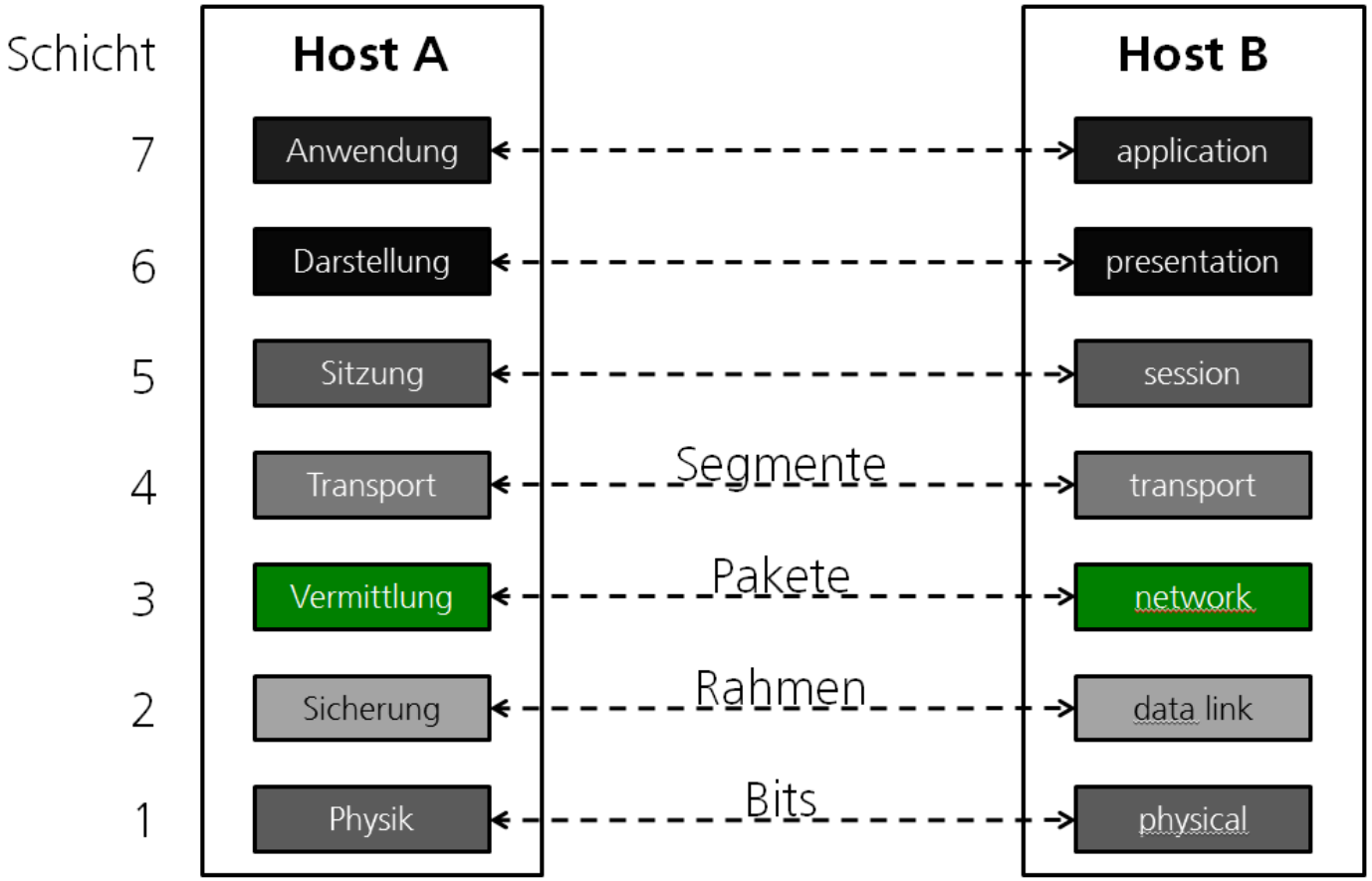


Vermittlungsschicht

Schicht 3: Netzwerk- oder Vermittlungsschicht

Ab Schicht 3 des OSI-Referenzmodells werden über physikalische Netzwerkgrenzen hinweg Nachrichten ausgetauscht, ein logischer Verbund von Teilnetzwerken entsteht.

Jeder Teilnehmer (Knoten) in diesem Verbund soll mit jedem anderen Knoten kommunizieren können. Es braucht daher

1. für jeden Knoten eine eindeutige Adresse, die ihn in dem Verbund identifiziert,
2. Mechanismen zur Vermittlung von Daten zwischen beliebigen Knoten des Netzwerks.

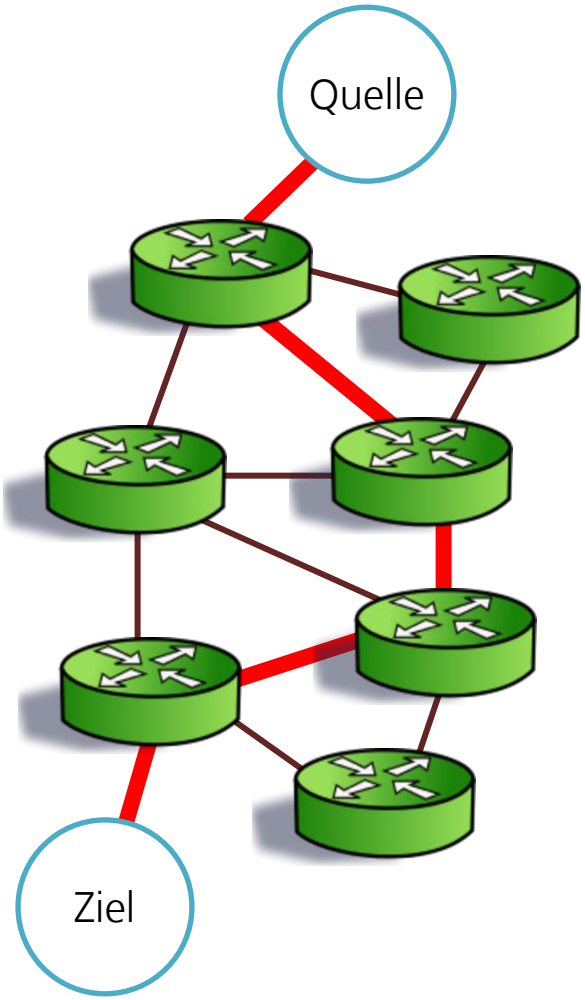
Ebendiese Aufgaben regelt u.a. das *Internet Protocol* (IP) *).

*) Es gibt auch andere Protokolle mit ähnlichen Aufgaben. IP ist das bekannteste.

Leitungsvermittlung

- Bei der Leitungsvermittlung wird für die Dauer des Datenaustauschs zwischen zwei Knoten durch Vermittlungsstellen eine **exklusive Verbindung** geschaltet.
- Auf einer Verbindung kann ein **Datenstrom unterbrechungsfrei** und **reihenfolgeerhaltend** übertragen werden.
- Eine aufgebaute Verbindung kann nur von den **beteiligten Knoten** genutzt werden, d.h. die physikalischen Leitungen sind für andere Knoten in dieser Zeit blockiert.
- Die beteiligten Leitungen sind auch dann blockiert, wenn keiner der beteiligten Kommunikationspartner Daten sendet,
- Der **Hardwareaufwand** ist hoch.

Leitungsvermittlung

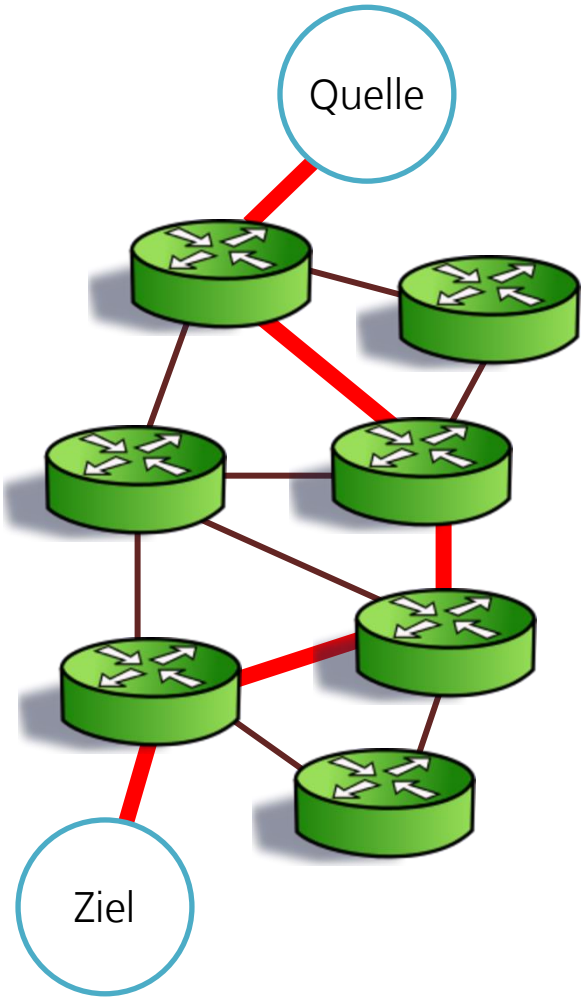


Leitungsvermittlung

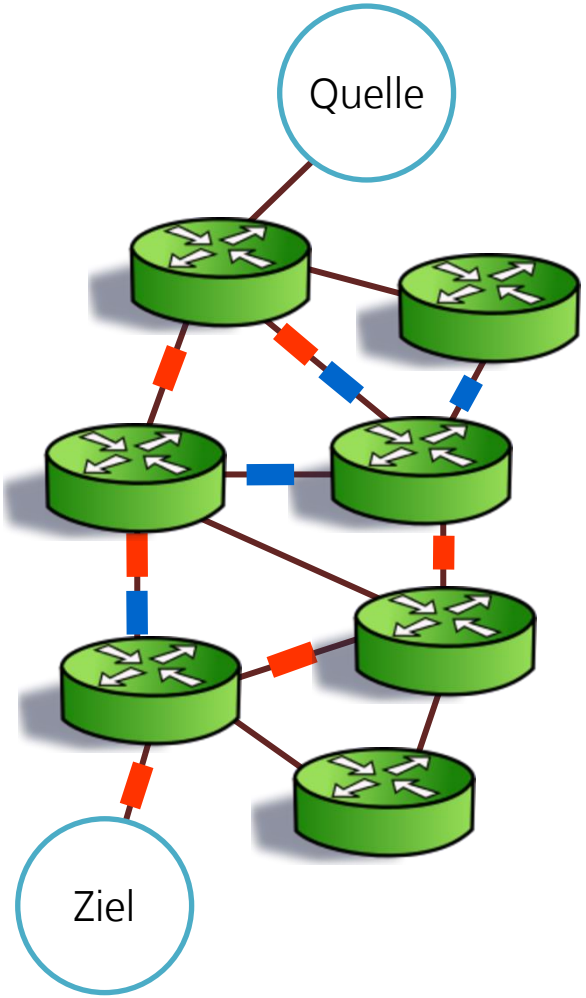
Paketvermittlung

- Datenströme werden **paketweise** übertragen.
- Physikalische Leitungen können von Paketen **mehrerer Absender** genutzt werden.
- Pakete werden **individuell** durch die Vermittler **weitergeleitet** und können daher **unterschiedliche Wege** durch das Netz nehmen.
- Auf Empfängerseite muss ggf. die Reihenfolge der Pakete wiederhergestellt werden (Aufgabe höherer Schichten).
- Pakete können verloren gehen (*fire-and-forget*).
- **Hardwareaufwand** merklich geringer als bei Leitungsvermittlung.

Leitungsvermittlung und Paketvermittlung



Leitungsvermittlung



Paketvermittlung

Das Internet Protocol (IP)

IP ist ein **verbindungsloses paketvermittelndes Protokoll** für logische Netzwerke, das auf der Sicherungsschicht aufsetzt. Es gruppiert Knoten mittels logischer Adressen in Netzwerke mit gemeinsamen Adressanteilen und erlaubt so effiziente Wegefindung.

IP wird aktuell in zwei Versionen eingesetzt:

- IPv4 mit 32-Bit Adressen (ergibt $\sim 4{,}3 \cdot 10^9$ Adressen),
- IPv6 mit 128-Bit Adressen (ergibt $\sim 3{,}4 \cdot 10^{38}$ Adressen)

Der Übergang von IPv4 zu IPv6 findet fließend statt und wird voraussichtlich noch einige Jahre andauern.

Adressierung und Wegefindung im Internet-Protocol

Wir betrachten zunächst nur IPv4:

- Jede Adresse umfasst 32 Bit. Zur einfacheren Darstellung werden diese in vier durch Punkte getrennten Teilen zu je 8 Bit (0-255) dezimal dargestellt:

1	1	0	1	1	0	1	0		0	1	1	0	0	0	1	0		1	1	1	1	0	0	1	1		0	0	0	0	0	0	0	1
218								.	98								.	243								.	1							

- Eine zusätzlich zur IP-Adresse auf jedem Knoten konfigurierte **Netzmaske** legt fest, welcher Adressbereich **innerhalb** eines logischen Netzwerks erreichbar ist (**Netzadresse**). Alle Bit der IP-Adresse, an deren Stelle in der Netzmaske eine 1 gesetzt ist, stimmen für alle Knoten in dem jeweiligen logischen Netzwerk überein.

Netzmaske - Beispiel

Adresse:

1	1	0	1	1	0	1	0		0	1	1	0	0	0	1	0		1	1	1	1	0	0	1	1		0	0	0	0	0	0	0	1
218								.	98								.	243								.	1							

Maske:

1	1	1	1	1	1	1	1		1	1	1	1	1	1	1	1		1	1	1	1	0	0	0	0		0	0	0	0	0	0	0	0
255								.	255								.	240								.	0							

Netz-adresse:

1	1	0	1	1	0	1	0		0	1	1	0	0	0	1	0		1	1	1	1	0	0	0	0		0	0	0	0	0	0	0	0
218								.	98								.	240								.	0							

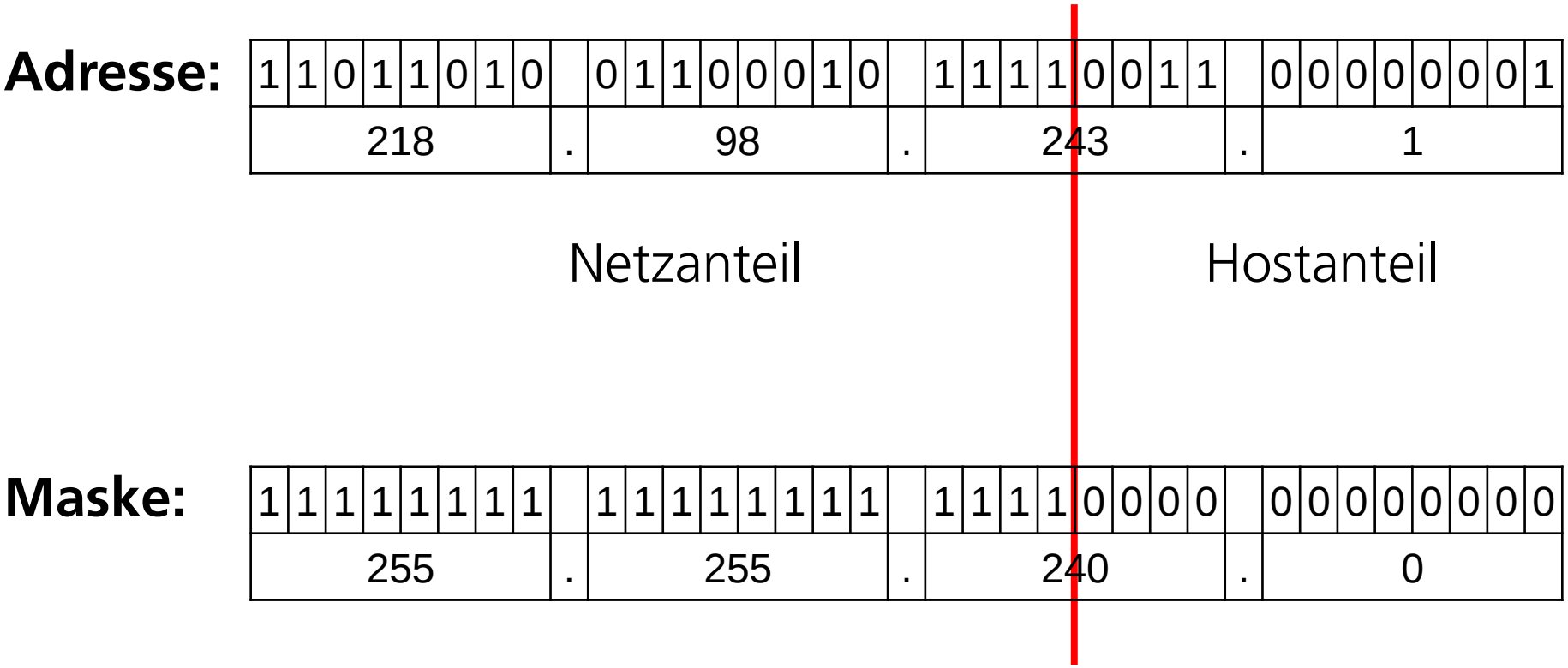
Hinweis: Es wird stets von rechts ausmaskiert, d.h. die Netzmaske besteht von links aus gesehen stets aus einer zusammenhängenden Gruppe von Einsen, gefolgt von einer zusammenhängenden Gruppe von Nullen.

Netzmaske - Beispiel

Netz-adresse:	1	1	0	1	1	0	1	0		0	1	1	0	0	0	1	0		1	1	1	1	0	0	0	0		0	0	0	0	0	0	0	0
	218								.	98								.	240								.	0						
entspricht:	1	1	0	1	1	0	1	0		0	1	1	0	0	0	1	0		1	1	1	1	0	0	0	0		0	0	0	0	0	0	0	0
	218								.	98								.	240								.	0						
bis	1	1	0	1	1	0	1	0		0	1	1	0	0	0	1	0		1	1	1	1	1	1	1	1		1	1	1	1	1	1	1	1
	218								.	98								.	255								.	255						

Kurzschreibweise: 218.98.240/20, d.h. die ersten 20 Bit identifizieren das Netz, die verbleibenden 12 Bit den Knoten innerhalb des Netzes.

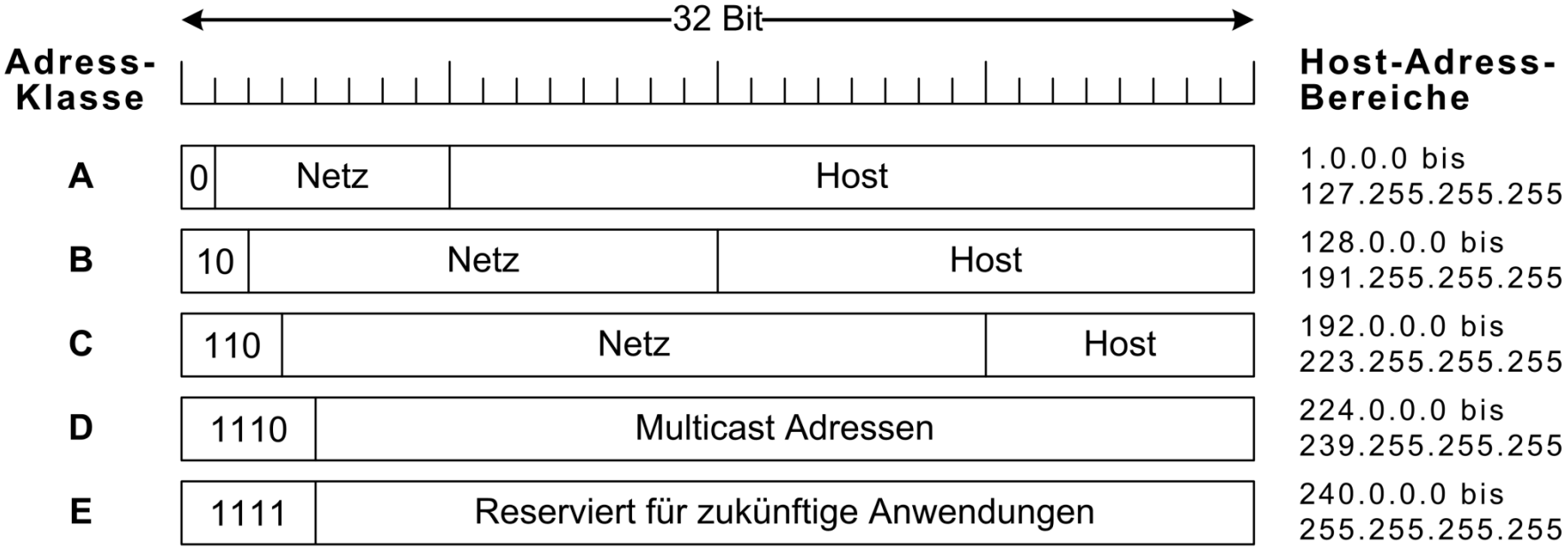
Netzmaske - Beispiel



Für eine konkrete IP-Adresse wird derjenige Teil, der durch die Netzmaske **ausmaskiert** wird (also dort, wo die Netzmaske Nullen aufweist), als **Hostanteil** bezeichnet. Der verbleibende Teil ist der **Netzanteil**.

Netzklassen

Bis 1993 wurden Netze und die daraus resultierenden Netzadressen in ihrer Größe standardisiert, sodass der Netzanteil nur Längen von 8, 16 oder 24 Bit aufwies. Dies wurde zugunsten höherer Flexibilität aufgegeben.



Wegefindung

- Um Wegefindung zu ermöglichen, werden Knoten eines **gemeinsamen physikalischen** Netzes ebenfalls in einem **gemeinsamen logischen** Netz organisiert, d.h. die Netzanteile ihrer Adressen stimmen überein.
- Jeder Knoten kennt zunächst nur seine **eigene** IP-Adresse.
- Will ein Knoten eine Nachricht versenden, so kann er anhand seiner IP-Adresse und Netzmaske feststellen, ob das **Ziel** der Nachricht **innerhalb** desselben Netzwerkes liegt oder **außerhalb**.

Wegefindung

- Soll die Nachricht **innerhalb** desselben **Netzwerkes** zugestellt werden, muss der Knoten zunächst ermitteln, welche MAC-Adresse der gesuchten Ziel-IP-Adresse zugeordnet ist, denn physikalisch werden weiterhin MAC-Adressen verwendet.

- Dazu fragt er per **Rundnachricht** in das **physikalische Netzwerk**, welche MAC-Adresse zu der Ziel-IP-Adresse gehört. Details hierzu regelt das *Address-Resolution-Protocol* (ARP).

- Erhält er eine passende Antwort, kann er die Nachricht über die **Sicherungsschicht** direkt zustellen. Andernfalls ist die Nachricht nicht zustellbar.

Wegefindung

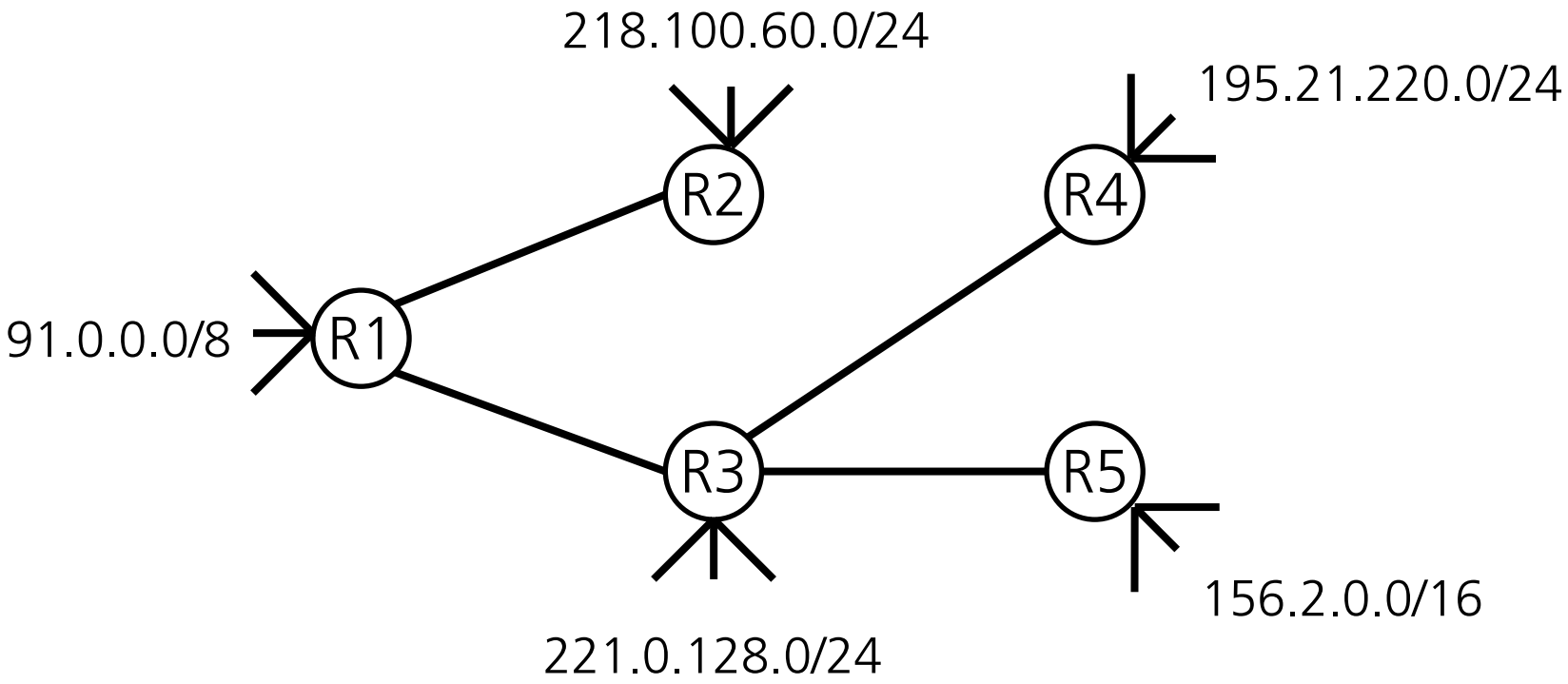
- Nachrichten an Knoten **außerhalb** des eigenen Netzwerkes werden an einen im lokalen Netz befindlichen **Vermittler (Router)** weitergegeben, der Verbindungen zu anderen Netzwerken besitzt, d.h. der im Regelfall über mindestens zwei physikalische Netzwerkschnittstellen verfügt.

- Der Router entscheidet anhand einer lokal hinterlegten **Routingtabelle**, in welche Richtung die Nachricht weitergeleitet wird.

- In dieser Routingtabelle sind für die Wegeentscheidungen typischerweise **Netzadressen** als Ziele hinterlegt. Einzelne Knoten haben hier nur ausnahmsweise einen Eintrag. Dadurch verringert sich die Komplexität (u.a. Speicherbedarf) erheblich.

Wegefindung

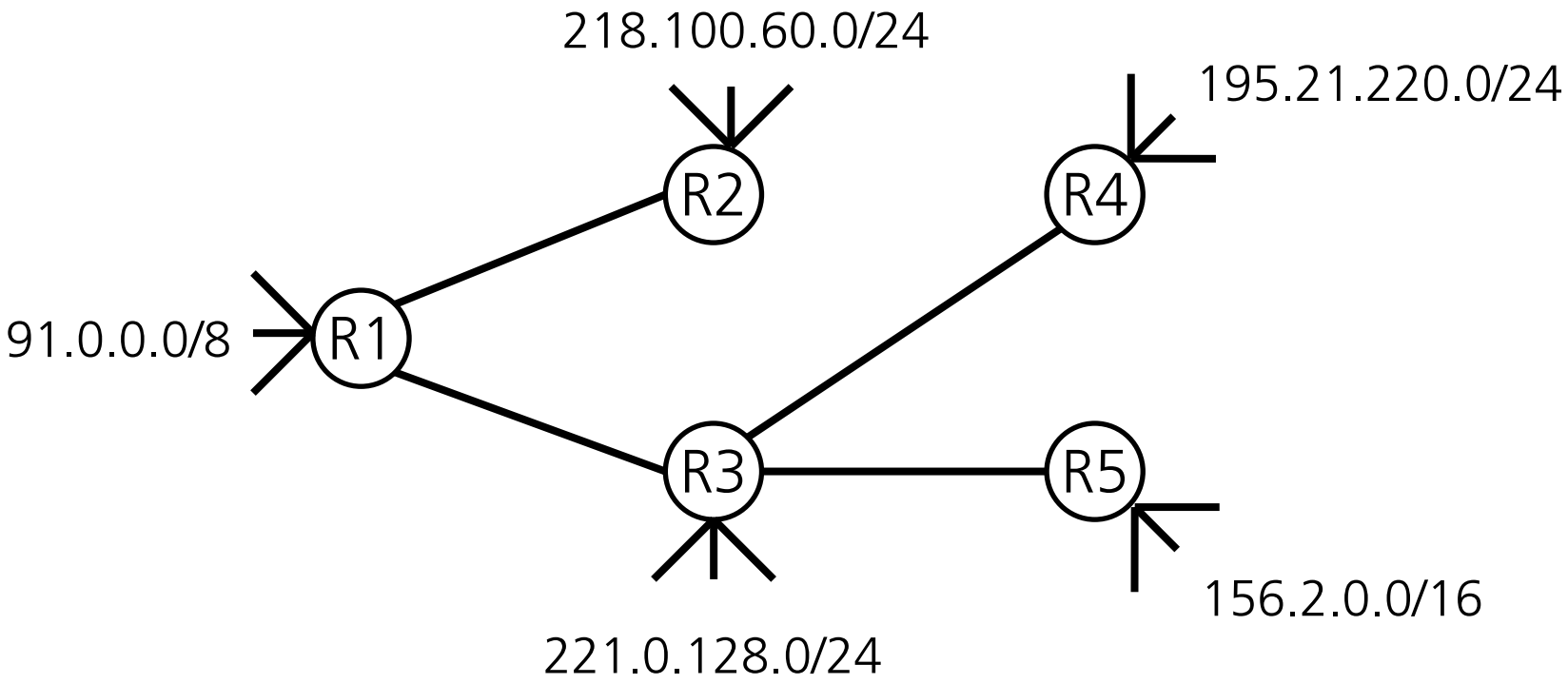
Im Beispiel: Fünf Netzwerke werden über mehrere Router miteinander verbunden. Aus Sicht der Router sind nicht die konkreten Knoten in den jeweiligen Netzwerken relevant, sondern nur deren Netzadresse.



Wegefindung

Routingtabelle R1:

Zielnetz	Wegpunkt
218.100.60.0/24	R2
195.21.220.0/24	R3
156.2.0.0/16	R3
221.0.128.0/24	R3



Internet

- Diejenige Struktur, die die unter privater, gewerblicher oder öffentlicher Kontrolle befindlichen lokalen Netzwerke erdumspannend miteinander verbindet, wird als **Internet** (Zwischennetz) bezeichnet.

- Zugang zu dieser Struktur bieten gewerbliche und öffentliche **Internetdienstanbieter** (*Internet Service Provider*, ISP). Diese betreiben eigene Routernetzwerke für die Paketvermittlung zwischen den eigenen Kunden sowie zu anderen ISP.

- Für ein lokales Netzwerk **Internetzugang** herzustellen ist äquivalent zur Bereitstellung von **Vermittlungskapazität** zu den anderen an das Internet angebundenen lokalen Netzwerken.

Autonome Systeme

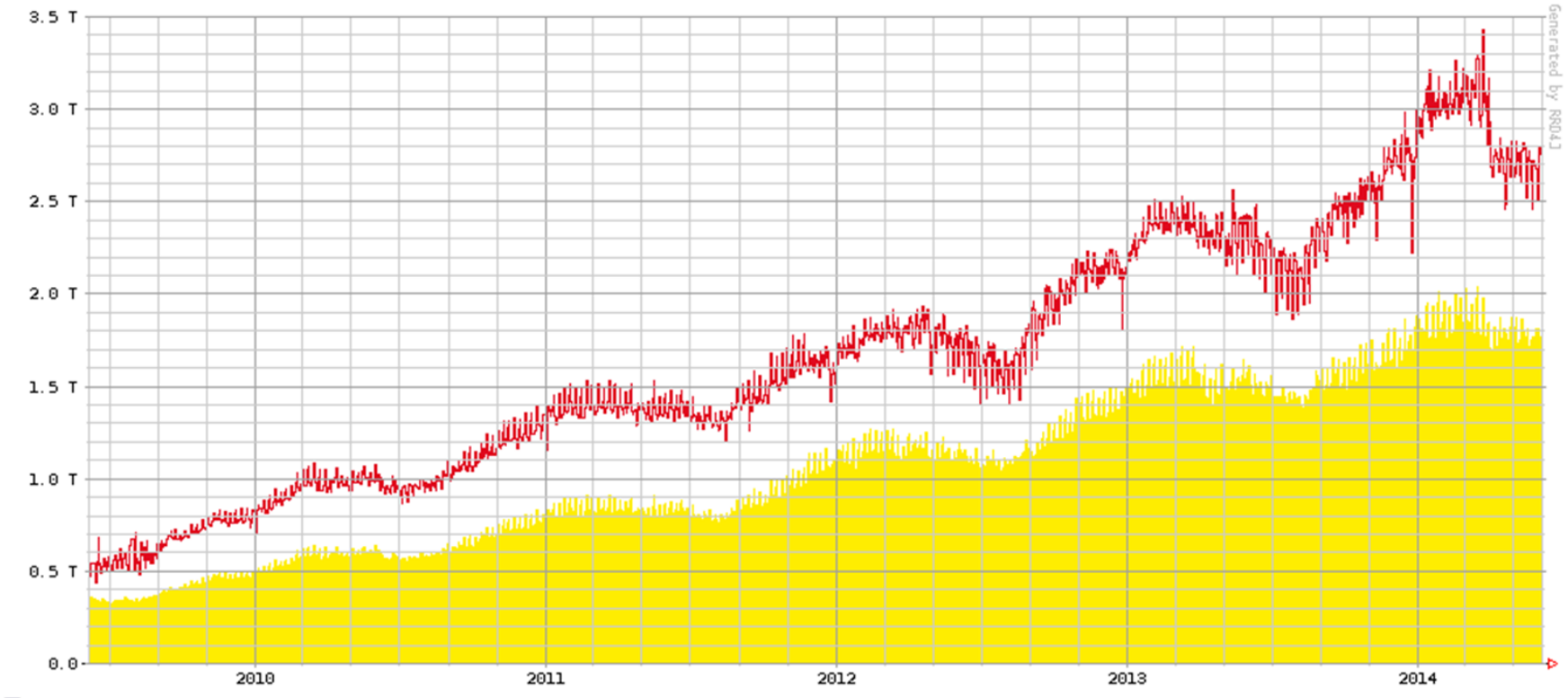
- Die Netzwerke der Internetdiensteanbieter werden auch als **Autonome Systeme** (AS) bezeichnet, sofern sie
  - in der Lage sind, Nachrichten auch in andere AS zu vermitteln,
  - nach außen ein weltweit standardisiertes Protokoll zum Austausch von Vermittlungsinformationen zwischen den beteiligten Routern verwenden. Dieses Protokoll ist das *Border Gateway Protocol* (BGP).
- Autonome Systeme erhalten eine eindeutige Kennung *(Autonomous System Number*, ASN), die auf Routern an die Stelle von Netzadressen als Ziel treten kann. Dies reduziert die Komplexität der auszutauschenden Daten erheblich.

Internet

- Internetdienstanbieter werden nach ihrer Größe klassifiziert:
  - Rang 3 Anbieter (engl.: *tier 3*) agieren lokal oder regional
  - Rang 2 Anbieter (*tier 2*) agieren überregional
  - Rang 1 Anbieter (*tier 1*) agieren weltweit.
- Anbieter gleichen Ranges sind z.T. direkt untereinander vernetzt. Den Zugang zu Netzwerken, die durch diese Kooperation nicht erreichbar sind, erfolgt durch höherrangige Anbieter.
- Die Vernetzung zwischen den Anbietern kann entweder direkt oder durch sogenannte **Internetknoten** (*Internet Exchange Point*, IXP) erfolgen, an die jeweils mehrere ISP angebunden sind.

DE-CIX

- Der weltweit **durchsatzstärkste IXP** wird in Deutschland von DE-CIX betrieben. Aktuell leistet dieser Knoten Spitzenübertragungsraten von über 3 Tbit/s.

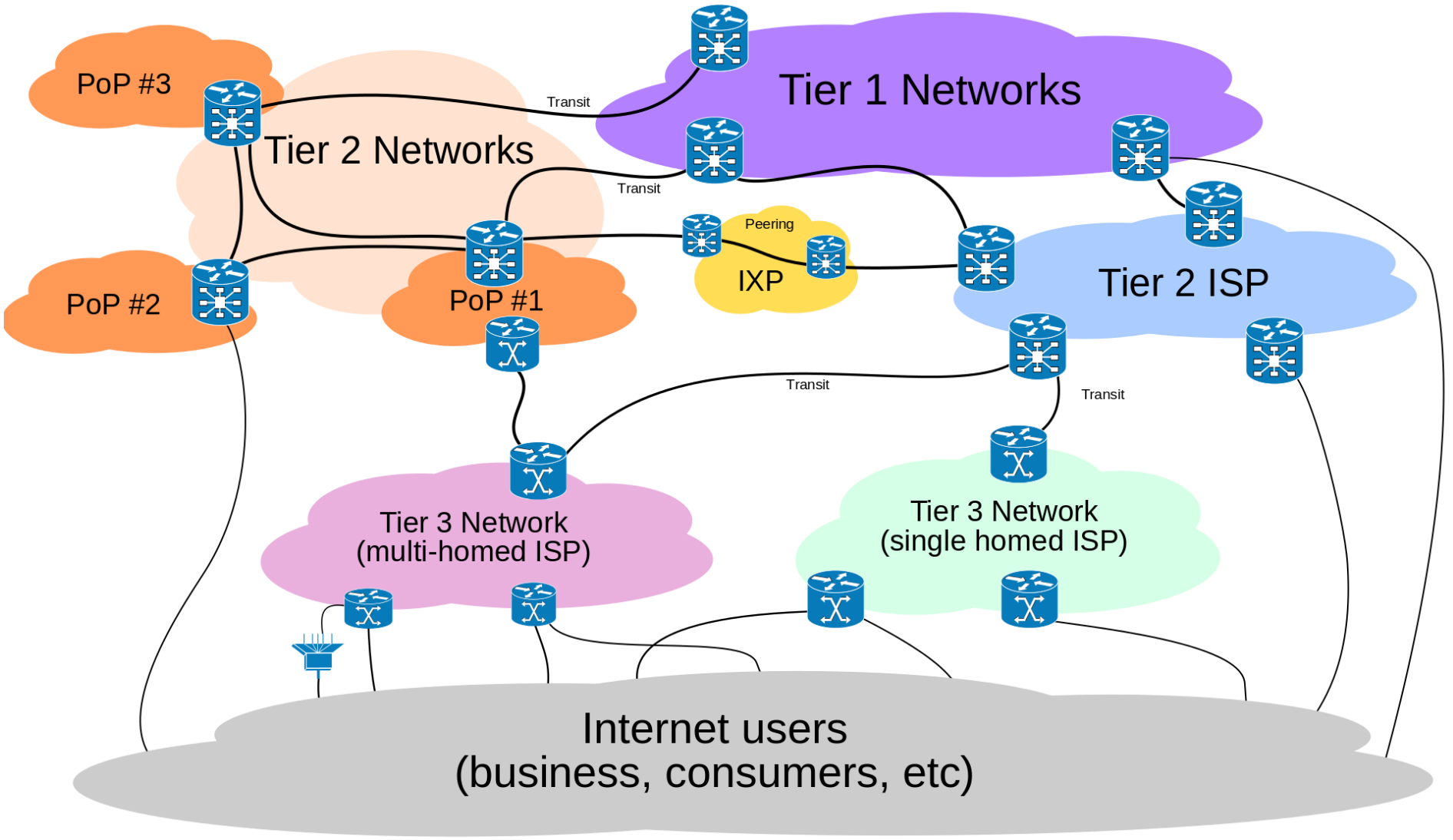


Quelle: DE-CIX

Peering und Transit

- ISP sind darauf angewiesen, dass andere ISP Pakete **weitervermitteln**, die sie selbst nicht zustellen können. Hierfür existieren zwei Modelle:
  - Beim **Peering** (engl.: *Partner*) leiten die beteiligten ISP kostenneutral den Verkehr untereinander weiter. Vermittlung in Fremdnetze erfolgt nicht.
  - Beim **Transit** vermittelt ein ISP kostenpflichtig auch den Durchgangsverkehr in die Netze anderer ISP.
- Typischerweise besteht **Peering** zwischen Anbietern **gleichen Rangs**, während **Transit** bei Anbietern höheren Rangs **eingekauft** werden muss. Tier 1 Anbieter kaufen keinen Transit ein und betreiben ausschließlich Peering mit anderen Tier 1 Anbietern.

Internet – Hierarchie der autonomen Systeme im Internet



IP Adressvergabe

IP-Adressen können nicht frei gewählt werden. Um Doppelbelegungen zu vermeiden, muss die Zuteilung **internetweit koordiniert** werden.

Die **Internet Assigned Numbers Authority** (IANA) verwaltet den weltweit verfügbaren IP-Adressraum. Sie teilt einzelne Adressbereiche den **Regional Internet Registries** (RIR) zu:

- *Réseaux IP Européens Network Coordination Centre*
- *American Registry for Internet Numbers*
- *Asia-Pacific Network Information Centre*
- *Latin American and Caribbean Internet Addresses Registry*
- *African Network Information Centre*

Regional Internet Registries

IP Adressvergabe

Die Regional Internet Registries vergeben einzelne Adressbereiche an **National Internet Registries**, die für einzelne Staaten zuständig sind*).

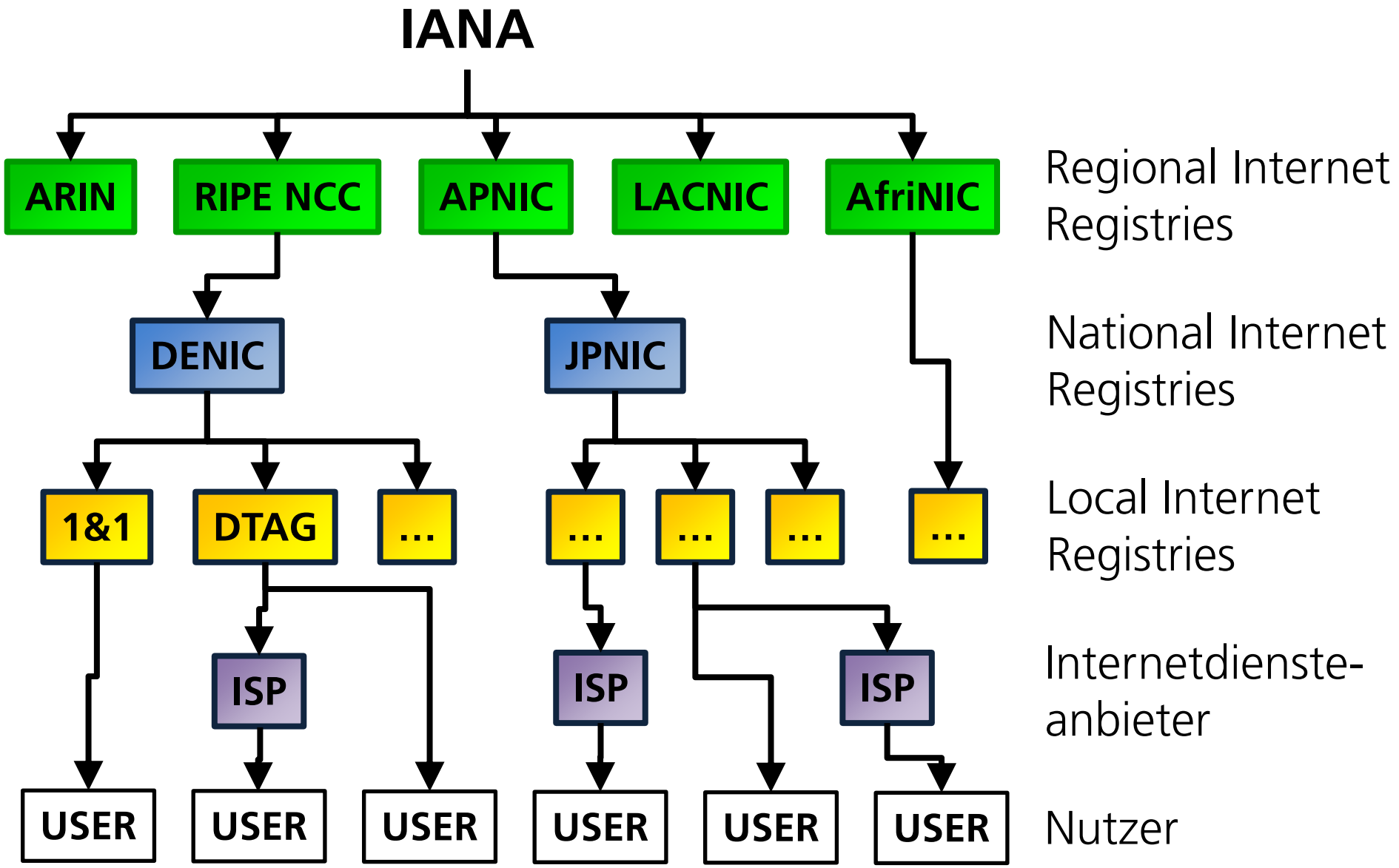
Diese wiederum teilen ihren Adressbereich **Local Internet Registries** zu, die typischerweise von Internetdiensteanbietern oder Universitäten betrieben werden.

Von hier aus erfolgt schließlich die Zuteilung an Endgeräte bzw. zu Kunden.

Jede dieser Zuteilungen ist **gebührenpflichtig**, auch wenn diese Gebühren beim Kunden nicht explizit ausgewiesen werden.

*) In einigen Fällen fehlen diese nationalen Registraturen.

IP Adressvergabe



Private Netze

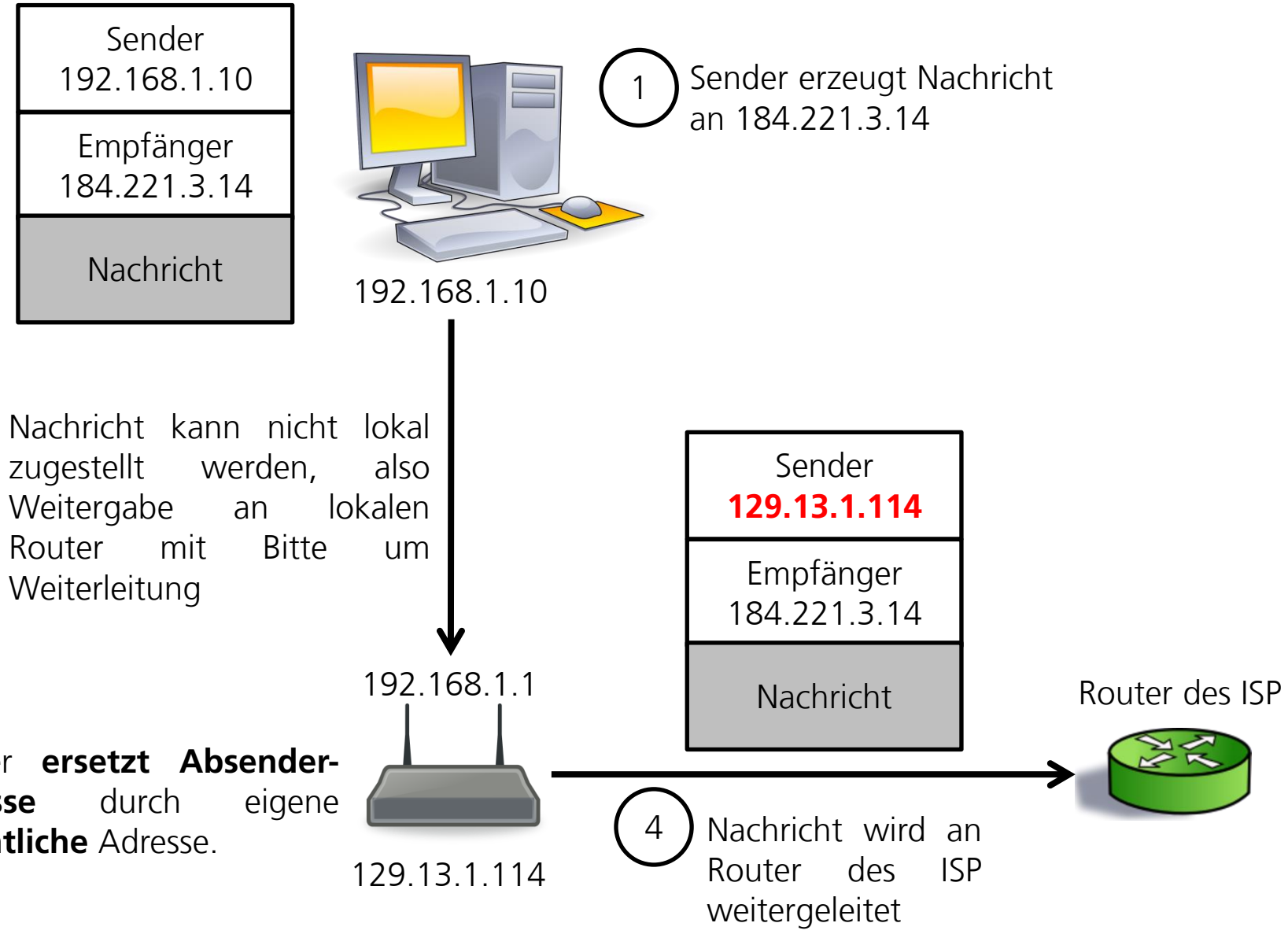
- Jede bei der IANA registrierte (d.h. öffentliche) IP-Adresse verursacht für deren Nutzer direkte oder indirekte Kosten.
- Zwei Adressbereiche 192.168.0.0/16 und 10.0.0.0/8 sind explizit für den **privaten** Gebrauch vorgesehen. Nachrichten an diese Adressen werden durch Router im Internet **nicht weitervermittelt**.
- Bekanntes Anwendungsgebiet: Im **Heimnetz** besitzt nur der Router eine öffentliche IP-Adresse, alle anderen Knoten sind mit privaten Adressen versehen.
- **Frage:** Wie wird für die Knoten mit privaten Adressen der Netzzugang sichergestellt?

Network Address Translation (NAT)

- **Konzept:** Der Router ersetzt bei jedem Weiterleitungs-vorgang **aus dem lokalen Netz heraus** die Absenderadresse durch seine **eigene** Adresse *). Danach wird die Nachricht wie gewohnt an die Zieladresse geleitet.
- Kommt eine **Antwort** zurück ersetzt er die **Zieladresse** der Antwort (also seine eigene Adresse) durch die Adresse des Knotens im lokalen Netz und leitet die Nachricht an dieses weiter.
- Für den Knoten im lokalen Netz ist dieser Vorgang **transparent**.

*) Präzise wird deswegen hier von Source-NAT gesprochen (*source=quelle*).

Network Address Translation (NAT)



Network Address Translation (NAT)

192.168.1.10

3 Nachricht wird an Original-sender zugestellt

Sender
184.221.3.14
Empfänger
192.168.1.10
Antwort

192.168.1.1

Sender
184.221.3.14
Empfänger
129.13.1.114
Antwort

Router des ISP

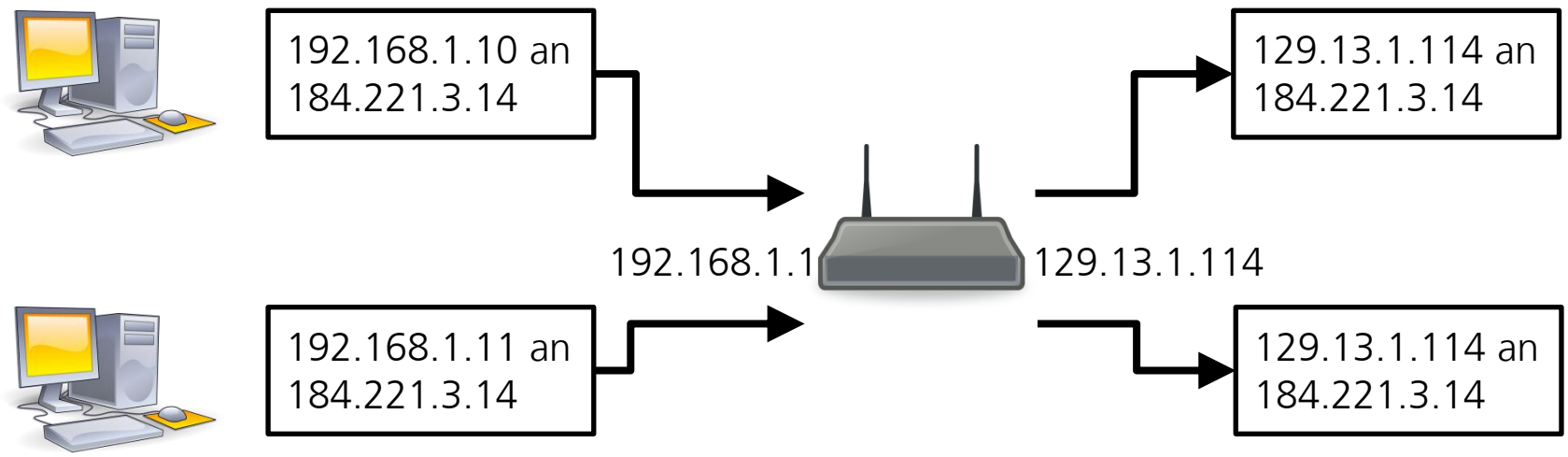
2 Router **ersetzt Empfänger-adresse** durch private Adresse des Originalsenders.

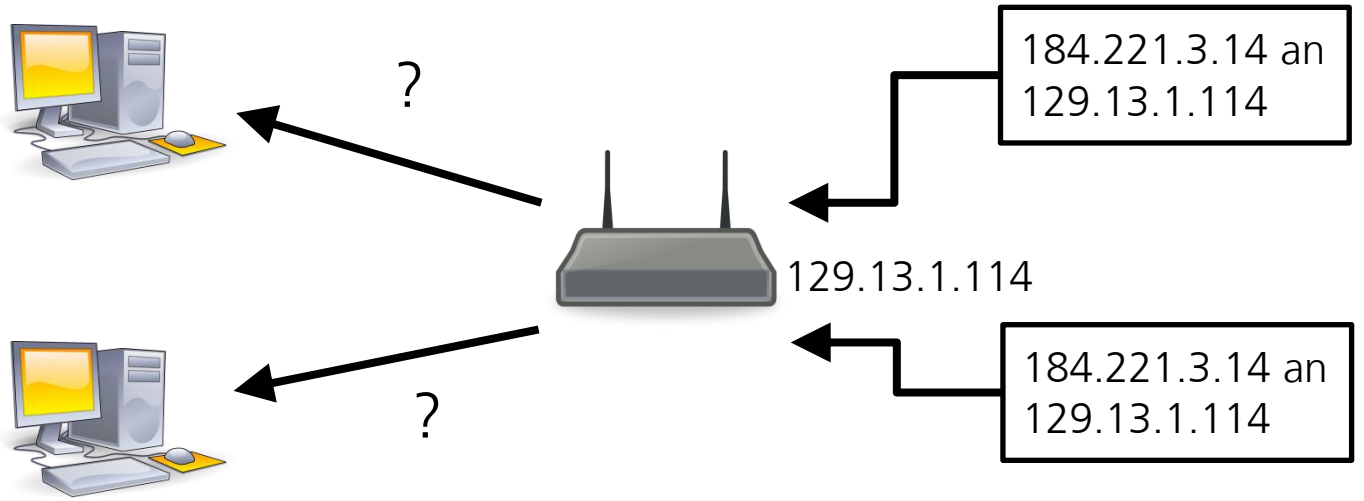
129.13.1.114

1 Lokaler Router erhält Rückantwort über Router des ISP

Network Address Translation (NAT)

- In einem Netzwerk mit NAT-Router befinden sich typischerweise **mehrere** Geräte, die ggf. parallel Nachrichten an **denselben Zielknoten** verfassen können.
- Bei einer **Rückantwort** muss der NAT-Router erkennen, an welchen der Quellknoten er diese zustellen muss.
- Ohne **Zusatzinformation** ist dies nicht möglich, denn die Antwortnachricht enthält in beiden Fällen denselben Sender und Empfänger.


192.168.1.10 an
184.221.3.14
129.13.1.114 an
184.221.3.14
192.168.1.1
129.13.1.114
192.168.1.11 an
184.221.3.14
129.13.1.114 an
184.221.3.14


?
184.221.3.14 an
129.13.1.114
129.13.1.114
?
184.221.3.14 an
129.13.1.114

Übungsfrage

Wie kann dieses Problem gelöst werden?

Antwort

Antwort: Gar nicht, ohne die Grenzen von IP zu verlassen!

In der Praxis werden Informationen aus Schicht 4 hinzugezogen und manipuliert (siehe nachfolgende Kurseinheit). Erst dadurch wird eine eindeutige Zuordnung der Rückantworten zu den Knoten im lokalen Netz wieder möglich.

NAT – Bewertung

- NAT kann die Sicherheit in lokalen Netzwerken erhöhen, da die Knoten hinter dem NAT-Router **verborgen** bleiben.
- Ebendiese fehlende Erreichbarkeit führt jedoch bei einigen Protokollen (**Internettelefonie**, etc.) zu **Komplikationen**, wenn Knoten von außen kontaktiert werden sollen.
- NAT durchbricht die **Schichtenhierarchie**.
- **1-zu-1 Zuordnung** von Knoten zu IP-Adresse wird durchbrochen. Dies ermöglicht **neue Angriffsszenarien**.
- Zu guter Letzt: NAT stellt ohnehin nur eine **Notlösung** dar, weil IPv4 schlicht nicht genügend Adressen bereithält, um den aktuellen Bedarf in lokalen Netzwerken zu decken.

Transition zu IPv6

- Ehemals getrennte Netze wachsen durch IP zu logischen Netzen zusammen (Festnetztelefonie, TV, Mobilgeräte).
- Dieser an sich positive Effekt führt aber dazu, dass der **Adressraum** von **IPv4** mittlerweile praktisch **ausgeschöpft** ist, ein größerer **Adressraum** ist nötig.
- Übergang zu IPv6, wo 128-Bit Adressen verwendet werden („Jedes Sandkorn erhält eine Adresse“).
- **Vorteil:** Jedes Gerät kann dauerhaft und stabil mit einer Adresse versorgt werden (analog zu Telefonnummern).
- **Nachteil:** Tracking einzelner Geräte wird dadurch weiter vereinfacht, Anonymität geht verloren.

Wegefindung

Routingprotokolle

- Router brauchen für ihre Entscheidungsfindung Informationen über mögliche **Ziele** und die **Wege** dorthin. Ebendiese Informationen werden in der lokal auf dem Router vorgehaltenen **Routingtabelle** hinterlegt.
- In kleinen Routernetzwerken können diese Tabellen manuell aufgebaut werden.
- In **großen Routernetzwerken** und dementsprechend auch im Internet ist dies aufgrund von Informationsmenge und Dynamik der Netze nicht möglich, d.h. die Routingtabellen müssen **automatisiert** erstellt werden.
- Dies ist Aufgabe der **Routingprotokolle** und der darin verwendeten Algorithmen.

Routingprotokolle

- Allen Routingprotokollen liegt (mindestens) eine **Metrik** zugrunde, anhand derer **Verbindungen** zwischen Routern **bewertet** werden.
- Als Basis hierzu können **Verbindungsparameter** herangezogen werden:
  - Aktuell verfügbare **Bandbreite**
  - **Fehlerrate**
  - **Verzögerungen**
  - etc.
- Mittels der verwendeten Metrik wird jeder Verbindung zwischen zwei Routern ein **Kostenwert** zugeordnet.

Kürzeste Pfade

- Die Aufgabe der Routingalgorithmen ist es, die Wege mit **geringstmöglichen** aufsummierten **Kosten** zwischen den Routern zu ermitteln.
- Dies entspricht dem Problem der Ermittlung der **kürzesten Pfade** zwischen den Knoten eines gewichteten Graphen.
- Hierzu existieren **zentrale** und **dezentrale** Verfahren.
- Sind einem Router die kürzesten Pfade zu allen anderen Routern bekannt, kann er aus diesen Informationen seine **Routingtabelle konstruieren**.

Zentrale und dezentrale Verfahren

- Bei **zentralen Verfahren** zur Ermittlung der kürzesten Pfade von einem Knoten zu allen anderen Knoten besitzt jeder Knoten **vollständiges Wissen** über die **Struktur** des Graphen. Für Routernetzwerke bedeutet dies: Jeder Router hat Kenntnis über jeden anderen Router und dessen Verbindungen.

- Bei **dezentralen Verfahren** werden Informationen nur zwischen **benachbarten** Knoten ausgetauscht. Knoten lernen nur, welche **Ziele** ihre Nachbarn zu welchen **Kosten** erreichen können, erhalten aber keine expliziten Informationen über die **Netzwerktopologie**.

- Zentrale Verfahren eignen sich für kleinere Netzwerke, dezentrale auch für größere.

Zentrale und dezentrale Verfahren

- Bei zentralen Verfahren tauschen die Router untereinander Informationen über ihre eigenen **Verbindungen** aus. Man bezeichnet sie daher auch als **Link-State-Verfahren**.

- Im Gegensatz dazu tauschen die Router bei dezentralen Verfahren Informationen über **Distanzen** zu anderen, weiter entfernten Routern aus. Man spricht hier von **Distanzvektorverfahren**.

Grundlagen

- Jedes Netzwerk wird als Graph $G = \{E, V\}$ betrachtet. Die Knoten $v \in V$ entsprechen den Routern, die Kanten $e \in E$ entsprechen den Verbindungen zwischen ihnen.

- Jeder Kante ist ein **Gewicht** zugeordnet. Dies entspricht den Kosten der jeweiligen Verbindung. Formal erfolgt dies durch eine **Kostenfunktion** $c(v_i, v_j)$, die die Kosten zwischen den benachbarten Knoten $v_i \in V$ und $v_j \in V$ angibt.

- Die **Distanz** zwischen zwei beliebigen Knoten entspricht der kleinstmöglichen Summe von Einzelkosten aller Verbindungen zwischen diesen beiden Knoten. Sie entspricht also den Gesamtkosten des (im Sinne der Kosten) **kürzesten Pfades** zwischen den beiden Knoten.

Zentrales Verfahren nach Dijkstra

- **Aufgabe:** Finde für einen bestimmten Knoten die kürzesten Pfade zu allen anderen Knoten des Graphen.
- **Voraussetzung:** Der Netzwerkgraph ist bereits bekannt.
- **Initialisierung:** Für alle Distanzen wird der Wert ∞ angenommen. Es wird eine Liste bereits bearbeiteter Knoten angelegt, die zunächst leer ist.
- **Ansatz:** In jedem Iterationsschritt wird ein Knoten ausgewählt. Für diesen wird analysiert, welche Verbindungen er besitzt und welche neuen Pfade sich daraus für die Distanztabelle ergeben. Danach wird er in die Liste der bereits betrachteten Knoten eingefügt. Gibt es keine weiteren Knoten zu betrachten, endet der Algorithmus.

Zentrales Verfahren nach Dijkstra

- **Knotenauswahl:** Es wird stets derjenige Knoten als der nächste zu betrachtende Knoten ausgewählt, der die **kürzeste Distanz** zum **Startknoten** hat und noch nicht betrachtet wurde.

- **Erster Knoten:** Der Startknoten wird als erster Knoten betrachtet. Dies ist im Einklang mit dem o.g. Auswahlkriterium, denn er hat die Distanz 0 zu sich selbst.

- Für jeden Eintrag in der Distanztabelle wird auch derjenige Knoten abgelegt, über den die Vermittlung zum Zielknoten erfolgen soll. Dieser **Wegpunkt** ist bei zentralen Verfahren der **Vorgänger des Zielknotens**, also der vorletzte Knoten im Vermittlungspfad.

Zentrales Verfahren nach Dijkstra – formale Betrachtung

Initialisierung

$S = \{\}$

$\forall v \in V \backslash v_s$

$D(v) = \infty$

Startknoten

$S = \{v_s\}$

$\forall\ v_n \in {N_g}^+(v_s)$

$D(v_n) = c(v_s, v_n)$

Hinweis: Initialisierung und Betrachtung des Startknotens werden i.d.R. zusammengefasst.

Algorithmus

Wiederhole

wähle $v \in V, v \notin S$ **so, dass** $D(v) = min$

$S = \{v_s\} \cup v$

$\forall\ v_n \in {N_g}^+(v)$

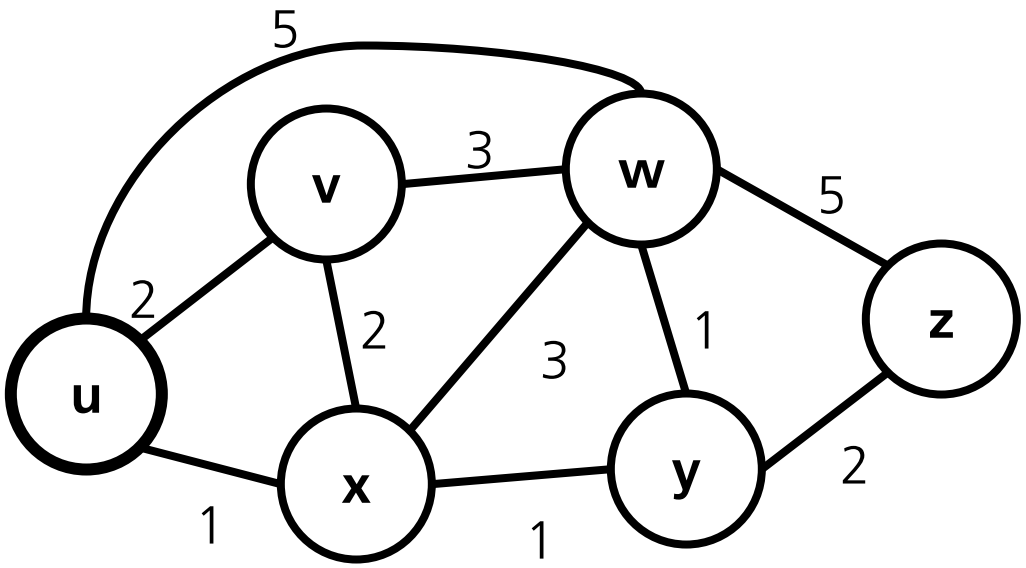
Wenn $D(v_n) > D(v) + c(v,\ v_n)$

Dann $D(v_n) = D(v) + c(v,\ v_n)$

bis $S = V$

Dijkstra-Algorithmus - Beispiel

$V = \{u, v, w, x, y, z\}$



Initialisierung:

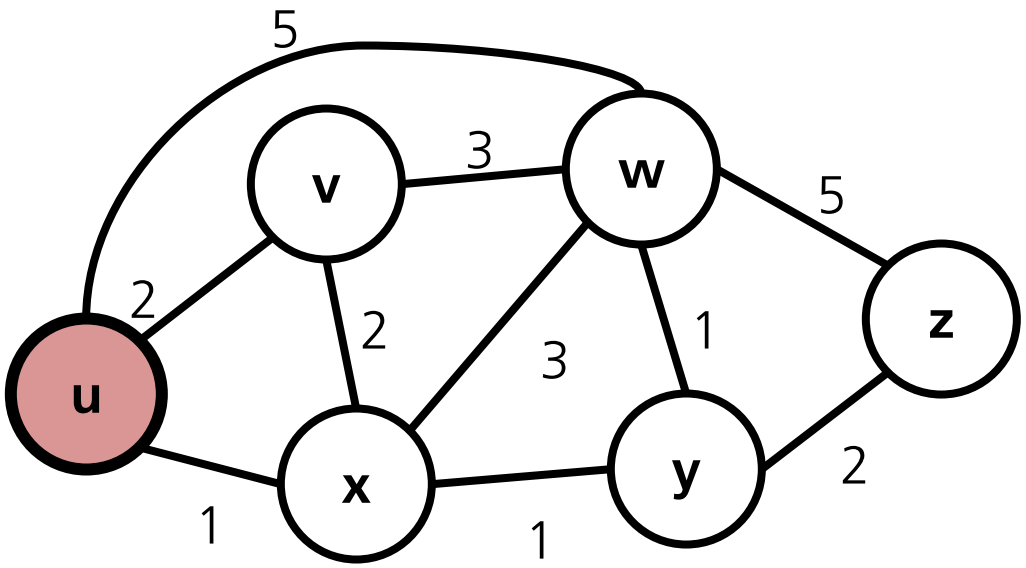
$S = \{\}$

Knoten	Distanz	Weg-punkt
v	∞	
w	∞	
x	∞	
y	∞	
z	∞	

- Menge betrachteter Knoten ist leer
- Distanztabelle ist angelegt und für alle Knoten mit ∞ initialisiert

Dijkstra-Algorithmus - Beispiel

$V = \{u, v, w, x, y, z\}$



Startknoten:

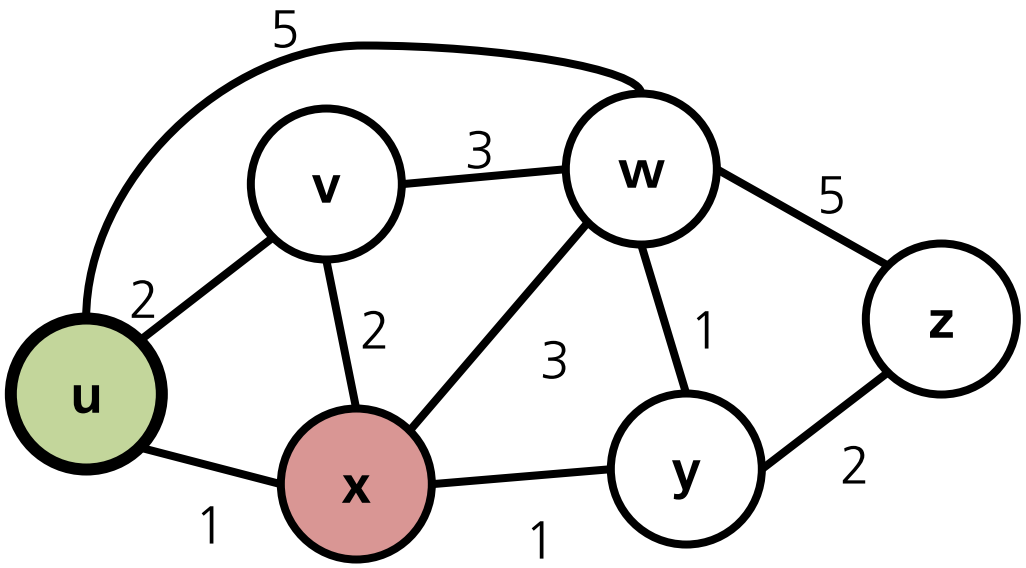
$S = \{u\}$

Knoten	Distanz	Weg-punkt
v	**2**	u
w	**5**	u
x	**1**	u
y	∞	
z	∞	

- Startknoten ist benachbart mit v, x und w
- entsprechende Distanzen werden eingetragen

Dijkstra-Algorithmus - Beispiel

$V = \{u, v, w, x, y, z\}$



Iteration 1:

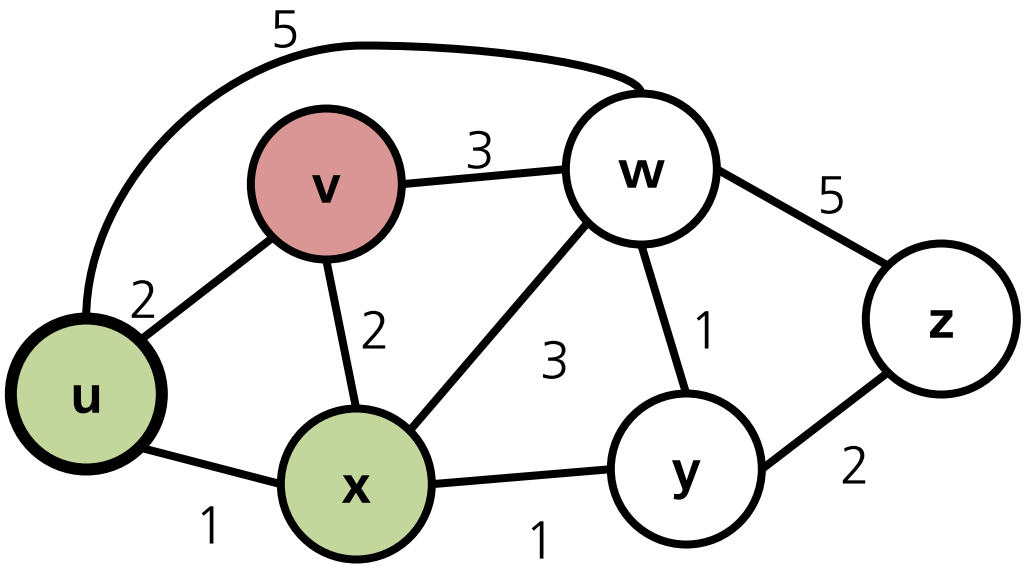
$S = \{u, x\}$

Knoten	Distanz	Weg-punkt
v	2	u
w	**4**	**x**
x	1	u
y	**2**	**x**
z	∞	

- Knoten x ist derjenige, mit der kleinsten Distanz zu u
- x hat kürzeren Weg zu w und Weg zu Knoten y
- Weg zu v ist länger als der bereits bekannte

Dijkstra-Algorithmus - Beispiel

$V = \{u, v, w, x, y, z\}$



Iteration 2:

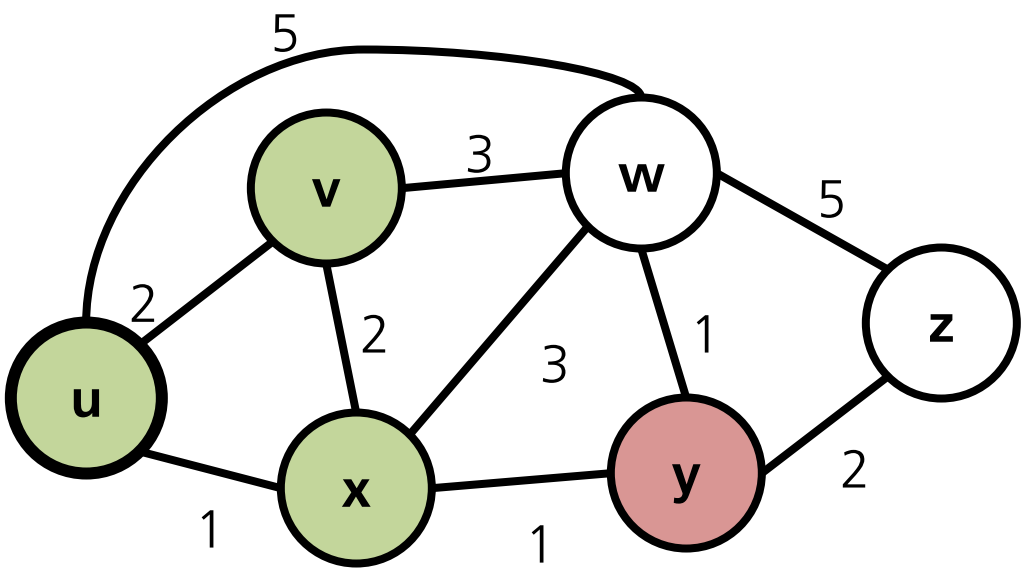
$S = \{u, x, v\}$

Knoten	Distanz	Weg-punkt
v	2	u
w	4	x
x	1	u
y	2	x
z	∞	

- Jetzt könnte entweder v oder y betrachtet werden, beide haben eine Distanz von 2 zu u. Für das Ergebnis ist die Reihenfolge nicht relevant, wir wählen v.
- Von v lernen wir keine kürzeren Pfade oder neue Knoten.

Dijkstra-Algorithmus - Beispiel

$V = \{u, v, w, x, y, z\}$



Iteration 3:

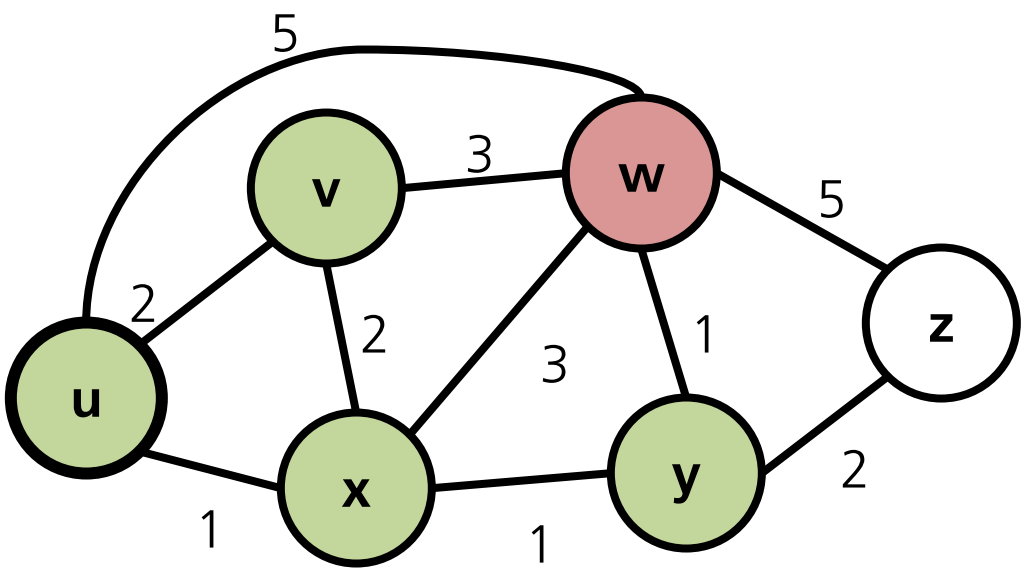
$S = \{u, x, v, y\}$

Knoten	Distanz	Weg-punkt
v	2	u
w	**3**	**y**
x	1	u
y	2	x
z	**4**	**y**

- Nun wird y betrachtet.
- z wird als neuer Knoten gelernt.
- Die Distanz zu w über y ist besser als über den bisherigen Weg über x.

Dijkstra-Algorithmus - Beispiel

$V = \{u, v, w, x, y, z\}$



Iteration 4:

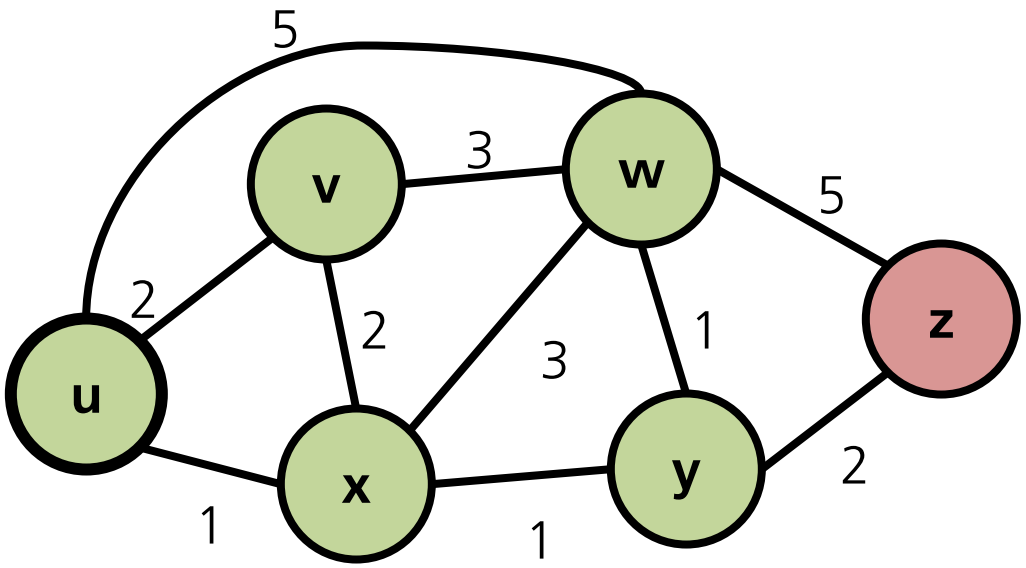
$S = \{u, x, v, y, w\}$

Knoten	Distanz	Weg-punkt
v	2	u
w	3	y
x	1	u
y	2	x
z	4	y

- w wird betrachtet.
- Es werden über w keine kürzeren Pfade oder neue Knoten gefunden.

Dijkstra-Algorithmus - Beispiel

$V = \{u, v, w, x, y, z\}$ ← $S = V$ → $S = \{u, x, v, y, w, z\}$



Iteration 5:

Knoten	Distanz	Weg-punkt
v	2	u
w	3	y
x	1	u
y	2	x
z	4	y

- Schließlich wird z betrachtet, auch hier werden keine neuen Informationen gewonnen.
- z war der letzte noch nicht betrachtete Knoten, d.h. $S = V$. Der Algorithmus terminiert.

Distanzvektoralgorithmus nach Bellman-Ford

Der Distanzvektoralgorithmus nach Bellman-Ford ist ein dezentrales Verfahren zum Aufbau der Routingtabellen.

Prinzip:

- Jeder Router teilt Veränderungen seiner Routingtabelle nur seinen direkten Nachbarn mit.
- Aus den Mitteilungen anderer Router können Änderungen der eigenen Routingtabelle resultieren, die dann wiederum an die eigenen Nachbarn propagiert werden.
- Ändert sich in der eigenen Routingtabelle nichts, werden auch keine Mitteilungen an Nachbarn gesendet.

Distanzvektoralgorithmus nach Bellman-Ford

Initialisierung: Jeder Router ermittelt für seine direkten Nachbarn die Kosten und trägt diese in seine Routingtabelle ein. Alle nicht direkt erreichbaren Knoten werden mit der Distanz ∞ initialisiert.

Der Vektor aus bekannten Zielen und zugehörigen Distanzen (der Distanzvektor) wird nun an alle Nachbarn weitergeleitet.

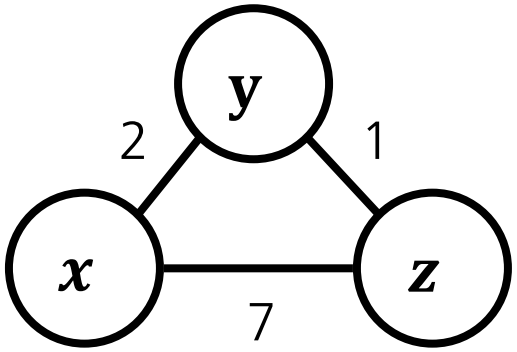
Algorithmus: Erhält ein Router einen Distanzvektor von einem seiner Nachbarn oder ändern sich Linkkosten zu seinen Nachbarn, prüft er, ob sich aus diesen Informationen Änderungen der Routingtabelle ergeben. Diese trägt er ein.

Eventuelle Änderungen seiner Routingtabelle werden als Distanzvektor an seine Nachbarn weitergeleitet.

Distanzvektoralgorithmus nach Bellman-Ford

Wegpunkte: Auch bei diesem Verfahren wird für jeden Eintrag in der Distanztabelle derjenige Knoten vermerkt, über den vermittelt werden Soll. Im Gegensatz zum zentralen Verfahren ist dieser Wegpunkt jedoch stets der **Nachfolger** des **aktuellen** Knotens.

Algorithmus von Bellman & Ford – Beispiel



x

Wegpunkt \ Ziel	x	y	z
x			
y			
z			

y

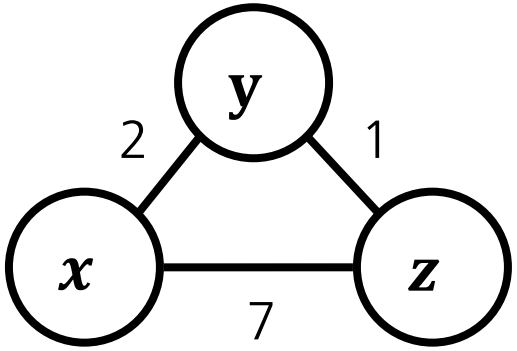
Wegpunkt \ Ziel	x	y	z
x			
y			
z			

z

Wegpunkt \ Ziel	x	y	z
x			
y			
z			

- Ein kleines Netzwerk mit den Knoten x, y und z sei gegeben.
- Jeder Knoten besitzt eine Distanztabelle (Routingtabelle), in der die Kosten zu jedem anderen Knoten abgelegt werden sollen.

Algorithmus von Bellman & Ford – Beispiel



x

Wegpunkt \ Ziel	x	y	z
x			
y			
z			

y

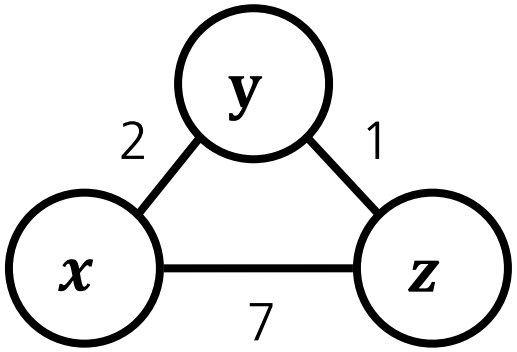
Wegpunkt \ Ziel	x	y	z
x			
y			
z			

z

Wegpunkt \ Ziel	x	y	z
x			
y			
z			

- Dabei wird für jeden Zielknoten jeder mögliche **Knoten als Wegpunkt** in Betracht gezogen, sodass im Beispiel drei Zeilen statt nur einer Zeile je Tabelle benötigt werden.

Algorithmus von Bellman & Ford – Initialisierung



x

Wegpunkt \ Ziel	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

y

Wegpunkt \ Ziel	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

z

Wegpunkt \ Ziel	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

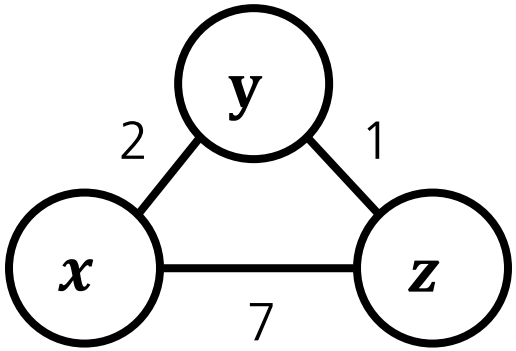
- Jeder Knoten ermittelt die Kosten zu seinen **direkten Nachbarn** und trägt diese in seine Distanztabelle ein.
- Er kennt aktuell nur **sich selbst** als **Wegpunkt**, daher werden alle anderen Wege mit ∞ initialisiert.

Algorithmus von Bellman & Ford – Iteration 1

y sendet

x	y	z
2	0	1

an x



x

Wegpunkt \ Ziel	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

y

Wegpunkt \ Ziel	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

z

Wegpunkt \ Ziel	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

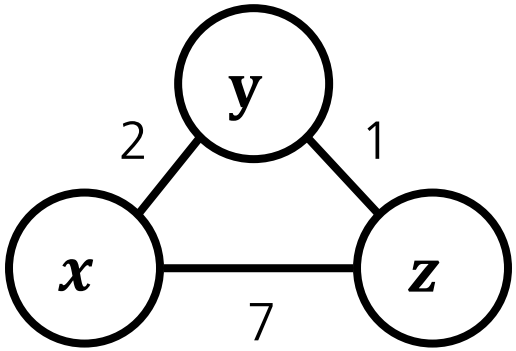
- Jeder Knoten sendet nun alle ihm bekannten Wege an seine Nachbarn. Wir beginnen mit y.
- x lernt von y, dass y
  - zu Kosten von 2 zu x,
  - zu Kosten von 0 zu y (also sich selbst) und
  - zu Kosten von 1 zu z routen kann.

Algorithmus von Bellman & Ford – Iteration 1

y sendet

x	y	z
2	0	1

an x



x

Wegpunkt \ Ziel	x	y	z
x	0	2	7
y	**4**	**2**	**3**
z	∞	∞	∞

y

Wegpunkt \ Ziel	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

z

Wegpunkt \ Ziel	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

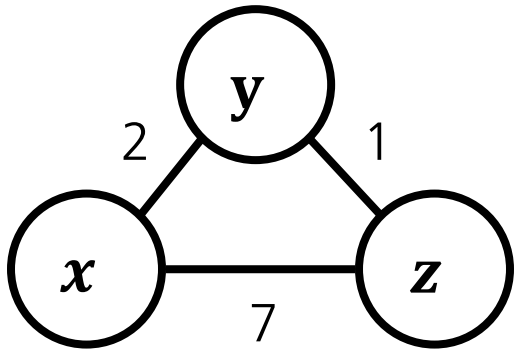
- Alle Routen, die y als Wegpunkt erhalten, wurden von x bis dahin mit Kosten ∞ angenommen. Daher wird der empfangene Distanzvektor in die Distanztabelle übernommen.
- Dabei muss x jeweils die **Linkkosten** zu y **addieren**.

Algorithmus von Bellman & Ford – Iteration 1

x sendet

x	y	z
0	2	7

an y



x

Wegpunkt \ Ziel	x	y	z
x	0	2	7
y	4	2	3
z	∞	∞	∞

y

Wegpunkt \ Ziel	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

z

Wegpunkt \ Ziel	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

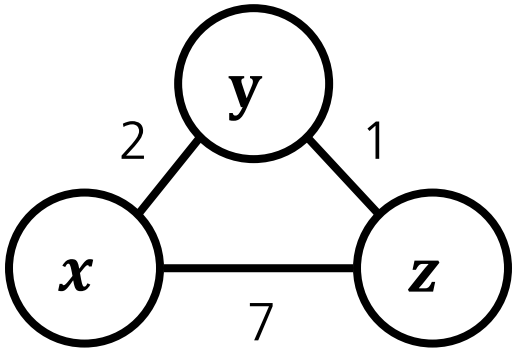
- **Gleichzeitig** sendet x seinen Distanzvektor aus der Initialisierung auch an y.
- y lernt daher von x, dass x
  - zu Kosten von 0 zu x,
  - zu Kosten von 2 zu y (also sich selbst) und
  - zu Kosten von 7 zu z routen kann.

Algorithmus von Bellman & Ford – Iteration 1

x sendet

x	y	z
0	2	7

an y



x

Wegpunkt \ Ziel	x	y	z
x	0	2	7
y	4	2	3
z	∞	∞	∞

y

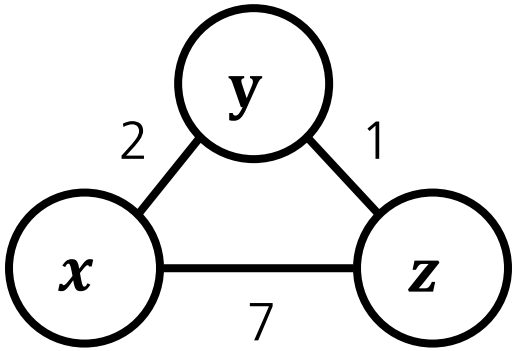
Wegpunkt \ Ziel	x	y	z
x	**2**	**4**	**9**
y	2	0	1
z	∞	∞	∞

z

Wegpunkt \ Ziel	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

- y pflegt diese Information in seine Distanztabelle ein

Algorithmus von Bellman & Ford – Iteration 1



x

Wegpunkt \ Ziel	x	y	z
x	0	2	7
y	4	2	3
z	**14**	**8**	**7**

y

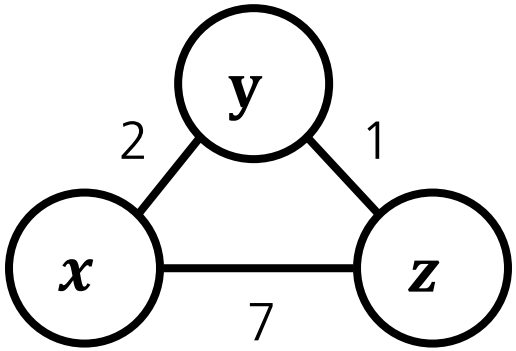
Wegpunkt \ Ziel	x	y	z
x	2	4	9
y	2	0	1
z	**8**	**2**	**1**

z

Wegpunkt \ Ziel	x	y	z
x	**7**	**9**	**14**
y	**3**	**1**	**2**
z	7	1	0

- Nach demselben Prinzip tauschen auch x und z sowie y und z Informationen aus und passen ihre Distanztabellen an.

Algorithmus von Bellman & Ford – Iteration 1



x

Wegpunkt \ Ziel	x	y	z
x	0	2	7
y	4	2	**3**
z	14	8	7

y

Wegpunkt \ Ziel	x	y	z
x	2	4	9
y	2	0	1
z	8	2	1

z

Wegpunkt \ Ziel	x	y	z
x	7	9	14
y	**3**	1	2
z	7	1	0

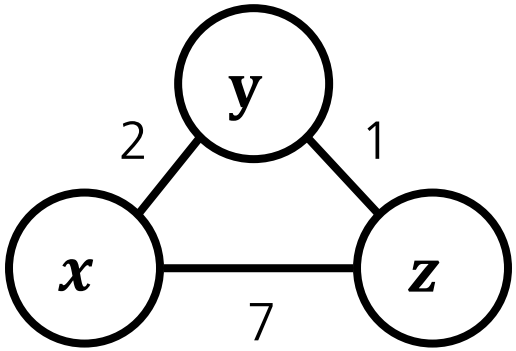
- **Bessere Routen** als vorher haben
  - x zu z über y und
  - z zu x über y

  gelernt, jeweils mit einer Verbesserung von 7 auf 3.
- Iteration 1 ist nun beendet.

Algorithmus von Bellman & Ford – Iteration 2

x sendet $\begin{array}{|c|}\hline z \\ \hline 3 \\ \hline \end{array}$ an y und z

z sendet $\begin{array}{|c|}\hline x \\ \hline 3 \\ \hline \end{array}$ an x und y



x

Wegpunkt \ Ziel	x	y	z
x	0	2	7
y	4	2	3
z	14	8	7

y

Wegpunkt \ Ziel	x	y	z
x	2	4	**5**
y	2	0	1
z	**4**	2	1

z

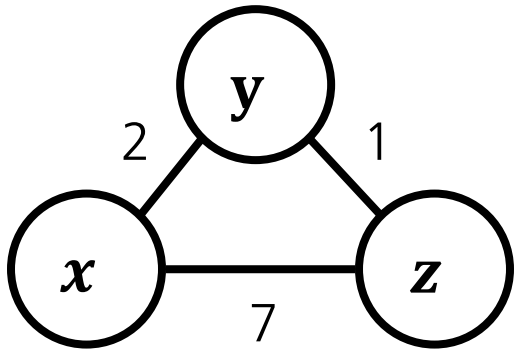
Wegpunkt \ Ziel	x	y	z
x	7	9	14
y	3	1	2
z	7	1	0

- Es werden nur **neue beste Distanzen** propagiert.
- y lernt daher von x, dass dieser z nun zu Kosten 3 erreichen kann, d.h. y kann z über x zu Kosten 5 erreichen.
- Ebenso lernt y von z, dass x über z nun zu Kosten von 4 erreichbar ist.

Algorithmus von Bellman & Ford – Iteration 2

x sendet | z / 3 | an y und z

z sendet | x / 3 | an x und y



x

Wegpunkt \ Ziel	x	y	z
x	0	2	7
y	4	2	3
z	14	8	7

y

Wegpunkt \ Ziel	x	y	z
x	2	4	**5**
y	2	0	1
z	**4**	2	1

z

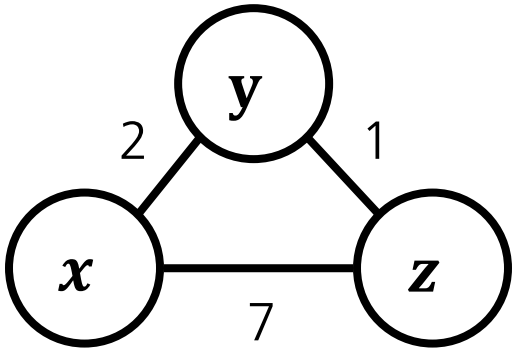
Wegpunkt \ Ziel	x	y	z
x	7	9	14
y	3	1	2
z	7	1	0

- Da y keine Verbesserung erlernt hat (die bekannten Routen zu x und z sind weiterhin besser als die neu erlernten), sendet y kein Update an seine Nachbarn.

Algorithmus von Bellman & Ford – Iteration 2

x sendet | z | 3 | an y und z

z sendet | x | 3 | an x und y



x

Wegpunkt \ Ziel	x	y	z
x	0	2	7
y	4	2	3
z	**10**	8	7

y

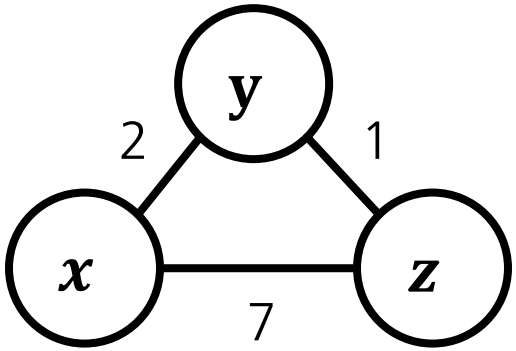
Wegpunkt \ Ziel	x	y	z
x	2	4	5
y	2	0	1
z	4	2	1

z

Wegpunkt \ Ziel	x	y	z
x	7	9	**10**
y	3	1	2
z	7	1	0

- Analog lernt x, dass er sich selbst mit z als nächsten Knoten nun zu Kosten 10 erreichen könnte (Route $\mathrm{x} \rightarrow z \rightarrow y \rightarrow x$).
- Schließlich lernt z, dass er sich selbst über x ebenfalls zu Kosten 10 erreichen könnte (Route $\mathrm{z} \rightarrow x \rightarrow y \rightarrow z$)
- Für keinen der beiden Knoten ergibt sich eine neue beste Route.

Algorithmus von Bellman & Ford – Iteration 2



x

Wegpunkt \ Ziel	x	y	z
x	0	2	7
y	4	2	3
z	10	8	7

y

Wegpunkt \ Ziel	x	y	z
x	2	4	5
y	2	0	1
z	4	2	1

z

Wegpunkt \ Ziel	x	y	z
x	7	9	10
y	3	1	2
z	7	1	0

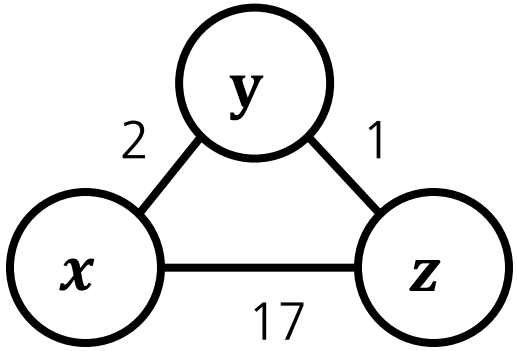
- Da keiner der Knoten eine neue beste Route gefunden hat, sendet keiner eine weitere Aktualisierungsnachricht.
- Der Algorithmus ist konvergiert.

Das Count-To-Infinity-Problem

- Das **Count-to-Infinity-Problem** taucht beim Distanzvektorverfahren immer dann auf, wenn ein Link sich verschlechtert oder gar komplett ausfällt (Kosten ∞).
- Der betroffene Router „meint“, dass seine Nachbarn eine leicht schlechtere Route haben, die aber tatsächlich wieder über **ihn selbst** führen würde.
- Diese leicht schlechtere Route wird zwischen den Routern hin- und hergereicht, jedes mal mit einer durch den zusätzlichen Link aufaddierten Kostensteigerung.
- Existiert keine Alternativroute. endet dieses Wechselspiel nicht (*count-to-infinity*). Bestehende Alternativrouten werden erst sehr spät berücksichtigt.

Das Count-To-Infinity-Problem – Beispiel

Ausgangslage:



y

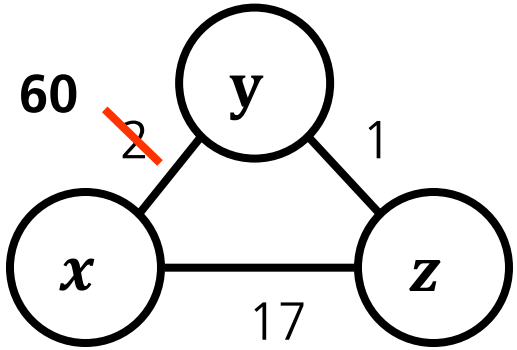
	x
über x	2
über z	4

z

	x
über x	17
über y	3

Das Count-To-Infinity-Problem – Beispiel

Link verschlechtert sich



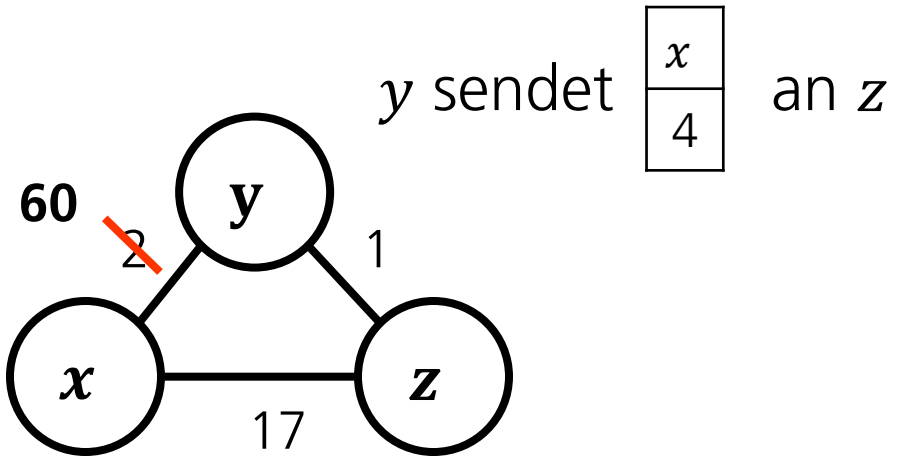
y

über	x
x	**60**
z	4

z

über	x
x	17
y	3

Das Count-To-Infinity-Problem – Beispiel



y

	x
x	60
z	**4**

über

z

	x
x	17
y	3

über

y geht fälschlicherweise davon aus, dass z eine Alternativroute zu x mit Kosten 3 hätte. Tatsächlich führt diese aber wieder über y. Diese Information teilt z y aber nicht mit (nicht Teil des Protokolls)

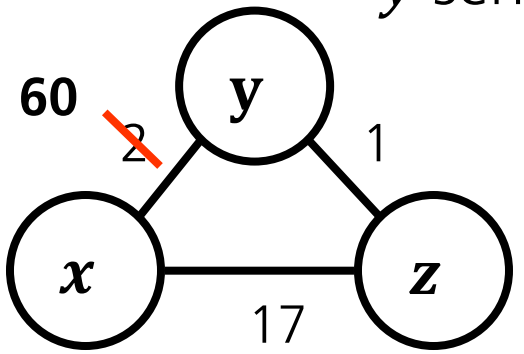
Das Count-To-Infinity-Problem – Beispiel

y trägt neue beste Route ein und teilt diese z mit

y sendet

x
4

an z



y

	x
x	60
z	**4**

über

z

	x
x	17
y	3

über

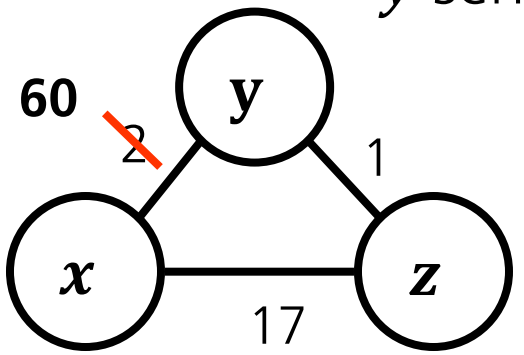
Das Count-To-Infinity-Problem – Beispiel

z passt Distanz an

y sendet

x
4

an z



y

	x
über x	60
über z	**4**

z

	x
über x	17
über y	3

z

	x
über x	17
über y	**5**

z weiß, dass seine beste Route zu x über y geht. Da y gerade eine Verschlechterung bekanntgegeben hat (von 2 auf 4), passt z seine Kosten an.

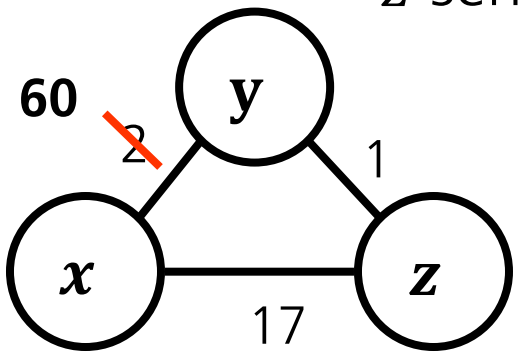
Das Count-To-Infinity-Problem – Beispiel

und sendet seinerseits eine Aktualisierung

z sendet

x
5

an y



y

über	x	60
	z	**4**

z

über	x	17
	y	3

z

über	x	17
	y	**5**

Diese Kostenänderung propagiert z nun wieder an y.

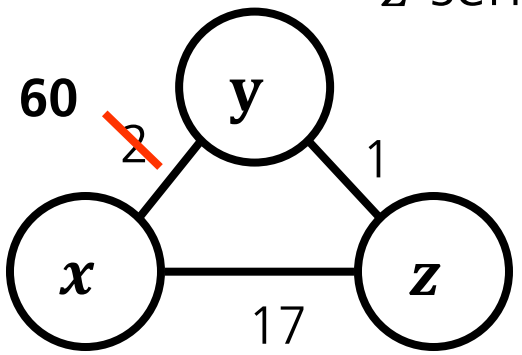
Das Count-To-Infinity-Problem – Beispiel

y passt Distanz an

z sendet

x
5

an y



y

	x
über x	60
über z	**4**

y

	x
über x	60
über z	**6**

z

	x
über x	17
über y	3

z

	x
über x	17
über y	**5**

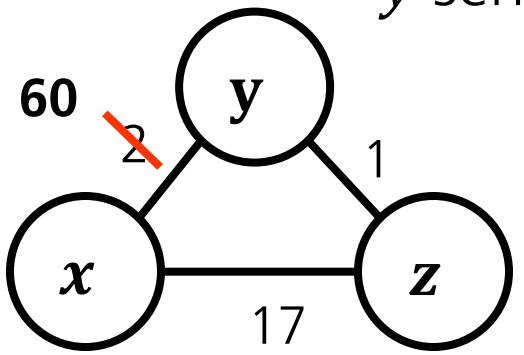
Das Count-To-Infinity-Problem – Beispiel

und sendet seinerseits eine Aktualisierung

y sendet

x
6

an z



y

	x
x	60
z	**4**

über

y

	x
x	60
z	**6**

über

z

	x
x	17
y	3

über

z

	x
x	17
y	**5**

über

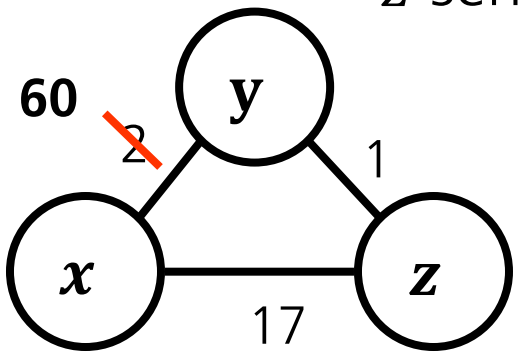
Das Count-To-Infinity-Problem – Beispiel

z sendet

x
7

an y

z passt an und sendet Aktualisierung



y

über	x
x	60
z	**4**

y

über	x
x	60
z	**6**

z

über	x
x	17
y	3

z

über	x
x	17
y	**5**

z

über	x
x	17
y	**7**

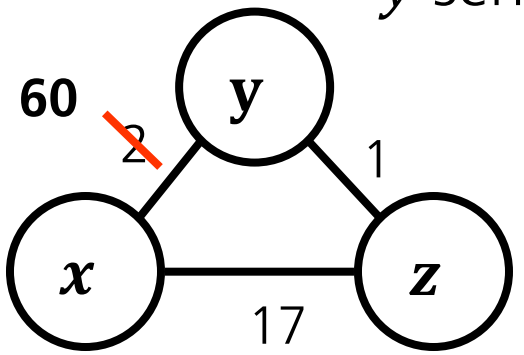
Das Count-To-Infinity-Problem – Beispiel

y passt an und sendet Aktualisierung

y sendet

x
8

an z



y

	x
über x	60
über z	**4**

y

	x
über x	60
über z	**6**

y

	x
über x	60
über z	**8**

z

	x
über x	17
über y	3

z

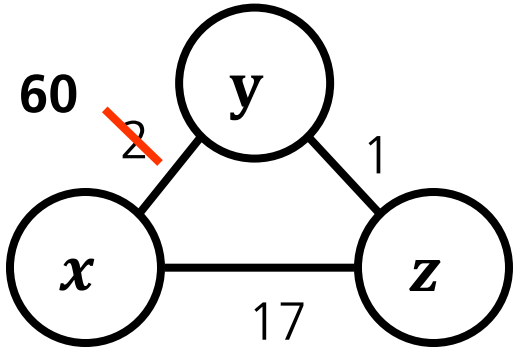
	x
über x	17
über y	**5**

z

	x
über x	17
über y	**7**

Das Count-To-Infinity-Problem – Beispiel

Weiterzählen bis zur Unendlichkeit



y

über	x
x	60
z	**4**

z

über	x
x	17
y	3

z

über	x
[illegible]	17
y	**5**

y

über	x
x	60
z	**6**

z

über	x
[illegible]	17
y	**7**

y

über	x
x	60
z	**8**

z

über	x
x	17
y	**9**

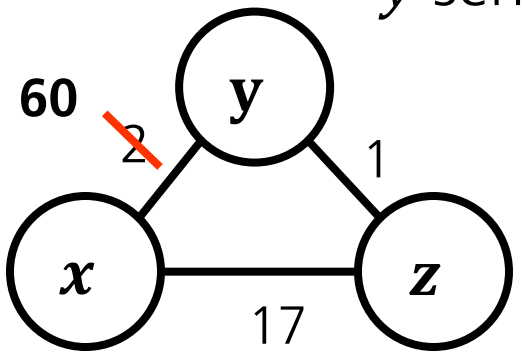
Das Count-To-Infinity-Problem – Beispiel

Der direkte Link von z zu x setzt sich erst spät durch.

y sendet

x
17

an z



...

y

	x
x	60
z	**17**

über

z hat bessere Route mit Kosten 17

z

	x
x	17
y	**16**

über

z

	x
x	**17**
y	18

über

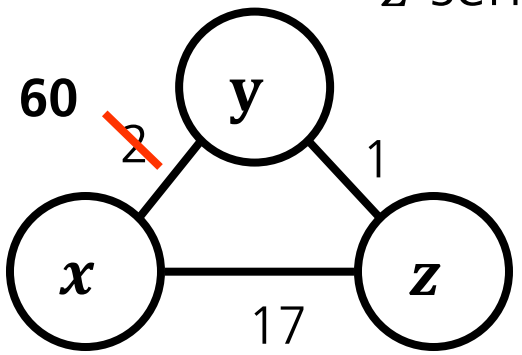
Das Count-To-Infinity-Problem – Beispiel

Der direkte Link von z zu x setzt sich erst spät durch.

z sendet

x
17

an y



...

y

	x
x	60
z	**17**

über

z

	x
x	17
y	**16**

über

z

	x
x	**17**
y	18

über

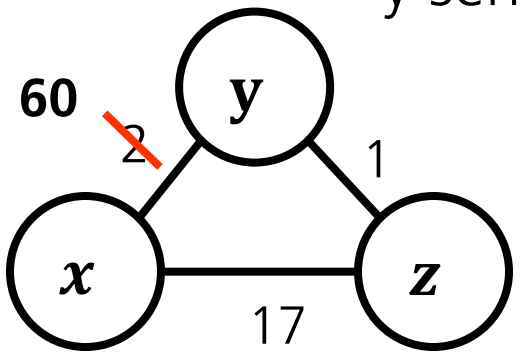
Das Count-To-Infinity-Problem – Beispiel

Der direkte Link von z zu x setzt sich erst spät durch.

y sendet

x
18

an z



...

y

	x
x	60
z	**17**

über

y

	x
x	60
z	**18**

über

z

	x
x	17
y	**16**

über

z

	x
x	**17**
y	18

über

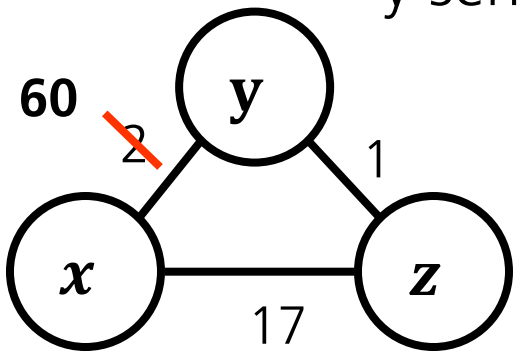
Das Count-To-Infinity-Problem – Beispiel

Erst jetzt terminiert der Algorithmus.

y sendet

x
18

an z



...

y

	x
x	60
z	**17**

über

y

	x
x	60
z	**18**

über

z

	x
x	17
y	**16**

über

z

	x
x	**17**
y	18

über

z

	x
x	**17**
y	18

über

Das Count-To-Infinity-Problem

- Gute Nachrichten verbreiten sich sofort, schlechte Nachrichten nur zögerlich.
- Erster Lösungsansatz: Routinginformationen werden nicht in die Richtung propagiert, aus der sie empfangen wurden. (Split-Horizon). Dieser Ansatz hilft aber nur lokal.
- Weitere Verfahren sind komplexer und werden hier nicht weiter betrachtet.

Zusammenfassung

- Die Vermittlungsschicht bietet logische Netzwerke über physikalische Netzwerkgrenzen hinweg.
- Entscheidende Aufgaben: Adressierung und Wegefindung.
- Diese Aufgaben werden vom Internet Protocol (IP) übernommen, es ist die Grundlage unserer heutigen weltweiten Vernetzung.
- Vermittler (Router) benötigen geeignete Protokolle, um lokale Weiterleitungsinformationen aktuell zu halten. Es gibt zentrale und dezentrale Verfahren.