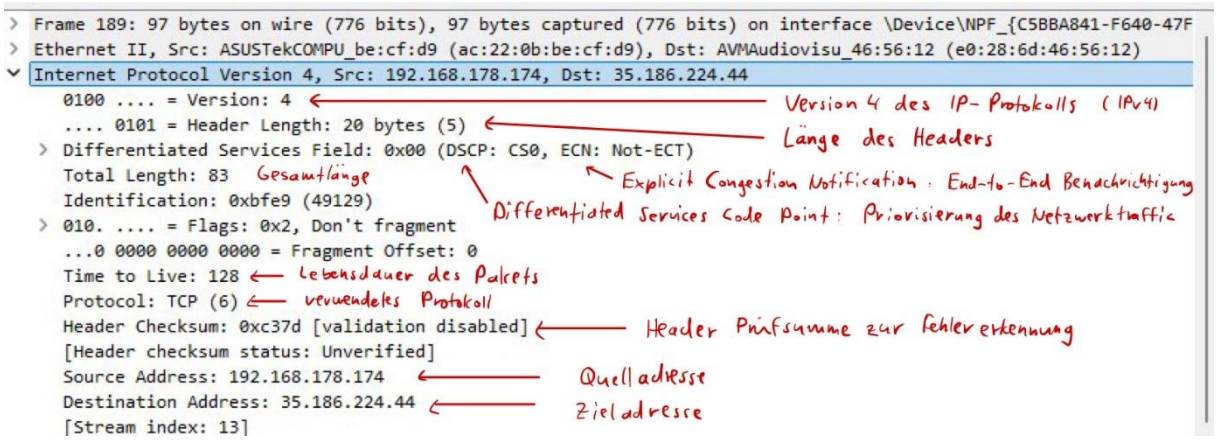
4. Hausaufgabe

Aufgabe 1: Protokoll-Header

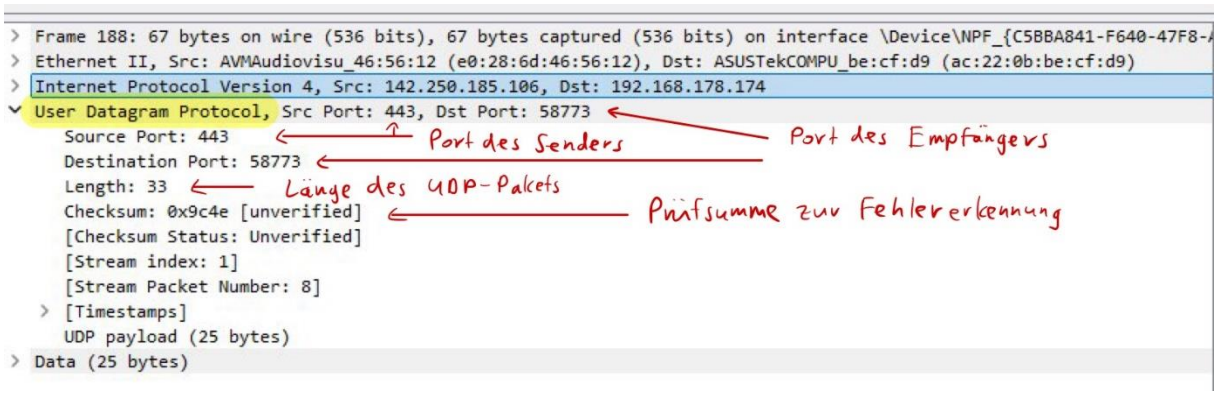
In der Vorlesung wurden die Kategorien der IPv4-Header vorgestellt. Finden Sie in Wireshark ein beliebiges IPv4-Paket. Ordnen Sie Elemente des gefundenen Paketes den Kategorien des Headers zu (Version, Header Length, Type of Service, etc.).

Finden Sie anschließend jeweils ein UDP – und TCP-Paket und ordnen Sie auch dort die Elemente des Pakets den Kategorien zu. Beachten Sie, dass die Header dieser beiden Protokolle anders aufgebaut sind als der IPv4-Header.

IPv4-Paket:

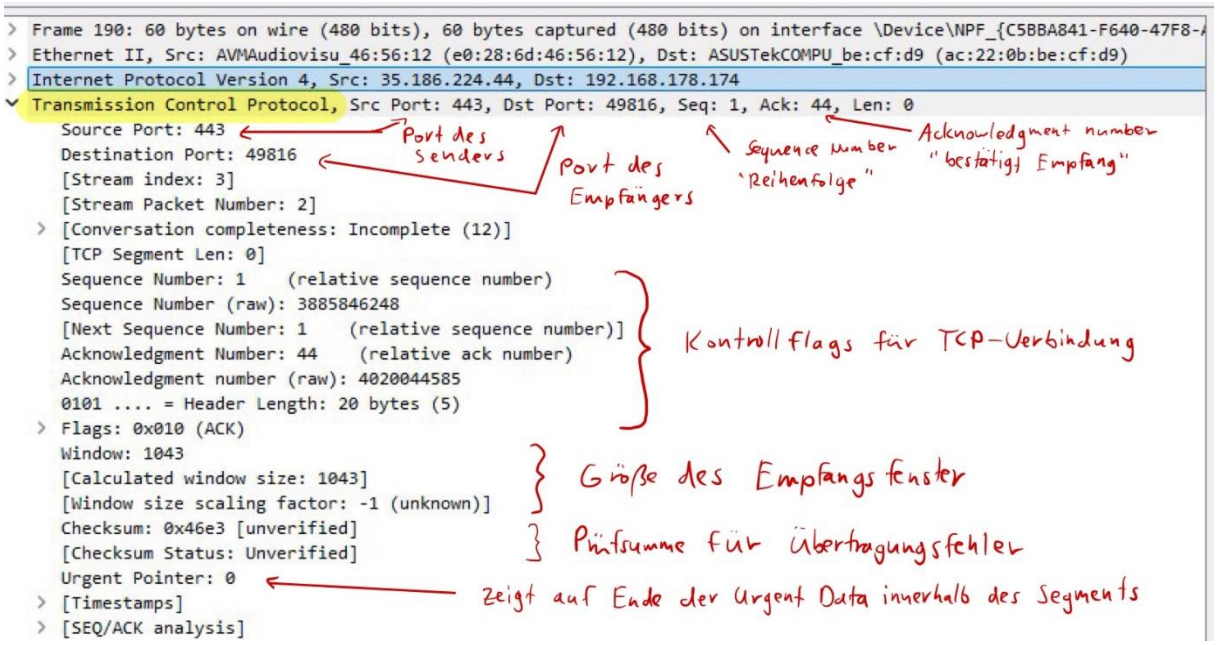


UDP-Paket:



UDP ist ein **verbindungsloses** Protokoll, sodass man im Paket nur die grundlegenden Informationen für die Übertragung findet. Ganz anders sieht es da beim **verbindungsorientierten TCP-Protokoll** aus.

TCP-Paket:



TCP als **verbindungsorientiertes** Protokoll enthält viele Kontroll-Flags für den Verbindungsaufbau und die Aufrechterhaltung, sodass das Paket fast doppelt so lang ist, wie das UDP-Paket, welches wir zuvor betrachtet haben.

Aufgabe 2: CIDR

Beschreiben Sie 103.161.122.83/18. Was bedeuten jeweils die 103.161.122.83 und die 18? Wie kann man daraus Subnetzmaske, Broadcastadresse und Netzwerkadresse ermitteln? Liegt die 103.161.122.83/18 im selben Netz wie 103.161.193.83/18? Notieren Sie ihre Ergebnisse.

Beschreibung:

- 103.161.122.83 ist die **IP-Adresse**, welches jedes Gerät im Netzwerk identifiziert
- /18 ist die *Classless Inter-Domain Routing* (**CIDR**), welche die Anzahl der Bits angibt, die die Netzwerkadresse darstellen, in diesem Fall also 18, während die restlichen Bits den Host im Netzwerk identifizieren

Subnetzmaske:

- Berechnung: die ersten 18 Bit der 32 Bit IPv4 auf 1 setzen, Rest auf 0 (32-18 = 14)
  Umrechnung: 11111111.11111111.11000000.00000000 blockweise in Dezimalsystem
- => 255.255.192.0

Netzwerkadresse:

- Berechnung: Bitweise AND-Operation zwischen Subnetzmaske & IP-Adresse 01100111.10100001.01111010.01010011 (binär)
- Umrechnung: 01100111.10100001.01000000.00000000 blockweise in Dezimalsystem
- => 103.161.64.0

Broadcastadresse:

- Berechnung: die letzten 14 Bits (Host-Bits) der Netzwerkadresse (binär) auf 1 setzen
- Umrechnung: 01100111.10100001.01111111.11111111 blockweise in Dezimalsystem
- => 103.161.127.255

Selbes Netzwerk:

103.161.122.83/18 und 103.161.193.83/18 liegen **nicht** im selben Netzwerk. Berechnet man die Netzwerkadressen erhält man:

103.161.122.83/18	103.161.64.0
103.161.193.83/18	103.161.192.0

Da wir unterschiedliche Netzwerkadressen erhalten, sieht man direkt, dass sie nicht im selben Netzwerk liegen.

Aufgabe 3: Kommunikation zwischen Implementationen

Nutzen Sie das Chat-Programm der letzten Übung. Ob Sie dabei Ihre eigene Implementierung, oder die Ihrer KommilitonInnen, nutzen, spielt keine Rolle.

Versuchen Sie, das UDP-Chat-Programm mit Implementierungen von KommilitonInnen kommunizieren zu lassen. Probieren Sie es anschließend auch mit dem TCP-Chat-Programm aus. Dokumentieren Sie Probleme, die dabei auftreten und überlegen Sie sich Lösungen dafür.

UDP:

Bei **UDP** handelt es sich um ein **verbindungsloses** Protokoll. Das bedeutet, dass man einmal das Skript im Servermodus startet und danach auf eingehende Pakete wartet.

Diese schlanke Kommunikation vereinfachte den Aufwand beim Anpassen der beiden Programme enorm. Es musste lediglich der Vorgang der Registrierung, die Kommandos im Chat und die Syntax beim Senden von Nachrichten angepasst werden, sodass beide Programme miteinander kompatibel waren.

TCP:

Bei **TCP** handelt es sich um ein **verbindungsorientiertes** Protokoll, sodass der Verbindungsaufbau und die Erhaltung eine viel zentralere Rolle spielen, als bei UDP.

Dies bedeutete, dass beide TCP-Programme schon von Beginn an beim Verbindungsaufbau und der Registrierung identisch laufen mussten. Dies erforderte einige Änderungen. Nachdem die Verbindung aufgebaut wurde und stabil lief, waren noch Anpassungen bei den Kommandos im Chat, sowie bei der Syntax beim Senden von Nachrichten erforderlich, bevor auch die TCP-Chat Programme kompatibel waren.

Fazit:

Wie zu erwarten war es bedeutend schwieriger, die TCP-Verbindung aufzubauen und aufrecht zu erhalten, als die UDP-Verbindung zwischen zwei Clients herzustellen.

Überaschenderweise war es kein einziges Mal nötig (weder bei UDP, noch bei TCP) eine Firewall-Ausnahme oder sonstige Anpassungen (z.B. deaktivieren des Virenschutz, etc.) vorzunehmen. Ab dem Moment, wo beide Programme kompatibel waren, funktionierte die Kommunikation ohne Probleme.

Aufgabe 4: Programmierung

Erweitern Sie das Chat-Programm um folgende Methoden:

a) Sende Nachricht an alle bekannten Clients (UDP & TCP-Server)
b) Anfrage, Nachricht an alle bekannten Clients schicken (TCP-Client)
c) Sendung der Liste der bekannten Clients (TCP-Server)
d) Anfrage, Liste der bekannten Clients zu schicken (TCP-Client)
e) Wenn bestimmte, vordefinierte Fragen gestellt werden, soll automatisch eine bestimmte, vordefinierte Antwort an den Fragesteller gesendet werden, z.B. „Was ist deine MAC-Adresse?“, „Sind Kartoffeln eine richtige Mahlzeit?“, etc. (UDP & TCP-Client).

Wie können Sie sicherstellen, dass diese Funktionen auch bei der Kommunikation mit Clients/Servern von anderen Implementierungen funktioniert? Nutzen Sie dazu dieses Google Docs-Dokument. Einigen Sie sich auf bestimmte Spezifikationen, so dass Ihre Implementierungen mit denen Ihrer KommilitonInnen funktionieren werden.

UDP:

a) Methode ***send_all()*** hinzugefügt
e) Erweiterung in ***__init__()*** um die vordefinierten Fragen und in ***process_incoming()***, sowie Ergänzung um ***get_curent_time()*** für die Abfrage der Systemzeit

TCP:

a) Methode ***send_all_server()*** hinzugefügt & Eingabe in Serverfenster ergänzt
b) Methode ***handle_client()*** um Abfrage nach Befehl *broadcast* ergänzt
c) Funktion ***broadcast_client_list()*** hinzugefügt & ***server(port)*** um *send_all_server()* ergänzt
d) Funktion ***handle_client()*** um Abfrage nach Befehl *list* ergänzt
e) 2 Importe ergänzt, Methode ***get_own_ip()*** hinzugefügt & ***handle_client()*** ergänzt um den Fall, dass vordefinierte Fragen gesendet werden