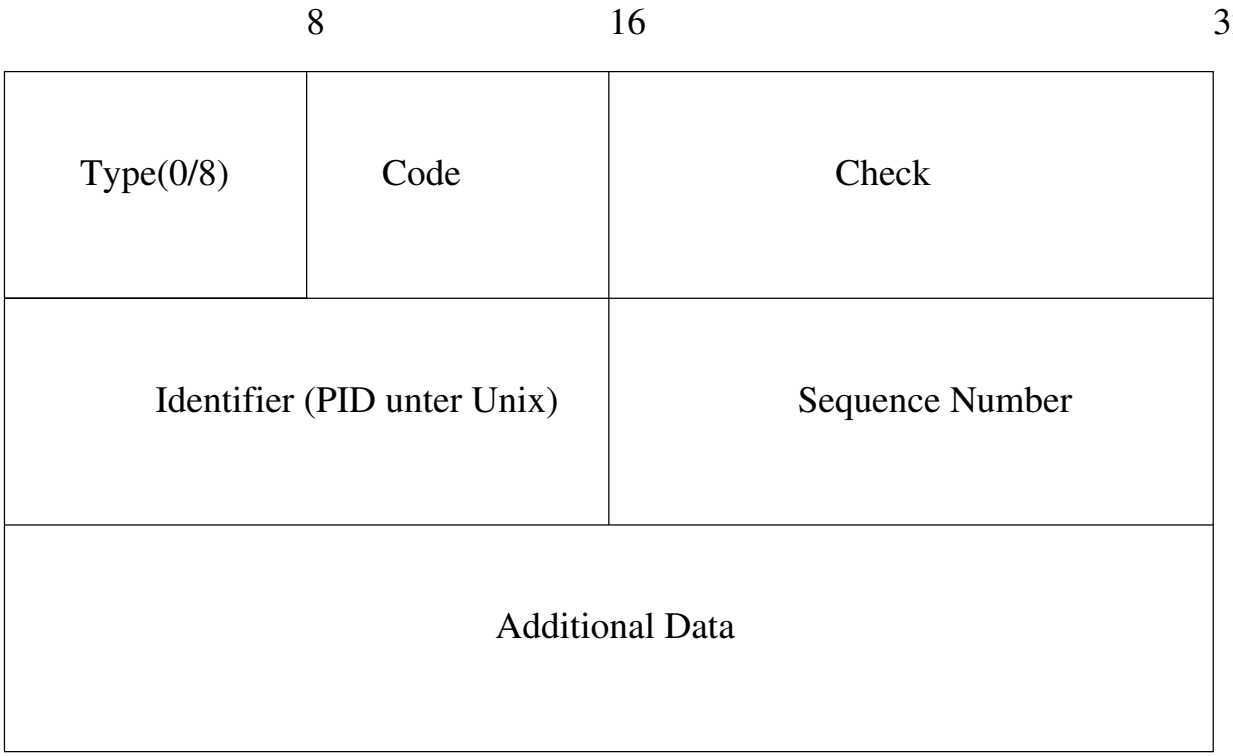
Wichtige ICMP Typen

Typ	Name
0	Echo Reply
3	Destination Unreachable
4	Source Quench
5	Redirect
8	Echo Request
9	Router Advertisement
10	Router Solicitation
11	Time Exceeded
12	Parameter Problem
13	Timestamp Request
14	Timestamp Reply
17	Address Mask Request
18	Address Mask Reply

ICMP Echo Request/Reply

Test, ob die Gegenseite auf IP-Ebene erreichbar ist:
Type 8, Code 0 für Request, Type 0, Code 0 für Reply



ICMP Echo Request/Reply

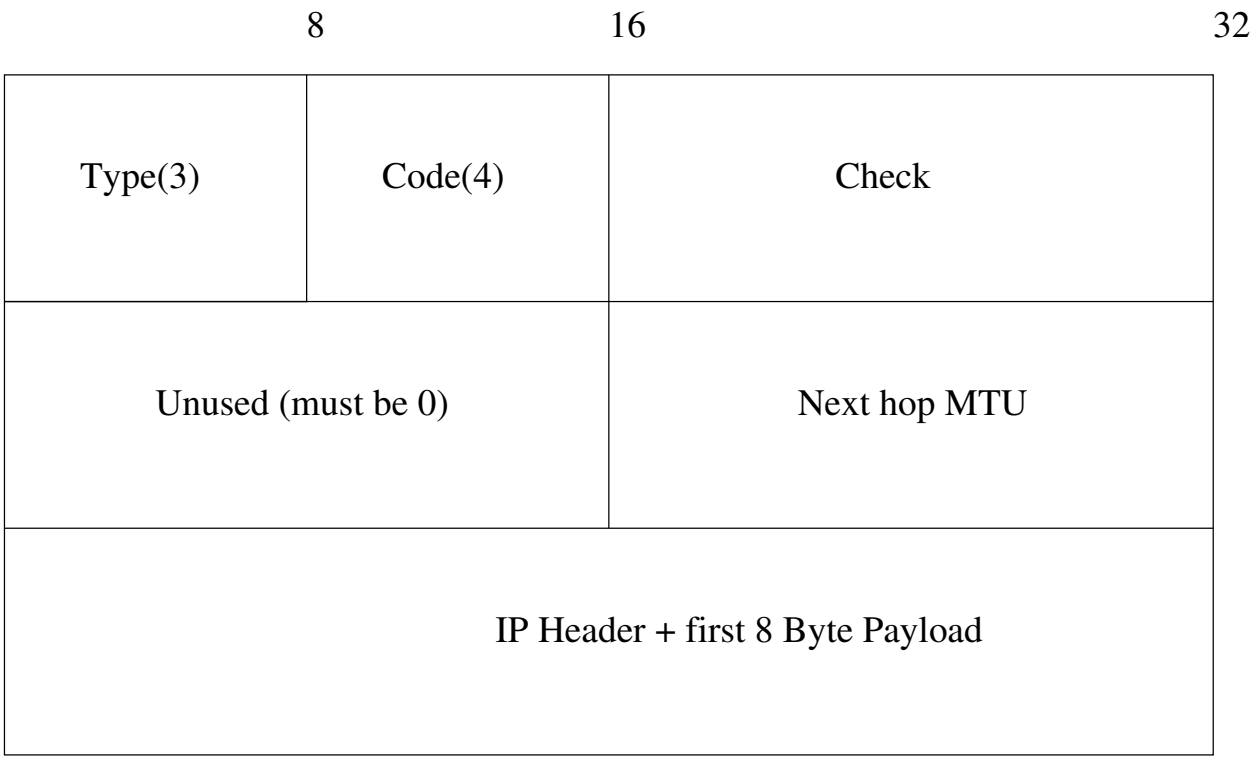
- Ein Host wird einen ICMP Echo Request beantworten, indem er die Felder **Indentifier**, **Sequence Number** und **Additional Data** in die Antwort kopiert.
- **Identifier:** Dieses Feld wird benutzt, damit ein Reply dem Sender des Requests zugeordnet werden kann. Unix benutzt oft die Process ID (Endianess beachten!).
- **Sequence Number:** Fortlaufende Nummer, dei benutzt wird, um Request und Reply einander zuzuordnen. **ping** benutzt das zur Laufzeitmessung.
- **Additional Data:** Beliebige Daten, die vom Server zurückgeschickt werden.

ICMP Destination Unreachale

Code	Name
0	network unreachable
1	host unreachable
2	protocol unreachable
3	port unreachable
4	fragmentation needed but don't-fragment bit set
5	source route failed
6	destination network unknown
7	destination host unknown
9	destination network administratively prohibited
10	destination host administratively prohibited
11	network unreachable for TOS
12	host unreachable for TOS
13	communication administratively prohibited by filtering

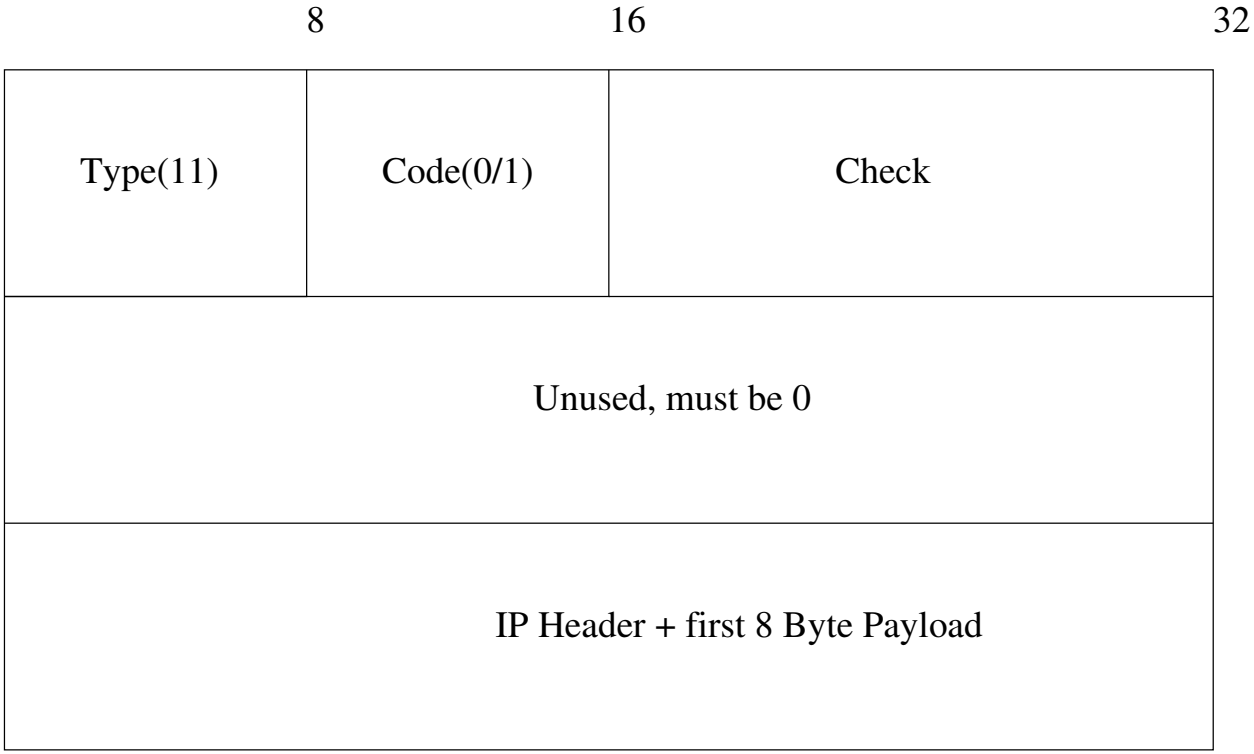
Beispiel: Fragmentation Required

Fragmentation Required (Code 4) wird erzeugt, wenn die MTU des nächsten Links nicht ausreicht, um das IP Paket zu übertragen, aber DF gesetzt ist.



ICMP Time Exceeded

Time Exceeded (Typ 11) wird erzeugt, wenn in einem Router der Wert des TTL Feldes auf 0 fällt (Code 0), oder zu lange auf ein Fragment gewartet werden muß (Code 1).



Traceroute, Van Jacobson, 1988

- Programm zu Bestimmung der Route zu einem Zielhost
- Windows Implementation (tracert) benutzt ICMP Echo Request (Unix Impementationen benutzen UDP)
- **Algorithmus:**

```
Setze TTL = 1
Bis ICMP Echo Reply empfangen wurde:
  Sende drei ICMP Echo Request
  Gebe Laufzeit bis zu den Antworten aus
  TTL = TTL + 1
```

tracert Beispiel

Die Ausgabe von **tracert** von einem Windows Host:

```
C:\>tracert -d 10.1.180.33

Tracing route to 10.1.180.33
1 <10 ms <10 ms <10 ms 172.16.199.1
2 <10 ms <10 ms 16 ms 10.1.195.2
3 <10 ms <10 ms <10 ms 10.1.180.33

Trace complete.
```

tracert Netzwerktrace

Jeder der drei Dialoge wurde je dreimal aufgezeichnet mittels `tcpdump -ttt -vn`:

```
000000 IP (ttl 1) 172.16.199.122 > 10.1.180.33:
  ICMP echo request, id 512, seq 10240, length 72
000196 IP (ttl 64) 172.16.199.1 > 172.16.199.122:
  ICMP time exceeded in-transit, length 100

000000 IP (ttl 2) 172.16.199.122 > 10.1.180.33:
  ICMP echo request, id 512, seq 11520, length 72
001224 IP (ttl 254) 10.1.195.2 > 172.16.199.122:
  ICMP time exceeded in-transit, length 36

000000 IP (ttl 3) 172.16.199.122 > 10.1.180.33:
  ICMP echo request, id 512, seq 12032, length 72
000204 IP (ttl 253) 10.1.180.33 > 172.16.199.122:
  ICMP echo reply, id 512, seq 12032, length 72
```

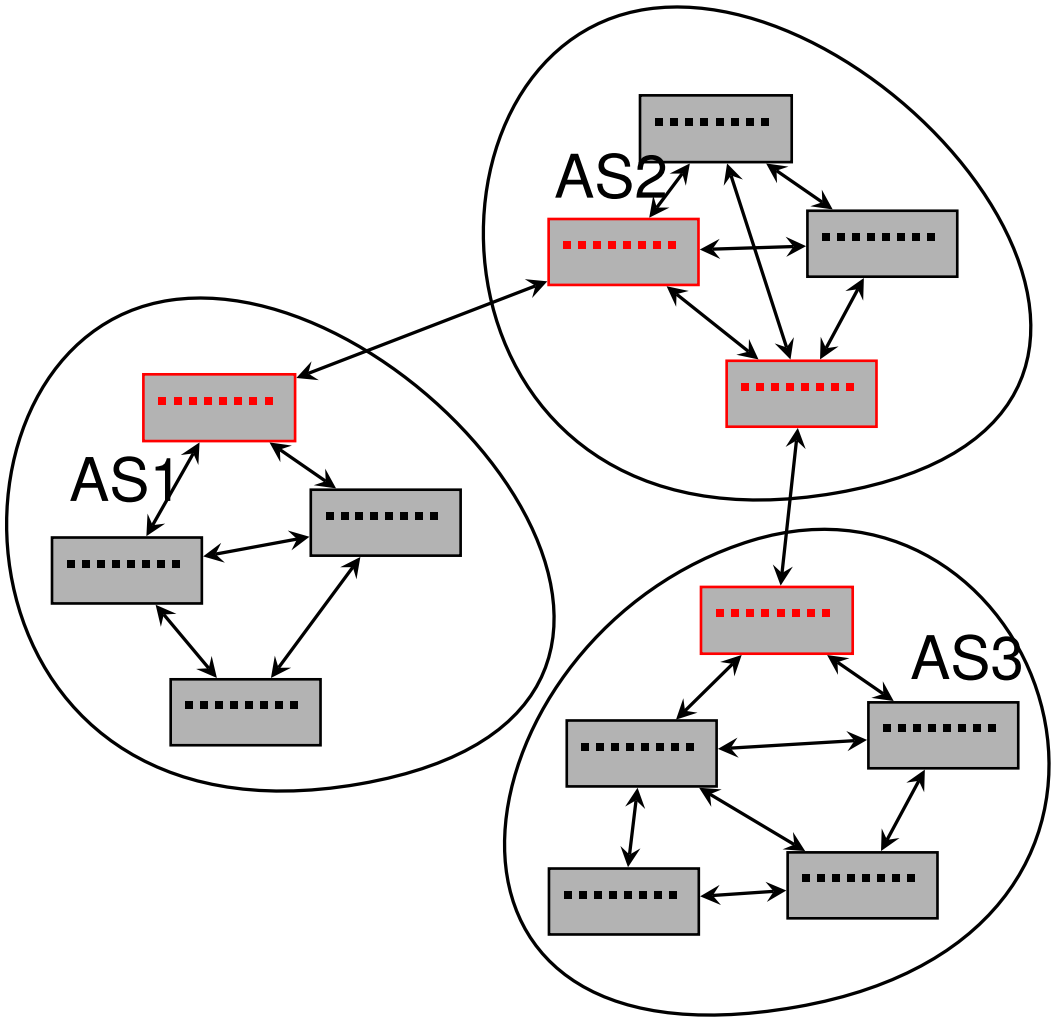
Grundlage

- Das Internet gliedert sich in Bereiche unterschiedlicher administrativer Verantwortung (z.B. Veratwortung eines ISPs), sog. **autonome Systeme** (**AS**).
- Es muß unterschieden werden zwischen Routing innerhalb eines AS (intra-AS) und zwischen verschiedenen AS (inter-AS), da hier unterschiedliche Anforderung an den Austausch von Routinginformation gestellt werden.
- Die Knoten an den Verbindungsstellen zwischen AS heißen **Gateway Router**.

Anforderungen an Routingprotokolle

- **Skalierbakeit:** Besonsers ein inter-AS Routingprotokoll muß mit wachsender Anzahl AS skalieren können, da es keine Möglichkeit gibt, die Anzahl zu kontrollieren/reduzieren.
- **Leistungsfähigkeit:** Das Protokoll sollte in der Lage sein, schnell auf Änderungen in der Netzstruktur zu reagieren, administrative Vorgaben in die Routenplanung zu integrieren und gute Routen zu generieren.
- **Flexibilität:** Das Protokoll muß in der Lage sein, administrative Vorgaben in die Planung einzubauen. Besonders wichtig ist das beim inter-AS Routing, da z.B. keine/unterschiedliche vertragliche Beziehung zwischen den Betreibern einzelner AS bestehen.

Beispielnetz



Bezeichnungen aus der Graphenthorie

- **Graph:** Ein 2-Tupel $G = (V, E)$ ist ein (ungerichteter) Graph, falls $V \neq \emptyset$ und $E = \{ \{v_1, v_2\} \mid v_1, v_2 \in V\}$.
- **Gewichteter Graph:** Ein Graph $G = (V, E)$ mit einer Funktion $c : E \rightarrow \mathbb{R}$ heißt gewichteter Graph.
- **adjazent:** Zwei Ecken $v_1, v_2 \in V$ heißen adjazent, falls $\{v_1, v_2\} \in E$.
- **zusammenhängend:** Ein Graph heißt zusammenhängend, falls es zu jedem Paar $w_1, w_2 \in V$ eine Folge $v_1, \ldots, v_n \in V$ gibt, so daß $v_i, v_{i+1}\ \forall i$ und w_1, v_1 bzw. v_n, w_2 adjazent sind.

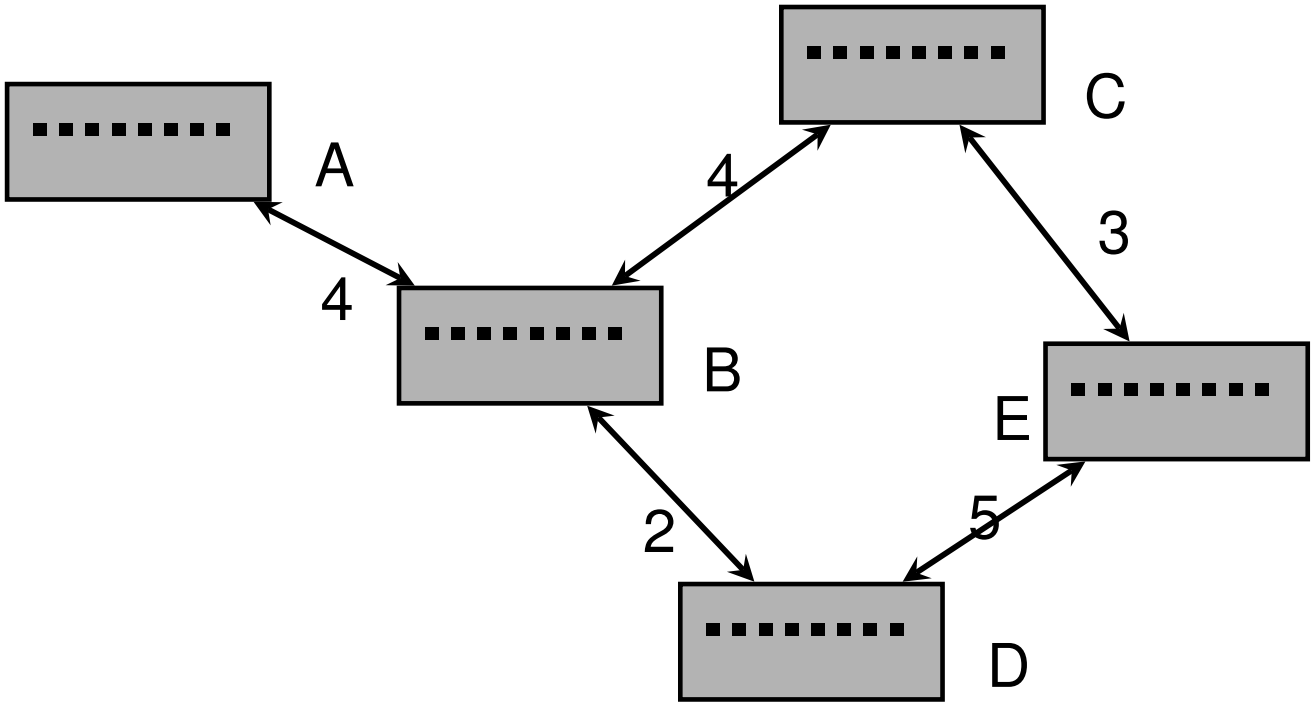
V ist die Menge der Ecken (Vertices) eines Graphen, E die Menge der Kanten. Die Funktion c ordnet jeder Kante $e \in E$ ein Gewicht zu. Oft läßt sich das Gewicht als Kosten oder Distanz interpretieren.

Übersicht über Routingverfahren

- **Distance Vector Algorithms:** Benutzt den “Verteilten Bellman Ford Algorithmus”, hat Probleme in großen Netzen. Kommunikation ist nur mit unmittelbaren Nachbarn notwendig.
- **Link State Algorithms:** Jeder Router bildet einen gewichteten Graph des gesamten Netzes. Die Routingtabelle ergibt sich durch den Shortest Path Algorithm. Das Verfahren hat Probleme in großen Netzen und erzeugt hohe Netz- und CPULast.
- **Path Vector Protocol:** Ähnlich den Distance Vector Algorithms, aber es werden nur ausgewählte Hosts (Speaker Node) einbezogen.

Beispiel eines AS

- 5 Router innerhalb des autonomen Systems
- 5 Links mit unterschiedlichen Kosten



Bellmann-Ford Algorithmus: Initialisierung

Wir betrachten ungerichteten Graphen $G = (V, E)$ mit Gewicht c. Ziel ist, zu jeder Ecke v die minimale Distanz $D_v(w)$ zu jeder anderen Ecke w zu bestimmen.

Initialisierung:

$$D_v(w) := \begin{cases} 0 & \text{falls } v = w \\ c(\{v, w\}) & \text{falls } v, w \text{ adjazent} \\ \infty & \text{sonst} \end{cases}$$

$D_w(y) := \infty$ für alle $w \in V,\ v, w$ adjazent.
Sende Distanzvektor $\mathbf{D}_v := (D_v(y),\ y \in V)$ an alle Nachbarn von v.

Bellmann-Ford Algorithmus: Hauptschleife

Der Algorithmus besteht aus drei Schritten:

1. Empfange Distanzvektor $\mathbf{D}_w$ von Nachbar w.
2. Für alle $y \in V$ setzte
$$D_v(y) := \min\left\{ c(\{w, y\}) + D_w(y),\ w \in V \right\}$$
3. Falls $\mathbf{D}_v$ verändert wurde, sende $\mathbf{D}_v$ an alle Nachbarn.

Wann immer ein Distanzvektor von einem Nachbarn w empfangen wird, prüfe, ob sich dadurch ein besserer Weg zu einem der Knoten im Netz ergibt, d.h. ist die Summe

- Weg nach w
- Weg von w zum Ziel

günstiger als der bisher bekannte Weg.

Bellmann-Ford im Beispiel AS

	A	B	C	D	E
A	0	**4**	∞	∞	∞
B	**4**	0	**4**	**2**	∞
C	∞	**4**	0	∞	**3**
D	∞	**2**	∞	0	**5**
E	∞	∞	**3**	**5**	0

C sendet Distanzvektor an B und E:

	A	B	C	D	E
A	0	**4**	∞	∞	∞
B	**4**	0	**4**	**2**	7(C)
C	∞	**4**	0	∞	**3**
D	∞	**2**	∞	0	**5**
E	∞	7(C)	**3**	**5**	0

Bellmann-Ford im Beispiel AS

B sendet Distanzvektor an A,C,D:

	A	B	C	D	E
A	0	**4**	8(B)	6(B)	11(B)
B	**4**	0	**4**	**2**	7(*C*)
C	8(B)	**4**	0	6(B)	**3**
D	6(B)	**2**	6(B)	0	**5**
E	∞	7(C)	**3**	**5**	0

D sendet Distanzvektor an B,E:

	A	B	C	D	E
A	0	**4**	8(B)	6(B)	11(B)
B	**4**	0	**4**	**2**	7(*C*)
C	8(B)	**4**	0	6(B)	**3**
D	6(B)	**2**	6(B)	0	**5**
E	11(D)	7(C)	**3**	**5**	0

Dijkstra Algorithmus

Seien G und c wie im Bellmann-Ford Algorithmus. Sei ferner $u \in V$ eine Ecke des Graphen.

Initialisierung:

Setze $N := \{u\}$ und

$$D(v) := \begin{cases} 0 & \text{falls } u = v, \\ c(\{u, v\}) & \text{falls } u, v \text{ adjazent}, \\ \infty & (\textit{sonst}) \end{cases}$$

Dijkstra Algorithmus

do
finde $w \in V \setminus N$, so daß $D(w) < D(w') \; \forall \; w' \in V \setminus N$
$N := N \cup \{w\}$
Für alle v mit $v \in V \setminus N$, v, w adjazent bilde
$D(v) := \min\{D(v), D(w) + c(\{v, w\})\}$
while $N \neq V$

- Während des Ablaufes gibt $D(v)$ stets für alle $v \in V$ die beste bis dahin gefundene Distanz zwischen u und v an.
- N enthält die Ecken, für die das Ergebnis schon feststeht.
- Initial ist $N = \{u\}$ mit Distanz $D(u) = 0$.
- In jedem Schritt wird die nächstgelegene Ecke außerhalb N betrachtet und geprüft, ob Routing über diese Ecke den Weg zu anderen Ecken verkürzt.

Dijkstra Algorithmus im Beispiel AS

Wir betrachten Ecke A als Ausgangspunkt, in der Tabelle sind für jeden Schritt die aktuellen Werte von $D(v)$ für $v \in \{ABCDE\}$ aufgetragen.

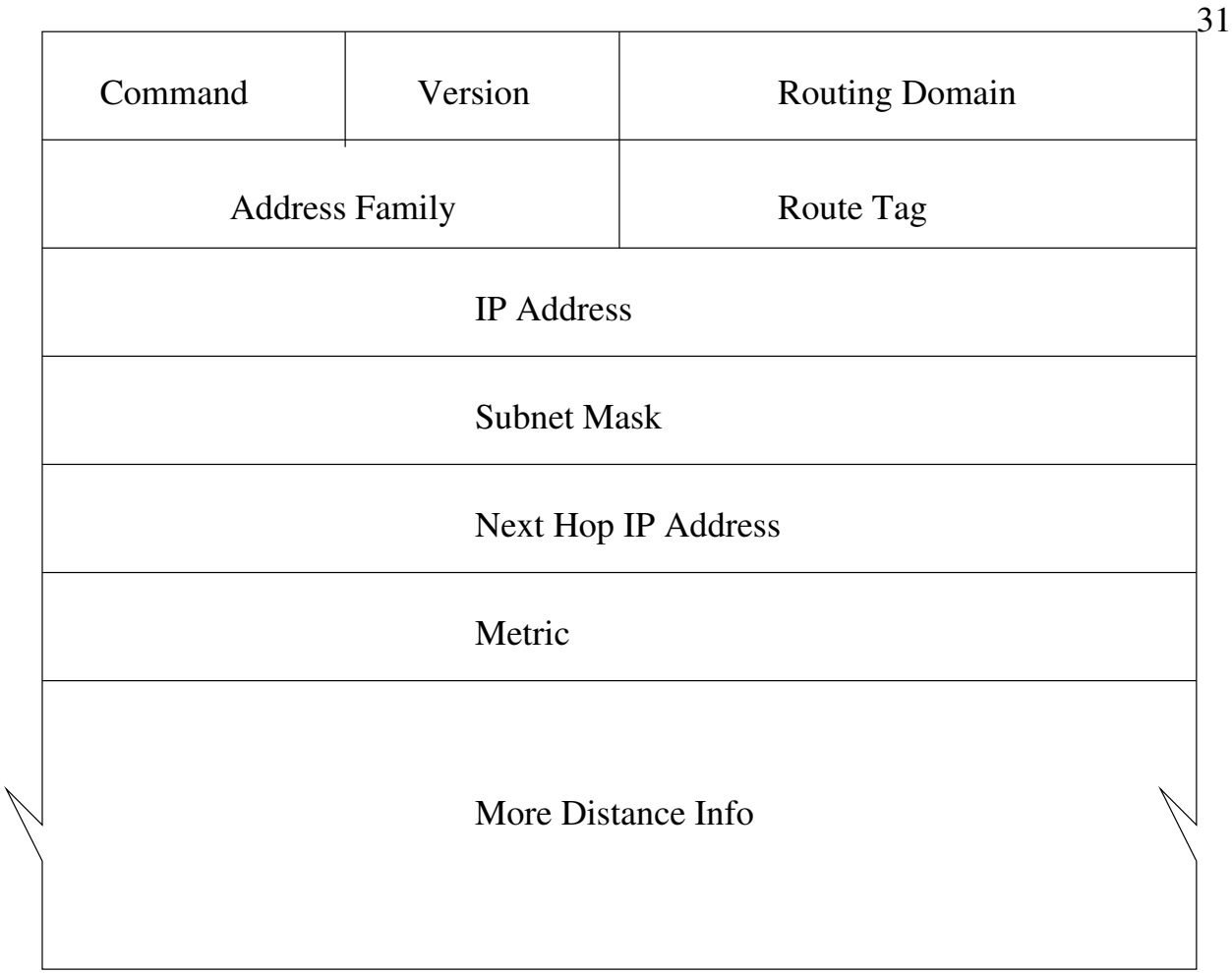
Schritt	A	B	C	D	E	N
1	0	4(B)	∞	∞	∞	$\{A\}$
2	0	4(B)	8(B)	6(B)	∞	$\{A, B\}$
3	0	4(B)	8(B)	6(B)	11(B)	$\{A, B, D\}$
4	0	4(B)	8(B)	6(B)	11(B)	$\{A, B, C, D\}$
5	0	4(B)	8(B)	6(B)	11(B)	$\{A, B, C, D, E\}$

Dijkstra Algorithmus im Beispiel AS

Entsprechend für Ecke C als Ausgangspunkt:

Schritt	A	B	C	D	E	N
1	∞	4(B)	0	∞	3(E)	$\{C\}$
2	∞	4(B)	0	8(E)	3(E)	$\{C, E\}$
3	8(B)	4(B)	0	6(B)	3(E)	$\{B, C, E\}$
4	8(B)	4(B)	0	6(B)	3(E)	$\{B, C, D, E\}$
5	8(B)	4(B)	0	6(B)	3(E)	$\{A, B, C, D, E\}$

RIP Version 2 (vgl. RFC2453)



RIP Version 2 ist ein Distance Vector basiertes Verfahren.

- **Command:** Entweder Request(1) oder Reply(2)
- **Version:** RIP Version 2 (RIP Version 1 unterstützt kein CIDR, das Rahmenformat ist gleich)
- **Address Family:** Für IP immer 2
- **Route Tag:** Tag zur Identifikation des AS (ASN, RFC1930)
- **IP Address:** Addresse des Subnetzes, zu dem die Information gehört
- **Subnet Mask:** Netzmaske des Subnetzes
- **Next Hop IP Address:** IP Addresse des Routers, über den das Subnetz erreicht werden kann
- **Metric:** Kosten, bei RIP immer Anzahl Hops zum Ziel, max. 15
- Pro Rahmen können bis zu 25 Distanzinformationen von je 20 Byte gesendet werden.

- Bei Start schickt der Router einen RIP2 Request mit Address Family 0 (statt 2) auf allen Interfacen. Dieser fordert von allen Routern den kompletten Satz Distanzvektoren an.
- In einem Request mit Address Family 2 wird jeder Eintrag im Rahmen bearbeitet. Falls eine Route vorhanden ist, setze Metric, andernfalls setze Metric auf 16 (unendlich).
- Wird ein Reply empfangen, verwende den Distance Vector Algorithm zum Neuaufbau der Routingtabelle.
- Alle 30 Sekunden wird die komplette Routingtabelle an alle Nachbarn verschickt.
- Bei jeder Veränderung der eigenen Routingtabelle werden die Änderungen der Metric übertragen.
- Jeder Routingeintrag verfällt nach max. 2 Minuten, wenn er nicht erneuert wird.

Open Shortest Path First (vgl. RFC2328)

- OSPF Daten werden im IP Rahmen übertragen (Protocol 89)
- OSPF benutzt den Link State Algorithmus zur Berechnung der Routingtabelle.
- Anwendung im Intra-AS Routing
- Bei OSPF werden die Linkstati eines Routers zu allen Routern des AS übertragen.
- Jeder Router berechnet mit dem Dijkstra Algorithmus die Route geringster Kosten zu allen anderen Routern.
- Links werden mit OSPF Paketen auf Funktionalität geprüft.

Eigenschaften von OSPF

- OSPF unterstützt echte Authentifizierung.
- Mehrere Pfade mit gleichen Kosten können parallel benutzt werden (Load Balancing).
- OSPF unterstützt verschiedene Routen abhängig vom TOS Feld.
- Kosten eines Links sind dimensionslos, unterschiedliche Kosten können für unterschiedliche Werte des TOS Feldes benutzt werden.
- Routen müssen nicht durch IP Adressen identifiziert werden (vgl. PPP)
- OSPF erlaubt eine hierarchische Aufteilung des AS in kleinere AS, die durch **Area Border Router** mit dem **Backbone AS** verbunden sind.

Border Gateway Protocol Version 4 (vgl. RFC4271)

- Beispiel eines Path-Vector Protokolls
- Dient als Inter-AS Routing Protokoll im Internet
- BGP4 muß von jedem ISP eingesetzt werden, um Routing zu anderen AS zu unterstützen.
- In Benutzung seit 1994 (Version 4 unterstützt im Gegensatz zu Version 3 CIDR)
- Routingentscheidungen zwischen AS basieren nicht auf Kosten sondern auf Regeln.

Bezeichnungen

- **BGP Speaker:** Knoten, der das BGP Protokoll implementiert.
- **eBGP Kummunikation:** BGP Datenaustausch zwischen AS
- **iBGP Kummunikation:** Datenaustausch zwischen BGP Speaker und seinem AS
- **Präfix:** Netwerkidentifikation bestehend aus IP und Maske, z.B. 134.130.35.128/25
- **BGP Attribut:** Zusätzliche Information zu einem Präfix, besonders wichtig sind die Attribute AS-Path und Next-Hop
- **Route:** Kombination von Präfix und zugehörigen Attributen

Generelle Arbeitsweise

- Jede Router hat eine Liste seiner Nachbarn, zu denen er eine Verbindung aufbaut.
- Verbindungen werden durch einen Keepalive Mechanismus permanent überwacht.
- Gateway Router tauschen via eBGP Kummunikation die verwendeten Routen mit ihren Nachbarn aus.
- Router innerhalb des AS informieren sich gegenseitig die verwendeten Routen via iBGP Kommunikation
- Aus den möglichen Routen wählt jeder Router anhand seiner Konfiguration die für ihn beste aus.

Routenerzeugung

- Wird ein Präfix von einem Router via eBGP an einen anderen Router gemeldet, fügt er seine ASN (vgl. RFC1930) an das AS-Path Attribut an und setzt das Next-Hop Attribut auf die IP Adresse des externen Interfaces.
- Jeder Router innerhalb eines AS wird eine Route zum Präfix über den Next-Hop berechnen. Dazu wird ein Intra-AS Routingverfahren (z.B. OSPF) verwendet.

Routenselektion

Ein Router kann verschiedene Routen zum selben Ziel über verschiedene BGP Speaker empfangen.

- Kann der Next-Hop nicht erreicht werden, verwerfe den Pfad
- Wähle die Route mit der höchsten Präferenz (wird über Regeln vom Administrator festgelegt).
- Bei gleicher Präferenz, wähle die kürzeste Route, d.h. mit dem kürzesten AS-Path Attribut.
- Unter den verbliebenen Routen, wähle die mit den günstigsten Kosten zum Next-Hop.