Ausführung (B) ist My_IP=127.0.0.1 und Remote_IP=127.0.0.2.

Ein Mitschnitt der ausgetauschten Pakete in Tabelle 1 dargestellt.

1. Tragen Sie in die erste Spalte von **Fehler! Verweisquelle konnte nicht gefunden werden.** ein, welche Code-Zeile bei welcher Ausführung des Codes (A oder B) diese Paketübertragung bewirkt hat, sofern diese explizite Zuordnung möglich ist d.h. wenn die Übertragung des Pakets umittelbar durch eine Code-Zeile ausgelöst wird.
2. Tragen Sie in die zweite Spalte von **Fehler! Verweisquelle konnte nicht gefunden werden.** ein, welche „blockierende" Befehlszeile bei welcher Ausführung des Codes (A oder B) erfolgreich (ohne Fehler) „beendet" wird, wenn das Paket empfangen wird. Auch hier soll natürlich nur dann ein Eintrag erfolgen, wenn der Empfang dieses Pakets die „Fertigstellung" eines Socket-Befehls bewirkt.

Hinweis: Die Eintragungen in die Tabelle sollen also beispielweise die Form A10 haben, wenn Programmzeile 10 der ersten Ausführung (A) des Codes die Übertragung des Pakets bewirkt hat, oder B10 wenn dementsprechend die zweite Ausführung (B) des Codes die Übertragung des Pakets bewirkt hat.

Abbildung 1 Code Listing

Ausführung (A) ist My_IP=127.0.0.2 und Remote_IP=127.0.0.1

Ausführung (B) ist My_IP=127.0.0.1 und Remote_IP=127.0.0.2

```
1   import socket
2   socket.setdefaulttimeout(30)
3
4   My_IP = '127.0.0.2' / '127.0.0.1'
5   My_PORT = 50000
6   Remote_IP= '127.0.0.1' / '127.0.0.2'
7   Remote_PORT=50000
8
9   def start_task(sock,message):
10      sock.send(message.encode('utf-8'))
11      msg=sock.recv(1024)
12      sock.close()
13
14  def start_server():
15      sock=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
16      sock.bind((My_IP, My_PORT))
17      sock.listen(1)
18      try:
19          conn, addr = sock.accept()
20          start_task(conn,"Thx for connecting!!!")
21      except socket.timeout:
22          pass
23
24  sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
25  try:
26      sock.connect((Remote_IP, Remote_PORT))
27      start_task(sock,"Thx for accepting!!!");
28  except socket.error:
29      start_server(
```

Ausführung A	Ausführung B	
A26		Socket Connect to Server
A28		Socket Error (Server not running)
A29		Request start Server
A16		Start Server on IP: 127.0.0.2
A19		Waiting on Clients to Connect
	B26	Client (127.0.0.1) connect to Server 127.0.0.2 / 127.0.0.2
	B27/10	Start Task and Send Msg "Thx for accepting" to Server [Len: 20]
	B11	Waiting on Server response
A20/10		Start Task and Send Msg "Thx for Connecting" to Client [Len: 21]
	B12	Close Client Socket
A11		Waiting on Client resond
A21		Waiting till Timeout or Connections close

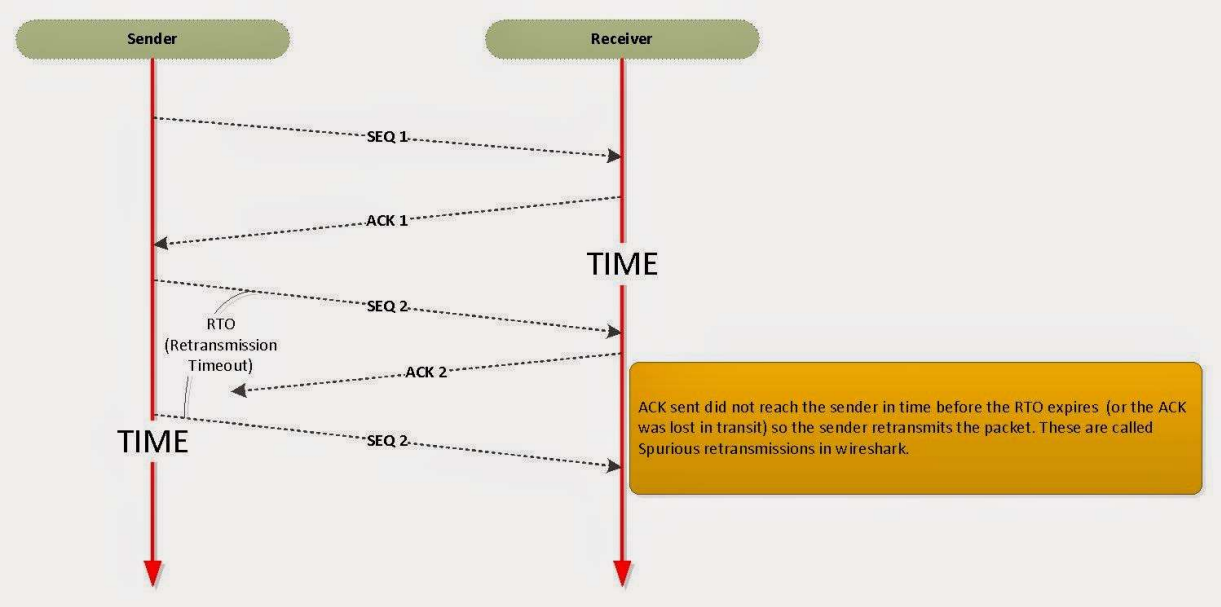

Tabelle 1: TCPdump

Info	Übertragung ausgelöst von Programmzeile m	Empfang bewirkt "Fertigstellung " von Prog. Zeile	No.	Time	Source	Destination	Protocol	Length	Src Port	Dst Port	
C. try connect to Server No 1	**A26**		1	1,340	127.0.0.2	127.0.0.1	TCP	52	56835	50000	56835 >
Reset from IP-Stack			2	1,340	127.0.0.1	127.0.0.2	TCP	40	50000	56835	50000 >
C. try connect to Server No 2	**A26**		3	1,842	127.0.0.2	127.0.0.1	TCP	52	56835	50000	[TCP Spu
Reset from IP-Stack			4	1,842	127.0.0.1	127.0.0.2	TCP	40	50000	56835	50000 >
C. try connect to Server No 3	**A26**		5	2,342	127.0.0.2	127.0.0.1	TCP	48	56835	50000	[TCP Spu
Res. from IP-St. -> Run Server			6	2,342	127.0.0.1	127.0.0.2	TCP	40	50000	56835	50000 >
Client Request Connection	**B26**		7	3,646	127.0.0.1	127.0.0.2	TCP	52	56835	50000	56837 >
Server Accept Connection	**A19**	**B26**	8	3,646	127.0.0.2	127.0.0.1	TCP	52	50000	56835	50000 >
Client ACK Connection	**B26**	**A19**	9	3,646	127.0.0.1	127.0.0.2	TCP	40	56835	50000	56837 >
Send "Thx for accepting"	**B10**		10	3,646	127.0.0.1	127.0.0.2	TCP	60	56835	50000	56837 >
ACK from Server to Client		**A11**	11	3,646	127.0.0.2	127.0.0.1	TCP	40	50000	56835	50000 >
Send "Thx for Connecting"	**A10**		12	3,646	127.0.0.2	127.0.0.1	TCP	61	50000	56835	50000 >
ACK from Client to Server		**B11**	13	3,646	127.0.0.1	127.0.0.2	TCP	40	56835	50000	56837 >
Close Connection from C to S	**B12**		14	3,646	127.0.0.1	127.0.0.2	TCP	40	56835	50000	56837 >
ACK Connection Close to C	**A12**		15	3,646	127.0.0.2	127.0.0.1	TCP	40	50000	56835	50000 >
Ack Connection Close to S			16	3,646	127.0.0.1	127.0.0.2	TCP	40	56835	50000	56837 >

Verbinfungsaufbau mittels Three-Way Handshake

Verbindungsabbau OHNE Three-Way Handshake

Siehe auch: https://www.elektronik-kompendium.de/sites/net/2009211.htm