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: Yasmin Hoffmann

Team-Mitglied 2: Chris Jakob

Team-Mitglied 3: Axel Schwarz

Abbildung 1

In dieser Aufgabe geht es darum, den Zusammenhang von Socket-Befehlen und übertragenen Pakete zu verstehen. Dazu ist in **~~Fehler! Verweisquelle konnte nicht gefunden werden.~~** 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 ist in Tabelle 1 dargestellt.

1. Tragen Sie in die erste Spalte von Tabelle 1 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 Tabelle 1 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

```
1   import socket
2   socket.setdefaulttimeout(30)
3
4   My_IP = '127.0.0.1'
5   My_PORT = 50000
6   Remote_IP='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()
```

SOCK_DATA = UDP

\ TCP

Client requests connection by sending SYN (synchronize)
Server acknowledges by sending SYN-ACK
Client responds with ACK (acknowledge)
=> connection established

FIN : triggers graceful connection termination
RST : triggers forceful connection termination
PSH : tells client/server to push bytes to application layer
ACK : informs client about last received byte by the server

	Übertragung ausgelöst von Programmzeile	Empfang bewirkt "Fertigstellung" von Programmzeile	No.	Time	Source	Destination	Protocol	Length	Src Port	Dst Port	Info
connect	A26		1	1,340	127.0.0.2	127.0.0.1	TCP	52	56835	50000	56835 > 50000 [SYN] Seq=0
			2	1,340	127.0.0.1	127.0.0.2	TCP	40	50000	56835	50000 > 56835 [RST, ACK] Seq=1 Ack=1
			3	1,842	127.0.0.2	127.0.0.1	TCP	52	56835	50000	[TCP Spurious Retransmission] 56835 > 50000 [SYN] Seq=0
			4	1,842	127.0.0.1	127.0.0.2	TCP	40	50000	56835	50000 > 56835 [RST, ACK] Seq=1 Ack=1
			5	2,342	127.0.0.2	127.0.0.1	TCP	48	56835	50000	[TCP Spurious Retransmission] 56835 > 50000 [SYN] Seq=0
		A26	6	2,342	127.0.0.1	127.0.0.2	TCP	40	50000	56835	50000 > 56835 [RST, ACK] Seq=1 Ack=1
connect	B26		7	3,646	127.0.0.1	127.0.0.2	TCP	52	56837	50000	56837 > 50000 [SYN] Seq=0
accept			8	3,646	127.0.0.2	127.0.0.1	TCP	52	50000	56837	50000 > 56837 [SYN, ACK] Seq=0 Ack=1
		326	9	3,646	127.0.0.1	127.0.0.2	TCP	40	56837	50000	56837 > 50000 [ACK] Seq=1 Ack=1
send	B10		10	3,646	127.0.0.1	127.0.0.2	TCP	60	56837	50000	56837 > 50000 [PSH, ACK] Seq=1 Ack=1 Len=20
receive		B10	11	3,646	127.0.0.2	127.0.0.1	TCP	40	50000	56837	50000 > 56837 [ACK] Seq=1 Ack=21 Len=0
send	A10		12	3,646	127.0.0.2	127.0.0.1	TCP	61	50000	56837	50000 > 56837 [PSH, ACK] Seq=1 Ack=21 Len=21
receive		A10	13	3,646	127.0.0.1	127.0.0.2	TCP	40	56837	50000	56837 > 50000 [ACK] Seq=21 Ack=22 Len=0
close	B12		14	3,646	127.0.0.1	127.0.0.2	TCP	40	56837	50000	56837 > 50000 [FIN, ACK] Seq=21 Ack=22 Len=0
			15	3,646	127.0.0.2	127.0.0.1	TCP	40	50000	56837	50000 > 56837 [FIN, ACK] Seq=22 Ack=22 Len=0
close		B12	16	3,646	127.0.0.1	127.0.0.2	TCP	40	56837	50000	56837 > 50000 [ACK] Seq=22 Ack=23 Len=0

Tabelle 1: TCPdump