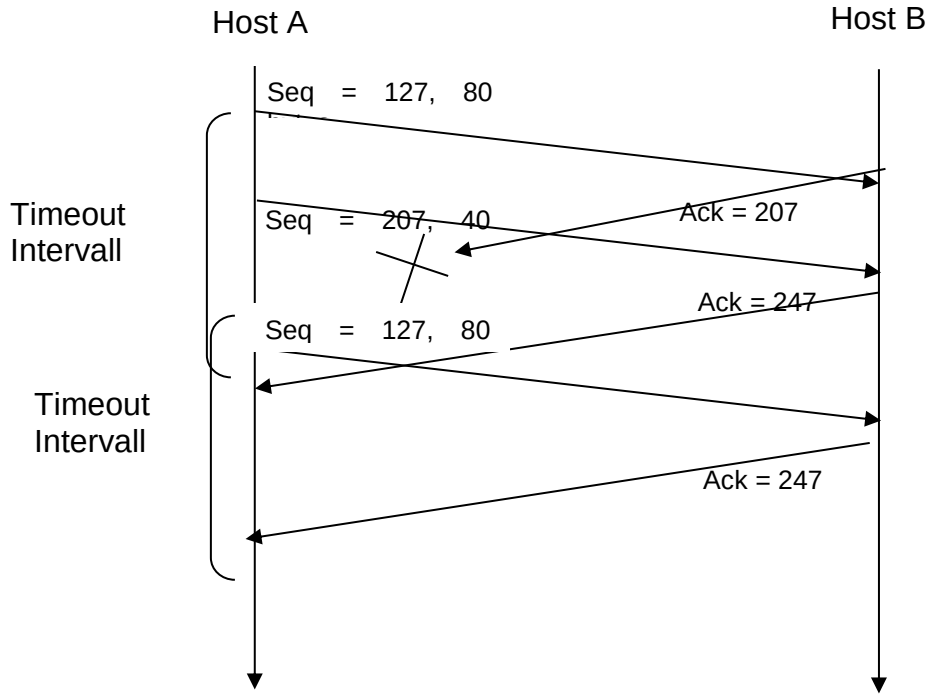
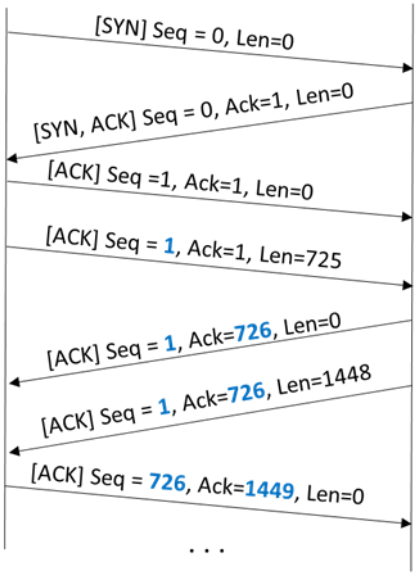
Lösung 11: TCP

Aufgabe 1: TCP – Unidirektionale Kommunikation

a) Seq = 207, Src = 302, Dst = 80
b) Src = 80, Dst = 302, Ack = 207 (nächstes erwartetes Byte)
c) Ack 127 (kumulative ACKs; als nächstes wird Byte Nummer 127 erwartet.
d) Hinweis: TCP ist eine Mischung aus Go-Back-N und Selective Repeat. Im Gegensatz zum reinen Go-Back-N Ansatz wird bei einem Timeout zunächst nur das älteste noch unbestätigte Paket erneut gesendet.



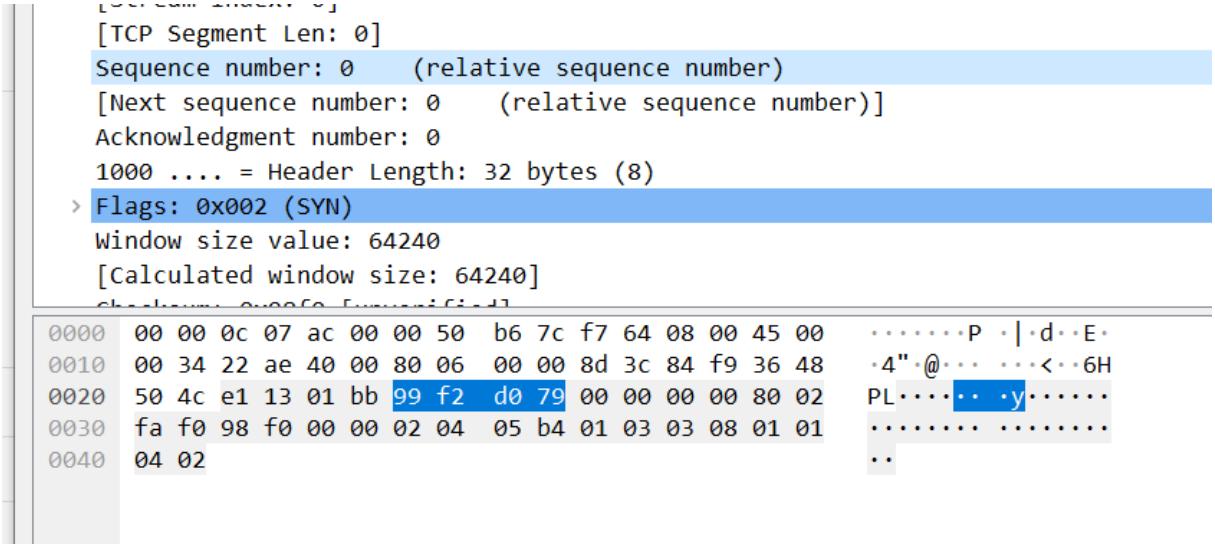
Aufgabe 2: TCP – Bidirektionale Kommunikationf



Aufgabe 3: TCP Connection Management und Flow Control

a) Immer die gleiche Sequenznummer zu wählen ist problematisch, wenn die Verbindung schnell hintereinander ab- und wieder aufgebaut wird. Dann wäre es nicht zu unterscheiden, ob die Pakete von einer alten Verbindung stimmen oder von einer neuen. Deshalb „empfiehlt" die RFC die erste Sequenznummer zufällig zu wählen. In der Praxis wird das auch gemacht.

Wireshark erweckt jedoch den Eindruck, dass die initiale Sequenznummer 0 ist. Schaut man genauer hin, erkennt man, dass es Wireshark „relative Sequenznummern" angibt. Im Beispiel wird 99 f2 d0 79 als „initiale Sequenznummer" verwendet. Wireshark zeigt aber relativ immer 0 an. Das erhöht die Lesbarkeit.



b) TCP Client: CLOSED, SYN SENT und ESTAB
TCP Server: CLOSED, LISTEN, SYN RCVD, ESTAB

c) A wird daraufhin keine Nutzdaten mehr zu B senden, da sein Sendefenster durch den erhaltenen Wert *rwnd* nach oben begrenzt ist.
Hinweis: Problematisch wird es dann, wenn B in umgekehrter Richtung keine Daten zum Senden an A hat. Wie soll B dann jemals an A mitteilen, dass in seinem Empfangspuffer nun wieder Platz ist? In vielen TCP Implementierungen muss deshalb der Receiver ein aktives Update von rwnd senden (auch ohne Nutzdaten).

d) Das FIN-Paket wird bei B ignoriert, vermutlich wird B ein TCP RST senden. Der Grund: Das Paket stammt von einer anderen Source IP Adresse (nicht von A!) und gehört deshalb logischerweise zu einem anderen Socket bzw. zu einer anderen TCP Verbindung (wenn es diese überhaupt gibt). Es wird also die bestehende Verbindung nicht beeinflussen.

Aufgabe 3: TCP in Wireshark

a)

	TCP/HTTP Client	TCP/HTTP Server
IP Adresse	192.168.1.102	128.119.245.12
Portnummer	1161	80

b) Wireshark markiert das Paket #199 als HTTP Post Paket. Zur Übertragung des HTTP Posts wurden 122 TCP Pakete benötigt.

c) Seq = 0 (müsste nicht zwingend 0 sein). SYN Flag ist auf 1 gesetzt

d) Seq = 0, Ack = 1, Syn/Ack Flag gesetzt. Interessant: Der TCP Server bestätigt mit ACK=1 anstatt wie erwartet mit ACK=0. Der Grund warum man ACK=0 erwarten würde: Im SYN-Paket werden noch keine Nutzdaten versendet. Für das SYN-Paket gilt jedoch eine Ausnahme.

e) Paket #4, Seq=1

f)

	Paketnummer in Wireshark	Sequenznummer	Paketnummer des dazugehörigen ACKs
1. Segment	4	1	6
2. Segment	5	566	9
3. Segment	7	2026	12
4. Segment	8	3486	14

g) In der SYNACK Nachricht ist das *Receive Window* 5840 Bytes groß. Es wird dann schnell vergrößert bis auf 62780 Bytes, aber nie verkleinert.
Schön sieht man das in Wireshark, wenn man die Entwicklung des Empfangsfensters plottet: *Statistiken, TCP Stream Graph, Window Skalierung.*

h) Jede Segmentnummer kommt nur einmal vor (mit Ausnahme des Verbindungsabbaus). Die Verbindung war also stabil ohne Retransmissions.