

Computernetzwerke

Vermittlungsschicht

Sebastian Bauer

Wintersemester 2022/2023

Computer Engineering Curriculum



Hybrides Referenzmodell: Vermittlungsschicht



Pakete

- Protokollelemente, die hier ausgetauscht werden, heißen Pakete (engl. *packets*)
- Enthalten schichtspezifische Quell- und Zieladressen
 - Beim IPv4-Protokoll die IPv4-Adressen
 - Beim IPv6-Protokoll die IPv6-Adressen
- Pakete werden im Adressraum über das gesamte Kommunikationsnetz durch *Router* vermittelt

Outline

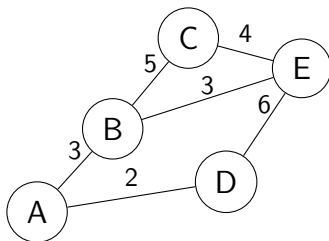
1 Adressen

2 Routing

3 IP

Adressierung im Kontext der Vermittlungsschicht

- Jeder Knoten im Netz hat mind. eine Adresse
- Innerhalb der direkten Erreichbarkeit sind diese eindeutig
- Adressen gehören zu den Netzwerkadaptern
- Netz kann mathematisch als Graph betrachtet werden:
 - Knoten sind Hosts oder Router (dahinter können auch andere Netzwerke sein)
 - Kanten sind Netzverbindungen mit Kostengewichtung

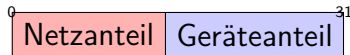


Internet-Protokoll

- Sind konkrete Implementierungen der Vermittlungsschicht
- Wird für das Internet genutzt
- Schicht heißt im TCP/IP-Referenzmodell Internet-Schicht
- Verbindungsloses und unzuverlässiges Protokoll (Pakete können verloren gehen)

Aufbau von IPv4-Adressen

- 32-stellige Binärzahl
- Aufgeschrieben in *dotted decimal notation*
 - Vier Zahlen zw. 0 und 255 durch Punkt getrennt
 - Beispielsweise: 123.22.23.43
- Besteht aus einem Netzanteil und einem Geräteanteil
 - Netzanteil (ein Präfix) kennzeichnet ein Teilnetz
 - Geräteanteil ein Gerät oder Host im Teilnetz



Pakete werden geroutet, genau dann wenn Quell- und Ziel-IP nicht im selben Netz sind.

IPv4-Subnetzmaske

- Netzanteil einer Adresse spezifiziert durch *Subnetzmaske*
- Bitweise Und-Verknüpfung ergibt Netzanteil

```

IPv4-Adresse 11000000 10101000 00000001 10000001 192.168.1.129
UND Netzmaske 11111111 11111111 11111111 00000000 255.255.255.0
= Netzwerkeil 11000000 10101000 00000001 00000000 192.168.1.0
  
```

- Adresse des Netzwerks endet immer mit 0-Bits
- Diese steht keinem Gerät zur Verfügung

Welche boolsche Funktion nutzt man, um die Geräteadresse aus Subnetmaske und IP-Adresse zu ermitteln? Führe diese auf das Beispiel aus.

IPv4-Subnetzmaske

- Netzanteil einer Adresse spezifiziert durch *Subnetzmaske*
- Bitweise Und-Verknüpfung ergibt Netzanteil

```

IPv4-Adresse 11000000 10101000 00000001 10000001 192.168.1.129
UND Netzmaske 11111111 11111111 11111111 00000000 255.255.255.0
= Netzwerkeil 11000000 10101000 00000001 00000000 192.168.1.0

```

- Adresse des Netzwerks endet immer mit 0-Bits
- Diese steht keinem Gerät zur Verfügung

Welche boolsche Funktion nutzt man, um die Geräteadresse aus Subnetmaske und IP-Adresse zu ermitteln? Führe diese auf das Beispiel aus.

```

IPv4-Adresse 11000000 10101000 00000001 10000001 192.168.1.129
UND ~Netzmaske 00000000 00000000 00000000 11111111 0.0.0.255
= Geräteadresse 00000000 00000000 00000000 10000001 0.0.0.129

```

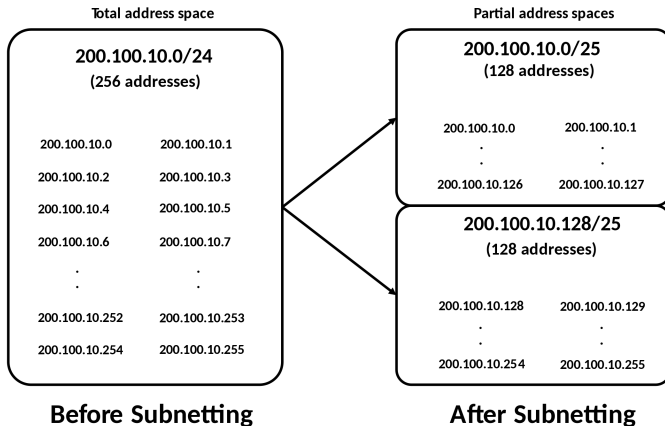
Classless Inter-Domain Routing (CIDR)

- CIDR-Notation enthält nur die Anzahl der gesetzten Bits der Subnetzmaske, z. B. /24 steht für 255.255.255.0
- Um IP- und Netzteil anzugeben, kann man schreiben: 192.168.0.1/24

Netzwerkklassen (Vorläufer zu CIDR, obsolet seit 1993)

Klasse	Präfix	CIDR	Bereich
A	0b0	/8	0.0.0.0 – 127.255.255.255
B	0b10	/16	128.0.0.0 – 191.255.255.255
C	0b110	/24	192.0.0.0 – 223.255.255.255
D	0b1110		224.0.0.0 – 239.255.255.255
E	0b1111		240.0.0.0 – 255.255.255.255

Unterteilung der Netze in weitere Subnetze



Quelle: <https://en.wikipedia.org/wiki/Subnetwork>

Broadcast IPv4-Adressen

- Alle Bits der Geräteadresse sind gesetzt
- Verlassen nicht das eigene Netzwerk

Welche bitweise Verknüpfung führt man aus, um aus Netzwerkadresse und Subnetzmaske auf die Broadcast-Adresse zu kommen?

Private IPv4-Adressen

- Sind vom Kontext des Internets nicht sichtbar
- Erlauben volle IP-Konnektivität im abgeschirmten
- Verbundene Rechner bilden ein *Intranet*
- Anschluss ans Internet via Router/Gateway und Netzwerkadressübersetzung

CIDR	Bereich
10.0.0.0/8	10.0.0.0 – 10.255.255.255
172.16.0.0/12	172.16.0.0 – 172.31.255.255
192.168.0.0/16	192.168.0.0 – 192.168.255.255

Link-Local-Adressen

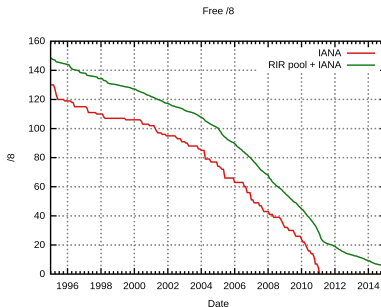
- Weiterer privater Adressraum: 169.254.0.0/16
- Geräte können sich selbst Adressen aus diesem Bereich zuweisen (wenn z. B. DHCP versagt)
- Definiert in [RFC 3927](#)
 - 1 Wähle eine zufällige IP-Adresse im Bereich
 - 2 Schaue, ob anderer Host die Adresse hat (ARP-Probe)
 - Sender-IP: 0.0.0.0
 - Sender-Mac: wie beim Netzwerkadapter
 - Target-IP: die zufällige IP-Adresse
 - Target-Mac: alles Nullen
 - 3 Gehe zu 1 zurück, falls sich ein Host meldet
 - 4 Ansonsten ist die IP-Adresse jetzt Hostadresse
- Teil von Zeroconf

Loopback

- Klasse A Netzwerk 127.0.0.0/8 ist reserviert für Loopback (Schleifeninterface)
 - Pakete mit einer solchen Adresse als Quelle dürfen den Rechner niemals verlassen!
 - Wird genutzt, um den Senderechner selbst zu adressieren
- ⇒ Transparente Nutzung der Software auch auf nur lokalen Rechner möglich
- IPv4-Adresse ist häufig 127.0.0.1
 - Symbolischer Name dazu häufig localhost

Adressknappheit

- Internet Assigned Numbers Authority (IANA) verteilte IP-Netze an 5 Regional Internet Registries (RIRs)
- Letzte größere Block wurde 2011 vergeben
- Durch Vergabepaxis entstand auch hohe Fragmentierung



- Entwicklung von IPv6 seit 1998 mit [RFC 2460](#) im Gang

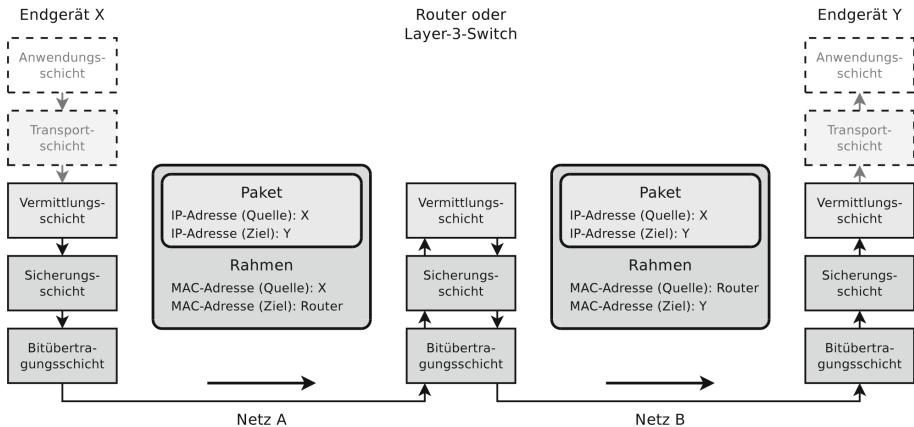
Outline

1 Adressen

2 Routing

3 IP

Endgeräte in zwei verschiedenen Netzen



aus Computernetze kompakt (2018)

Paketweiterleitung

- Aufgabe von Routern ist es, Pakete von einem Host zum anderen Host *weiterzuleiten*
- Oft hat ein Router Verbindung nur zu weiteren Routern und nicht zum Zielgerät
 - Pakete erreichen also nur über Zwischengeräte ihr Ziel
= Hop-by-hop-Routing
- Router wählt anhand der Zieladresse des Pakets und der *Weiterleitungstabelle* die Netzwerkschnitte aus, an der das Paket weitergeleitet wird

Weiterleitungstabelle

Einträge enthalten mindestens:

- Netzwerkanteile (Präfixe), müssen nicht präfixfrei sein
→ Präfixe können andere Präfixe enthalten
- Lokale Weiterleitungsschnittstelle bzw. Interface

Beispiel

Eintrag	CIDR	Interface
1	198.51.100.0/24	1
2	198.51.100.64/26	2
3	198.51.100.128/26	3

Präfix von Eintrag 1 ist in den Präfixen von 2 und 3 enthalten.

Weiterleitungstabelle enthält Teilmenge der *Routing-Tabelle*.
Sie ist optimiert für möglichst schnelle Entscheidungen.

Auswahl der Schnittstelle

Zur Zieladresse eines Pakets wird der Eintrag mit dem *längsten Präfixübereinstimmung* ermittelt. Das Paket wird zu dieser Schnittstelle weitergeleitet.

Es soll ein Paket mit Zieladresse 198.51.100.78 weitergeleitet werden. Welches Interface wird gewählt?

CIDR	Binärsuffix	Interface
198.51.100.78/32	...01100100.01001110	
198.51.100.0/24	...01100100.00000000	1
198.51.100.64/26	...01100100.01000000	2
198.51.100.128/26	...01100100.10000000	3

Routing-Tabelle

- Ist Resultat von Routingalgorithmen und -protokollen
- Übermenge der Weiterleitungstabellen
 - Mehr Informationen über die Topologie
 - Metriken für die Kosten der Verbindung
- Können statisch oder dynamisch erstellt werden

Mit `netstat -r` kann man sich Routing-Tabellen anschauen

```
$ netstat -r
```

```
Kernel IP routing table
```

Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
default	fritz.box	0.0.0.0	UG	0 0		0	wlan0
172.17.0.0	0.0.0.0	255.255.0.0	U	0 0		0	docker0
192.168.178.0	0.0.0.0	255.255.255.0	U	0 0		0	wlan0
fritz.box	0.0.0.0	255.255.255.255	UH	0 0		0	wlan0

Routing

- Prozess, der die Routing-Tabelle befüllt
- Routing-Protokolle setzen *verteilten Algorithmus* um
 - Distanzvektor-Protokolle
 - Link-State-Routing-Protokolle
- Ziel: immer einen möglichst preiswerten Weg nehmen

Distanzvektor-Protokolle

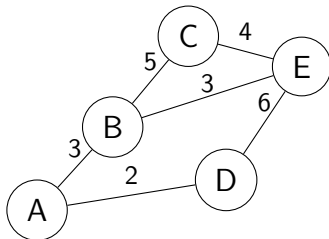
- Dynamischer Routing-Algorithmus
- Idee: Teile Deinen Nachbarn mit, wie Du die Welt siehst
 - 1 Erzeuge Graph, der zu Beginn nur die Knoten und direkten Nachbarn erhält
 - 2 Schicke die Info an *alle* direkten Nachbarn
 - 3 Warte bis Nachbarn ihre Info schicken
 - 4 Aktualisiere Graph
 - 5 Ändert sich dabei etwas, gehe zu Schritt 2, andernfalls Schritte 3
- Vorgehen entspricht dem **Bellman-Ford-Algorithmus**
- Vertreter: Routing Information Protocol (RIP)

Link-State-Routing-Protokolle

- Dynamischer Routing-Algorithmus, der Wissen von Nachbarknoten mit allen Knoten im Netzwerk teilt
- Jeder Knoten
 - ermittelt Zustand und Kosten der Nachbarverbindungen
 - erstellt Datenbank mit Topologie-Informationen
 - übermittelt die Information dann an *alle* anderen
 - Aktualisiert entsprechend der anderen Informationen, den eigenen Graphen (eigene Sicht des Netzes)
 - berechnet die preiswertesten Pfade auf dem Graphen
- Höhere Bandbreite, dafür bessere Dynamik als Distanzvektorverfahren
- Vertreter: Open Shortest Path First (OSPF)

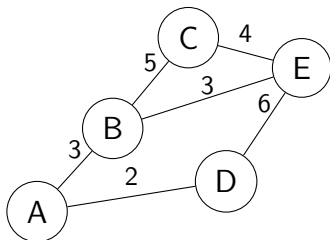
Pfadfindung: Problembeschreibung

Sei $G = (V, E)$ ein Graph mit Knotenmenge V , Kantenmenge E und mit Kostenfunktion $d : E \rightarrow \mathbb{R}$. Finde den preiswertesten Pfad von einem Knoten v zu beliebigen Zielknoten u .



Welcher Pfad ist der preiswerteste, um von A nach E zu gelangen?

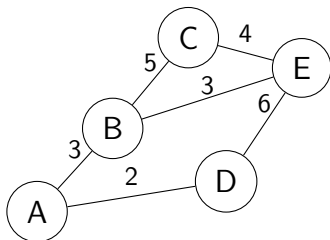
Pfadfindung: Naiver Algorithmus



- 1 Liste alle Pfade von A nach E auf
- 2 Bestimme die Kosten der Pfade und wähle preiswertesten

1) A – B – C – E:

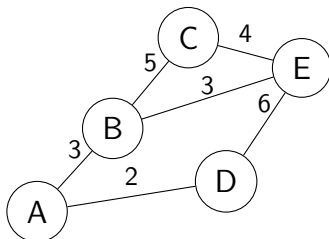
Pfadfindung: Naiver Algorithmus



- 1 Liste alle Pfade von A nach E auf
- 2 Bestimme die Kosten der Pfade und wähle preiswertesten

1) A – B – C – E: $3 + 5 + 4 = 12$; 2) A – B – E:

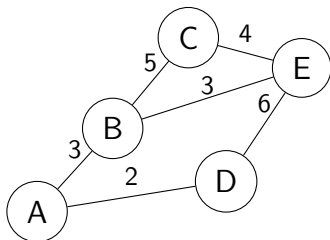
Pfadfindung: Naiver Algorithmus



- 1) Liste alle Pfade von A nach E auf
- 2) Bestimme die Kosten der Pfade und wähle preiswertesten

1) A – B – C – E: $3 + 5 + 4 = 12$; 2) A – B – E: $3 + 3 = 6$
3) A – D – E:

Pfadfindung: Naiver Algorithmus

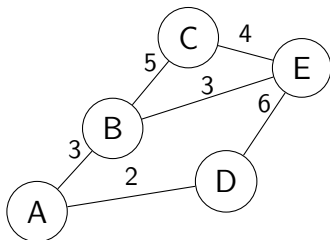


- 1) Liste alle Pfade von A nach E auf
- 2) Bestimme die Kosten der Pfade und wähle preiswertesten

1) A – B – C – E: $3 + 5 + 4 = 12$; 2) A – B – E: $3 + 3 = 6$
3) A – D – E: $2 + 6 = 8$

Sehr ineffizient bei großen Graphen! Wie viele Pfade zw. zwei Knoten beim vollständigen Graphen?

Pfadfindung: Naiver Algorithmus



- 1) Liste alle Pfade von A nach E auf
- 2) Bestimme die Kosten der Pfade und wähle preiswertesten

1) A – B – C – E: $3 + 5 + 4 = 12$; 2) A – B – E: $3 + 3 = 6$
3) A – D – E: $2 + 6 = 8$

Sehr ineffizient bei großen Graphen! Wie viele Pfade zw. zwei Knoten beim vollständigen Graphen? $(n - 1)!$

Pfadfindung: Dijkstra

Dijkstra: Finde kürzesten Pfad vom Start- zum Endknoten

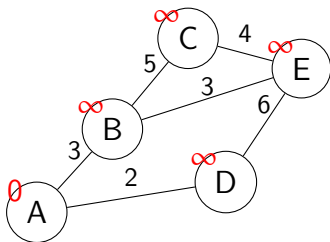
Knoten haben Eigenschaften: *Distanz* und *Vorgänger*

- ① Setze *Distanz* des Startknotens auf 0 und die aller anderen auf ∞
- ② Solange es unbesuchte Knoten gibt:
 - ① Wähle v als denjenigen mit minimaler Distanz
 - ② Berechne neue Distanzen zu allen Nachbarn von v
 - ③ Ist dabei neue Distanz kleiner als bisheriger Wert
 - ① Merke neue *Distanz* zu Nachbarn
 - ② Merke v als *Vorgänger* zu Nachbarn
 - ④ v ist jetzt besucht

Komplexität: $O(|V| \log |V| + |E|)$

Pfadfindung: Initialisierung beim Beispielgraph

Nach Initialisierung

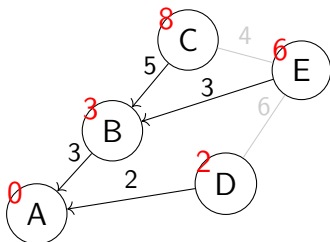


Pfadfindung: Ergebnis beim Beispielgraph

Nach letzter Iteration

Pfadfindung: Ergebnis beim Beispielgraph

Nach letzter Iteration



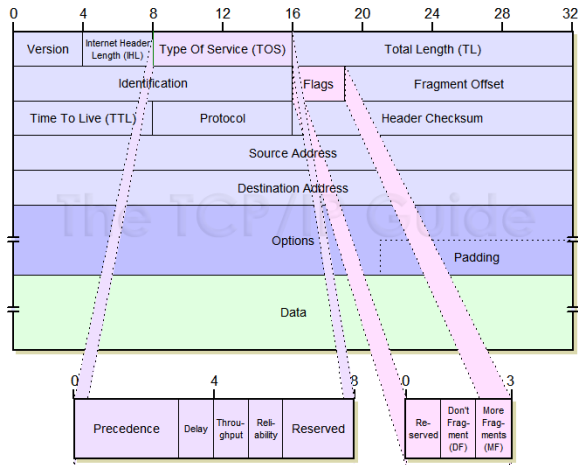
Outline

1 Adressen

2 Routing

3 IP

IPv4: Headerformat



- Ursprünglich definiert in [RFC 791](#)
- TOS jetzt Differentiated Services Codepoint (DSCP)

IPv4: Beschreibung der Felder

Beschreibung der Felder z. B. hier:

- [RFC 791](#) (lokal)
- <https://tools.ietf.org/html/rfc791>
- http://www.tcpipguide.com/free/t_IPDatagramGeneralFormat.htm
- <https://www.elektronik-kompodium.de/sites/net/2011241.htm>

IPv4: Checksumme

RFC 791 definiert:

The checksum field is the 16-bit one's complement of the one's complement sum of all 16-bit words in the header. For purposes of computing the checksum, the value of the checksum field is zero.

Addition im 1er-Komplement: Bei Übertrag muss 1 dazu addiert werden.

Bestimme die Checksumme vom IP-Header mit Inhalt: 4500
0073 0000 4000 4011 XXXX c0a8 0001 c0a8 00c7

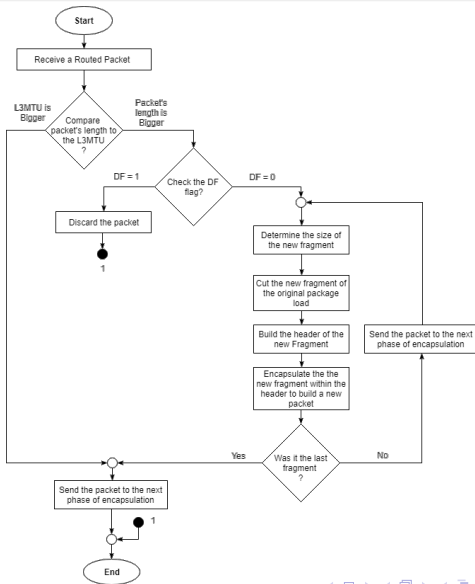
XXXX ist die Stelle, an der die Checksumme eintragen wird.

Wie verifizieren?

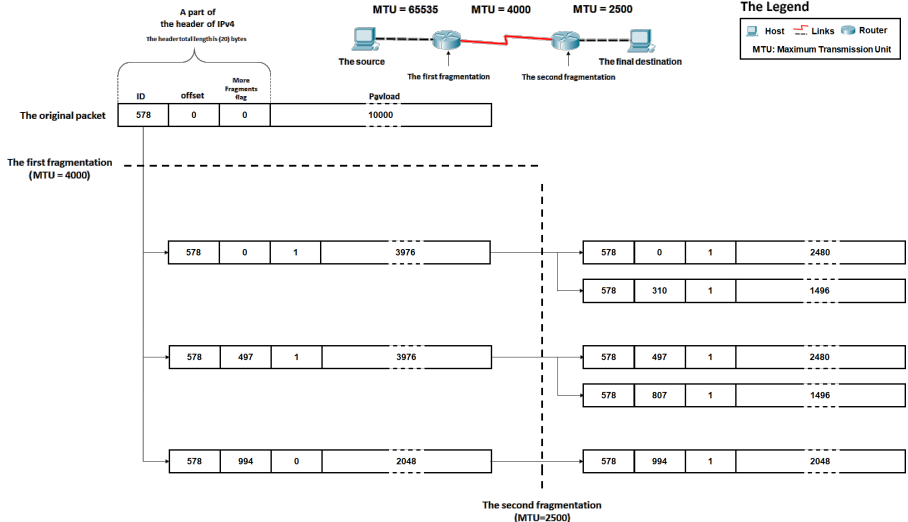
Fragmentieren von IPv4 Paketen

- IPv4 Pakete können bis zu 64 KiB groß werden
- Untere Schichten verarbeiten meistens weniger
- *Maximum Transmission Unit* (MTU) gibt Grenze an
 - Ethernet: 1500 B
 - PPPoE (bei DSL): 1492 B
 - WLAN: 2312 B
- Pakete mit größeren Nutzdaten werden deshalb je nach DF-Bit von der Vermittlungsschicht in MTU-Große Fragmente unterteilt oder verworfen
- Beim Empfänger werden sie wieder zusammengefügt

Fragmentieren: Algorithmus



Fragmentieren: Beispiel



Probleme durch das Fragmentieren

- Obere Schichten können nicht in Erfahrung bringen, ob Paket fragmentiert wird oder war
- Wenn Übertragung größerer Nutzdaten fehlschlägt, müssen ggf. nach Timeout gesamte Nutzdaten neu übertragen werden
- Implementation waren oft Buggy ([Teardrop Attacken](#))
- Don't Fragment Flag

ICMP

- Verkapselt Informations- und Fehlermeldungen über IPv4
- Ist selbst Payload in IPv4 Paketen
- Gehört trotzdem zur selben Schicht
- Wird z. B. von Ping oder Traceroute genutzt
- Beschreibung der Felder:

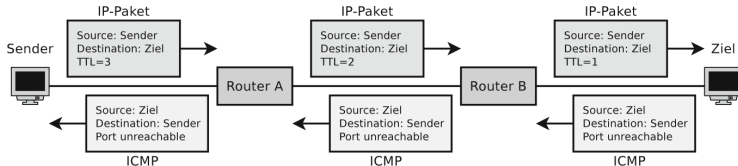
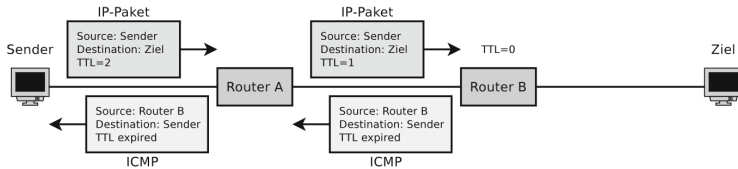
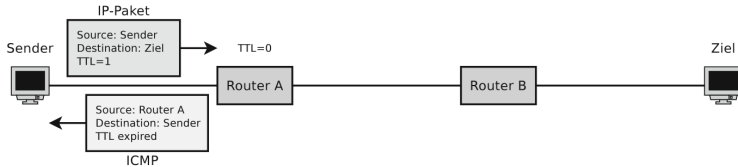
http://www.tcpipguide.com/free/t_ICMPCommonMessageFormatandDataEncapsulation.htm

Ping Trace: Request

No.	Time	Source	Destination	Protocol	Length	Info
74	8.317372499	141.45.46.191	141.45.66.214	ICMP	98	Echo (ping) request id=0x263d, seq=1/256, ttl=64 (reply in 78)
78	8.319636105	141.45.66.214	141.45.46.191	ICMP	98	Echo (ping) reply id=0x263d, seq=1/256, ttl=63 (request in 74)
84	9.318836904	141.45.46.191	141.45.66.214	ICMP	98	Echo (ping) request id=0x263d, seq=2/512, ttl=64 (reply in 85)
85	9.319996726	141.45.66.214	141.45.46.191	ICMP	98	Echo (ping) reply id=0x263d, seq=2/512, ttl=63 (request in 84)
116	10.319899437	141.45.46.191	141.45.66.214	ICMP	98	Echo (ping) request id=0x263d, seq=3/768, ttl=64 (reply in 117)
117	10.321487542	141.45.66.214	141.45.46.191	ICMP	98	Echo (ping) reply id=0x263d, seq=3/768, ttl=63 (request in 116)

<ul style="list-style-type: none"> Frame 74: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0 Ethernet II, Src: IntelCor_0e:b0:dc (0c:54:15:0e:b0:dc), Dst: IETF-VRRP-VRID_ac (00:00:5e:00:01:ac) Internet Protocol Version 4, Src: 141.45.46.191, Dst: 141.45.66.214 <ul style="list-style-type: none"> 0100 = Version: 4 0101 = Header Length: 20 bytes (5) Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT) Total Length: 84 Identification: 0xa7f6 (42998) Flags: 0x4000, Don't fragment Time to live: 64 Protocol: ICMP (1) Header checksum: 0x06c3 [validation disabled] [Header checksum status: Unverified] Source: 141.45.46.191 Destination: 141.45.66.214 Internet Control Message Protocol <ul style="list-style-type: none"> Type: 8 (Echo (ping) request) Code: 0 Checksum: 0x3965 [correct] [Checksum Status: Good] Identifier (BE): 9789 (0x263d) Identifier (LE): 15654 (0x3d26) Sequence number (BE): 1 (0x0001) Sequence number (LE): 256 (0x0100) [Response frame: 78] 	<pre> 0000 00 00 5e 00 01 ac 0c 54 15 0e b0 dc 08 00 45 00 ..A...T....E. 0010 00 54 a7 f6 40 00 00 01 06 c3 8d 2d 2e bf 80 2d -T...@-..... 0020 42 d6 08 00 39 65 26 3d 00 01 82 48 d6 5d 00 00 B...9e&=...H.] 0030 00 00 74 e3 0c 00 00 00 00 00 10 11 12 13 14 15 -t..... </pre>
--	--

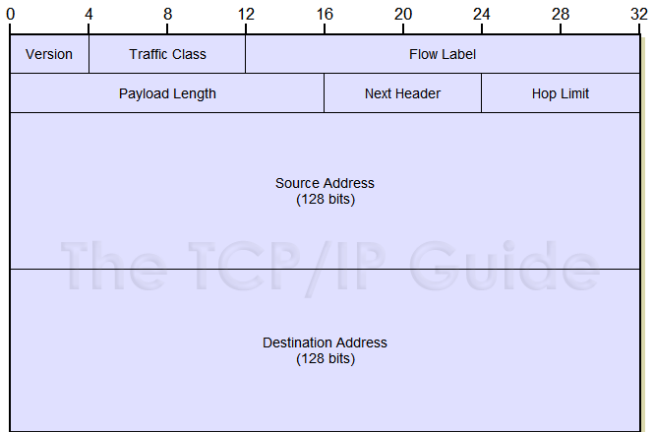
Traceroute via ICMP



Path MTU Discovery

Funktioniert nicht bei Firewall!

IPv6: Headerformat



- Ursprünglich definiert in [RFC 2460](#)

IPv6: Beschreibung der Felder

- RFC 2460 (lokal)
- <https://tools.ietf.org/html/rfc2460>
- http://www.tcpipguide.com/free/t_IPv6DatagramMainHeaderFormat.htm
- <https://www.elektronik-kompodium.de/sites/net/1902121.htm>

ICMPv6

IPv6: Fragmentierung

- Router müssen zu große Pakete mit ICMPv6-Nachricht zurückweisen

Entweder MTU-Path-Discovery

Oder Fragmentierung passiert nur noch beim Sender
(Fragment-Header-Extension)