

Datenkommunikation und Sicherheit

Kapitel 4: Vermittlungsschicht

Klaus Wehrle

Communication and Distributed Systems

Chair of Computer Science 4

RWTH Aachen University

<http://www.comsys.rwth-aachen.de>



IV-1

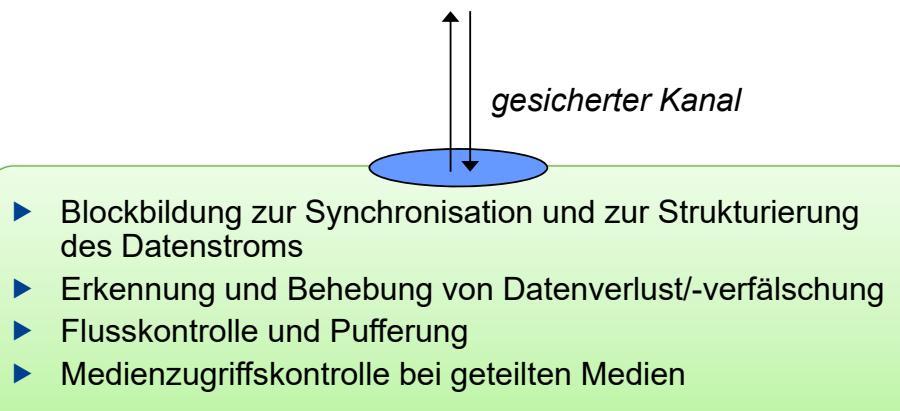
Themenübersicht

• Datenkommunikation und Sicherheit

- ▶ Einführung, Begriffe und allgemeine Grundlagen
- ▶ Technische und nachrichtentechnische Grundlagen: Medien, Signale, Bandbreite, Leitungscodes und Modulation, Multiplexing
- ▶ Lokale Netze: Strukturierung des Datenstroms, Fehlererkennung/-behebung, Flusssteuerung, Medienzugriff, Ethernet
- ▶ Internet und Internet-Protokolle:
 - Vermittlungsschicht: IP, Routing
 - Transportschicht: TCP
- ▶ Grundlagen der Sicherheit in/von Kommunikationsnetzen
 - Grundlagen: Verschlüsselung, Authentifizierung, Integrität
 - Sichere Internet-Protokolle

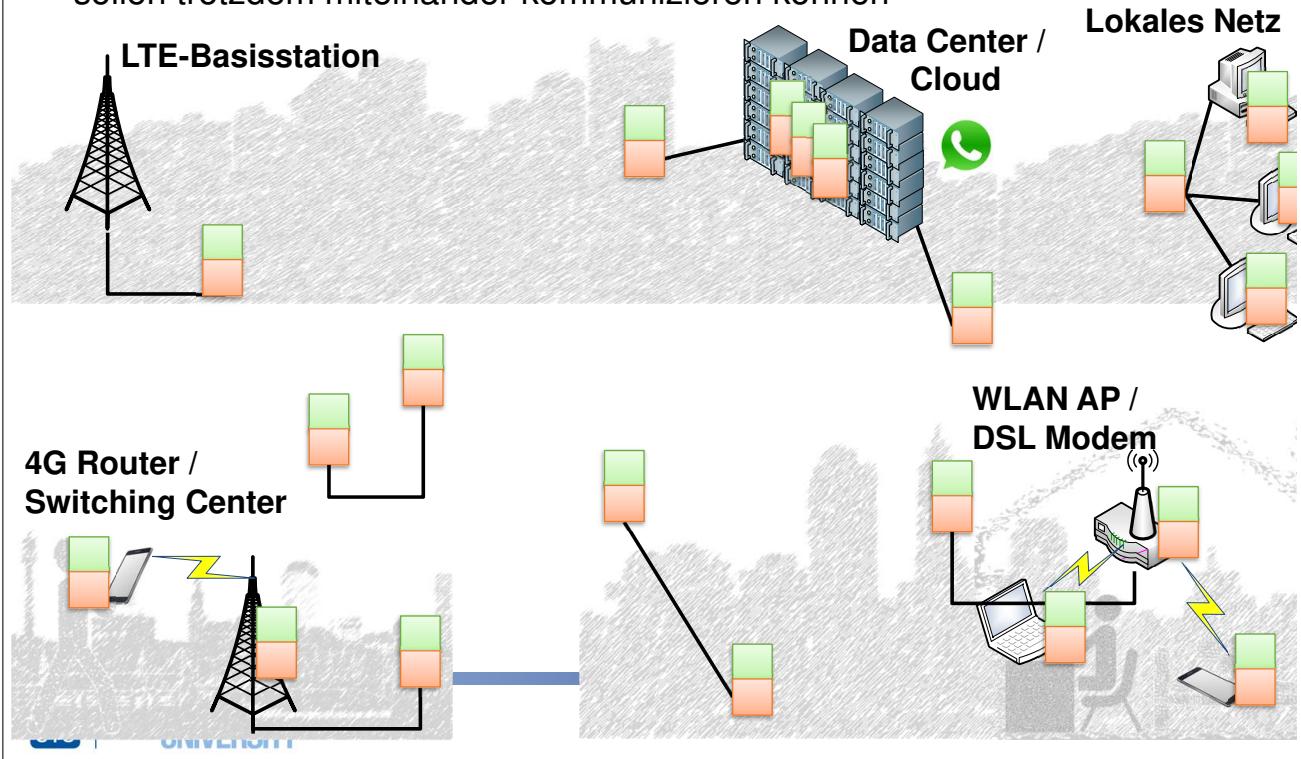
Aufgaben der Vermittlungsschicht

Schicht 2:
Gesicherter
Übertragung
von Rahmen



Vermittlungssysteme

- ▶ Systeme, die nicht direkt durch ein physikalisches Medium verbunden sind, sollen trotzdem miteinander kommunizieren können

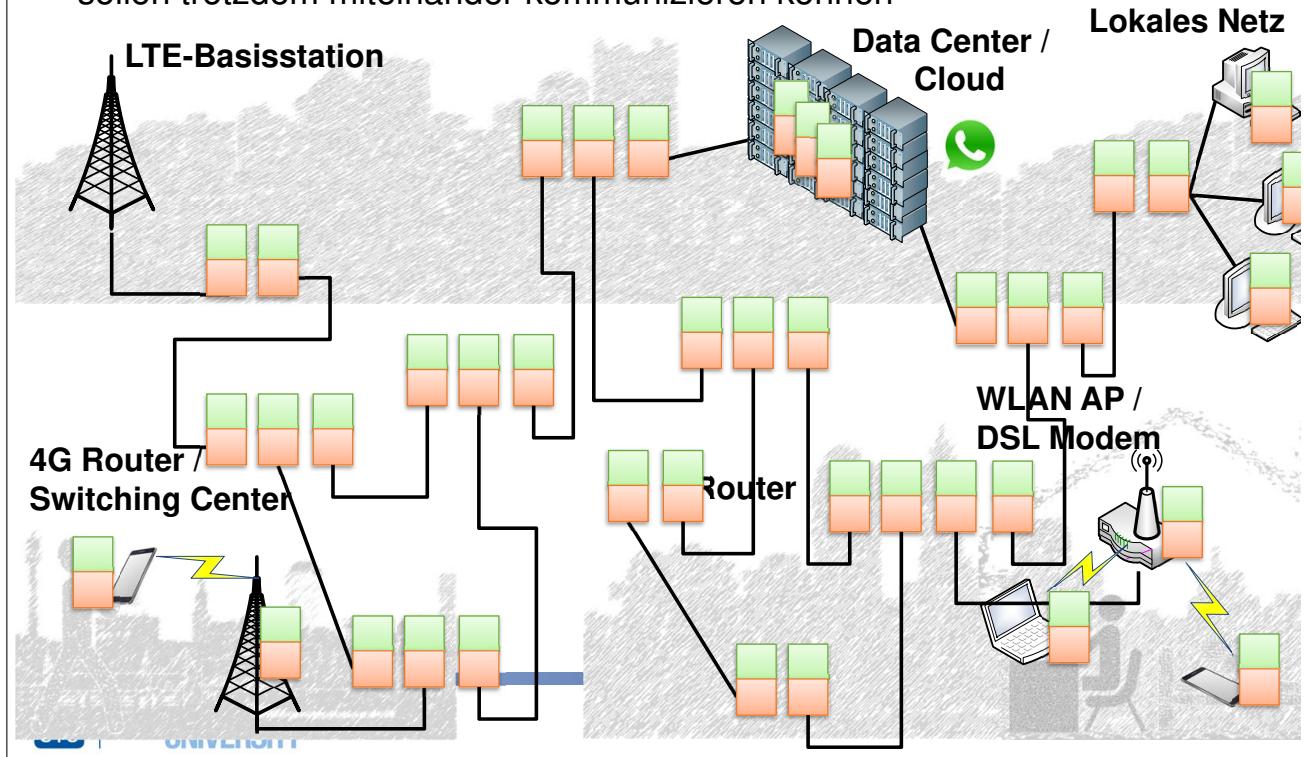


Die Hauptaufgabe der Vermittlungsschicht ist die Schaffung von Endsystemverbindungen zwischen beliebigen Endsystemen im weltweiten Netzwerk. Dazu ist die Kopplung aller lokalen Netze notwendig. Dies ließe sich zwar auf Schicht 2 durch Brücken erreichen, die auch eine Protokollwandlung vornehmen können, aber aus verschiedenen Gründen wäre dies problematisch.

Daher findet eine tatsächliche Kopplung von Netzen erst auf Schicht 3 statt, auf der „Vermittlungsknoten“ – im Internet: Router – zur Verbindung von LANs und zur Weiterleitung von Daten zwischen beliebigen LANs eingesetzt werden.

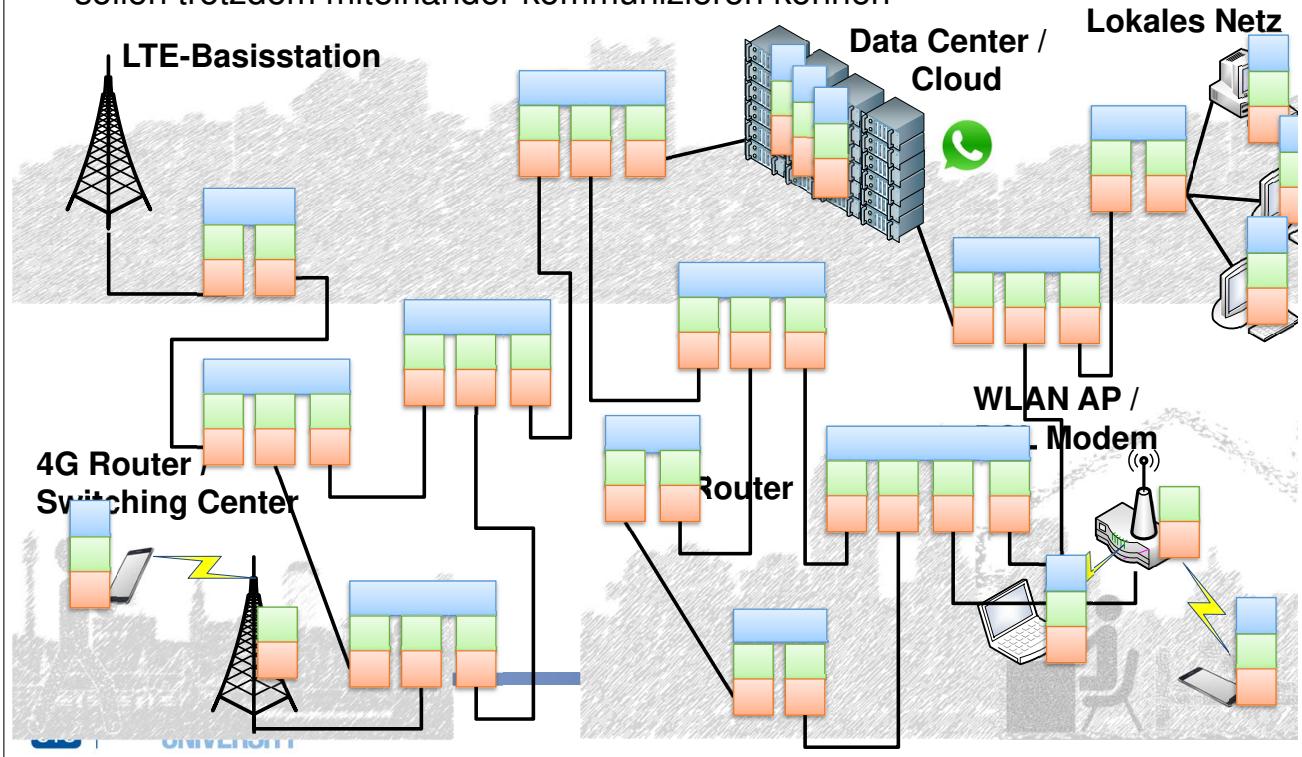
Vermittlungssysteme

- ▶ Systeme, die nicht direkt durch ein physikalisches Medium verbunden sind, sollen trotzdem miteinander kommunizieren können



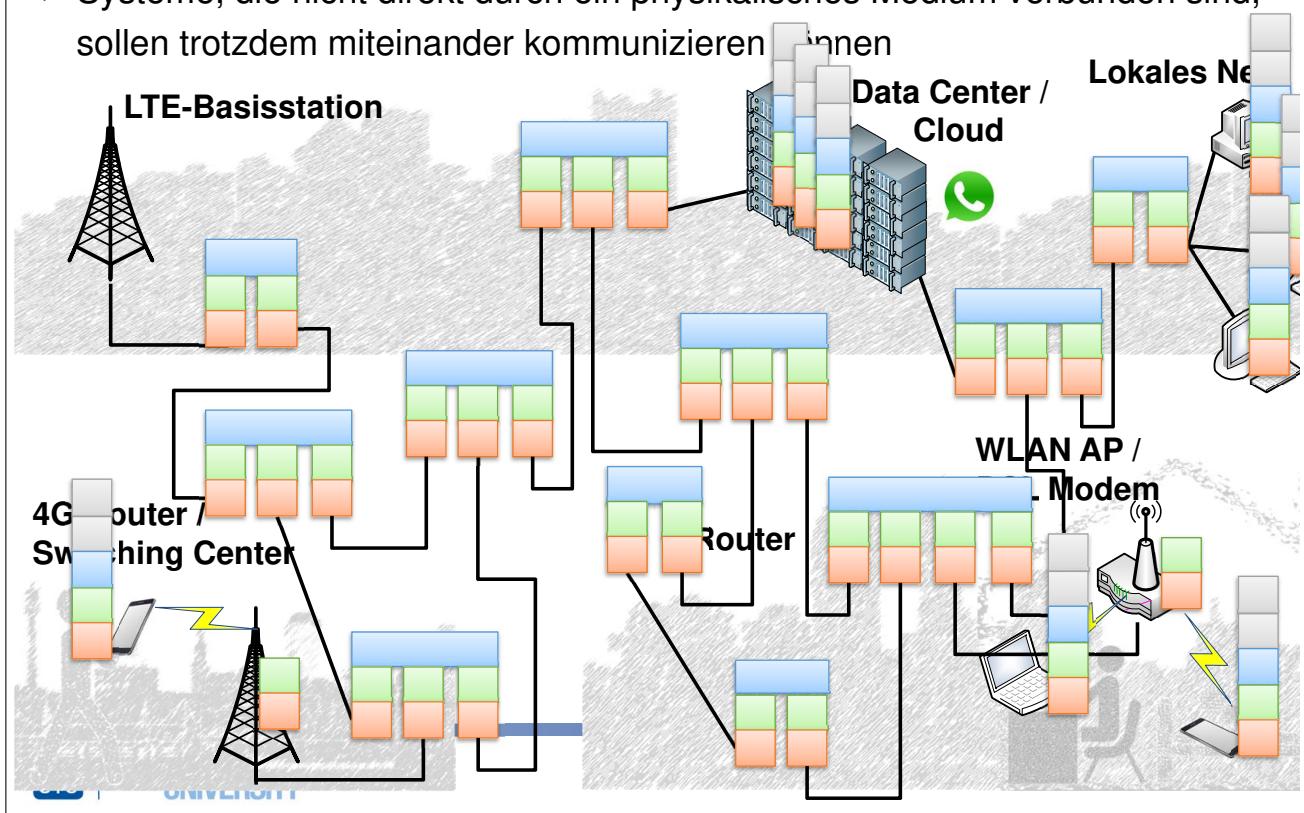
Vermittlungssysteme

- ▶ Systeme, die nicht direkt durch ein physikalisches Medium verbunden sind, sollen trotzdem miteinander kommunizieren können

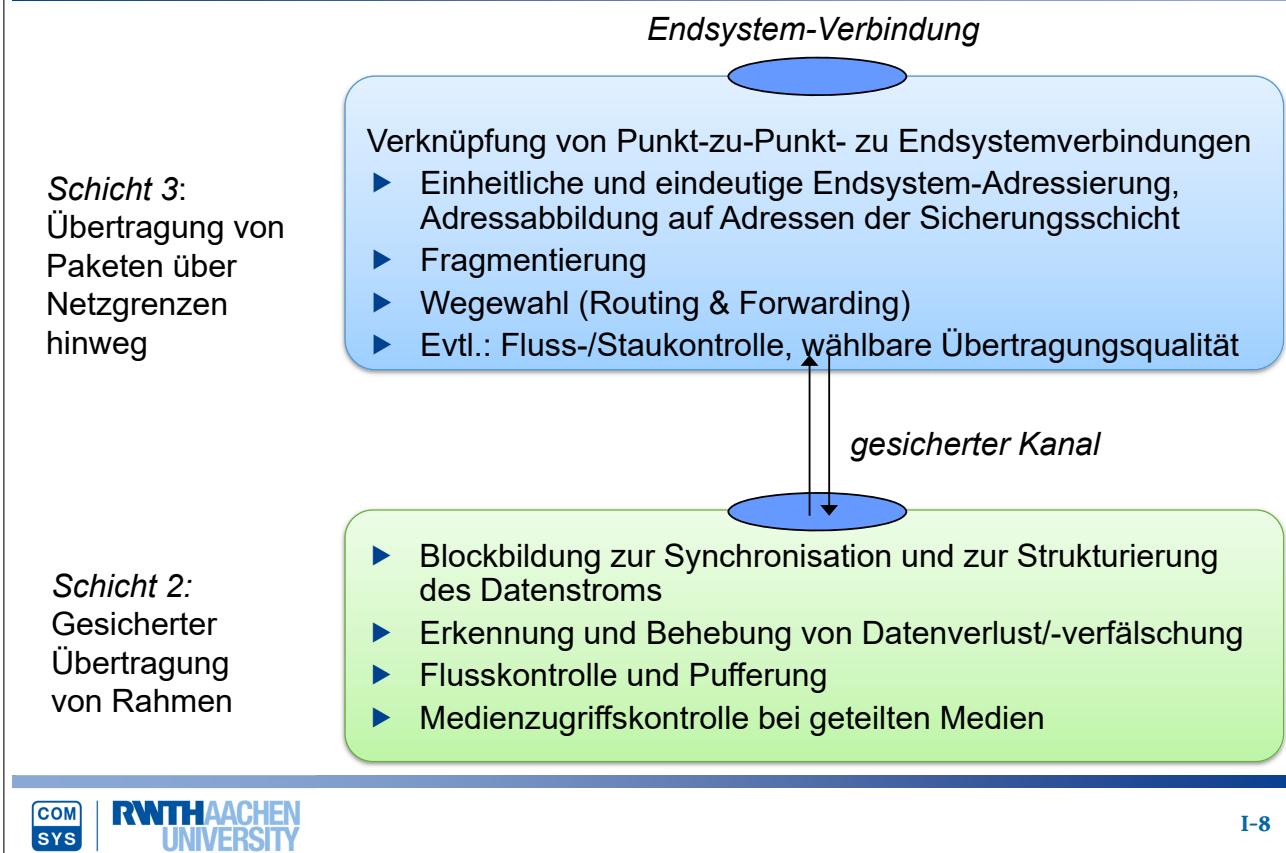


Vermittlungssysteme

- ▶ Systeme, die nicht direkt durch ein physikalisches Medium verbunden sind, sollen trotzdem miteinander kommunizieren können



Aufgaben der Vermittlungsschicht



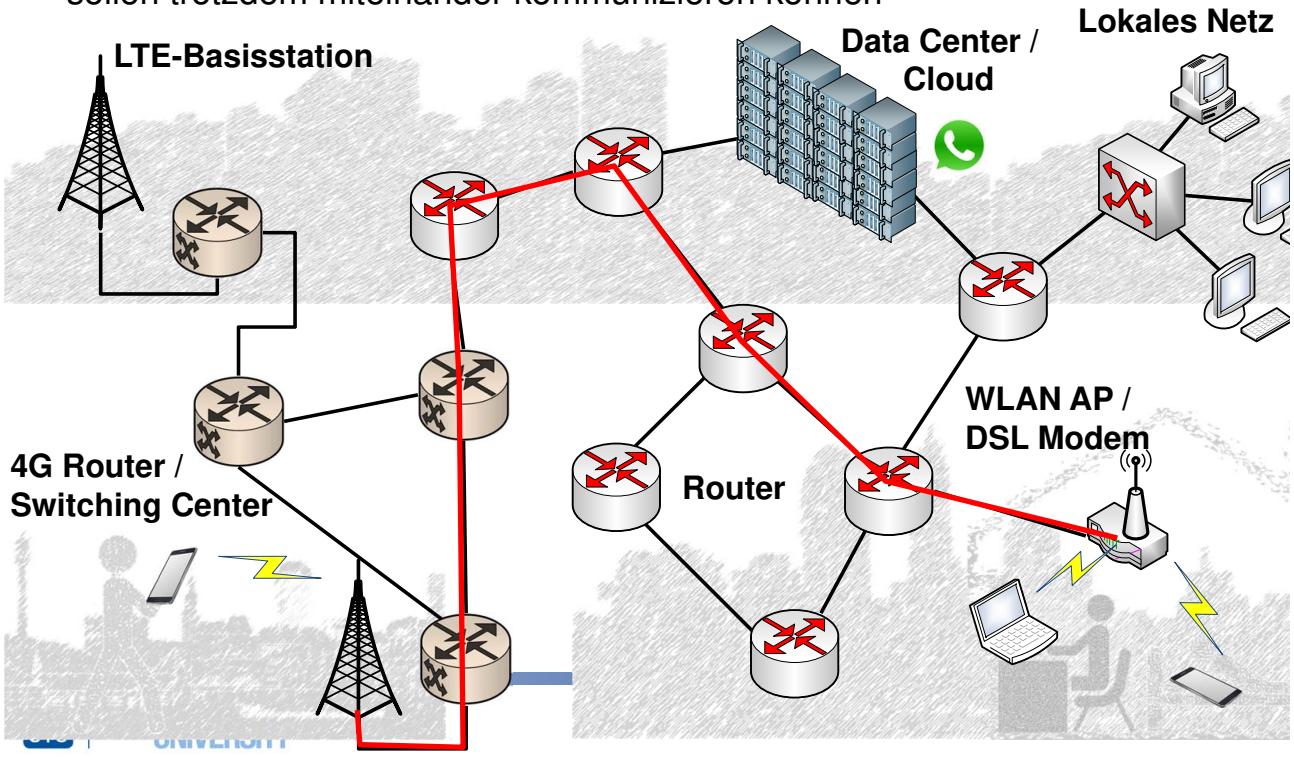
Die Vermittlungsschicht ist am engsten mit dem Begriff eines Netzwerkes verbunden und wird deshalb auch oft als Netzwerkschicht bezeichnet. Im TCP/IP-Modell heißt sie „Internet-Schicht“ (= die Schicht, die Kommunikation über Netzgrenzen hinweg ermöglicht). Dies liegt darin begründet, dass erst diese Schicht die lokalen Punkt-zu-Punkt-Verbindungen, d.h. die einzelnen physikalischen Links zwischen benachbarten Netzwerknoten, verknüpft und Verbindungen zwischen entfernten Systemen über eine Vielzahl von Zwischensystemen (Vermittlungsknoten, typischerweise Router) hinweg erlaubt (*Endsystemverbindung*). Hierzu stellt die Vermittlungsschicht eine Adressierung zur Verfügung, die die weltweit eindeutige Identifikation einzelner Systeme ermöglicht (in der Praxis IP-Adressen) und anhand derer eine geeignete Wegewahl durchgeführt werden kann. Im lokalen Netz (bzw. auf jeder Teilstrecke) muss diese Adressierung dann von der Vermittlungsschicht wieder auf die MAC-Adressen der Schicht 2 abgebildet werden.

Die Datenübertragung selbst findet weiterhin über die unteren Schichten 2 und 1 statt.

Viele der Protokollmechanismen der Schicht 2 können sich auch in der Vermittlungsschicht wiederfinden, so z.B. auch Verfahren zur Flusskontrolle und Datensicherung. Hierbei muss man sich den veränderten Bezug dieser Mechanismen vor Augen halten: im Gegensatz zur Schicht 2 sind die hierbei beteiligten Systeme nicht benachbart, sondern möglicherweise Tausende von Kilometern voneinander entfernt und es müssen neue Fehlerquellen berücksichtigt werden, die durch das Zusammenwirken unterschiedlicher Netze sowie durch die Vermittlungsknoten entstehen. Achtung: diese Mechanismen werden hier kaum behandelt und sind auf obiger Folie mit „Eventuell: ...“ bezeichnet, da diese Funktionen in der Praxis (IP) keine Verwendung finden.

Vermittlungssysteme

- ▶ Systeme, die nicht direkt durch ein physikalisches Medium verbunden sind, sollen trotzdem miteinander kommunizieren können



Vermittlung

- **Aufgabe: Bereitstellung eines Kommunikationswegs durch das Netz auf Anforderung**

- ▶ Schicht-3-Verbindung (*Endsystemverbindung*)
 - Temporäre oder permanente Bereitstellung möglich
 - Mit oder ohne Zustand
- ▶ Über eine Kette von *Vermittlungsstellen (Routern)*
 - Zwischen Vermittlungsstellen Austausch von Kontrollnachrichten (*Signalisierung*)
- ▶ Arten der Vermittlung:
 - Leitungsvermittlung oder *Speicher-/Paketvermittlung*
 - *Verbindungsorientierte/verbindungslose* Vermittlungstechniken

Aufgabe der Vermittlung ist es also, zum Zwecke einer Datenübertragung zwischen zwei Teilnehmern einen geeigneten Kommunikationsweg über eine Kette von Vermittlungsstellen bereitzustellen. Dies kann entweder verbindungsorientiert erfolgen (Verbindungsaufbauphase vor dem Datenaustausch, in der der Übertragungsweg festgelegt wird) oder verbindungslos (jedes Datenpaket wird isoliert betrachtet und zum Empfänger weitergeleitet).

Verbindungen auf Schicht 3 werden auch Endsystemverbindungen genannt, da sie zwei entfernte Endsysteme (Teilnehmer) koppeln. Wie wir in Kapitel 5 sehen werden, dient erst die Transportschicht (Schicht 4) dazu, innerhalb eines Endsystems nochmals zwischen einzelnen Anwendungen zu differenzieren (Ende-zu-Ende-Verbindungen).

Als Signalisierung wird die vermittelungstechnische Kommunikation zwischen einem Endgerät (Rechner) und dem Netz sowie die Kommunikation im Netzinnen bezeichnet.

In der Telekommunikation unterscheidet man zwei grundsätzlich verschiedene Formen der Vermittlung:

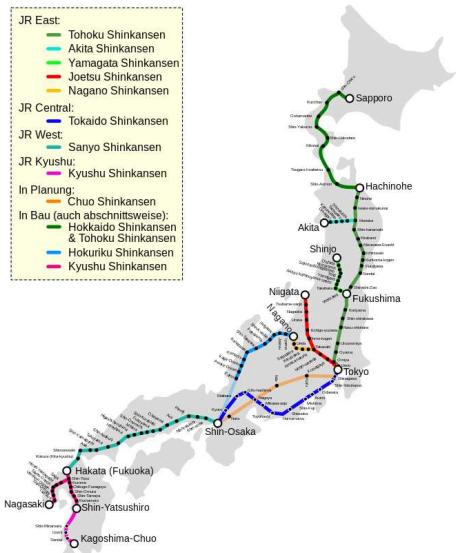
- Durchschaltevermittlung (Leitungsvermittlung)
- Speichervermittlung bzw. Paketvermittlung

Diese beiden Vermittlungsarten werden auf den nächsten Folien erläutert.

Leitungs- und Paketvermittlung

• Leitungsvermittlung

- ▶ Geplantes Schienennetz
- ▶ Pro Strecke eine Linie



• Paketvermittlung

- ▶ Bekannte Streckenführungen
- ▶ Keine globale Planung
- ▶ Z.B. Verwendung eines Navis, um dem für ein Fahrzeug aktuell besten Weg zu folgen



Kapitel 4: Vermittlungsschicht

• Vermittlungsverfahren

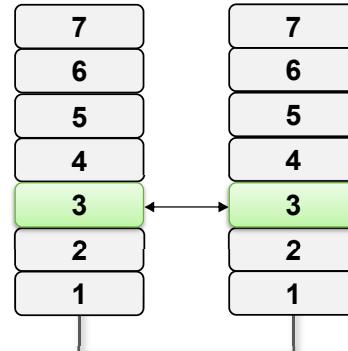
- ▶ Leitungsvermittlung, Speicher-/Paketvermittlung
- ▶ Beispiel ATM

• Die Vermittlungsschicht im Internet – IP

- ▶ IPv4: Adressen, Subnetze, CIDR, NAT
- ▶ IPv4-Header
- ▶ Hilfsprotokolle: ARP, DHCP, ICMP
- ▶ IPv4 vs. IPv6

• Wegewahlverfahren (Routing)

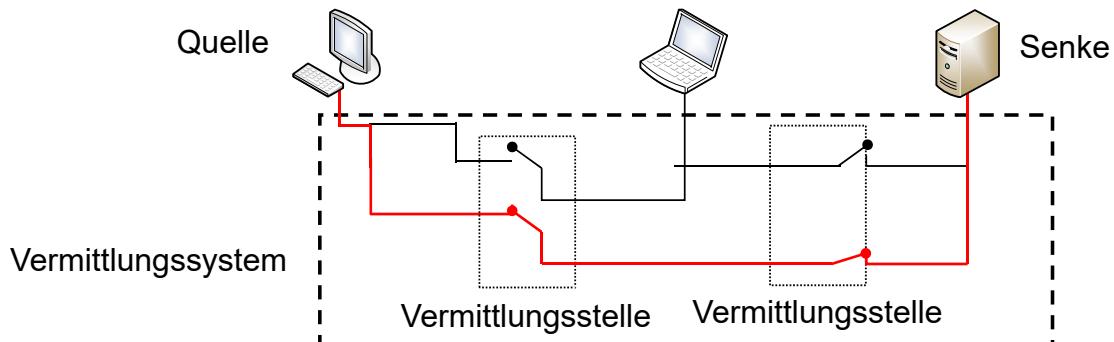
- ▶ Hierarchien, einfache Verfahren
- ▶ Distance Vector
- ▶ Link State



Leitungsvermittlung (Circuit Switching)

- Schalten einer *durchgängigen, dedizierten Verbindung*

- ▶ Von der Quelle zur Senke über das Vermittlungssystem
- ▶ Verbindung besteht aus nachrichtentechnischen Kanälen
 - Geschaltet für Dauer der Verbindung
 - Reihenfolgetreue Übertragung garantiert
- ▶ Z.B. klassisches Telefonnetz



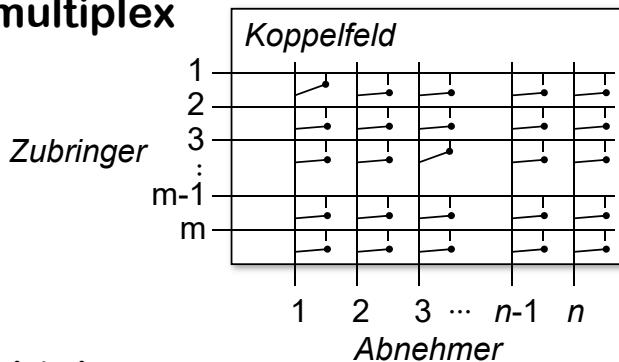
Bei der Leitungsvermittlung (engl. Circuit Switching, anderer Begriff: Durchschaltevermittlung) wird durch das Vermittlungssystem (d.h. die beteiligten Vermittlungsstellen) eine durchgängige dedizierte Verbindung von der Quelle zur Senke geschaltet. Dazu existiert eine Verbindungsauftauphase vor dem Datenaustausch, in der die Verbindung fest geschaltet wird (in dieser Phase werden auch die wichtigsten Steuerinformationen ausgetauscht, wie z.B. Adressierungs- und Abrechnungsinformation).

Das bekannteste Beispiel eines Kommunikationsnetzes, welches auf der Basis der Leitungsvermittlung arbeitet, ist das Telefonnetz. Wichtig ist, dass die nachrichtentechnischen Kanäle, die Bestandteil der geschalteten Verbindung sind, während der gesamten Dauer der Verbindung ausschließlich von den beteiligten Teilnehmern genutzt werden.

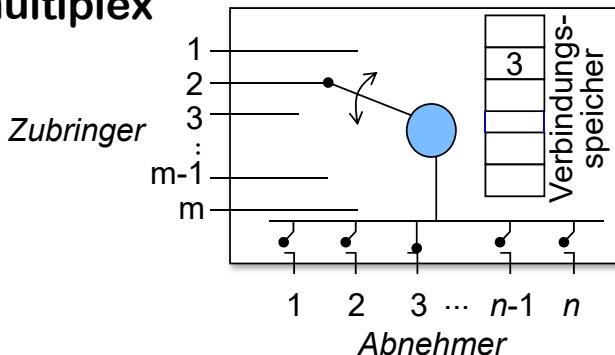
Die Kernfunktionalität der Leitungsvermittlung besteht in der Kopplung von Medien, und hier lassen sich zwei grundlegende Verfahren unterscheiden, wie auf der nächsten Folie gezeigt wird.

Leitungsvermittlung: Medienkopplungsverfahren

• Raummultiplex



• Zeitmultiplex



Die verallgemeinerte Aufgabenstellung der Medienzuordnung besteht in der Kopplung von m Zubringerleitungen an n Abnehmerleitungen.

Durch die Verwendung eines sogenannten *Raummultiplex*-Koppelfeldes kann jede Zubringerleitung auf beliebige Abnehmerleitungen geschaltet werden, indem die entsprechenden Schaltelemente der Matrix geschlossen werden. Diese beliebige Medienzuordnung ist möglich, weil jedes Schaltelement einzeln und unabhängig voneinander angesteuert werden kann.

Sehr viel eingeschränktere Möglichkeiten bietet hingegen das *Zeitmultiplex*-Verfahren, bei welchem zu jedem Zeitpunkt immer nur eine Zubringerleitung mit einer Abnehmerleitung nach Wahl verbunden werden kann. Jeder Zubringerleitung wird ein Zeitschlitz zugeordnet.

Im Verbindungsspeicher ist zu jeder Zubringerleitung vermerkt, auf welche Abnehmerleitung geschaltet werden soll.

Für beide Verfahren gilt: Ist ein durchgehender Übertragungskanal (eine Leitung) zwischen den Teilnehmern aufgebaut, so sind Übertragungsverzögerungen auf physikalisch bedingte signaltechnische Laufzeiten beschränkt. Bitfolgen werden reihenfolgetreu übertragen, damit wird die Reihenfolge von PDUs des Absenders beim Empfänger beibehalten. Dies mag logisch klingen, doch bei anderen Verfahren, die im Folgenden noch erläutert werden, kann es durchaus vorkommen, dass PDUs beim Empfänger nicht in der korrekten Reihenfolge ankommen – diese Eigenschaft ist also ein klarer Vorteil von Leitungsvermittlung.

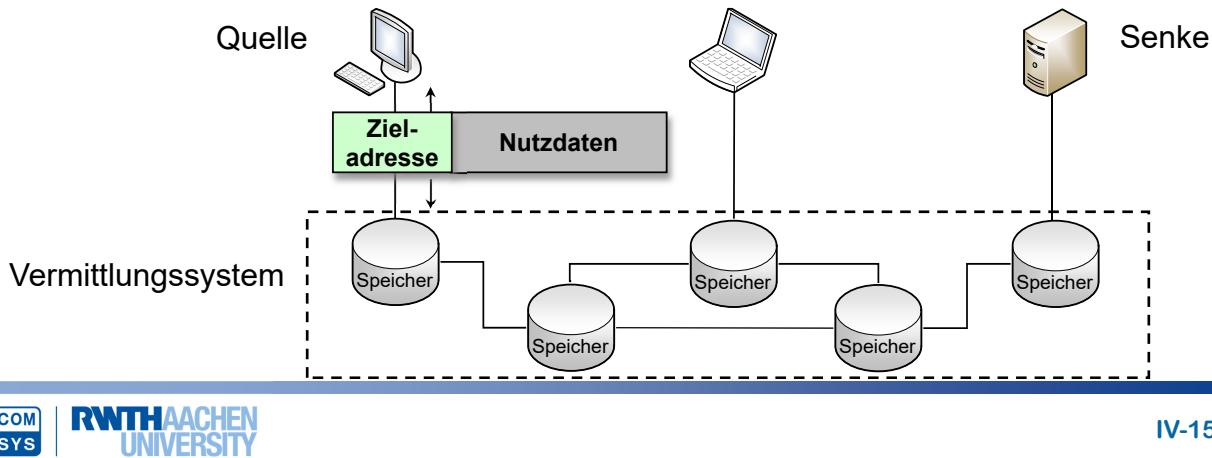
Streng genommen realisiert die Zeitmultiplex-Variante keine echte Leitungsvermittlung mehr, da Daten nicht direkt weitergeleitet werden können, sondern solange zwischengespeichert werden müssen, bis der passende Zeitslot zur Weiterleitung beginnt. Dies ist bereits ein Element der Speichervermittlung. Wird allerdings eine statische Zuordnung von Zeitslots zu Zubringerleitungen vorgenommen, behält die Weiterleitung trotz der Zwischenspeicherung alle Eigenschaften der Leitungsvermittlung – man kann dies als Vermittlung auf einer *virtuellen Leitung* bezeichnen.

Bei Verwendung des Zeitmultiplex gibt es auch die Möglichkeit, Zeitlots anforderungsgesteuert zuzuweisen (statistisches Zeitmultiplex). Dies führt uns zur Speichervermittlung.

Speicher-/Paketvermittlung (Packet Switching)

• „Store & Forward“-Prinzip

- ▶ Vermittlungsstellen puffern Daten (Pakete) und leiten sie später weiter
 - Zeitlich unabhängige Weiterleitung je Paket
- ▶ Keine „Besetzt“-Fälle durch fest geschaltete Leitungen
- ▶ Bessere Bandbreitenausnutzung
- ▶ Verlust von Daten, Zustellung in falscher Reihenfolge möglich



Bei der Speichervermittlung werden die zu vermittelnden Daten in einem Netzknoten zwischengespeichert und bei freier Teilstrecke zu einem geeigneten benachbarten Netzknoten weitergeschickt, bis sie bei der Senke angekommen sind.

Im Englischen heißt diese Form der Vermittlung treffend *Store-and-forward Switching*, was genau die beiden Aktionen umfasst (speichern und weiterschicken), die ein Netzknoten auszuführen hat. Ein Beispiel für ein nicht rechnergestütztes Transportsystem, das gemäß dem Store-and-Forward-Prinzip arbeitet, ist die Briefpost, wobei die Knoten die Postämter (bzw. Brief- und Paketzentren) sind.

- Bei der Speichervermittlung treten *keine Besetzfälle* durch für andere geschaltete Leitungen auf, sondern nur eventuell Wartezeiten: ist die ausgehende Leitung besetzt oder stark belastet, so werden die Daten zwischengespeichert und erst verzögert weitergegeben.
- Bei der Speichervermittlung kann es durchaus zu *Reihfolgevertauschungen* von Daten kommen, z.B. durch Verwendung unterschiedlicher Routen für verschiedene Dateneinheiten. Es besteht keine Zeitbeziehung zwischen den Dateneinheiten!
- Man kann *variable Datenraten* benutzen, d.h. hat man z.B. burstartigen Verkehr, so kann bei einem Burst – vorausgesetzt, das Netz ist nicht stark belastet – die gesamte Kapazität ausgenutzt werden, bei Sendepausen steht das Netz anderen zur Verfügung. Im Gegensatz dazu hätte man bei der Leitungsvermittlung dauerhaft eine feste Datenrate zur Verfügung.
- Während bei der Leitungsvermittlung eine Leitung durchgeschaltet ist, der alle Daten folgen, muss bei der Speichervermittlung in jeder Dateneinheit der Empfänger *adressiert* werden, damit die Vermittlungsstelle die einzelnen Pakete zustellen kann – es besteht kein Bezug zwischen den verschiedenen Dateneinheiten eines Senders.
- Es kann auch zu *Verlusten von Daten* kommen, wenn ein Zwischenknoten mehr Dateneinheiten zugestellt bekommt, als er weiterleiten kann und irgendwann sein Speicher überläuft. Daher müssen bei den Endsystemen mehr Protokollmechanismen vorhanden sein als bei Leitungsvermittlung (Fehlererkennung und -behandlung, Reihenfolgeerhaltung, Paketverluste, ...).

Eine Ausprägung der Speichervermittlung stellt die sogenannte Paketvermittlung (engl. Packet Switching) dar. Bei diesem Verfahren werden Daten in Form einzelner Pakete (= PDU auf Schicht 3) übertragen (dabei ist entweder eine feste Länge oder eine maximale Länge vorgegeben) und in den Vermittlungsstellen entlang des Übertragungsweges zwischengespeichert. Um eine Weiterleitung zu ermöglichen, müssen die Pakete als PCI zumindest die Adresse des Zielrechners enthalten. Die weitergehenden Möglichkeiten, Paketvermittlung zu betreiben (verbundungsorientiert oder verbundungslos) werden im Folgenden erläutert.

Verbindungslose Paketvermittlung

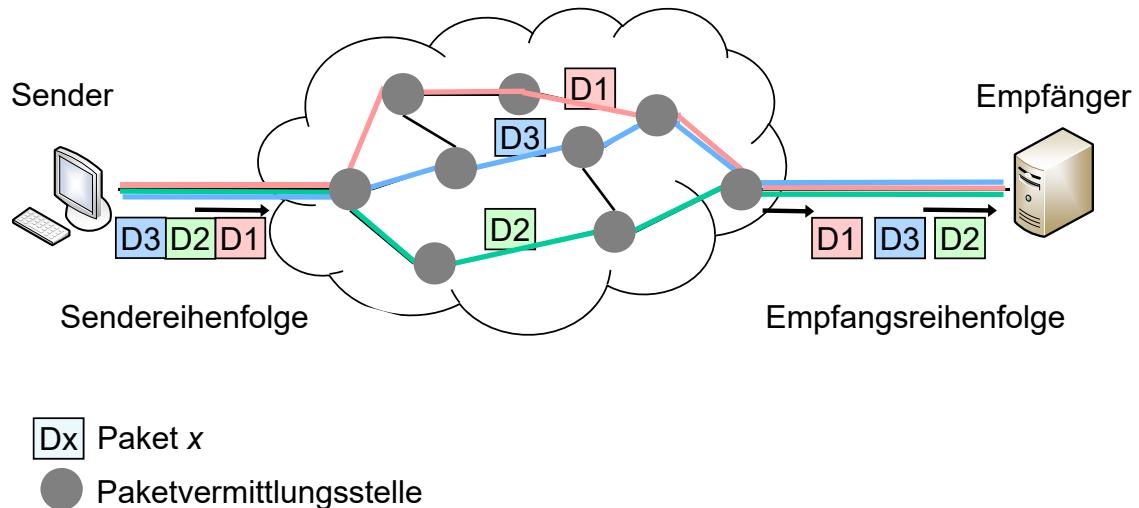
- **Pakete bilden geschlossene Vermittlungseinheiten**
 - ▶ U.A. mit Quell-/Zieladresse
 - ▶ Pakete mit gleicher Quell- und Zieladresse werden unabhängig voneinander weitergeleitet
- **Vermittlungsstellen besitzen keine Informationen über Verbindungskontext!**
 - + Verbindungsauftakt, -überwachung und -abbau entfallen
 - + Bessere Nutzung der Netzkapazität möglich
 - Einzelne Datagramme können unterschiedliche Wege nehmen
 - Die reihenfolgerichtige Auslieferung beim Empfänger ist nicht gesichert

Die verbindungslose Paketvermittlung basiert auf der isolierten Betrachtung jedes einzelnen Datenpakets. Jedes Paket enthält in den entsprechenden Kontrollfeldern im Paketheader ausreichende Adressierungsinformationen, die es den Vermittlungsstellen ermöglichen, Pakete dem gewünschten Empfänger zuzustellen. Verbindungsauftakt- und -abbau können somit komplett entfallen. Dies bringt aber auch Schwächen des Mechanismus mit sich: da zwei Pakete des selben Paketstroms auf unterschiedlichen Pfaden zum Ziel geleitet werden können, kommen sie unter Umständen in falscher Reihenfolge beim Empfänger an. Zudem können keine Garantien wie bei der verbindungsorientierten Vermittlung gegeben werden, wenn nicht jedes Paket im Header alle notwendigen Parameter enthält – und selbst dann ist nicht gewährleitet, dass Garantien erfüllt werden können, da jeder Vermittlungsknoten für sich selbst entscheidet, wie ein Paket weiterzuleiten ist und nicht weiß, ob der Vermittlungsknoten, der das Paket erhält, auch in der Lage sein wird, die Garantien zu erfüllen.

Verbindungslose Paketvermittlung – Ablauf

- Verkehr in einem paketvermittelten Netz

- ▶ Jedes Paket mit eigenem Weg
- ▶ Eine Reservierung von Verarbeitungskapazität ist nicht möglich



Gegenüberstellung

Leitungsvermittlung	Paketvermittlung
<p>Beispiel: Telefonnetz</p> <ul style="list-style-type: none">- Starre Leitungszuordnung (Ressourcenbindung)- Schlechtes Verhalten im Fehlerfall+ Exakte Leistungsvorhersage (Paketverlustrate, Datenrate, Latenz, ...)+ Reihenfolgetreue+ kein Header-Overhead- Verzögerung durch Leitungsreservierung- Überlast kann Verbindung verhindern+ geringe Implementierungs-komplexität	<p>Beispiel: Internet</p> <ul style="list-style-type: none">+ Flexible Zuordnung von Ressourcen (Effizienz)+ Flexible Fehlerbehandlung- Keine Leistungsvorhersage möglich: Paketverlust, schwankende Datenrate, ...- keine Reihenfolgetreue- Header-Overhead- Verzögerung durch Paketweiterleitung- Überlast erhöht Verzögerung+ geringe Implementierungs-komplexität

Grundsätzlich gilt, dass der Vorteil des einen Verfahrens (Leitungs-/Paketvermittlung) der Nachteil des anderen ist und umgekehrt.

Die Leitungsvermittlung kann grob als starr und einfach, die Speichervermittlung als flexibel und komplex zusammengefasst werden. Der wesentliche Unterschied, welcher alle weiteren positiven und negativen Eigenschaften nach sich zieht, liegt in der Art, wie die Vermittlungsverfahren die Kommunikationsressourcen, sprich den nachrichtentechnischen Kanal, belegen:

Die Leitungsvermittlung nimmt eine feste Reservierung der Kommunikationsressourcen während der gesamten Verbindsdauer vor, während die Speichervermittlung die Ressourcen immer nur für ein Paket bzw. eine Nachricht beansprucht.

Der Einsatz virtueller Leitungen kombiniert zwar die Vorteile beider Verfahren, aber auch ihre Nachteile. Zudem bringt dieses Verfahren eine höhere Komplexität mit sich und kann schnell ineffizient werden – siehe dazu wieder ATM: die geteilte Nutzung der Leitungen wird durch Zeitmultiplex ermöglicht, welches eine feste Zellgröße erfordert. Wählt man die Zellen zu klein, wird der Header-Overhead enorm groß und die Nutzdatenrate für Anwendungen geht in den Keller. Wählt man die Zellen zu groß, dauert die Weiterleitung einer einzelnen Zelle so lange, dass alle Zellen in den Switches stark verzögert werden können; zudem können die Ressourcen nicht mehr effizient genutzt werden, wenn Anwendungen die Zellen nicht komplett füllen können: niemand sonst kann die dadurch frei bleibende Netzkapazität nutzen. Dies war gerade das Problem mit ATM: die Zellgröße war ein Kompromiss zwischen der datenorientierten Internetfraktion, die große Zellen für Datenpakete haben wollte, und der Sprachfraktion, die kleine Zellen für die Echtzeitübertragung von Sprachinformationen forderten. Die gewählte Zellgröße stellte beide Fraktionen nicht zufrieden, und speziell für die datenorientierte Übertragung war ATM höchst ineffizient, da ein großer Teil der Netzkapazität für Headerinformationen verloren ging.

Verbindungsorientierte Paketvermittlung

- **Virtuelle Leitung (Virtual Circuit, VC)**
 - ▶ Bidirektonaler fester Übertragungsweg (vollduplex)
 - Wird bei Verbindungsaufbau festgelegt
 - ▶ Kann mehrere (Paket-)Vermittlungsstellen umfassen
- **Kombiniere Vorteile von Circuit- und Packet Switching**
 - ▶ Aufbau einer Verbindung, ähnlich Leitungsvermittlung
 - Möglichkeit von Garantien, z.B. reihenfolgetreue Übertragung
 - ▶ Teilung von Leitungen wie bei Paketvermittlung
 - Ressourceneffizienz, Flexibilität



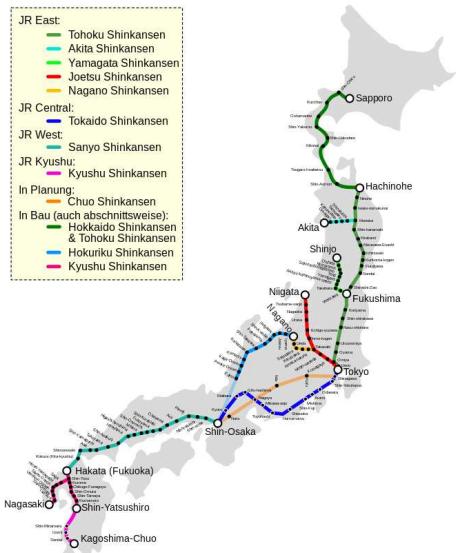
Bei der *verbindungsorientierten Paketvermittlung* findet vor dem eigentlichen Datenaustausch ein Verbindungsaufbau statt, der neben verschiedenen Verbindungsparametern (z.B. für die Dienstqualität) den Weg für alle zu dieser Verbindung gehörenden Pakete festgelegt. Für den Benutzer erscheint die Verbindung wie eine echte, durchgehende Leitung, obwohl es eine solche in der Realität nicht gibt - deshalb auch die Bezeichnung *virtuelle Verbindung*.

Wird ein Übertragungsweg vom Netzbetreiber längerfristig eingerichtet („Standleitung“), so spricht man auch von *fester virtueller Verbindung* (Permanent Virtual Circuit, PVC). Ist eine gesonderte Verbindungsaufbauprozedur erforderlich (Signalisierung), wird die Verbindung als *gewählte virtuelle Verbindung* bezeichnet (Switched Virtual Circuit, SVC).

Circuit Switching und Virtual Circuits

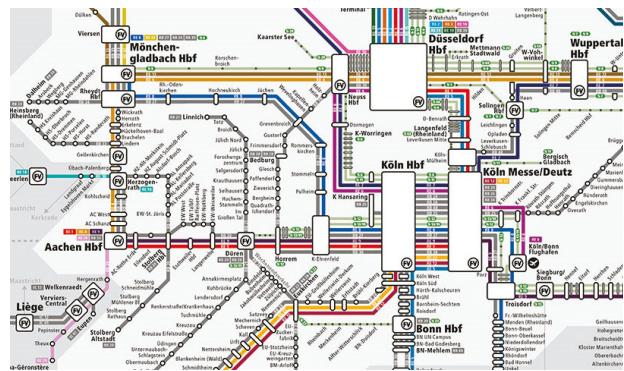
• Leitungsvermittlung

- Geplantes Schienennetz
- Pro Strecke eine Linie



• Virtuelle Leitung

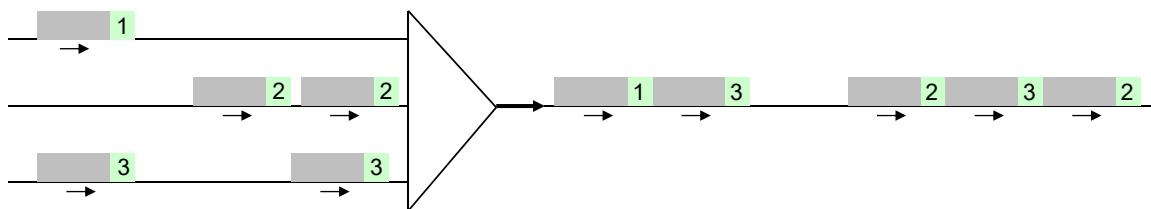
- Geplantes Schienennetz
- Eine Linie fährt immer die gleiche Strecke
- Aber Linien können Teilstrecken gemeinsam nutzen



Virtual Circuit: Asynchronous Transfer Mode (ATM)

• Zellenbasiertes Multiplexing und Switching

- ▶ **ATM-Zelle:** Paket fester Länge mit 5 Byte Header + 48 Byte Payload
- ▶ Schaltet *virtuelle Verbindungen*
- ▶ *Multiplexing* verschiedener Verbindungen auf eine Leitung
 - Konstante Zellgröße ermöglicht Weiterleitung in Zeitmultiplex-Raster
 - Reservierung von Kapazität möglich
 - *Admission Control* vor Annahme einer Verbindung, um Überlastung des Netzes zu verhindern
 - Bei Überlast werden Zellen verworfen

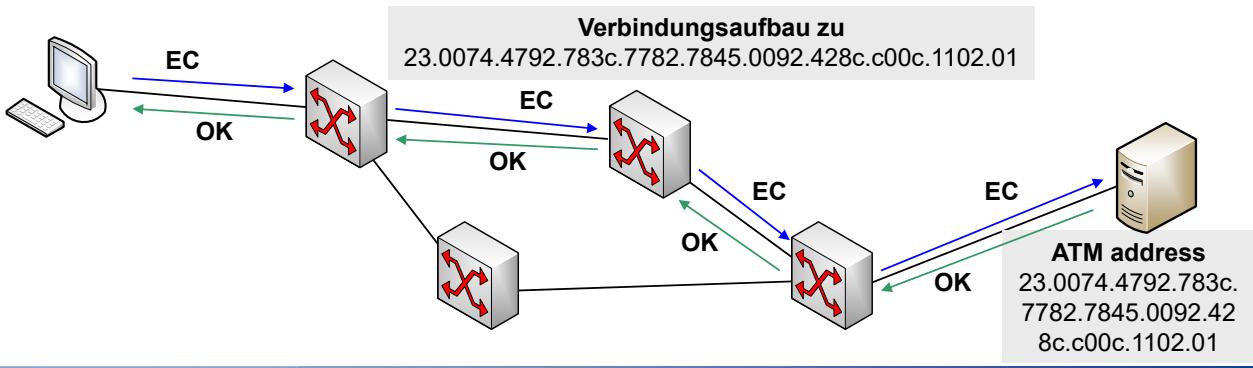


Ein Versuch, Leitungs- und Paketvermittlung zu kombinieren, wurde mit ATM vorgenommen. Typischerweise werden bei der Paketvermittlung die Ressourcen des Netzes geteilt zwischen mehreren Stationen genutzt und es ist schwer, konkrete Garantien zu geben, da Pakete unterschiedliche Längen haben können, wodurch indeterministische Verzögerungen bei der Weiterleitung von Paketen auftreten können. ATM verwendet Pakete fester Länge (Zellen), um ein Zeitmultiplexing mit fester Slotlänge einzusetzen zu können. Dadurch ist bei Bedarf die Reservierung einer garantierten Bandbreite möglich. Mechanismen wie Admission Control (Zugangskontrolle) sorgen dafür, dass nicht zu viele Teilnehmer gleichzeitig Garantien erfragen und das Netz überlasten können.

Verbindungsauflaufbau bei ATM

• Zellweiterleitung durch Switches

- ▶ Anhand von *Verbindungs-IDs*, nicht Rechneradressen
- ▶ Vor Beginn der Kommunikation: Verbindungsauflaufbau
 - Kontrollpaket ermittelt Pfad durchs Netz, Empfänger antwortet mit Quittung
 - Passierte Switches legen eine Verbindungs-ID an
 - Ermöglicht weniger (kleinere) Einträge in den Weiterleitungstabellen

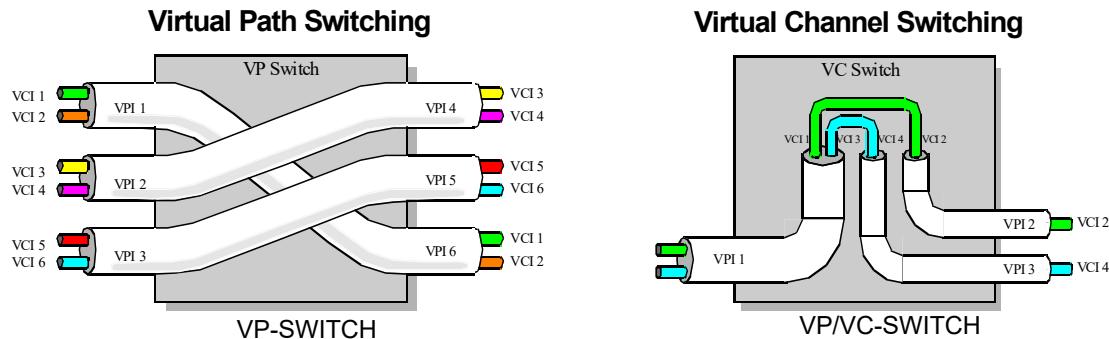


Vor Beginn einer Kommunikation erfolgt ein Verbindungsauflaufbau, bei dem auch Reservierungen von Bandbreite in den einzelnen Switches vorgenommen werden können. Bei Verbindungsauflaufbau bekommt eine Kommunikationssitzung eine ID zugewiesen; eine Weiterleitung der Zellen erfolgt dann nicht mehr aufgrund einer Wegewahl anhand der Zieladresse: die Vermittlungsknoten agieren als Switches, die in einem festen Eintrag die ID einer physikalischen Leitung zuordnen und die folgenden Zellen der Verbindung immer über diese Leitung weiterleiten.

Verbindungs-IDs: Pfad und Kanal

- **Hierarchischer Aufbau der Verbindungs-IDs**

- ▶ Identifier für virtuellen Pfad (*Virtual Path Identifier*, VPI)
- ▶ Identifier für virtuellen Kanal (*Virtual Channel Identifier*, VCI)
- ▶ Hierarchische Struktur ermöglicht Kombination mehrerer Routen-Informationen in der Weiterleitungstabelle durch Vergabe des gleichen VPI



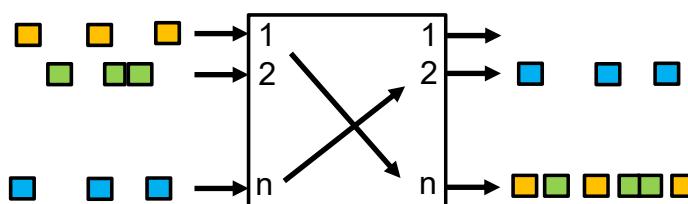
Ein Problem bei der Verwendung des Switchings ist die Explosion der Weiterleitungstabellen; für jede aktuell existierende Verbindung muss ein Switch einen eigenen Eintrag verwalten, was zu extrem großen Weiterleitungstabellen führt, die unperformant werden. Daher wird eine hierarchische ID-Struktur erstellt – soweit möglich, wird ein neuer Eintrag mit bereits existierenden Einträgen aggregiert, um die Anzahl der Weiterleitungseinträge zu minimieren.

Switching

• Weiterleitungstabellen in den Switches

- ▶ Old Header enthält VPI/VCI auf dem vorherigen Teilstück
- ▶ New Header enthält VPI/VCI zur Verwendung auf dem nächsten Teilstück
- ▶ In und out bezeichnen die Netzwerkkarten, über die ein Paket eingeht / ausgesendet werden muss

Eingangsleitungen Switch Ausgangsleitungen



Switching-Tabelle

In	Old Header	Out	New Header
1	a	n	a
2	c	n	d
...
n	b	2	e

Da die ID durch jeden Switch geändert wird, ist eine Modifikation des Headers notwendig. Zudem enthält der Header noch eine CRC-C checksumme, die der Korrektur von 1-Bit-Fehlern im Header dient. Diese CRC muss also vor der Weiterleitung auch neu berechnet werden. Dies führt zu einer Verzögerung der Zellen vor der Weiterleitung. Um den Prozess zu beschleunigen, werden die CRC-Prüfsummen bereits bei Verbindungsaufbau vorberechnet und mit in der Weiterleitungstabelle abgespeichert. Bei Empfang einer Zelle wird nun lediglich geprüft, auf welcher Leitung sie weitergesendet werden muss, der alte Zellheader verworfen und der vorberechnete Header aus der Weiterleitungstabelle eingefügt.

Gegenüberstellung

Leitungsvermittlung	Virtuelle Leitung	Paketvermittlung
<p>Beispiel: Telefonnetz</p> <ul style="list-style-type: none"> - Starre Leitungszuordnung (Ressourcenbindung) - Schlechtes Verhalten im Fehlerfall + Exakte Leistungsvorhersage (Paketverlustrate, Datenrate, Latenz, ...) + Reihenfolgetreue + kein Header-Overhead - Verzögerung durch Leitungsreservierung - Überlast kann Verbindung verhindern + geringe Implementierungs-komplexität 	<p>Beispiel: ATM</p> <ul style="list-style-type: none"> + Flexible Zuordnung von Ressourcen (Effizienz) ~ Beschränkte Fehlerbehandlung + Exakte Leistungsvorhersage (Paketverlustrate, Datenrate, Latenz, ...) + Reihenfolgetreue - Header-Overhead - Verzögerung durch Verbindungsauftakt und Paketweiterleitung - Überlast kann Verbindung verhindern und erhöht Verzögerung - hohe Implementierungs-komplexität 	<p>Beispiel: Internet</p> <ul style="list-style-type: none"> + Flexible Zuordnung von Ressourcen (Effizienz) + Flexible Fehlerbehandlung - Keine Leistungsvorhersage möglich: Paketverlust, schwankende Datenrate, ... - keine Reihenfolgetreue - Header-Overhead - Verzögerung durch Paketweiterleitung - Überlast erhöht Verzögerung + geringe Implementierungs-komplexität

Grundsätzlich gilt, dass der Vorteil des einen Verfahrens (Leitungs-/Paketvermittlung) der Nachteil des anderen ist und umgekehrt.

Die Leitungsvermittlung kann grob als starr und einfach, die Speichervermittlung als flexibel und komplex zusammengefasst werden. Der wesentliche Unterschied, welcher alle weiteren positiven und negativen Eigenschaften nach sich zieht, liegt in der Art, wie die Vermittlungsverfahren die Kommunikationsressourcen, sprich den nachrichtentechnischen Kanal, belegen:

Die Leitungsvermittlung nimmt eine feste Reservierung der Kommunikationsressourcen während der gesamten Verbindsdauer vor, während die Speichervermittlung die Ressourcen immer nur für ein Paket bzw. eine Nachricht beansprucht.

Der Einsatz virtueller Leitungen kombiniert zwar die Vorteile beider Verfahren, aber auch ihre Nachteile. Zudem bringt dieses Verfahren eine höhere Komplexität mit sich und kann schnell ineffizient werden – siehe dazu wieder ATM: die geteilte Nutzung der Leitungen wird durch Zeitmultiplex ermöglicht, welches eine feste Zellgröße erfordert. Wählt man die Zellen zu klein, wird der Header-Overhead enorm groß und die Nutzdatenrate für Anwendungen geht in den Keller. Wählt man die Zellen zu groß, dauert die Weiterleitung einer einzelnen Zelle so lange, dass alle Zellen in den Switches stark verzögert werden können; zudem können die Ressourcen nicht mehr effizient genutzt werden, wenn Anwendungen die Zellen nicht komplett füllen können: niemand sonst kann die dadurch frei bleibende Netzkapazität nutzen. Dies war gerade das Problem mit ATM: die Zellgröße war ein Kompromiss zwischen der datenorientierten Internetfraktion, die große Zellen für Datenpakete haben wollte, und der Sprachfraktion, die kleine Zellen für die Echtzeitübertragung von Sprachinformationen forderten. Die gewählte Zellgröße stellte beide Fraktionen nicht zufrieden, und speziell für die datenorientierte Übertragung war ATM höchst ineffizient, da ein großer Teil der Netzkapazität für Headerinformationen verloren ging.

Kapitel 4: Vermittlungsschicht

- **Vermittlungsverfahren**

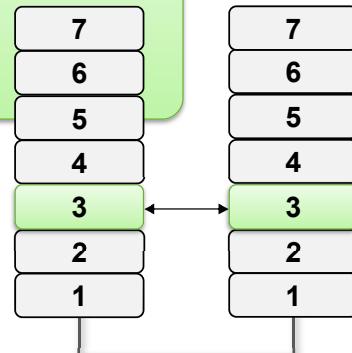
- ▶ Leitungsvermittlung, Speicher-/Paketvermittlung
- ▶ Beispiel ATM

- **Die Vermittlungsschicht im Internet – IP**

- ▶ IPv4: Adressen, Subnetze, CIDR, NAT
- ▶ IPv4-Header
- ▶ Hilfsprotokolle: ARP, DHCP, ICMP
- ▶ IPv4 vs. IPv6

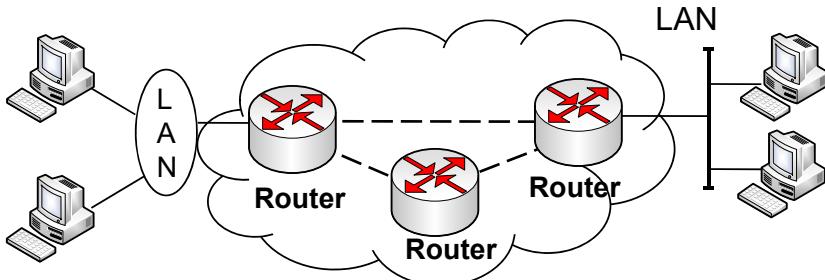
- **Wegewahlverfahren (Routing)**

- ▶ Hierarchien, einfache Verfahren
- ▶ Distance Vector
- ▶ Link State



Vermittlungsschicht im Internet: IP (Internet Protocol)

- **Historie:**
 - ▶ Entwickelt vom amerikanischen Verteidigungsministerium (Departement of Defense, DoD)
 - ▶ Bereits 1983 im damaligen ARPANET eingesetzt (anfangs vier Hosts)
- **Realisierung und Entwicklung:**
 - ▶ Verbreitet ist vorwiegend Version 4 ([IPv4](#))
 - ▶ Weiterentwicklung im Projekt IPng (IP next generation) der IETF (Internet Engineering Task Force) zu [IPv6](#)



Die Entwicklung der Protokolle der TCP/IP-Familie geht auf eine Initiative des amerikanischen Verteidigungsministeriums (Department of Defense, DoD) in den 60er Jahren zurück, welche das primäre Ziel besaß, ein möglichst ausfallsicheres Netzwerk zu schaffen. Hierbei sollte die Kommunikation auch über große Entfernnungen hinweg in Krisenzeiten (z.B. Atomkrieg) sichergestellt werden. Dahingehende Untersuchungen wurden von der ARPA (Advanced Research Project Agency) durchgeführt, die sich schließlich für ein paketvermitteltes Netzwerk entschied und das hierzu entwickelte Protokoll IP erstmals 1983 einsetzte (die beteiligten Rechner bildeten das sogenannte ARPANET). Nach der Abspaltung des militärischen Teils in das sogenannte MILNET entstand aus dem ARPANET schließlich das heute als Internet bekannte Netzwerk.

Von da an entwickelte sich das Internet weiter wie kein anderes Netzkonzept mit dem Ergebnis, dass es heute ein weltweites (Daten-)Kommunikationsnetz ist, mit über einer Milliarde angeschlossener Rechner.

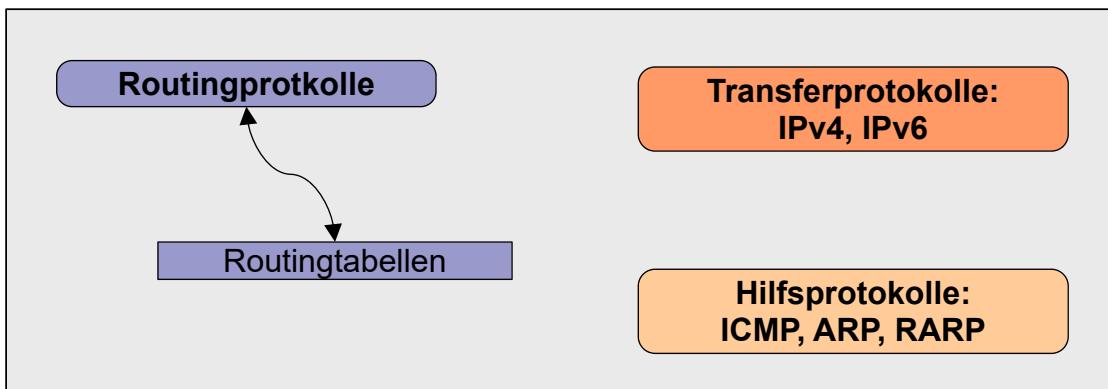
Die ursprüngliche entwickelte Version von IP war IPv4 (Versionen 1 – 3 gab es nie). Diese Version ist jetzt bereits seit ungefähr 35 Jahren im Einsatz. Die Entwicklung von IP ging allerdings weiter, die letzten größeren Arbeiten an IP erfolgten im Rahmen der Arbeitsgruppe „IP Next Generation“, aus der IPv6 hervorging (eine Version 5 gibt es ebenfalls nicht).

Die zentrale Aufgabe des IP-Protokolls bzw. der dieses Protokoll realisierenden Protokollinstanzen ist die Zustellung von Daten an einen entfernten Rechner – dazu definiert es ein Adressschema, Regeln zur verbindungslosen Paketverarbeitung und -weiterleitung sowie entsprechende Headerinformationen, die eine vernünftige Weiterleitung ermöglichen.

Vermittlungsschicht im Internet

- **Grobe Aufteilung in drei Aufgabenbereiche:**

- ▶ **Datentransfer** über ein globales Netz: Adressierung, Paketformat, Regeln zur Paketverarbeitung (Forwarding)
- ▶ **Wegwahl** durch die Zwischenknoten (Routing)
- ▶ **Hilfsprotokolle** z.B. zum Austausch von Informationen über den Netzstatus (Fehlermeldungen, Signalisierung)



IP wirkt zusammen mit einigen Hilfs- und Kontrollprotokollen, die Teilaufgaben wahrnehmen, sowie mit Routing-Protokollen, die festlegen, wie die Router (= Vermittlungsstellen im Internet) untereinander Informationen austauschen können, um bei der Weiterleitung eines Pakets eine Wegewahl zu treffen. Die Weiterleitung der Pakete selbst wird als Forwarding bezeichnet.

Eigenschaften des Internet Protocol (IPv4)

• Paketvermittelt, verbindungslos

- ▶ Ungesicherte Übertragung:
 - Ein Paket kann verloren gehen
 - Ein Paket kann dupliziert werden
 - Pakete können in falscher Reihenfolge ankommen
 - Pakete können endlos im Netz zirkulieren
- ▶ Nicht behebbare Fehler der zugrundeliegenden Schicht 2 können von IP im Allgemeinen ebenfalls nicht behandelt werden
 - Mit dem Protokoll *ICMP (Internet Control Message Protocol)* existiert jedoch eine Möglichkeit zur Fehleranzeige
- ▶ Keine Flusskontrolle, keine Staukontrolle

Die zentrale Eigenschaft von IP besteht darin, dass es sich um ein unzuverlässiges, verbindungsloses Protokoll handelt. Dementsprechend werden also Probleme wie Paketverlust, Duplizierung von Paketen oder Reihenfolgeumstellung im Paketstrom von IP nicht behoben. Diese Aufgabe wird den höheren Schichten überlassen.

Selbst Fehler, die in der Schicht 2 nicht behoben werden konnten und deshalb nach oben in die IP-Schicht durchschlagen, werden meist unbehandelt nach oben zur Transportschicht weitergegeben.

Aufgrund der Verbindungslosigkeit bietet IP auch keine Flusskontrolle.

Die einzige Form der Reaktion auf erkannte Fehlersituationen kann darin bestehen, dass eine IP-Instanz dem Sender des IP-Pakets über das Protokoll ICMP einen aufgetretenen Fehler mitteilt. Diese Information kann für den Sender wichtig sein, damit er weiß, warum seine Nachricht nicht beim Empfänger angekommen ist.

IP kann sowohl zur Kopplung von Teilnetzen z.B. in einem lokalen unternehmensweiten Netz genutzt werden, als auch zum Aufbau eines öffentlichen Netzes, wie es das Beispiel des Internet deutlich macht – der Trend hat sich sogar dahin entwickelt, alles über IP laufen zu lassen, also z.B. auch Telefonnetze auf IP umzustellen.

IP: Aufgaben

- Transparente Ende-zu-Ende-Kommunikation zwischen Rechnern auch über unterschiedliche Netztypen hinweg
 - ▶ Bereitstellung *weltweit eindeutiger Adressen*
 - IPv4: 32 Bit
 - Hierarchische Struktur
 - ▶ Definition von *Verarbeitungs-/Weiterleitungsregeln* samt eines zugehörigen *Paketformats*
 - Header mit Kontrollinformationen
 - Maximale Paketgröße: 64 kByte (in der Praxis: 1500 Byte)
 - ▶ Zusammenspiel mit *Routing* (Wegermittlung)
 - ▶ Zusammenspiel mit den tieferen Schichten (*ARP*)

- Eindeutige IP-Adressen für jeden Host und jeden Router

- ▶ IP-Adressen (IPv4) sind 32 Bit lang
 - *Dotted Decimal Notation*: Darstellung einer Adresse als 4 Dezimalwerte, die jeweils 8 Bit der Adresse entsprechen
- ▶ *Hierarchisch strukturiert und netzbezogen*, d.h. Maschinen mit Anschluss an mehrere Netze haben mehrere IP-Adressen
- ▶ Struktur der Adresse: *Netzwerk*-Anteil für physikalisches Netz (z.B. **137.226**.0.0) und *Host*-Anteil für einen Rechner (z.B. 137.226.**12.221**) in diesem Netz
- ▶ Um unterschiedlich große Netze installieren zu können, wurden ursprünglich mehrere Adressklassen definiert
 - Heute wird zwar noch die Terminologie verwendet, aber nicht mehr die Klassenaufteilung!

Um IP-Pakete weiterleiten zu können, muss ein Router alle möglichen Endsysteme kennen, und wissen, wie er diese erreichen kann. Jedes Netzwerkinterface ist eindeutig über eine MAC-Adresse adressierbar – aber da MAC-Adressen über keine Struktur verfügen, müsste jeder Router einen Eintrag pro Netzwerkkarte weltweit haben, wodurch Weiterleitungstabellen unhandhabbar groß werden. Daher definiert IP eigene Adressen. Gewählt wurde ein 32 Bit großer Adressraum, um ausreichend viele Adressen für das weltweite Netz zur Verfügung zu haben. Da es somit aber immer noch 2^{32} mögliche Adressen gibt, wären die Weiterleitungstabellen immer noch zu groß, um sie verwalten zu können.

Deshalb benutzt man eine Adressstruktur, die es nicht erforderlich macht, dass jeder Router jeden Host kennt. Man unterteilt die Adresse in einen Netzteil und einen Hostteil. Somit muss ein Router nur noch wissen, wie er ein bestimmtes Netz erreichen kann, es ist also nur noch ein Eintrag pro Netz nötig. Für sein eigenes Netz, das über den Router erreicht werden kann (LAN, MAN oder WAN), muss er zusätzlich wissen, wie er die einzelnen Hosts erreichen kann.

Allerdings wird keine starre Aufteilung der IP-Adresse in Netz- und Hostteil vorgenommen. Da Router nur noch einen einzigen Eintrag für ein ganzes Netz abspeichern, können Adressen mit gleichem Netzwerk-Anteil nur an einem einzigen Standort verwendet werden. Ungenutzte Adressen mit diesem Netzwerk-Anteil können nirgendwo sonst verwendet werden. Da reale Netze sehr unterschiedliche Größe haben können, würde eine einheitliche Festlegung der Netzgröße dazu führen, dass in einigen Netzen viele Adressen brachliegen und dass andererseits sehr große Netze eventuell nicht mit genügend Adressen versorgt werden können. Daher entschloss man sich dazu, unterschiedliche Netzgrößen, d.h. unterschiedlich lange Netzwerk-Anteile zuzulassen. Entsprechend der Netzgröße wurden mehrere Adressklassen definiert. (Bitte beachten: der klassenbasierte Adressierungsansatz hat sich als nicht praktikabel erwiesen und wird nicht mehr verwendet – die Klassenterminologie wird allerdings immer noch im Sprachgebrauch benutzt.)

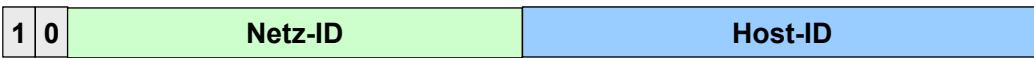
IP-Adressen / Adressklassen

• Ursprüngliches Adressschema: Adressklassen

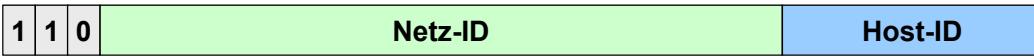
- ▶ Class A für Netze mit bis ca. 16 Mio. Hosts



- ▶ Class B für Netze mit bis zu 65.534 Hosts



- ▶ Class C für Netze mit bis zu 254 Host



- ▶ Class D für Gruppenkommunikation (Multicast)



- ▶ Class E, reserviert für zukünftige Anwendungen



Da nicht alle Netze gleich groß sind, d.h. die gleiche Anzahl Rechner umfassen, hat man zu Beginn ein Klassenschema eingeführt: die ersten Bits der Adresse definieren, welcher Klasse ein Netz angehört. Es wurde definiert, dass es drei unterschiedliche Netzgrößen geben kann (Klasse A – C):

- Klasse A: Die ersten 8 Bit der IP-Adresse identifizieren ein Netz; ein Netz dieser Klasse kann bis zu 2^{24} Hosts besitzen. Adressen dieser Klasse beginnen immer mit einer 0, daher gibt es 2^7 solche Netze.
- Klasse B: Die ersten 16 Bit der IP-Adresse identifizieren ein Netz; ein Netz dieser Klasse kann bis zu 2^{16} Hosts besitzen. Adressen dieser Klasse beginnen immer mit 10, daher gibt es 2^{14} solche Netze.
- Klasse C: Die ersten 24 Bit der IP-Adresse identifizieren ein Netz; ein Netz dieser Klasse kann bis zu 2^8 Hosts besitzen. Adressen dieser Klasse beginnen immer mit 110, daher gibt es 2^{21} solche Netze.

Soll ein IP-Paket weitergeleitet werden, liest ein Router zunächst nur die ersten Bits der Ziel-IP-Adresse aus und weiß daraufhin, zu welcher Klasse das Netz gehört, in dem der Zielrechner liegt und für welches Präfix der Adresse (8, 16 oder 24 Bit) er einen passenden Eintrag in seiner Weiterleitungstabelle suchen muss.

Des weiteren gibt es eine Klasse für Multicast-Adressen, d.h. Adressen, über die man Daten mit einem Paket an mehrere Empfänger zustellen kann. Und wie es bei Protokollentwicklern üblich ist, wurde ein Teil des Adressraums freigehalten (Klasse E), um eventuell später für neue Arten von Anwendungen weitere Adresstypen definieren zu können.

- Klasse D: Diese Klasse umfasst alle Adressen, die mit 1110 beginnen und ist für Multicast-Gruppen reserviert. Es sind 2^{28} Gruppen möglich.
- Klasse E: Alle Adressen, die mit 1111 beginnen; reserviert für zukünftige Verwendung.

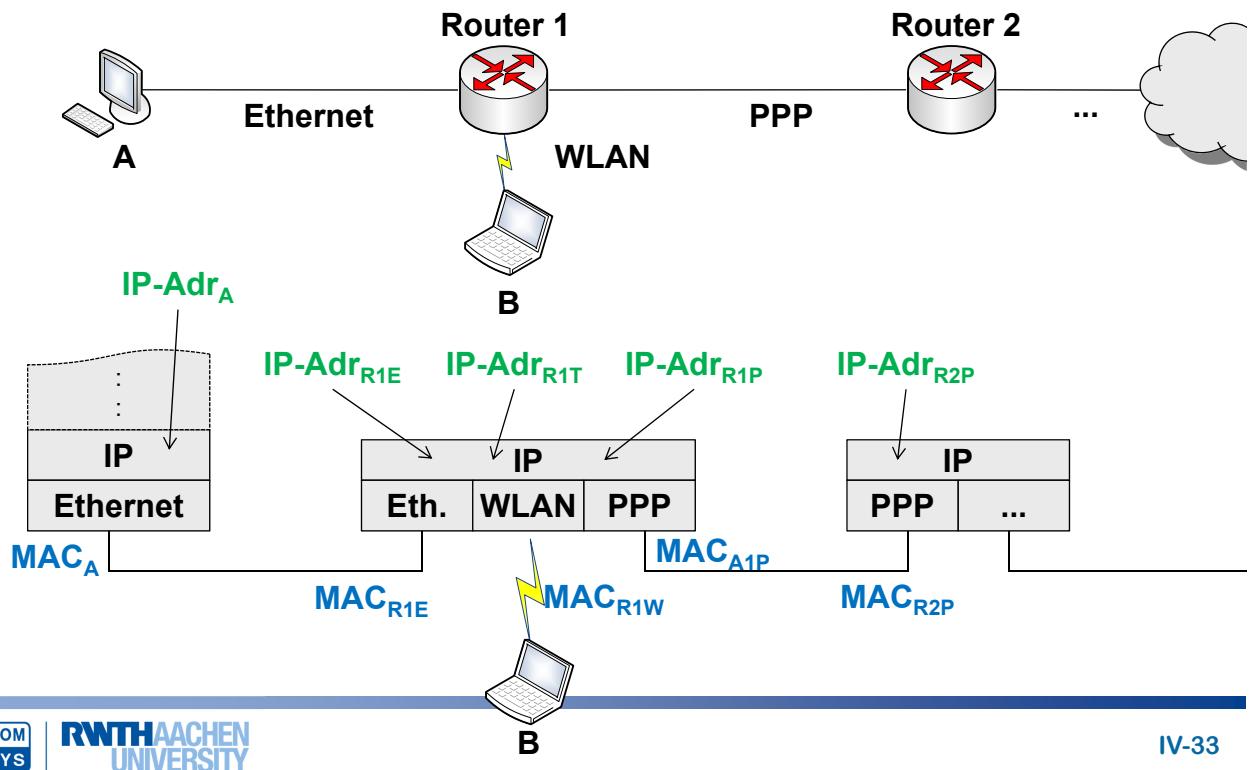
In der Praxis sind meist noch einige Adressen pro Klasse für spezielle Zwecke reserviert, wie z.B. Host-ID = 1...1 für Broadcast-Pakete (werden an alle Rechner innerhalb eines Netzes zugestellt) oder 127.0.0.1 als Loopback-Adresse, bei der keine Ausgabe der Daten aufs Netz erfolgt, sondern die Daten auf dem eigenen Rechner wieder durch den Protokollstack nach oben durchgereicht werden. Damit hat man eine Standardadresse zur Verfügung, über die eine Anwendung auf einem Rechner ohne Kenntnis der eigenen IP-Adresse über IP mit einer anderen lokalen Anwendung kommunizieren kann. Die Adresse wird daher auch als „localhost“ bezeichnet.

Außerdem gibt es einige Konventionen – so ist z.B. die höchste Adresse in einem Netz für Broadcast in diesem Netz reserviert; an diese Adresse versendete Pakete werden von jedem Rechner des lokalen Netzes entgegengenommen und verarbeitet. Router leiten Broadcast-Pakete nicht weiter, so dass ein Broadcast auf das eigene lokale Netz beschränkt ist. Die niedrigste Adresse ist für das Netz reserviert und wird üblicherweise an keinen Rechner vergeben.

Eine 4-Byte-IP-Adresse kennzeichnet einen Rechner weltweit eindeutig. Damit es keine Konflikte gibt, werden die Netz-IDs von NICs (Network Information Centers) vergeben.

IP-Adressen

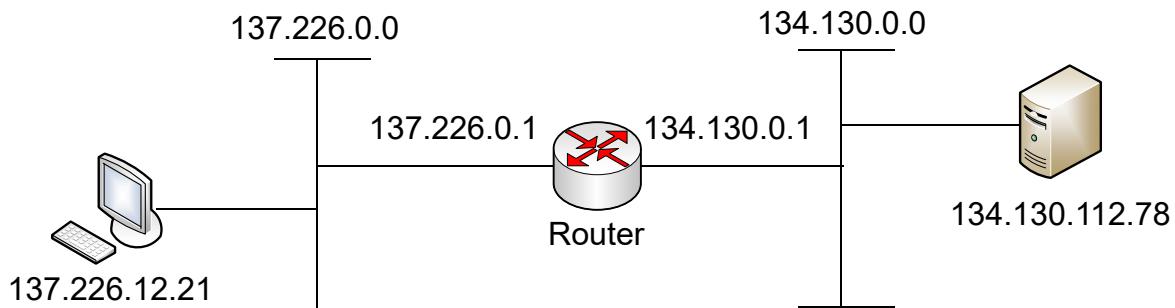
- Jedes Interface benötigt eine eigene IP-Adresse:



IP-Adressen

- Einrichten eines IP-Netzes

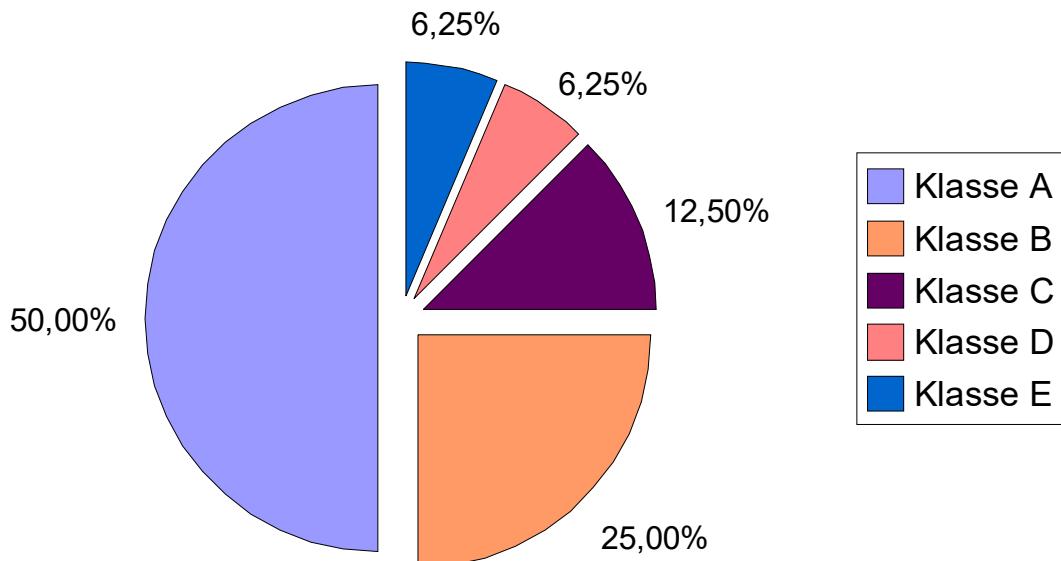
- ▶ Kaufen/mieten eines Adressblocks (z.B. Klasse B: 137.226.x.x)
- ▶ Jeder Host hat (wenigstens) eine weltweit eindeutige IP-Adresse
- ▶ Router oder Gateways, die mehrere Netze miteinander verknüpfen, haben für jedes angeschlossene Netz eine IP-Adresse



Binärformat	10001001 11100010 00001100 00010101
Dotted Decimal Notation	137.226.12.21

Aufteilung des Adressraums:

- Aufteilung der IP-Adressen auf die Klassen

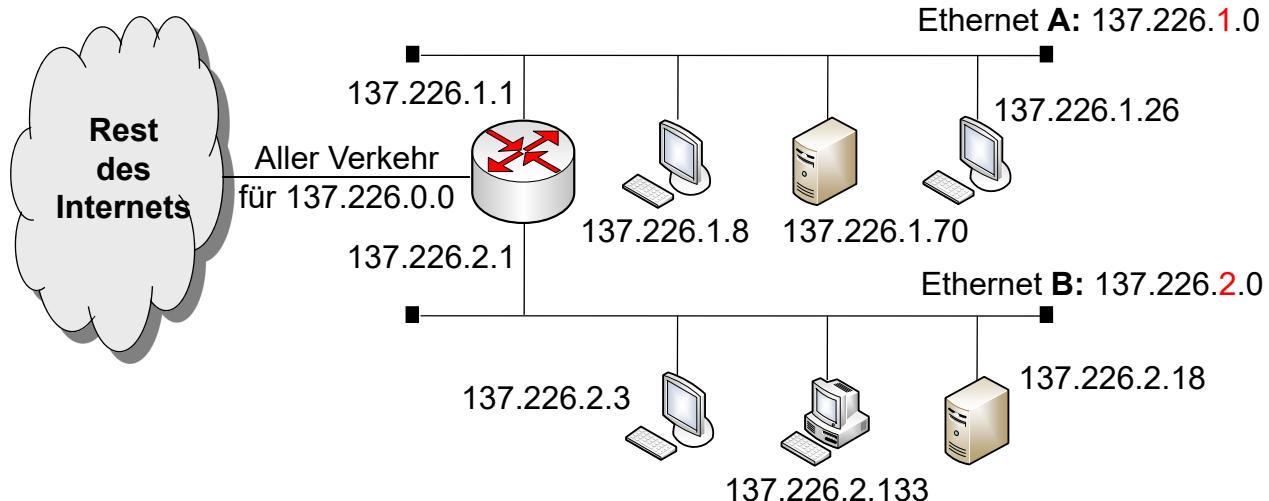


In den ersten Jahren des Internets wurden recht großzügig Klasse-A-Netze vergeben, da niemand mit einem solchen Anwachsen des Internets gerechnet hatte. Auch Klasse-B-Netze wurden bereits freizügig verteilt – was macht man beispielsweise, wenn man ein Netz mit 500 Rechnern aufbauen will? Ein Klasse-C-Netz wäre zu klein, also besorgt man sich einen Klasse-B-Adressblock. Daher war recht bald abzusehen, dass man in eine Adressknappheit hineinlief.

Als Lösung, den Adressraum effizienter aufteilen zu können, wurde daher das Konzept der *Subnetze* eingeführt.

- Erweiterung der Adresshierarchie: **Subnetze**

- ▶ Zerlege ein durch ein Präfix identifiziertes Netz in kleinere Netze
- ▶ Zerlegung identifiziert durch **Subnetz-Maske**
- ▶ Beispiel: Subnetz-Maske 255.255.255.0



Durch eine Subnetzmaske lässt sich ein großes Netz in mehrere kleinere Netze zerlegen. Dazu wurde das Konzept der Subnetzmasken eingeführt, über die zusammengehörige Adressen innerhalb eines größeren Adressraums identifiziert werden können.

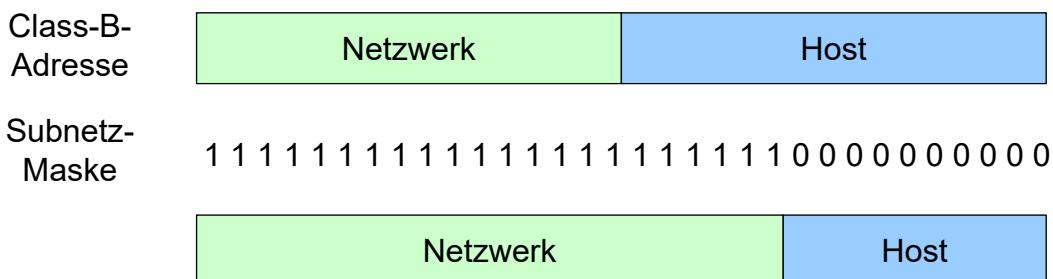
Bitte beachten: die 137.226.1.0 und 137.226.2.0 werden „Netzadressen“ genannt – auch wenn das Netz selbst gar keine Adresse hat. (Nur Hosts/Router haben Adressen.)

Das, was man als „Netzadresse“ bezeichnet, ist die Basisadresse des gesamten Netzes, die in Routingtabellen verwendet wird – siehe dazu Folie 33 – 35.

IP-Subnetze

- „Zerlegung“ eines Netzes in Subnetze

- ▶ Einige Bits der Host-Adresse werden als Netzwerk-ID genutzt
- ▶ Eine Subnetz-Maske identifiziert die Bits, die zur Identifizierung des Netzes genutzt werden
- ▶ Router ermitteln durch Kombination einer IP-Adresse und einer Subnetz-Maske, in welches Teilnetz ein Paket geschickt werden muss



- ▶ Schreibweise: **Netzwerk-ID/22** (= 22 Einsen in der Subnetz-Maske)

Subnetze führen eine dritte Hierarchiestufe ein. Ein außenstehender Router weiß nichts von der Einteilung in Subnetze und muss daher nicht mehr Informationen speichern. Ein netzinterner Router muss jedoch sein Netz in Subnetze unterteilen. Dabei hängt jedes Subnetz an einem physikalischen Ausgangsport des Routers.

Subnetze identifiziert der Router anhand der Subnetzmaske. Diese muss bei allen Hosts eines Netzes gleich sein. Die Subnetzmaske besteht aus einer Folge von x Einsen, die angeben, welcher Teil der IP-Adresse als Netzidentifikator verwendet wird. Dieser 1er-Folge folgen (32-x) 0en.

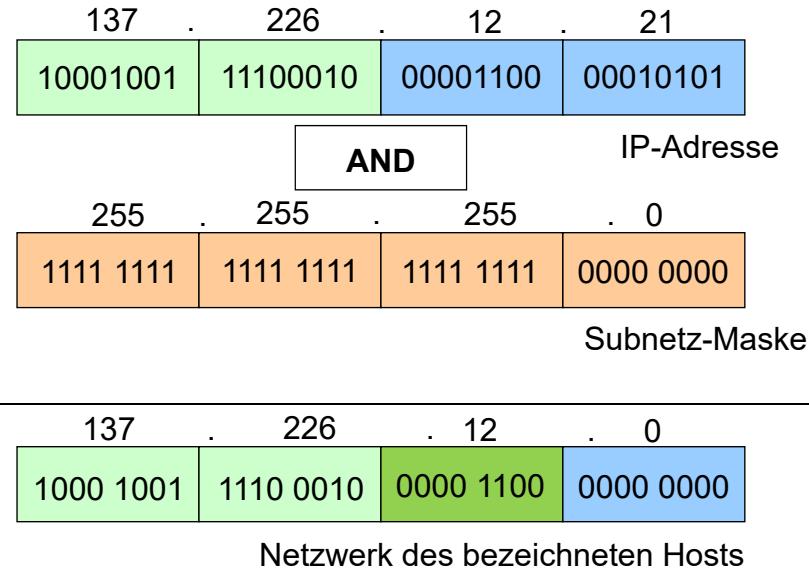
Beispiel:

- Klasse-B-Netz: Basisadresse 137.226.0.0
- Subnetzmaske: 255.255.240.0 (255.255.1111 0000.0000 0000)
- (137.226.aaaa bbbb.bbbb bbbb) a = Subnetz-Adresse b = Hostadresse innerhalb des Subnetzes

Damit kann man 16 (= 2^4) Subnetze bilden, die jeweils aus 4094 (= $2^{12}-2$) Hosts bestehen können. Die niedrigste und höchste Adresse eines Subnetzes sind für Netz-ID und für Broadcast im Subnetz reserviert. Durch die Größe der Subnetzmaske kann man sich also Varianten von vielen kleinen Subnetzen bis hin zu wenigen großen Subnetzen aussuchen. Nicht alle Subnetze müssen gezwungenermaßen die gleiche Größe haben.

IP-Subnetze - Berechnung des Zielhosts

Der Eingangs-Router der RWTH, der das IP-Paket empfängt, berechnet, wo sich Host '137.226.12.21' befindet



Der Router berechnet das Subnetz '137.226.12.0' und sucht den entsprechenden Eintrag in seiner Weiterleitungstabelle

Unterschiedliche Router können auch unterschiedliches Wissen über die Zerteilung eines Adressbereichs haben. Router weltweit brauchen die interne Subnetz-Struktur der RWTH nicht zu kennen – sie benötigen nur eine Weginformation hin zu 137.226.0.0/16, welches das Netz der RWTH identifiziert. Innerhalb der RWTH ist allerdings dieser Adressbereich in Subnetze mit der Maske /24 unterteilt; daher wird der Eingangsrouter der RWTH zur Ermittlung des Zielnetzes bei Ankunft eines Pakets für z.B. den Rechner 137.226.12.21 eine Verknüpfung dieser Zieladresse mit /24 vornehmen und sucht in seiner Weiterleitungstabelle nach dem Eintrag 137.226.12.0/24. Die Verknüpfung ist eine einfache AND-Operation.

Wegwahl bei IP

- Jedes System besitzt eine Routingtabelle
- Anhand der Zieladresse wird ein Eintrag bestimmt, der die Weiterleitung festlegt:
 1. Durchsuche Host-Adressen
 2. Durchsuche Netzwerkadressen
 3. Suche nach Default-Eintrag
- Zwei Möglichkeiten:
 1. Rechner direkt erreichbar (direct route)
 2. Rechner indirekt erreichbar (indirect route)



Die Wegwahl, die von IP durchgeführt wird, basiert auf dem Vorhandensein einer Weiterleitungstabelle (auch: Routingtabelle) in jedem Netzknoten (Router und Hosts), der IP implementiert. Ausgehende IP-Pakete indizieren dabei anhand der Zieladresse *einen* Eintrag dieser Tabelle und können so einem Netzadapter (z.B. Ethernet oder SDH) zugeordnet werden, auf dem der gewünschte Zielrechner entweder direkt (*direct route*) oder über einen zwischengeschalteten Router (*indirect route*) erreichbar ist (siehe Beschreibung des Flags G auf der nächsten Folie). In beiden Fällen wird das IP-Paket in einen entsprechenden MAC-Rahmen verkapselt - bei der direct route wird die MAC-Adresse des gewünschten Zielrechners eingesetzt, bei der indirect route die MAC-Adresse des entsprechenden Routers - die IP-Adresse wird natürlich nicht verändert.

Um die Weiterleitungsinformationen nicht für jeden Rechner einzeln in der Tabelle eintragen zu müssen, kann als „Destination“ auch ein ganzes Subnetz eingetragen sein (erkennbar am nichtvorhandenen H-Flag, siehe nächste Folie). Jedoch werden volle Rechneradressen stets vor Subnetz-Adressen betrachtet. Danach erst kommt der Default-Eintrag (*default-Router*) zum Einsatz.

Wegwahl bei IP

Beispiel: manuell konfigurierter Rechner 137.226.12.184:

Destination	Gateway (Next Hop)	Netmask	Flags	Ref	Use	Interface
137.226.12.0	*	255.255.255.0	U	0	0	eth0
default	137.226.12.1	0.0.0.0	UG	0	0	eth0

Beispiel: automatisch konfigurierter Rechner 137.226.12.98:

Destination	Gateway (Next Hop)	Netmask	Flags	Ref	Use	Interface
137.226.12.0	*	255.255.255.0	US	0	0	eth0
default	137.226.12.1	0.0.0.0	UGS	0	0	eth0
127.0.0.0	127.0.0.1	255.0.0.0	UHS	0	0	lo0
localhost	127.0.0.1	255.255.255.255	UHS	0	0	lo0
137.226.12.98	127.0.0.1	255.255.255.255	UHS	0	0	lo0

Die Weiterleitungstabelle eines Rechners kann mithilfe des UNIX-Befehls netstat -r angezeigt werden.

Destination beinhaltet den DNS-Namen/die IP-Adresse des Zielrechners, **Gateway** die Adresse des nächsten Routers/des lokalen Netzwerkadapters auf dem Weg zum Zielrechner (abhängig von den Flags) und **Interface** den Namen des Netzadapters, über welchen der Zielrechner bzw. nächste Router erreicht werden kann. Ein „*“ bei **Gateway** soll anzeigen, dass sich der Zielrechner im gleichen Netz befindet und keine Weiterleitung über Router hinweg mehr erfolgen muss. (In diesem Fall werden die Daten direkt an den Zielrechner zugestellt, wofür die MAC-Adresse des Zielrechners benötigt wird; diese wird über ARP ermittelt, siehe später im Kapitel.)

Flags:

- **U:** Up, d.h. verfügbar.
- **G:** Gateway, d.h. der in der Spalte **Gateway** angegebene Knoten ist ein IP-Router, der das Paket weitervermitteln soll (indirect route). Bei Einträgen ohne G bezeichnet diese Spalte die Adresse des lokalen Netzadapters. (Oft herrscht das Missverständnis, dass jeder Rechner im Internet eine IP-Adresse benötigt. Tatsächlich muss sogar *jeder Netzwerkadapter* eine IP-Adresse besitzen. Router, d.h. Rechner mit mehr als einem Netzadapter, besitzen somit mehrere IP-Adressen.)
- **S:** statische Route (keine Veränderung durch Routingprotokolle)
- **H:** Host, d.h. angegebene Adresse ist Endsystemadresse (ansonsten wird ein Subnetz beschrieben)

Ref gibt an, welche Routen aktuell von Kommunikationsvorgängen genutzt werden, **Use** zählt die Anzahl der gesandten Pakete auf der jeweiligen Route.

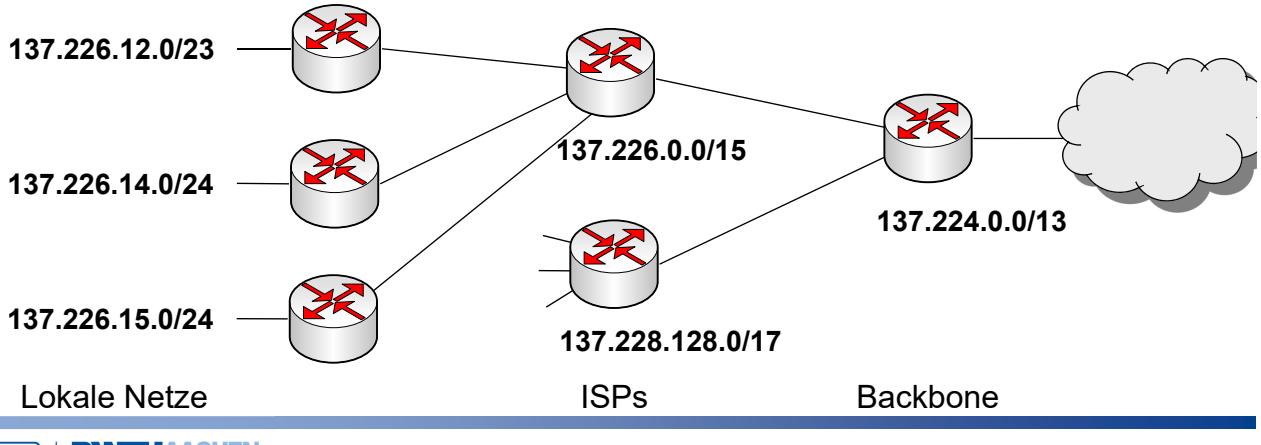
Wird kein passender Eintrag gefunden, greift auf jeden Fall der Eintrag 0.0.0.0/0 – dies ist die *Default-Route*, über die alle Pakete versendet werden, für die keine näheren Weginformationen vorliegen. Der Sinn ist, nicht für jedes Netz weltweit einen Eintrag haben zu müssen, sondern eine Vielzahl von Paketen in eine Richtung weiterleiten zu können, von der man weiß, dass dort zentrale Knotenpunkte stehen, die genauere Informationen besitzen. (Z.B. muss ein Router, der nur ein lokales Netz eines Lehrstuhls der RWTH anschließt, alles, was nicht für das eigene LAN gedacht ist, an den Gebäuderouter weiterleiten – unabhängig vom Zielnetz.) Wie die Routing-Tabellen berechnet werden, wenn Einträge sich dynamisch an die Netzsituation anpassen sollen, regeln Routing-Protokolle – die später im Kapitel behandelt werden.

„Gateway“ wird in der Routing-Terminologie auch als „Next Hop“ bezeichnet – der nächste Hop auf dem Weg zum Ziel.

Classless Inter-Domain Routing (CIDR)

- **CIDR:**

- ▶ Schlechte Ausnutzung des Adressraums selbst mit Subnetzeinteilung
- ▶ Ersetzen der festen Klassen durch *Netzwerk-Präfixe variabler Länge*
 - Es können auch kleine Klasse-C-Netze zu größeren Subnetzen vereint werden
 - *Route Aggregation* – Router kombinieren Einträge weitestmöglich



Die Einführung von Subnetzen brachte allerdings auch einen Nachteil mit sich. Router benötigen für jedes Zielnetz einen eigenen Weiterleitungseintrag. Durch die Aufsplittung großer Netze in kleinere wuchsen Routingtabellen stark an.

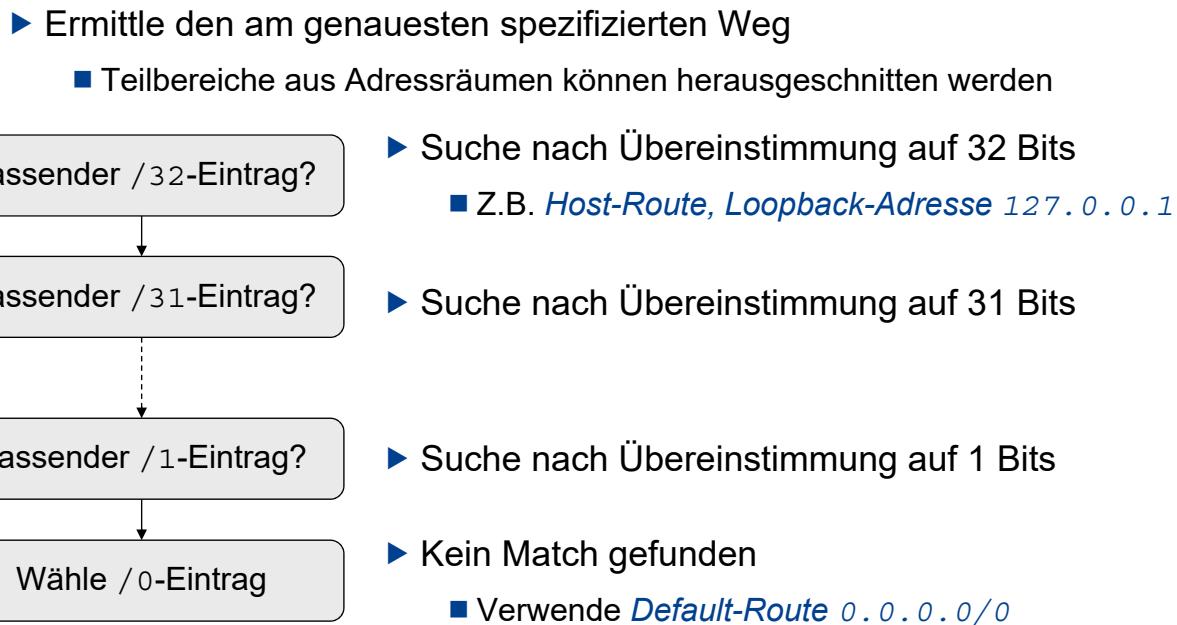
In den 90er Jahren wurde das bisherige “classful routing” zugunsten von “classless routing” aufgegeben. Dies geschah vor allem, um die Routingtabellen handhabbarer zu machen. Es wurde festgelegt, dass Subnetzmasken quasi beliebige Längen haben können und dadurch auch mehrere kleinere Netze durch einen Weiterleitungseintrag erschlagen werden können, wenn ihr Adress-Präfix gleich ist (Route Aggregation).

Einige Beispiele wären:

- Backbone-Router (z.B. an Transatlantik-Link) kann beispielsweise so konfiguriert werden, dass er nur die ersten 13 Bit der IP-Adressen betrachtet. Dadurch werden die Routing-Tabellen relativ klein gehalten und es entsteht wenig Rechenaufwand zur Bestimmung der Ausgangsleitung, über die ein Paket weitergeleitet wird.
- Router eines Providers betrachtet z.B. nur die ersten 15 Bit.
- Router in Firmennetz betrachtet z.B. die ersten 25 Bit.

Longest Prefix Match

- Suche nach dem Routing-Eintrag mit der **größten Netzwerkpräfix-Überdeckung** der Zieladresse



Es kann durch die flexible Aufteilung des Adressraums vorkommen, dass mehrere Einträge der Routingtabelle auf die Zieladresse eines weiterzuleitenden Pakets passen – würde der Lehrstuhl für Informatik 4 beschließen, die RWTH zu verlassen und nach Köln zu ziehen aber sein Subnetz des RWTH-Netzes mitnehmen, wäre dies möglich, allerdings hätten Router außerhalb der RWTH das Problem, dass Daten an 137.226.0.0/16 nach Aachen, Daten an 137.226.12.0/23 nach Köln weitergeleitet werden müssten. Dazu wurde das Longest-Match-Verfahren eingeführt: enthält eine Routingtabelle mehrere passende Einträge, wird derjenige mit dem längsten übereinstimmenden Präfix (d.h. der längsten passenden Subnetzadresse) verwendet. Wird keiner gefunden, passt wieder auf jeden Fall 0.0.0.0/0, also die kürzest mögliche Subnetzmaske – die Default-Route.

Longest Prefix Match

- **Beispiel:**

Ziel	11.1.2.5	= 00001011.0000001.00000010.00000101
------	----------	--------------------------------------

Route #1	11.1.2.0/24	= 00001011.0000001.00000010.00000000
----------	-------------	--------------------------------------

Route #2	11.1.0.0/16	= 00001011.0000001.00000000.00000000
----------	-------------	--------------------------------------

Route #3	11.0.0.0/8	= 00001011.0000000.00000000.00000000
----------	------------	--------------------------------------

- **Flexiblere Aufteilung und Anordnung von Netzbereichen**

- ▶ Z.B. Mitnahme des Adressblocks der Firma bei Umzug an einen neuen Standort
- ▶ Aber: Vergrößerung von Routingtabellen
- ▶ Und trotzdem: freie Adressbereiche sind ausgegangen...

- **Notwendigkeit der Adressierung**

- ▶ Internet: jeder Rechner benötigt eine *eindeutige IP-Adresse*, um global mit anderen Rechnern kommunizieren zu können
- ▶ Lokales Netz: findet Kommunikation nur innerhalb des LAN statt, kann man sich einen *beliebigen Adressbereich* wählen
 - Pakete dürfen das LAN nicht verlassen!

- **Private Adressen**

- ▶ Standardisiert: *private Adressblöcke*, die jeder intern frei nutzen kann:
 - 10.0.0.0 - 10.255.255.255
 - 172.16.0.0 - 172.31.255.255
 - 192.168.0.0 – 192.168.255.255
- ▶ Private Adressen werden im Internet nicht geroutet

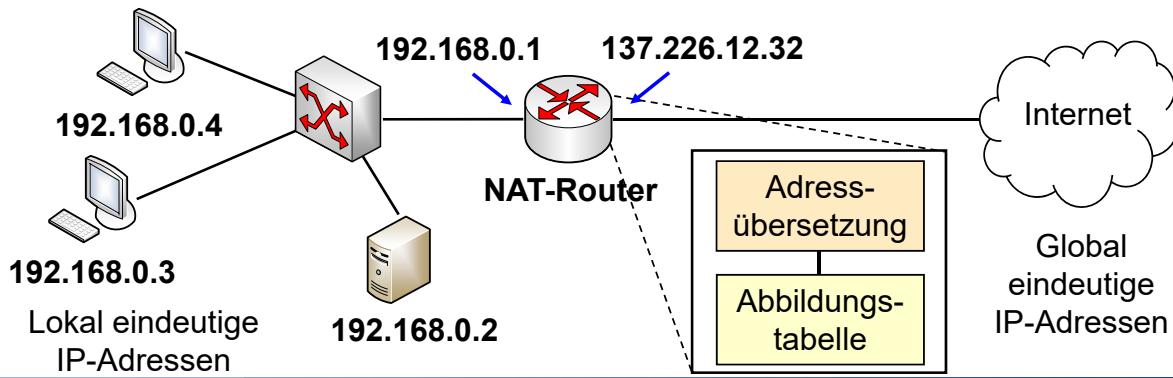
Drei Adressblöcke bei IP sind als private Blöcke freigehalten – jeder kann sich mit diesen Blöcken ein eigenes lokales Netz aufbauen, ohne dafür offiziell Adressen erwerben zu müssen. Allerdings kommt es dadurch dazu, dass mehrere unabhängige LANs die gleichen Adressen nutzen. Damit ist keine eindeutige Adressierung über das lokale Netz hinaus mehr gegeben, daher verwerfen Router Pakete mit Zieladressen aus diesen Bereichen einfach. Man kann sich also ein eigenes IP-Netz aufbauen, allerdings nur mit Rechnern aus dem eigenen Netz kommunizieren.

Diese Adressblöcke hat man sich zu Nutze gemacht, um mit der Adressknappheit bei IP umzugehen.

Network Address Translation (NAT)

- **Prinzip von NAT:**

- ▶ Rechner im lokalen Netz verwenden private Adressen
- ▶ Das Netz bekommt eine global eindeutige Adresse zugewiesen
- ▶ Zugangsrouter nimmt *transparente Umsetzung* zwischen Adressen vor
 - Speicherung in Abbildungstabelle
 - Keine Änderungen an Endgeräten erforderlich

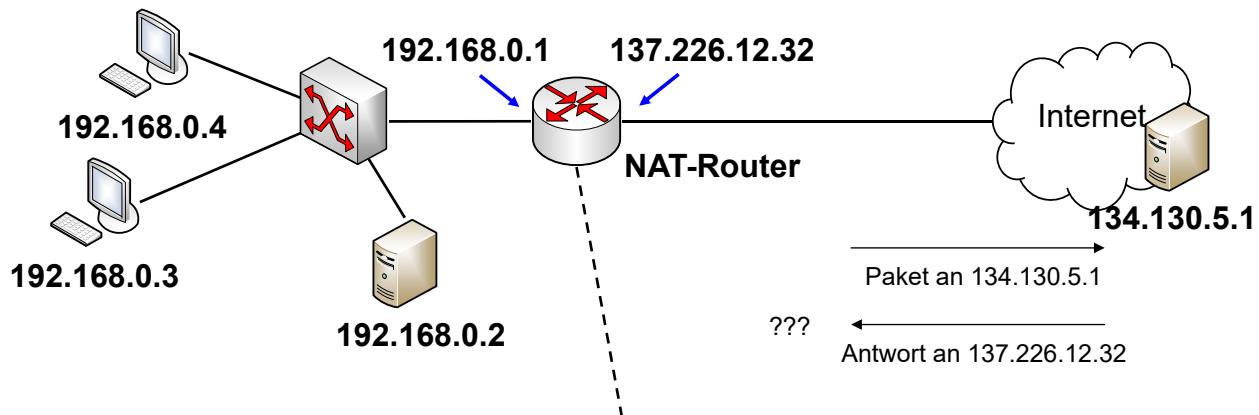


Ziel von NAT ist es, Netze mit privaten (und somit nicht routbaren) IP-Adressen an das Internet anzubinden (die Verwendung routbarer Adressen ist zwar ebenfalls möglich, es werden im Normalfall aber nur private Adressen verwendet, um Fehler durch falsch konfigurierte Zugangsrouter zu vermeiden). Hierzu nimmt der Router des lokalen Netzes eine Adressumsetzung vor, falls an der internen Netzwerkschnittstelle Pakete mit einer privaten Absenderadresse empfangen werden, die für ein externes Netz bestimmt sind. Diese Adressumsetzung kann dabei statisch (1:1) oder dynamisch (n private Adressen auf m öffentliche Adressen) erfolgen. Im Normalfall sind damit Rechner innerhalb des Netzes von außen nicht adressierbar. (Aber eine entsprechende statische Abbildung in umgekehrter Richtung kann angelegt werden, falls es notwendig ist.)

Somit ergeben sich zwei wesentliche Vorteile dieser Technologie: zum einen sind die Rechner im privaten Adressbereich geschützt, da sie von außen nicht zugreifbar sind, zum anderen wird nur eine geringe Anzahl an öffentlichen IP-Adressen benötigt.

Dabei kann ein ganzer privater Adressbereich auf nur eine einzige öffentliche IP-Adresse abgebildet werden.

NAT – Beispiel



Protokoll	lokal		global		Ziel	
	IP-Adresse	Port	IP-Adresse	Port	IP-Adresse	Port
TCP	192.168.0.4	53211	137.226.12.32	53211	134.130.5.1	80
TCP	192.168.0.3	48331	137.226.12.32	48331	134.130.5.1	80
TCP	192.168.0.4	48331	137.226.12.32	48332	141.18.62.55	25
TCP	192.168.0.2	49999	137.226.12.32	48333	123.12.123.12	12345

Problem bei NAT: was passiert, wenn mehrere Rechner des lokalen Netzes gleichzeitig nach Außen kommunizieren wollen? Alle verwenden die gleiche globale Adresse, so dass keine eindeutige Rückübersetzung möglich ist, wenn ein Antwortpaket von außen kommt. Hier könnte man natürlich einen Broadcast der Antwortpakete im lokalen Netz vornehmen (was in manchen Fällen auch gemacht wird), was aber im allgemeinen nicht effizient und meist auch nicht gewünscht ist.

Üblich ist daher, in der Abbildungstabelle noch Zusatzinformationen zu speichern, die eine eindeutige Rückübersetzung erlauben. Hierzu nimmt man im lokalen Netz eine Adresserweiterung vor, die dem eigentlichen Schichtenkonzept widerspricht: man verwendet die *Schicht-4-Adresse* eines ausgehenden Pakets zur Identifizierung des Rechners. Dies ist der sogenannte *Port* (TCP/UDP – Kapitel 5), der die sendende *Anwendung* auf dem sendenden Rechner identifiziert. Dadurch schafft man einen neuen Adressraum von 16 Bit, anhand dessen die lokalen Rechner unterschieden werden können. Dieses Prinzip wird auch NAT-Overloading genannt.

Sollte es vorkommen, dass zwei Rechner den gleichen Port nutzen, um Daten zu versenden, kann der NAT-Router nicht nur die IP-Adresse, sondern auch den Port vor der Weiterleitung des Pakets modifizieren (und muss natürlich die entsprechenden Informationen mit in seiner Abbildungstabelle speichern). Dieses Verfahren wird auch bei der Anbindung von Haushalten durch ISPs vorgenommen – man bekommt eine einzige IP-Adresse zugewiesen, der heimatische NAT-Router schließt alle lokalen Rechner mit privaten Adressen an.

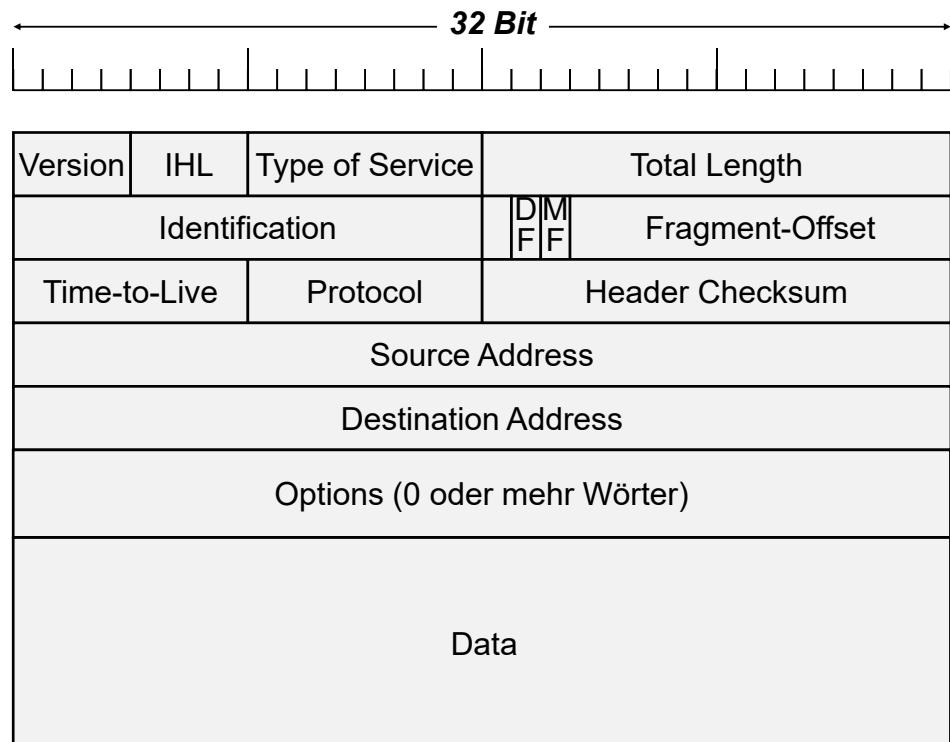
Im Beispiel auf der Folie sieht man dieses Prinzip. Ohne die Hinzunahme des Ports als Adressinformation wüsste der NAT-Router bei der Antwort von 134.130.5.1 nicht, an welchen lokalen Rechner – 192.168.0.4 und 192.168.0.3 – das Paket weitergeleitet werden müsste.

(Achtung: die Speicherung der Zieladressen ist nicht unbedingt notwendig; die in der Tabelle blau hinterlegten Einträge alleine sind bereits ausreichend, eine eindeutige Rückübersetzung vornehmen zu können. Ob die Zieladressen mit abgespeichert werden oder nicht, hängt von der Implementierung ab.)

An der Tabelle zeigen sich die Möglichkeiten von NAT: in Zeile 1 und 2 wird jeweils beim ausgehenden IP-Paket nur die Absender-IP-Adresse ersetzt. In Zeile 3 verwendet Rechner 192.168.0.4 den gleichen Port wie 192.168.0.3 zuvor bereits; hier wird auch die Portnummer ersetzt, um Einträge eindeutig zu halten. Eine weitere Möglichkeit zeigt Zeile 4: es gibt auch NAT-Implementierungen, die den Port immer ersetzen, egal, ob sie bereits einen identischen Eintrag haben oder nicht, und beginnend mit einer bestimmten Nummer einfach alle Ports aufsteigend verwenden.

Der Eintrag „Protokoll“ ist zwar nicht für das Mapping wesentlich, wird allerdings z.B. mit eingetragen, damit nur Pakete ins lokale Netz gelassen werden, die das korrekte Transportprotokoll verwenden (Sicherheitsaspekte) oder eine einfache Konfiguration des NAT-Routers ermöglichen (Definition von Regeln, die auf alle TCP-Verbindungen angewendet werden).

Der Aufbau eines IP-Pakets



Der IP-Header

- **Version**

- ▶ IP-Versionsnummer (hier immer 4)
- ▶ Erlaubt den gleichzeitigen Einsatz mehrerer Versionen
 - Anhand des Werts ermittelt ein Router den Aufbau des folgenden Headers

- **IHL**

- ▶ IP-Header-Length (in Worten á 4 Byte; zwischen 5 und 15, je nach Optionen)

- **Total Length**

- ▶ Länge des gesamten Datagramms (in Byte, $\leq 2^{16}-1 = 65535$ Bytes)
- ▶ In der Praxis üblicherweise maximal 1500 Byte

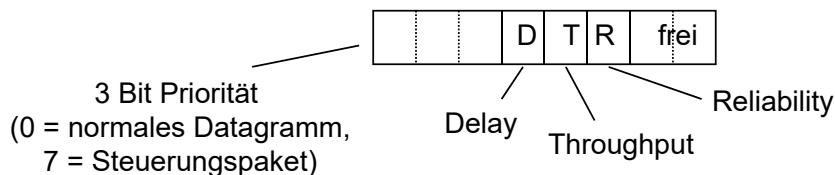
- **Source Address / Destination Address**

- ▶ IP-Adressen von sendendem und empfangendem Rechner

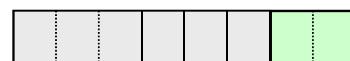
Der IP-Header

• Type of Service

- ▶ Angabe der gewünschten Eigenschaften bei der Übertragung: ist Geschwindigkeit, Durchsatz oder Zuverlässigkeit wichtig? Welche Priorität hat das Paket?



- ▶ Nicht eingesetzt, darum heute: Neudefinition der Bedeutung zur Unterstützung von Quality of Service und Stauerkennung



Das Type-of-Service-Feld sollte dazu dienen, Pakete nach Typ des Payloads klassifizieren zu können: z.B. Echtzeitdaten, Dateiübertragung, ...

Routerhersteller haben es allerdings stets ignoriert, da jede weitere Überprüfung eines Headerfeldes und entsprechende Aktionen zur unterschiedlichen Behandlung von Paketen zu einer Erhöhung der Bearbeitungszeit pro Paket führen. Stattdessen werden alle Pakete gleich (und einfach) behandelt, um die Anzahl weitergeleiteter Pakete pro Sekunde maximieren zu können.

Im Laufe der Jahre wurde die Bedeutung des Feldes daher umdefiniert. Die vorderen 6 Bits dienen der QoS-Unterstützung, behalten also im Wesentlichen ihren ursprünglichen Zweck bei, allerdings mit anderer, effizienterer Implementierung (hier nicht behandelt). Die letzten beiden Bits wurden der Stauerkennung zugeordnet, um TCP eine effizientere Arbeit zu ermöglichen (siehe Kapitel 5).

Der IP-Header

- **Time-to-Live (TTL)**

- ▶ Lebenszeit von Datagrammen begrenzen auf maximal 255 Hops
(verhindert endloses Kreisen von Paketen im Netz)
- ▶ Der Zähler wird von jedem Router um 1 verringert
- ▶ Bei 0 wird das Datagramm verworfen

- **Protocol**

- ▶ Welches Transportprotokoll wird im Datenteil verwendet?
- ▶ An welchen Transportprozess ist das Paket daher beim Empfänger weiterzugeben?

- **Header Checksum**

- ▶ Prüfsumme über den Header
- ▶ Muss bei jedem Hop neu berechnet werden (da sich TTL ändert)

TTL: war ursprünglich als tatsächliche Zeitangabe gedacht, wurde aber zur einfacheren Implementierung so modifiziert, dass jeder empfangende Router den Zähler um 1 verringert, unabhängig von der tatsächlich verbrachten Zeit im Router bzw. auf einem Link. Der Router, der den Wert auf 0 runterzählt, verwirft das Paket, d.h. der initiale Wert gibt an, wie viele Teilstrecken ein Paket maximal zurücklegen darf.

Header Checksum: ist keine CRC, sondern einfacher: der Header wird in 16-Bit-Blöcke zerlegt und diese (Einerkomplement) aufsummiert. Vom Ergebnis wird wieder das Einerkomplement gebildet.

Der IP-Header

- **Identification**

- ▶ Eindeutige Kennzeichnung eines Pakets

- **DF: Don't Fragment**

- ▶ Das Paket darf nicht fragmentiert werden
 - ▶ Zu große Pakete mit gesetztem DF-Flag werden verworfen

- **MF: More Fragments**

- ▶ "1" - es folgen weitere Fragmente
 - ▶ "0" - letztes Fragment eines Pakets

- **Fragment Offset**

- ▶ Folgenummern der Fragmente eines Pakets
 - ▶ Sagt aus, an welche Stelle des Datenteils eines Pakets (gerechnet in 8-Byte-Stücken) ein Fragment gehört

Anmerkung: es gibt drei Flag-Bits: MF, DF und eins, welches für zukünftige Zwecke reserviert wurde.

Fragmentierung

• Verwendung von Fragmentierung

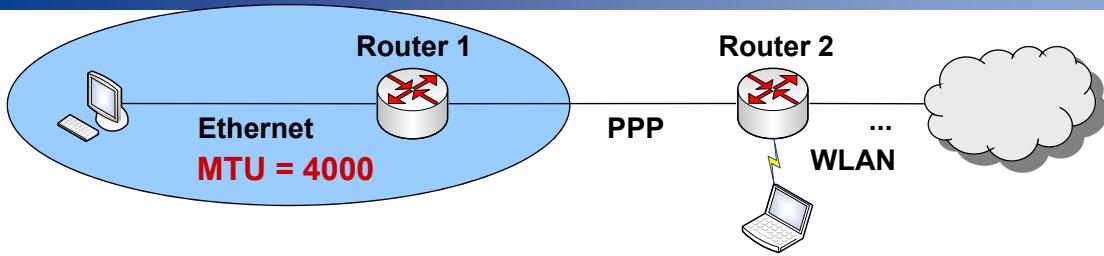
- ▶ In jedem Netz gibt es eine Maximallänge für Dateneinheiten
 - *MTU – Maximum Transfer Unit*
 - Z.B. Ethernet: Datenteil darf maximal 1500 Byte groß sein
- ▶ Jeder Router prüft, ob er ein IP-Paket über die ermittelte Leitung senden kann oder ob es dort die Maximallänge überschreitet
 - Zu große IP-Pakete werden fragmentiert, damit sie weitergeleitet werden können
- ▶ Aber: der Datenteil des Pakets formt eine logische Einheit
 - Wie kann diese wieder zusammengesetzt werden, bevor der Zielrechner die Daten verarbeitet?

Eine Aufgabe von IP ist die Fragmentierung. Typischerweise wird der Sender Pakete maximaler Größe erstellen und versenden (1500 Byte). Allerdings hat jedes Schicht-2-Protokoll eine MTU (= maximale Größe des Payloads). Ist ein IP-Paket größer als die MTU, muss IP das Paket fragmentieren (= in Teilkörper zerlegen), damit es über mehrere Pakete verteilt weitergeleitet werden kann. Bei Ethernet ist die MTU 1500 Byte – was auch der Grund dafür ist, dass in der Praxis keine größeren IP-Pakete erstellt werden.

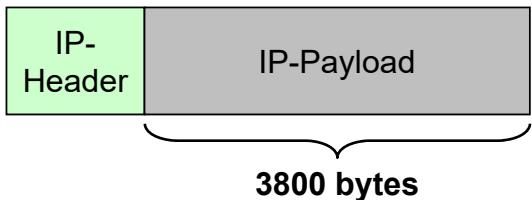
Trotzdem kann es sein, dass bereits der Sender oder ein Router auf dem Weg zum Ziel ein IP-Paket zerlegen muss, da es ansonsten nicht über das nächste Schicht-2-Protokoll weitergeleitet werden kann.

Damit der Empfänger Pakete wieder korrekt zusammensetzen kann, müssen während der Fragmentierung ein paar Kontrollinformationen hinzugefügt werden, die dem Empfänger mitteilen, an welcher Stelle des Originalpakets ein Teilkörper anzusiedeln ist.

Fragmentierung – Beispiel



TL=3820, ID=42, MF=0

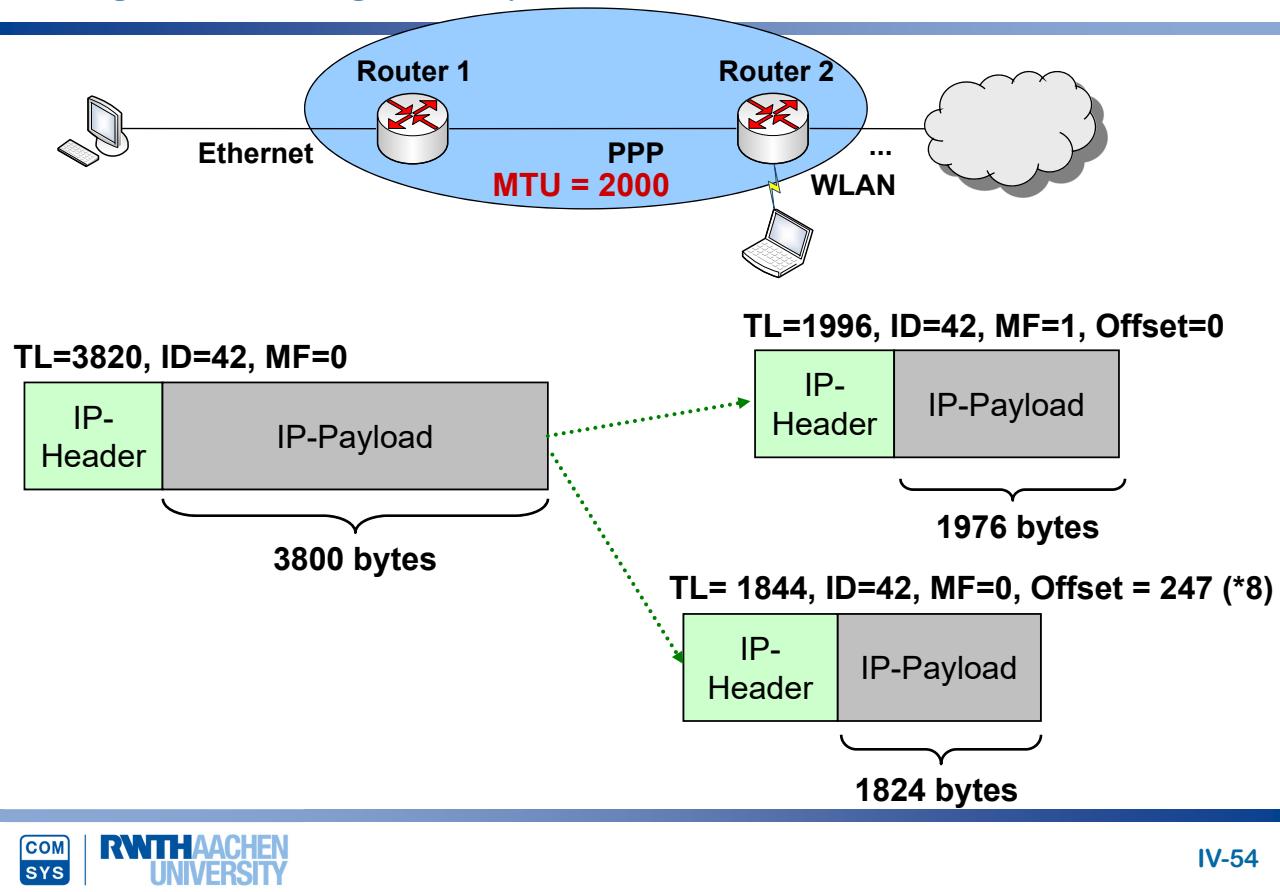


TL: Total Length

ID: Identification

MF: More Fragments

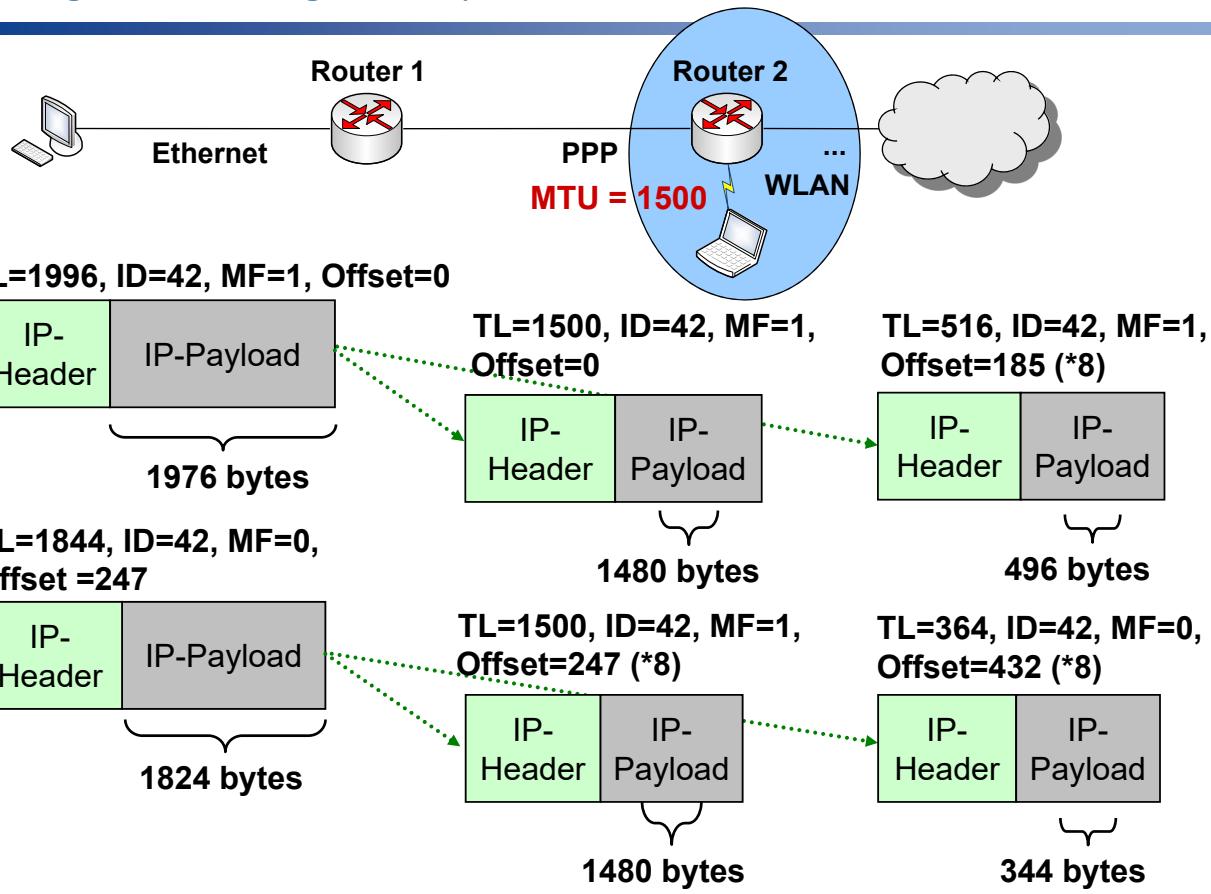
Fragmentierung – Beispiel



Die Kontrollinformationen zum Erkennen der Fragmentierung beim Empfänger sind

- ID: alle Fragmente eines Pakets tragen die ID des Ursprungspakets, damit der Empfänger identifizieren kann, welche Fragmente zusammengehören.
- MF: dieses Flag wird bei allen außer dem letzten gesetzt. Der Empfänger weiß bei Erhalt eines IP-Pakets mit gesetztem Flag, dass dies kein komplettes Paket ist, sondern er mit der Verarbeitung warten muss, bis alle Fragmente empfangen wurden. Im letzten Fragment ist das Bit auf 0 gesetzt, um anzugeben, dass keine weiteren Fragmente folgen.
- Offset: der Offset gibt (in Vielfachen von 8 Byte) an, an welcher Stelle im Originalpaket das Fragment anzusortieren ist. Hierdurch kann der Empfänger Fragmente in die richtige Reihenfolge sortieren und auch feststellen, ob einzelne Fragmente verloren gegangen sind.

Fragmentierung – Beispiel



Zusammensetzung der Fragmente

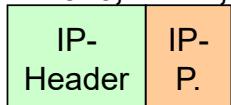
- Empfänger nutzt MF und Offset, um Fragmente zu erkennen und zusammenzusetzen

TL=1500, ID=42, MF=1, Offset=0



- ▶ IP-Pakete mit MF=1 oder mit MF=0 aber Offset>0 sind Fragmente
- ▶ Sortierung der Fragmente anhand des Offsets
- ▶ Unterscheidung der Fragmente unterschiedlicher Pakete durch ID

TL=516, ID=42, MF=1, Offset=185 (*8)



TL=1500, ID=42, MF=1, Offset=247 (*8)



TL=364, ID=42, MF=0, Offset=432 (*8)



- **Options: Spielraum für zukünftige Erweiterungen**

- ▶ Länge: Vielfaches von 4 Byte, daher ist möglicherweise Padding notwendig
- ▶ Fünf Optionen definiert, wenn auch nicht genutzt
 - **Security:** wie geheim sind die transportierten Informationen?
(z.B. zur Umgehung bestimmter Router)
 - **Strict Source Routing:** Vollständiger Pfad vom Quell- zum Zielhost, definiert durch die IP-Adressen der zu passierenden Router
 - **Loose Source Routing:** die aufgelisteten Router müssen in angegebener Reihenfolge durchlaufen werden, zusätzliche Router sind erlaubt
 - **Record Route:** Aufzeichnung der IP-Adressen der durchlaufenden Router (maximal 9 IP-Adressen möglich!)
 - **Time Stamp:** Aufzeichnung von IP-Adressen mit Zeitstempel für jeden Router (je 32 Bit)

Mittels Source Routing kann der Sender einer Nachricht festlegen, welchen Weg die Nachricht durch das Netz nehmen soll, d.h. nicht die Router treffen anhand ihrer Tabellen die Entscheidungen, sondern der Sender des Pakets. Beim Strict Source Routing gibt der Sender die exakte Route vor. Alle angegebenen Hosts/Router müssen durchlaufen werden, in der angegebenen Reihenfolge, und ohne dabei andere Hosts/Router zu durchlaufen. Beim Loose Source Routing wird zwar eine Route angegeben, und die in dieser Route aufgeführten Hosts müssen in der angegebenen Reihenfolge durchlaufen werden, dazwischen dürfen aber auch andere Hosts und/oder Router durchlaufen werden.

Source Routing wird kaum verwendet, und Pakete mit Source-Routing-Informationen werden häufig verworfen. Dies hat seinen Grund in Sicherheitsproblemen: so öffnet Source Routing z.B. die Möglichkeit, Router gezielt zu überlasten.

Zudem ist der Optionsteil auf maximal 40 Byte beschränkt – wovon ein bisschen bereits benötigt wird, um die verwendete Option anzuzeigen. Damit können maximal 9 Adressen eingetragen werden, was bei der heutigen Netzstruktur zu wenig ist. Gleches gilt für die Record-Route-Option und erst recht für die Timestamp-Option, bei der zu jedem Router insgesamt 8 Byte an Informationen abgespeichert werden müssen.

Überprüfung des IP-Headers

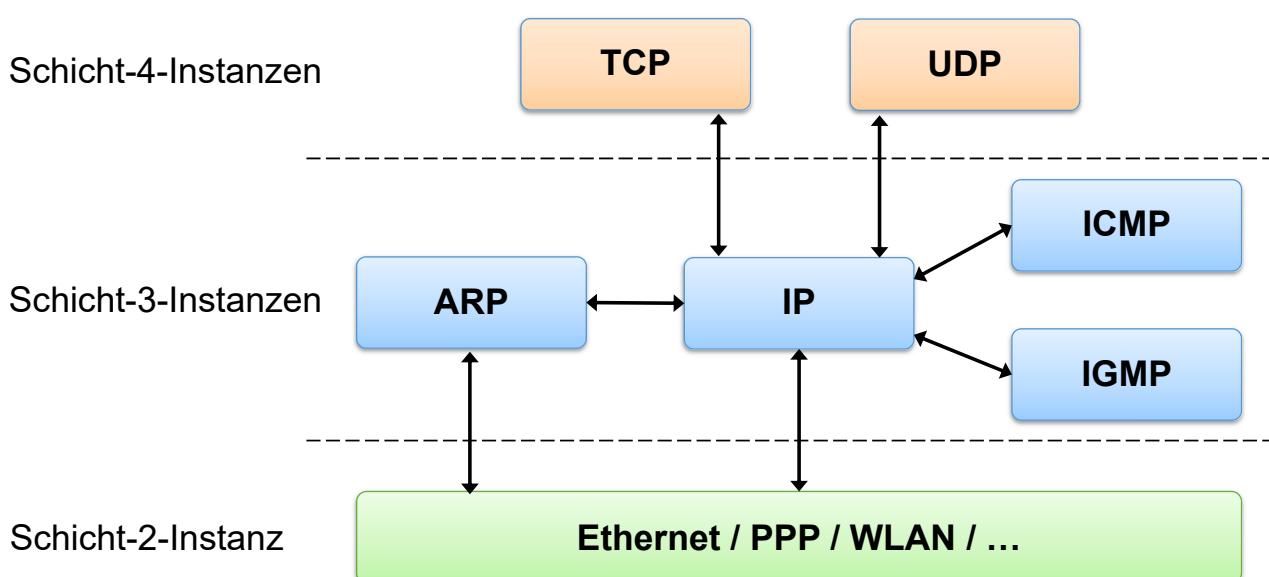
- Überprüfungen, die nach dem Empfang eines IP-Datagramms am Header durchgeführt werden:

- ▶ Test der IP-Versionsnummer
- ▶ Überprüfung der korrekten Länge des Headers und des Pakets
- ▶ Prüfsummenbildung über den IP-Header
- ▶ Überprüfung der Paketlebenszeit
- ▶ Überprüfung der Protokoll-ID
- ▶ Überprüfung der Gültigkeit von Quell- und Zieladresse
 - Private Adressen sind nicht erlaubt

- Negatives Ergebnis bei einer Überprüfung

- ▶ Paket verwerfen, Fehlermeldung per ICMP an Sender schicken

IP-Instanz und angrenzende Instanzen



Wir wollen die Ausführungen zur IP-Instanz mit der Beschreibung des Zusammenspiels mit den oben, unten und insbesondere seitlich angrenzenden Protokollinstanzen abschließen.

Nachdem im Protokollstack von oben, also üblicherweise von einer TCP- oder UDP-Instanz, die zu übertragenden Daten übergeben wurden, hat die IP-Instanz die Aufgabe, die Daten anhand einer IP-Adresse weiterzuvermitteln. IP-Adressen spannen allerdings nur einen logischen Adressraum auf – physikalisch kommunizieren alle Geräte im Netz immer noch über eine Netzwerkkarte, die anhand einer MAC-Adresse (Schicht-2-Adresse) adressiert werden muss. Zur Ermittlung dieser Adresse dient ARP. Die IP-Instanz beauftragt die lokale ARP-Instanz damit, zu der Ziel-IP-Adresse die zugehörige MAC-Adresse zu ermitteln.

Ist die gesuchte MAC-Adresse bestimmt, werden die Daten der lokalen Schicht-2-Instanz übergeben (also z.B. Ethernet) und von dieser an die entfernte Schicht-2-Instanz übertragen. Dort werden dann die Daten an die überliegende IP-Instanz (und im folgenden die TCP-/UDP-Instanz) weitergegeben.

Die beiden Hilfsprotokolle ICMP und IGMP werden von der IP-Instanz für besondere Zwecke genutzt. ICMP dient zur Übermittlung von Kontrollnachrichten auf IP-Ebene und wird üblicherweise verwendet, um bei Problemen während der Übermittlung eine entsprechende Benachrichtigung an den Absender zu übermitteln.

IGMP (Internet Group Management Protocol, hier nicht weiter behandelt) dient zur Übermittlung und Verwaltung von Informationen zu Gruppenzugehörigkeiten und ermöglicht damit Multicast.

Address Resolution Protocol (ARP)

- **Aufgabe:**

- ▶ Umsetzen IP-Adresse → Schicht-2-Adresse (MAC-Adresse)
- ▶ Beispiel Rechner messenger:
 - IP-Adresse: 137.226.13.40 → Ethernet-Adresse: 00:14:5E:67:99:C0

- **Vorgehensweise:**

- ▶ ARP erhält eine IP-Adresse zur Adressauflösung
- ▶ ARP sendet einen Broadcast im LAN unter Angabe der IP-Adresse
- ▶ Alle Stationen im Netz empfangen die Anfrage, doch nur diejenige, die ihre eigene IP-Adresse erkennt, antwortet
- ▶ Die Antwort wird bei der anfragenden Station gespeichert (ARP-Cache), um ein erneutes Anfragen zu vermeiden
- ▶ Dieser Eintrag muss nach einem festgelegten Zeitintervall wieder gelöscht werden



ARP (Address Resolution Protocol)

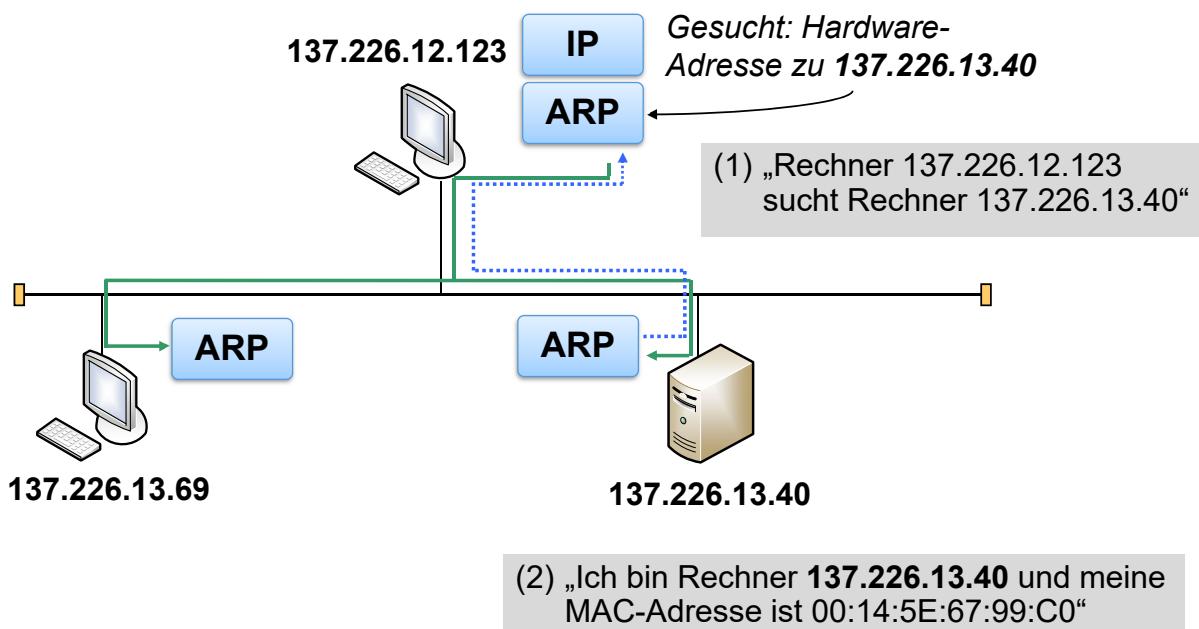
Beim Versenden eines IP-Pakets gibt die IP-Instanz das Ziel in Form einer IP-Adresse an. Um jedoch das Paket zum nächsten Router oder Host zu senden, muss man die physikalische Adresse dieser Station kennen. Das Problem besteht nun darin, die Zuordnung von MAC-Adresse zu IP-Adresse eindeutig zu lösen. Dieses Problem tritt nicht bei der Weiterleitung von Paketen zwischen Routern auf, da es hier eine dedizierte Leitung gibt, über die genau ein anderes Gerät (ein Router) angeschlossen ist. Sobald allerdings über eine Leitung mehrere Geräte angeschlossen sind, d.h. in einem LAN, muss vor der Weiterleitung des Pakets die Ziel-MAC-Adresse ermittelt werden.

Dabei gibt es drei Möglichkeiten:

- Man speichert in jedem Host die Zuordnung IP→MAC in einer Datei. Dies klappt nur für kleine Netze, da der Verwaltungsaufwand zu hoch ist, jede Änderung auf jedem Rechner zu korrigieren.
- Man unterhält einen speziellen Server in seinem Netz, der die Zuordnung vornimmt. Die MAC-Adresse dieses Servers wird in allen anderen Geräten abgespeichert. Dies bringt einen Verwaltungsaufwand mit sich, und zudem liegt das Netz lahm, wenn dieser Server ausfällt.
- Man löst das Problem dezentral, indem man den gewünschten Rechner durch Broadcast ermittelt.

Die ersten beiden Verfahren werden kaum benutzt. Die dezentrale Variante hat sich durchgesetzt und wird durch ARP implementiert.

ARP – Beispiel



Der Ablauf des ARP-Protokolls sieht folgendermaßen aus:

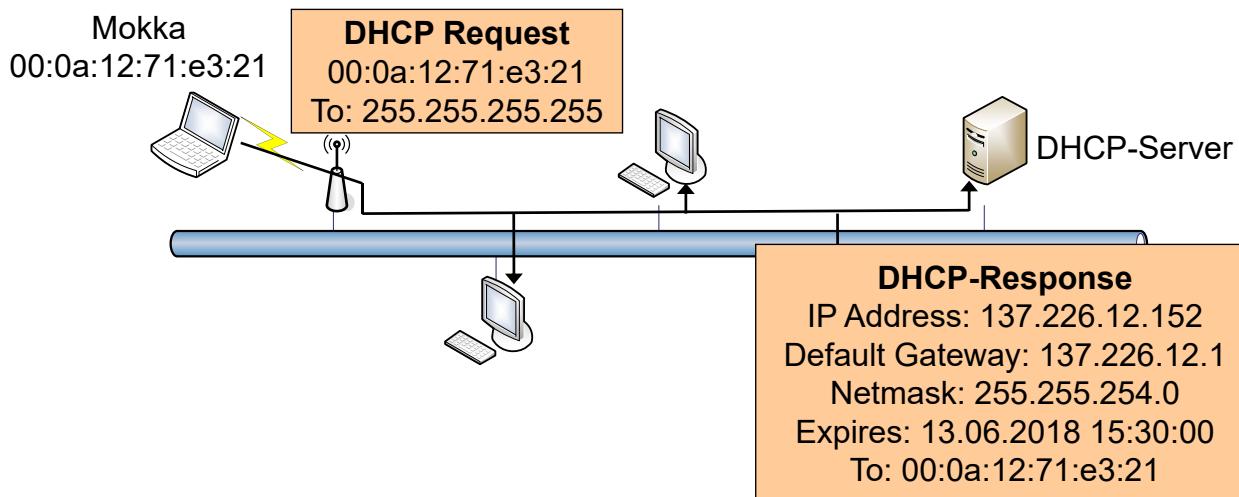
- Der Sender eines Pakets sieht in seinem ARP-Cache nach, ob er die MAC-Adresse zu der Ziel-IP-Adresse hat. Wenn ja, benutzt er diese. Die Werte im Cache werden nach einer gewissen Zeit (ohne Nachfrage) gelöscht, um zu vermeiden, dass veraltete Informationen weitergenutzt werden (falls sich z.B. durch einen Wechsel der Netzwerkkarte die MAC-Adresse eines Rechners ändert oder falls Zuordnungen durch Neukonfiguration der IP-Adressen ungültig werden).
- Wenn die MAC-Adresse nicht vorhanden ist, sendet er durch Broadcast (auf Schicht 2) einen ARP-Request, bei dem er seine eigene IP- und MAC-Adresse und die IP-Adresse des Ziel-Hosts versendet.
- Alle Rechner im LAN erhalten diese Nachricht und vergleichen die IP-Adresse mit ihrer eigenen. Zusätzlich vermerken sie die Zuordnung von IP-/MAC-Adresse des Senders in ihrem Cache; diese ist nun in allen ARP-Caches des LANs gespeichert. (In der Praxis ist es jedoch oft so, dass nur der betroffene Rechner diese Zuordnung speichert, alle anderen ignorieren die ARP-Nachricht.)
- Der Rechner, der die gesuchte IP-Adresse besitzt, sendet nun einen ARP-Reply an den Sender zurück (Unicast, da die MAC-Adresse des Senders bekannt ist), in dem seine MAC-Adresse steht. Der Sender speichert die Zuordnung in seinem Cache und sendet das eigentliche Paket ab.

Es gibt eine Sicherheitslücke bei diesem Verfahren: der ARP-Reply könnte von einem anderen Rechner kommen, der vorgibt, die gesuchte Adresse zu haben und somit alle Pakete erhält, die für den echten Rechner gedacht waren. Falls mehrere ARP-Replies kommen, verwirft der Initiator die zusätzlichen, so dass der erste Antwortende gewinnt.

DHCP: Dynamic Host Configuration Protocol

- **Einfache Konfiguration von vernetzten Rechnern**

- ▶ Weist IP-Adresse zu und liefert Informationen zur Netzkonfiguration:
DNS-Server-Adresse, Domain-Namen, Subnetz-Masken, Router etc.
→ *automatische Integration* eines Rechners in das Internet bzw. Intranet



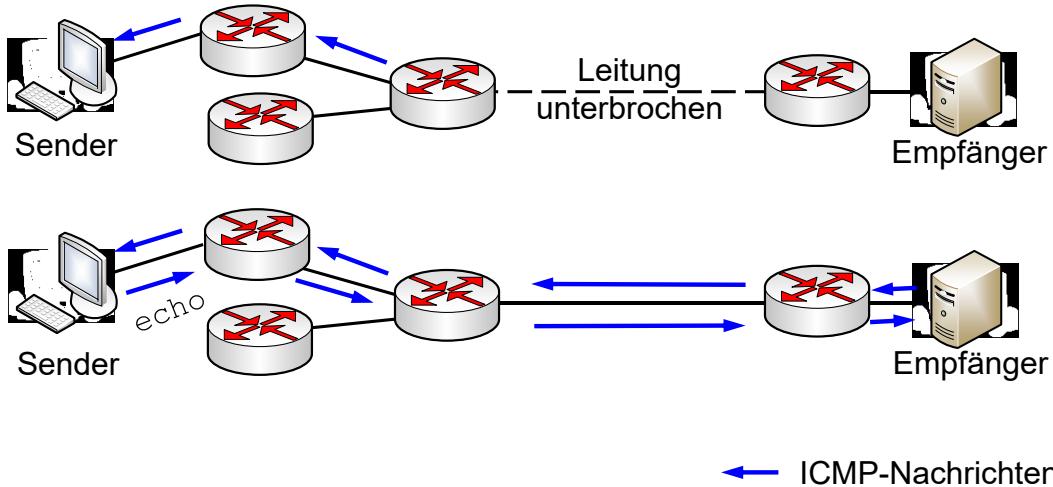
Ein Anwendungsprotokoll, das in diesem Kontext wichtig ist, ist das Dynamic Host Configuration Protocol (DHCP).

DHCP erlaubt das einfache Aufbauen eines Netzwerks mit Hosts, die eventuell nur zeitweise Teil dieses Netzes sein sollen und bei Bedarf IP-Adressen zugewiesen bekommen sollen. Dafür stellt der Netzwerkadministrator einen Bereich seiner verfügbaren Adressen einem speziellen DHCP-Server zur Verfügung. Wenn ein Client dem Netz hinzugefügt wird und eine IP-Adresse für die Kommunikation benötigt, sendet er eine DHCP-Anfrage als Broadcast auf Schicht 2. Der im Netz befindliche DHCP-Server wiest dem Client eine IP-Adresse zu und sendet ihm die gesamte Netzkonfiguration mit: die verwendete Subnetzmaske, die Adresse eines Standardgateways (Zugangsrouter des LANs, benötigt, um Daten an Hosts außerhalb des eigenen LANs versenden zu können), Adressen von DNS-Servern usw. Der Client verfügt nun über alle Informationen, um sowohl im Netz als auch aus dem Netz heraus zu kommunizieren.

Steuerung von IP: ICMP

- **IP: unzuverlässiger Datenaustausch**

- ▶ Für Fehlerfälle oder Testzwecke wird das **ICMP (Internet Control Message Protocol)** verwendet



IP ist ein unzuverlässiges Protokoll. Man bekommt keine Bestätigung über die Zustellung eines Pakets oder eine Meldung über verlorengegangene Pakete. Außerdem besitzt es keinen Mechanismus, der auf Überlast reagiert.

Mit ICMP können einige dieser Mängel behoben werden. Zwar werden keine Bestätigungen für die Zustellung von Paketen versendet, aber über ICMP kann eine Meldung versendet werden, falls z.B. ein Paket in einem Router verworfen wurde oder wenn ein Router stark belastet wird. Dies dient dazu, den Absender davon abzuhalten, denselben Fehler oft zu wiederholen. ICMP-Meldungen können sowohl von Routern als auch von Endsystemen verschickt werden.

Hosts können mittels ICMP auch ohne Vorliegen einer Fehlersituation eine ICMP-Nachricht an einen anderen Host versenden, beispielsweise, um zu prüfen, ob ein Host erreichbar ist (ping).

Man kann u.A. folgende Meldungen mit ICMP versenden:

- Zieladresse nicht erreichbar (destination unreachable): Ein Paket konnte wegen eines Ausfalls (Leitung, Router, oder Endsystem) nicht zugestellt werden.
- Zeit abgelaufen (time exceeded): Ein Paket wurde wegen Ablauf des TTL-Wertes vernichtet. Hinweis auf Kreisen des Pakets.
- Falscher Parameter (parameter problem): Ein Paket wurde wegen eines unzulässigen Wertes im IP-Header verworfen.
- Reduktion der Datenrate eines Senders (source quench): Ein überlastetes Kommunikationssystem fordert den Sender auf, die Übertragungsrate zu senken.
- Umleiten (redirect): Ein Paket sollte besser an einen anderen Router gesendet werden (z.B. da keine

geeigneten Routinginformationen bekannt sind).

Die einzelnen Meldungen können in der ICMP-Nachricht noch genauer erläutert, bspw. wird angezeigt, welches Gerät ausgefallen ist (Router oder Host), zudem kann der Header des IP-Pakets, welches die Meldung ausgelöst hat, mitgesendet werden, um dem Absender eine Analyse des Fehlers zu ermöglichen.

Wichtig zu merken ist, dass die sendende IP-Instanz zwar einen Bericht zur Analyse der Fehlerursache bekommt, ihn aber nicht dazu verwendet, die Zuverlässigkeit für die Paketübertragung zu erhöhen; beispielsweise wird keine Neuübertragung verworfener Pakete vorgenommen, sondern lediglich die Fehlerursache notiert.

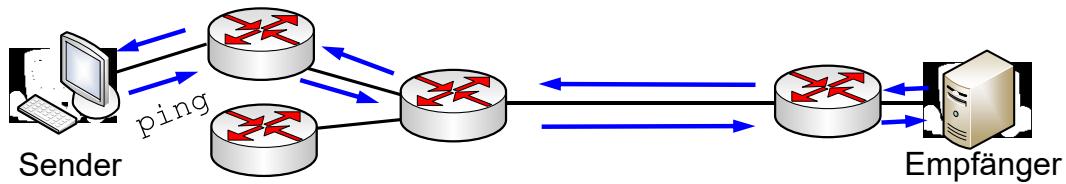
Außer den obigen Meldungen in einem Fehlerfall kann man mit ICMP auch Funktionen zur Diagnose einzelner IP-Rechner ausführen:

- Echo und Echoantwort (echo and echo reply): Der Sender sendet eine Echoanfrage an den Partner (ping) und kann dabei beliebige Daten mitsenden. Der Empfänger muss nun eine Antwort auf diese Echoanfrage zurücksenden. Dabei sendet er die Nutzdaten einfach wieder zurück. Die Nutzdaten dienen zur Analyse des Transports bei unterschiedlicher Nutzlast.
- Zeitstempel und Zeitstempelantwort (timestamp and timestamp reply): Bei dieser Anfrage sendet der Absender seine Systemzeit t_0 zum Zielhost. Dieser sendet wiederum diese Zeit, die Empfangszeit der ICMP-Anfrage t_1 und die Absendezzeit der Antwort t_2 zurück. Somit kann der Initiator Rückschlüsse über die Übertragungsverzögerung auf der Verbindung und die Belastung des Zielhosts ziehen. (Achtung: da zwei beliebige Rechner nicht unbedingt synchronisierte Uhren haben, sollte das Ergebnis mit Vorsicht betrachtet werden!)

ICMP benutzt IP zum Transport seiner Meldungen, man rechnet es jedoch trotzdem zur IP-Schicht. Enthält ein IP-Paket eine ICMP-Meldung, wird das Protocol-Feld im IP-Header entsprechend gesetzt (Wert = 1). Die ICMP-Meldung (bzw. ICMP-PDU) steht im Datenteil des IP-Pakets.

ICMP: ping

- ping: Testen der Erreichbarkeit eines Rechners und Mesung der RTT zu diesem Rechner
 - ▶ Einfache ICMP-Anfrage/-Antwort: echo
 - Ein Empfänger antwortet auf einen echo-Request so schnell wie möglich mit einem echo-Reply
 - ▶ Sender misst Zeit zwischen Aussenden der Anfrage und Erhalt der Antwort
 - ▶ Mehrmalige Wiederholung, Bildung von Mittelwerten



ping ist eine einfache Anwendung, mit der man schnell feststellen kann, ob ein bestimmter Rechner erreichbar ist. Zudem misst ping die aktuelle Round-Trip-Time zum Zielrechner.

Das Prinzip ist simpel: der anfragende Rechner sendet einen echo-Request aus und startet einen Timer. Der Empfänger eines echo-Requests beantwortet diesen schnellstmöglich mit einem echo-Response. Bei Erhalt der Antwort stoppt der anfragende Rechner seinen Timer.

Diese Anwendung dient der Netzadministration: man kann schnell ermitteln, ob Rechner erreichbar sind und welche Verzögerung die Netze aktuell erzeugen.

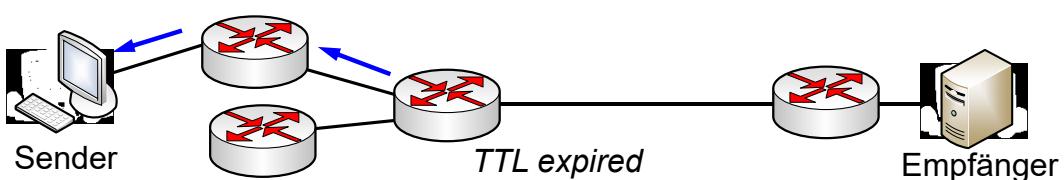
Beispiel: ping von Berlin nach Aachen

```
$ ping -c 10 www.rwth-aachen.de
PING www.rwth-aachen.de (137.226.107.63) 56(84) bytes of data.
64 bytes from www.vr.cms.rwth-aachen.de (137.226.107.63): icmp_seq=1 ttl=117 time=12.1 ms
64 bytes from www.vr.cms.rwth-aachen.de (137.226.107.63): icmp_seq=2 ttl=117 time=12.0 ms
64 bytes from www.vr.cms.rwth-aachen.de (137.226.107.63): icmp_seq=3 ttl=117 time=12.0 ms
64 bytes from www.vr.cms.rwth-aachen.de (137.226.107.63): icmp_seq=4 ttl=117 time=12.1 ms
64 bytes from www.vr.cms.rwth-aachen.de (137.226.107.63): icmp_seq=5 ttl=117 time=12.2 ms
64 bytes from www.vr.cms.rwth-aachen.de (137.226.107.63): icmp_seq=6 ttl=117 time=12.0 ms
64 bytes from www.vr.cms.rwth-aachen.de (137.226.107.63): icmp_seq=7 ttl=117 time=12.0 ms
64 bytes from www.vr.cms.rwth-aachen.de (137.226.107.63): icmp_seq=8 ttl=117 time=12.0 ms
64 bytes from www.vr.cms.rwth-aachen.de (137.226.107.63): icmp_seq=9 ttl=117 time=12.0 ms
64 bytes from www.vr.cms.rwth-aachen.de (137.226.107.63): icmp_seq=10 ttl=117 time=12.0 ms

--- www.rwth-aachen.de ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9010ms
rtt min/avg/max/mdev = 12.011/12.087/12.277/0.102 ms
```

- traceroute: **Ermittlung des Pfads zu einem Zielrechner**

- ▶ Versendung eines IP-Pakets an den Empfänger mit TTL=1
 - Erster Router auf dem Weg verwirft das Paket und sendet ICMP-Nachricht TTL expired zurück an den Sender
- ▶ Versendung des Pakets mit TTL=2
 - Zweiter Router auf dem Weg verwirft das Paket und sendet ICMP-Nachricht TTL expired zurück an den Sender
- ▶ ...



- ▶ Sender ermittelt alle Router auf dem Weg zum Empfänger

traceroute ist eine einfache Anwendung zur Ermittlung von Pfaden im Internet. An sich sollte diese Möglichkeit über die IP Option „Record Route“ bestehen – doch diese Option ist nicht verwendbar, da das Optionsfeld nur 40 Byte umfasst und maximal neun Routeradressen aufnehmen könnte.

Statt dessen macht sich traceroute ICMP zunutze. Der Sender versendet ein IP-Paket an den Empfänger, setzt aber absichtlich die TTL auf 1. Der erste Router auf dem Weg hin zum Empfänger zählt vor der Weiterleitung des Paktes die TTL runter – und verwirft das Paket, da die TTL nun 0 ist. Der Router sendet nun mittels ICMP eine Meldung an den Sender zurück, dass das Paket verworfen wurde. Der Sender gelangt auf diesem Weg an die IP-Adresse des ersten Routers.

Nun wird das Spiel mit TTL=2 wiederholt. Der erste Router leitet das Paket nun weiter, aber der zweite zählt die TTL auf 0 runter und verwirft es. Auch der zweite Router sendet eine entsprechende ICMP-Meldung an den Sender zurück.

Der Sender inkrementiert die TTL so lange und wiederholt den Prozess, bis der Empfänger erreicht wird. Der Sender kennt nun den Pfad, auf dem das Paket hin zum Empfänger geleitet wurde.

Das ist zumindest die Theorie – schaut man genauer in traceroute hinein, stellt man fest, dass man äußerst vorsichtig beurteilen sollte, was für einen Pfad man ermittelt hat.

Traceroute: Sample Output

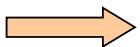
```
$ traceroute www.rwth-aachen.de
traceroute to www.rwth-aachen.de (137.226.107.63), 30 hops max, 60 byte packets
 1  130.149.221.190 (130.149.221.190)  0.127 ms  0.145 ms  0.172 ms
 2  ta-inet.gate.tu-berlin.de (130.149.235.193)  0.898 ms  0.984 ms  1.075 ms
 3  e-n-hft.gate.tu-berlin.de (130.149.126.57)  0.659 ms  0.791 ms  0.997 ms
 4  cr-tub1-te0-0-0-15.x-win.dfn.de (188.1.235.117)  0.929 ms  0.925 ms  0.999 ms
 5  cr-han1-hundredgige0-6-0-0-7.x-win.dfn.de (188.1.144.189)  5.641 ms  5.926 ms  5.923 ms
 6  kr-aac18-4.x-win.dfn.de (188.1.242.98)  12.222 ms  12.354 ms  12.247 ms
 7  fw-xwin-1-vl106.noc.rwth-aachen.de (134.130.3.229)  11.945 ms  11.815 ms  11.878 ms
 8  n7k-ww10-1-vl158.noc.rwth-aachen.de (134.130.3.243)  12.660 ms  12.670 ms  12.092 ms
 9  n7k-ww10-2-et1-1.noc.rwth-aachen.de (137.226.139.42)  12.913 ms n7k-sw23-2-et1-
1.noc.rwth-aachen.de (137.226.38.58)  12.604 ms n7k-ww10-2-et1-1.noc.rwth-aachen.de
(137.226.139.42)  12.860 ms
10  stone-rz-ids-pfo-1.noc.rwth-aachen.de (134.130.9.60)  12.023 ms  12.003 ms  11.963 ms
```

Im Trace sieht man in Zeile 9, dass die drei Antworten von zwei unterschiedlichen Rechnern kommen – zum Load Balancing werden hier einkommende Pakete auf mehrere Rechner verteilt (z.B. zur Prüfung, Firewalls), so dass nicht alle Pakete exakt den gleichen Weg nehmen. Genauso kann es sein, dass aufgrund es Routings schon im Kernnetz Pakete auf unterschiedlichen Wegen weitergeleitet werden, so dass man einen Pfad ermittelt, den es gar nicht gibt.

Auch sind Router nicht verpflichtet, ICMP-Nachrichten zu verarbeiten und zu verschicken. Da die Bearbeitung/Erstellung von ICMP-Nachrichten Rechenzeit braucht, die Router sinnvoller einsetzen könnten, kann es sein, dass sie bei Verwerfen eines Pakets keine ICMP-Nachricht zurücksenden. In dem Fall würde man keine Informationen über den entsprechenden Hop bekommen.

Das „neue“ IP – IPv6

IPv4 (September 1981)



IPv6 (Dezember 1995)

- **Warum ein neues Protokoll, wenn IPv4 gut funktioniert?**

- ▶ IP-Adressen gehen aus (NAT als Hilfslösung)
- ▶ Aufwand für Konfiguration von Rechnern (DHCP als Hilfslösung)
- ▶ Keine Unterstützung verschiedener Dienste (DiffServ als Hilfslösung)
- ▶ Keine Sicherheitsmechanismen wie Verschlüsselung und Authentifizierung (IPSec als Hilfslösung)
- ▶ Keine Unterstützung von Mobilität aufgrund ortsgebundener Adressen (Mobile IP als Hilfslösung)
- ▶ Relativ aufwändige Paketverarbeitung in Routern (Zeitaufwand)
- ▶ Großer Umfang der Routingtabellen (Verwaltungsaufwand)
- ▶ Kein vernünftiger Optionsmechanismus für Weiterentwicklungen

IPv4 hat sich als ein gut funktionierendes Protokoll über Jahre bewährt. Trotzdem hat sich die IETF bereits vor Jahrzehnten dazu entschlossen, IP zu überholen und eine neue Version zu etablieren. Folgende Gründe waren die entscheidenden für das Entwickeln eines neuen Protokolls:

- Anwachsen des Internets: Der große Erfolg des Internets führte zu einer stark wachsenden Benutzerzahl und vor allem auch zu einem starken Anstieg der Netzbelaistung. Die steigenden Benutzerzahlen führen dazu, dass der IP-Adressraum ausgeschöpft ist und die Einrichtung neuer Netze immer aufwändiger wird (NAT), und dass als Folge der nachträglichen Einführung von Subnetzen die Routingtabellen in Kernroutern sehr umfangreich sind, was den Weiterleitungsprozess von Paketen verlangsamt.
- Geänderte Anforderungen an die Datenübertragung: erfordern Vertraulichkeit und Authentifizierung, Mobilitätsunterstützung, usw. IP bietet solche Funktionalitäten nicht, daher wurden im Laufe der Zeit viele Erweiterungen entwickelt, die IPv4 um fehlende Funktionalitäten ergänzen sollen. Aufgrund des beschränkten Optionsmechanismus bietet IPv4 nicht viel Spielraum, solche Erweiterungen effizient zu integrieren.
- Hohe Datenraten: einige Protokollfunktionalitäten (bspw. Fragmentieren in einem Router) reduzieren die Effizienz der Weiterleitung von Paketen. Eine effizientere Paketverarbeitung in Routern ist unabdingbar, um die Kapazität der Netze zu erhöhen.

Dennoch verlief die Einführung von IPv6 nur schleppend. Grund: Zusätze zum alten IP (IPSec, Mobile IP, NAT, ...) erfüllen bereits die mit IPv6 gesetzten Erwartungen, so dass lange Zeit eigentlich kein direkter Handlungsbedarf bestand. Erst langsam verbreitet sich auch IPv6.

- Änderungen bei IPv6

- ▶ *Adressgröße 128 Bit* (8 Gruppen zu je 4 Hexadezimal-Zahlen)
 - Reduktion des Umfangs von Routing-Tabellen durch mehr Struktur in den Adressen
 - Multi-Homing: transparente Verwendung mehrerer Adressen gleichzeitig
 - *Autokonfiguration von Adressen* durch einen Rechner selbst (aber auch noch durch DHCP)
- ▶ *Vereinfachung des Protokolls (und in Folge des Headers)* zur beschleunigten Verarbeitung von IPv6-Paketen im Router
 - *Verbesserter Optionsmechanismus*, feste Headerlänge
 - Keine Checksum
 - *Keine Fragmentierung* im Router mehr, dadurch Entlastung der Router
- ▶ *Markieren von Paketen* für speziellen Verkehr
- ▶ *Zusatzdienste als Optionen*: z.B. Authentifizierung und Vertraulichkeit

- Beispiele für IPv6-Adressen

- ▶ 8 Hexadezimalwerte

- Beispiel: 2a00:8a60:1014:0000:0002:0000:0345:5f23

- ▶ Führende Nullen können weggelassen werden:

- 2a00:8a60:1014:0:2:0:345:5f23

- ▶ An einer Stelle können Nuller-Blöcke weggelassen werden

- 2a00:8a60:1014::2:0:345:5f23

- ▶ Integration des IPv4-Adresraums als ::ffff:ip4-address

- ::ffff:137.226.12.21 oder ::ffff:89e2:c15

- ▶ Hierarchische Struktur wie in IPv4, mit Präfixschreibweise

- 2a00:8a60:1014::/64

IPv6 – Adressen

- IPv6-Adressen können sein...

- ▶ *Unicast-Adressen* einzelner Hosts
- ▶ *Multicast-Adressen*: Adressieren alle Mitglieder einer Gruppe von Hosts
- ▶ *Anycast-Adressen*: Adressieren ein Mitglied einer Gruppe von Hosts
- ▶ Typ der Adresse wird durch das Präfix bestimmt:
 - Reserviert: 00000000 = **0000::/8** (= ::/8)
 - Unspezifizierte Adresse 00...00 (128 Bit) = ::/128
 - Loopback-Adresse: 00...01 (128 Bit) = ::1/128
 - IPv4-Adresse: 0....01...1 = ::ffff:0:0/96
 - Multicast-Adresse: 11111111 = **ff00::/8**
 - Broadcast ff01::1, ff02::1
 - Link-local Unicast-Adresse 1111111010 = **fe80::/10**
 - Unique local Unicast-Adresse 11111110 = **fc00::/7**
 - Global Unicast-Adresse alles andere

Wie auch bei IPv4 gibt es Unicast-Adressen, die einzelne Hosts eindeutig identifizieren. Broadcast-Adressen zur Versendung von Daten an alle Hosts in einem Netz gibt es der Terminologie nach nicht mehr – bei IPv6 wird dies auch als Multicasting bezeichnet, die Broadcast-Adresse eines Netzes ist nun eine spezielle Multicast-Adresse.

Anycast-Adressen sind ein neues Konzept, welches es bei IPv4 nicht gab: wie eine Multicast-Adresse adressiert man über eine Anycast-Adresse eine Gruppe von Hosts. Während allerdings Daten bei Multicasting an alle Gruppenmitglieder zugestellt werden, werden sie beim Anycasting nur an genau ein Gruppenmitglied versendet. Dies soll es ermöglichen, einen Anwendungsdienst (z.B. Suchmaschine, Streaming-Server) zu replizieren und mehrmals anzubieten (Beispiel Mirror-Server). Ruft ein Client den Anwendungsdienst auf, wird er nach einem bestimmten Prinzip (welches im Netz implementiert sein muss, z.B. als Teil des Routing-Algorithmus) mit einem Server verbunden. Dies wäre z.B. bei Video-on-Demand sinnvoll, um jeden Client mit dem geographisch nächsten Host zu verbinden, auf dem der Anwendungsdienst läuft (Reduktion der Latenz). Für Anycasting wird kein eigener Adressbereich reserviert, man verwendet Unicast-Adressen. Dadurch bleibt für einen Dienstnutzer verborgen, ob hinter einer IP-Adresse ein Empfänger oder eine Empfängergruppe steckt.

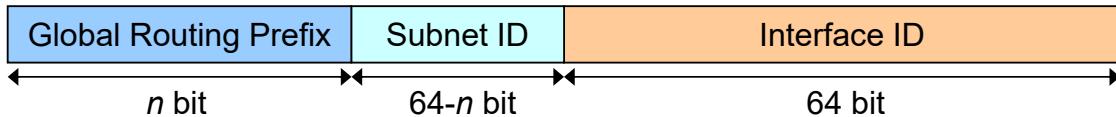
Dieses Prinzip findet auch bereits in IPv4-Netzen Einsatz, meist aber auf höheren Schichten implementiert.

Eine spezielle Adresse ist die Loopback-Adresse. Während in IPv4 ein ganzes Klasse-A-Netz dafür reserviert wurde, ohne Wissen der eigenen IP-Adresse Anwendungen auf dem gleichen Host ansprechen zu können, ist dies bei IPv6 nur noch eine einzige Adresse. (Da die Praxis gezeigt hat, dass auch in IPv4 nur eine einzige Adresse verwendet wurde.)

Um während einer Übergangszeit IPv4 und IPv6 parallel betreiben zu können, wurde auch vorgesehen, ein Subnetz in IPv6 vorzuhalten, welches den gesamten IPv4-Adressbereich integriert, so dass auch nur-IPv4-fähige Hosts von einem IPv6-fähigen Host aus angesprochen werden können.

Adressbeispiel

- **Global Unicast: dreigeteilte Hierarchie**



- ▶ Globales Routing-Präfix zur Reduktion des Umfangs von Routing-Tabellen (z.B. Organisation, geographischer Identifier)
- ▶ Subnet-ID als Adresse eines bestimmten Netzes
- ▶ Interface-ID als eindeutige Adresse eines Rechners, automatisch generiert z.B. aus der MAC-Adresse (*Autokonfiguration*):
 - MAC-Adresse 00:1D:72:8E:74:6C (48 bit)
 - Interface ID: 00:1D:72:**FF:FF**:8E:74:6C (64 bit)
 - (bzw. 001D:72FF:FF8E:746C zur Darstellung in der IP-Adresse)

- ▶ COMSYS-Webserver: 2a00:8a60:1014::142

SYS | UNIVERSITY

Die Entwicklung der Global-Unicast-Adressen wurde direkt mit mehr Hierarchiestufen vorgenommen als bei IPv4. Jedes Netz bekommt nach wie vor eine Netz-ID sowie eine Host-ID (Interface-ID). Beide Teile sind fix 64 Bit lang.

Die Netz-ID ist weiter strukturiert, um den Umfang von Routingtabellen zu reduzieren. Die ersten n (bis zu 32) Bit identifizieren eine Organisation (z.B. die RWTH Aachen) und werden von regionalen Vergabestellen (Regional Internet Registry) koordiniert vergeben. Geht die Vergabestelle geschickt vor, lassen sich hinterher eventuell sogar weniger als n Bit nutzen, um geographische Bereiche zu identifizieren.

Die Subnet-ID ist nur innerhalb der Organisation von Interesse und dient dazu, den zugewiesenen Adressraum auf mehrere unabhängige Einheiten zu verteilen (z.B. einzelne Lehrstühle an der RWTH). Die Aufteilung ist aber ganz dem Netzbereiber überlassen.

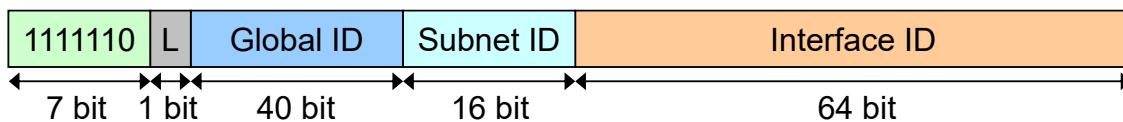
Die Interface-ID identifiziert eindeutig einen Rechner im Netz. Adressen können auch bei IPv6 manuell konfiguriert oder über DHCP bezogen werden, es gibt aber durch die sogenannte "Autokonfiguration", um eine einfachere Zuweisung von Adressen zu ermöglichen: ein Host generiert z.B. basierend auf seiner MAC-Adresse einen 64-Bit-Identifier. Auf der Folie dargestellt ist ein sehr einfaches Beispiel. In der Praxis sollten kryptographische Funktionen zur Generierung verwendet werden, damit niemand aus der IP-Adresse auf die physikalische Adresse eines Hosts schließen kann.

Adressbeispiel

- **Link-local Unicast:**

- ▶ Jeder Host hat zumindest eine Link-local-Adresse
- ▶ Schema: `fe80::/64 + interface ID`
- ▶ Z.B. für Austausch von Kontrollnachrichten mit dem lokalen Router

- **Unique local Unicast**



- ▶ Kommunikation zwischen Subnetzen
 - Global eindeutig, aber typischerweise nicht geroutet
 - L = 1: lokal zugewiesene, zufällige Global ID /
 - L = 0: global zugewiesene, eindeutige Global ID

Link-local-Unicast-Adressen sind nur innerhalb eines Netzes gültig. Sie werden IPv6-intern verwendet, z.B. zur Autokonfiguration.

Als Ersatz privater Adressen gibt es zudem noch die Unique-local-Unicast-Adressen. Hiermit kann man IP-basierte Netze erstellen, die keine Verbindung zum Internet benötigen. In diesen Adressen kann man durch die Subnet-ID weitere Strukturen schaffen, also auch mehrere Subnetze schaffen, die miteinander verbunden sind und private Adressen verwenden. Im Internet sollen diese Adressen allerdings nicht geroutet werden.

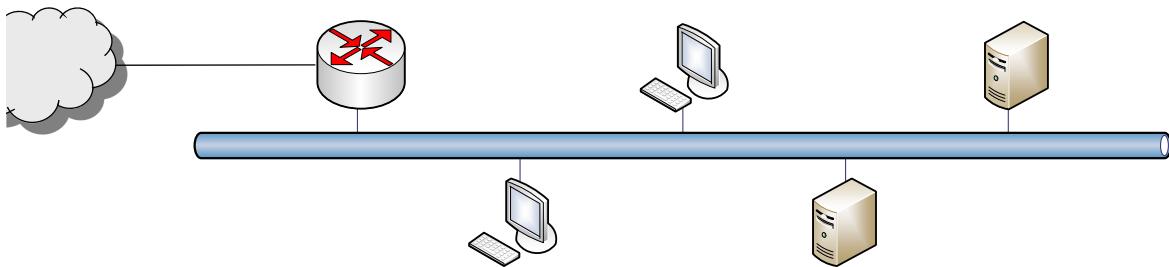
Die vorgeschaltete Global ID soll erreichen, dass auch all diese Netze unterschiedliche Adressen verwenden. Momentan ist aber immer L=1 gesetzt.

Es ist vorgesehen, dass ein Host gleichzeitig mindestens eine Link-local- und eine Global-Unicast-Adresse besitzt. Zudem kann ein Host noch auf mehrere Multicast-Adressen hören.

Autokonfiguration

- Ablauf der Autokonfiguration

1. Generiere Link-Local-Adresse



MAC-Adresse 00:1D:72:8E:74:6C
→ Link-Local-Adresse fe80::001D:72FF:FF8E:746C

In IPv6 gibt es drei Möglichkeiten, einem Host eine IP-Adresse zuzuweisen:
manuelle Konfiguration, Einsatz eines DHCPv6-Servers zur Adressvergabe,
Autokonfiguration.

Die Autokonfiguration eines Hosts umfasst drei Schritte

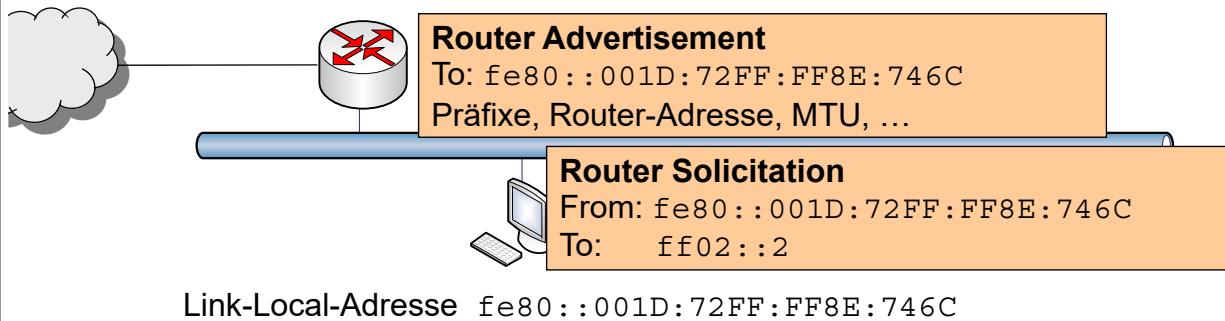
1. Zuweisung einer eigenen Link-local-Unicast-Adresse durch Berechnung der Interface-ID z.B. aus der eigenen MAC-Adresse
2. Ermittlung der Netzkonfiguration: Routeradressen, Netzwerkpräfixe, ...
3. Duplicate Address Detection zur Sicherstellung, dass die gewählte Adresse im Netz noch nicht verwendet wird (MAC-Adressen müssen nicht notwendigerweise eindeutig sein)

Autokonfiguration

• Ablauf der Autokonfiguration

2. Router Discovery

- „Router Solicitation“-Nachricht per Multicast an `ff02::2` versenden
- Router antworten mit „Router Advertisement“
 - Lokale Netzwerkpräfixe, Router-Adressen, MTU



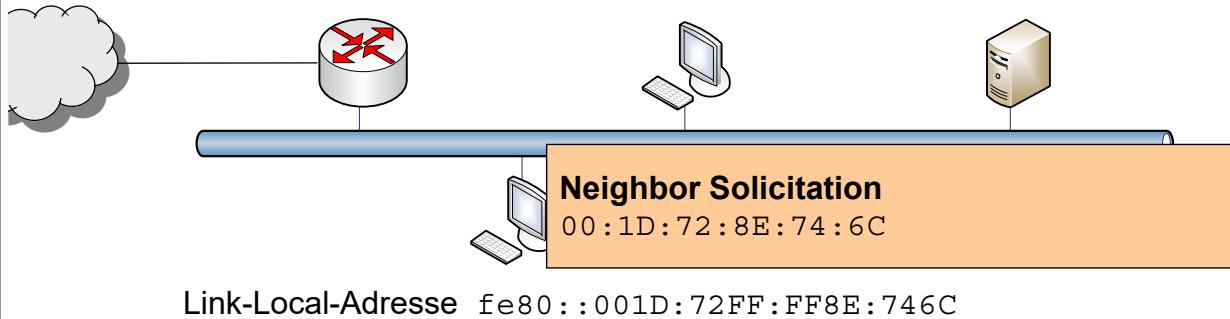
Schritte 2 und 3 werden mittels ICMPv6 ausgeführt; der speziell für diese Zwecke ausgelegte Teil von ICMPv6 wird als Neighbor Discovery Protocol bezeichnet und dient zum Auffinden anderer Rechner im eigenen Netz – sowohl andere Hosts als auch Router. Die wesentlichen Nachrichtentypen sind Router Solicitation, Router Advertisements, Neighbor Solicitation und Neighbor Advertisements.

Router Discovery ist notwendig, da eine Link-local-Unicast-Adresse keine Kommunikation mit Hosts außerhalb des eigenen Netzes erlaubt. Hierzu benötigt man das eigene Netzpräfix sowie Routeradressen (die man bei IPv4 zusammen mit der IP-Adresse über DHCP bezieht). Dazu wird eine Nachricht vom Typ „Router Solicitation“ an die Multicast-Adresse gesendet, auf der alle Router im Netz lauschen. Der Router wird eine Antwort schicken, in der er alle konfigurierten Netzwerkpräfixe, seine eigene Adresse, die lokale MTU und eine Gültigkeitsdauer der Angaben zurücksendet. Der Host kann sich nun eine Global-Unicast-Adresse generieren und weltweit kommunizieren.

- Ablauf der Autokonfiguration

- 3. Duplicate Address Detection

- „Neighbor Solicitation“ zur Abfrage der MAC-Adresse von Nachbarn
 - Sende eigene MAC-Adresse als Payload mit
 - Hosts mit der MAC-Adresse antworten mit „Neighbor Advertisement“
 - Keine Antwort = eindeutige Adresse gefunden



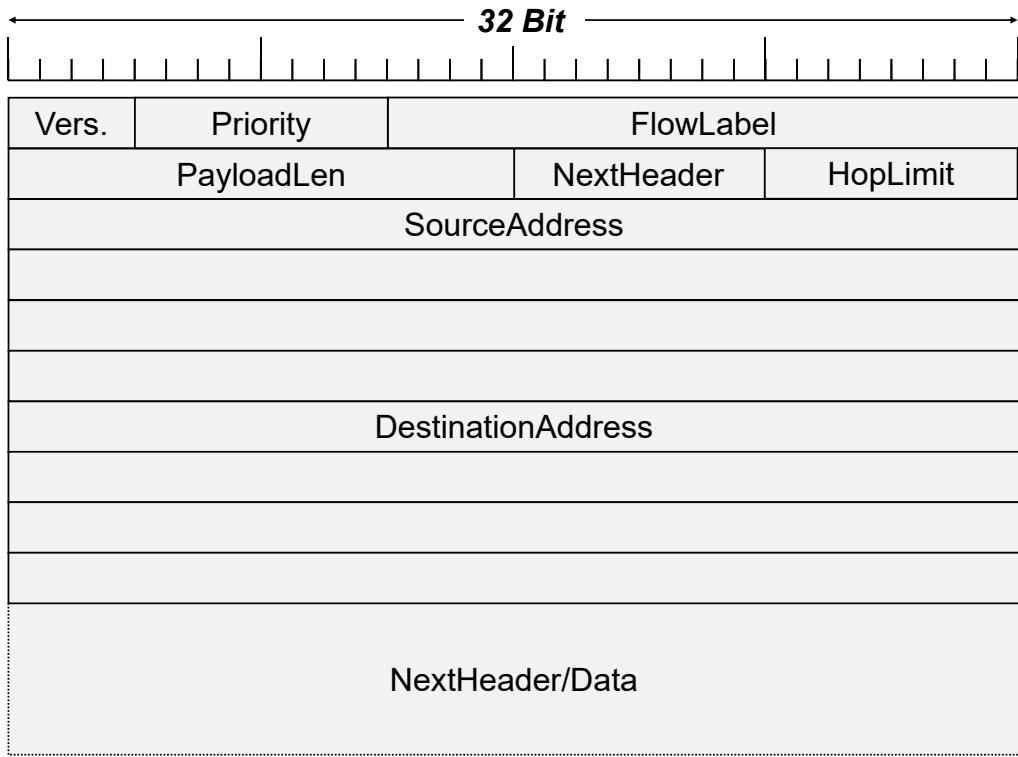
Dabei bleibt allerdings ein Problem: es ist nicht garantiert, dass die selbst generierte Interface-ID nicht bereits in Benutzung ist. Daher ist in einem letzten Schritt eine Prüfung notwendig, ob man eine bereits in Verwendung befindliche Adresse gewählt hat. Die Duplicate Address Detection ist sehr simpel: führe einen Multicast an alle bereits konfigurierten Hosts im eigenen Netz durch. Dabei wird die eigene MAC-Adresse mitgesendet. Hosts, die die gleiche MAC-Adresse verwenden, werden dazu aufgefordert, zu antworten. Kommt keine Antwort, kann man davon ausgehen, dass die autogenerierte Interface-ID eindeutig sein wird. Der zur Neighbor-Discovery eingesetzte Nachrichtentyp „Neighbor Solicitation“ ist aber auch für andere Zwecke einsetzbar; er ist generell dazu gedacht, die MAC-Adresse der Nachbarn abzufragen, um ein Bild des eigenen Netzes zu bekommen. Dadurch wird auch ARP ersetzt.

Path MTU Discovery

- **Vermeidung von Fragmentierung**

- ▶ *Erkunde minimale MTU* auf einem Pfad (PMTU)
 - Sende Pakete orientiert an lokaler MTU
 - Erster Router, der geringere MTU auf nächstem Hop hat, verwirft das Paket und sendet ICMP-Meldung „Packet too big“ zurück
 - Sender passt PMTU an und cached Wert für spätere Verwendung
- ▶ Zweck: passe Größe von Paketen an minimale MTU an, um Fragmentierung zu vermeiden
- ▶ Kleinstes unterstützte MTU: 1280 Byte
 - Keine weitere Path MTU Discovery falls Wert geringer ist
 - IPv6 geht immer von mindestens 1280 Byte MTU aus

IPv6 Haupt-Header



- **Version:** IP-Versionsnummer, Wert: 6
- **Priority / Traffic Class:** ursprünglich 4 Bit für Priorität (zur Definition unterschiedlicher Klassen von Übertragungsarten, die im Router unterschiedlich behandelt werden): z.B.: 1 – News, 4 – FTP, 6 – Telnet, 8 bis 15 – Echtzeitverkehr.
Mittlerweile auf 8 Bit erweitert und als „Traffic Class“ bezeichnet.
- **FlowLabel:** alle Pakete eines Datenstroms können das gleiche Label bekommen und von Routern anhand des Labels weitergeleitet werden (differenzierte Behandlung von Datenströmen)
- **PayloadLen:** Paketlänge nach dem 40-Byte-Header
- **NextHeader:** 8-Bit-Selektor. Gibt den Typ des folgenden Erweiterungs-Headers an (oder das verwendete Transportprotokoll, falls keine Erweiterungsheader folgen)
- **HopLimit:** Wird bei jedem Knoten um 1 dekrementiert. Bei Null wird das Paket verworfen
- **SourceAddress:** Die IPv6-Adresse des Senders des Pakets
- **DestinationAddress:** Die IPv6-Adresse des Empfängers (nicht unbedingt das endgültige Ziel, wenn es einen Routing-Erweiterungsheader gibt, z.B. zur Unterstützung von Mobilität).
- **NextHeader/Data:** Optionsmechanismus – hängt Optionen in Form beliebig vieler weiterer Header zwischen den IPv6-Header und den eigentlichen Payload (der auch mit einem Header, meist eines Transportschichtprotokolls beginnen wird).

Erweiterungs-Header

- **6 Klassen von Erweiterungs-Headern (Optionen):**

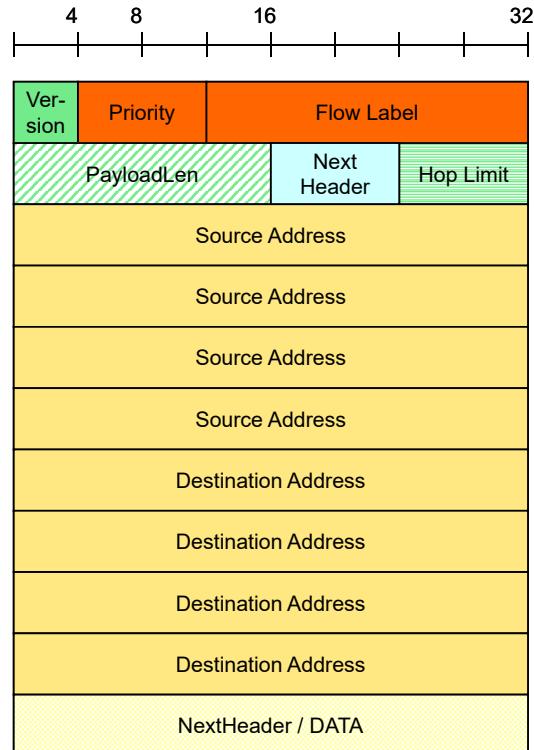
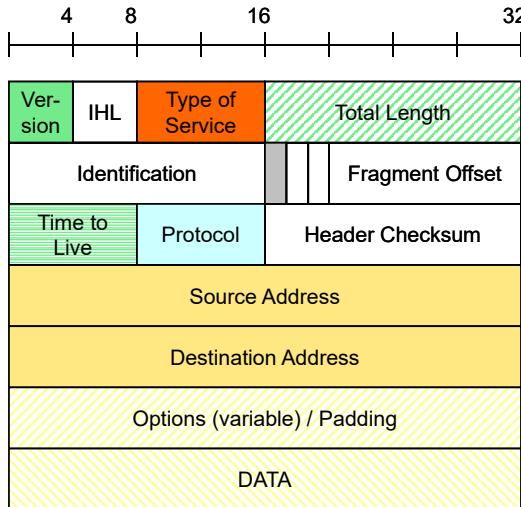
- ▶ Hop-by-Hop (Informationen für Teilstrecken)
 - Alle Router müssen dieses Feld prüfen
 - Z.B. Unterstützung von Jumbogrammen, d.h. Paketen mit Überlänge (Hierbei wird eine Längenangabe eingetragen)
- ▶ Zieloptionen (Zusatzinformationen für das Ziel)
- ▶ Routing (Definition einer vollen oder teilweise festgelegten Route)
- ▶ Fragmentierung (Verwaltung von Fragmenten)
 - Verwendung, wenn Sender Fragmentierung vornehmen muss
- ▶ Authentifikation (des Senders)
- ▶ Verschlüsselung (Informationen zur Verschlüsselung der Daten)
- ▶ Zieloptionen (Zusatzinformationen für das Ziel)

Die Auslagerung von Optionen in Erweiterungsheader ermöglicht es, beliebig viele Optionen einfach zu einem IP-Paket hinzuzufügen. Generell lassen sich alle Optionen in eine von sechs Klassen einordnen.

- Hop-by-Hop: jeder Router, der das Paket weiterleitet, muss die Inhalte dieser Erweiterungsheader prüfen. Eine Möglichkeit zum Einsatz wäre die effizientere Kommunikation zwischen Rechnern in kontrollierbarem Umfeld – so könnte man in einem Rechencluster zur Verringerung des Overheads durch Header sogenannte Jumbogramme versenden. Durch solch einen Erweiterungsheader werden alle Router angewiesen, die Längenangabe im Header zu ignorieren und stattdessen den folgenden Wert zu verwenden
- Fragmentierung: Auslagerung der Fragmentierungsinformationen in einen Erweiterungsheader, da Fragmentierung bei modernen Netzen als relativ selten notwendig eingeschätzt wurde. Durch Path MTU Discovery kann der Sender schon eine passende Fragmentierung vornehmen.
- Authentifizierung und Verschlüsselung dienen dazu, Informationen zur Authentizität des Absenders und zur Verschlüsselung des Paketinhaltes zu übermitteln. Beide Erweiterungsheader wurden als IPSec auch als Aufsätze zu IPv4 implementiert und werden in Kapitel 6 behandelt.
- Routing-Header können z.B. ein Source-Routing ermöglichen. Zusammen mit den Zieloptionen realisiert Mobile IPv6 eine Mobilitätsunterstützung, bei der man seine IP-Adresse behalten kann und unter dieser Adresse auch erreichbar ist, wenn man das Netz gewechselt hat (siehe Vorlesung “Mobile Internet Technology”)

Die Optionen für das Ziel sind zwei Mal aufgelistet, da die Reihenfolge auf der Folie auch diejenige ist, in der die Erweiterungsheader aneinandergehängt werden sollten; je nach Zielinformation bieten sich zwei Stellen an, an denen sie eingebettet werden.

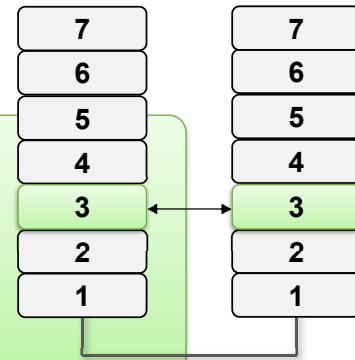
IPv4 vs. IPv6: Header



Der IPv6-Header ist zwar länger, doch dies liegt nur an den längeren Adressen. Ansonsten ist er „aufgeräumt worden“, was die Paketverarbeitung in den Routern beschleunigt.

Kapitel 4: Vermittlungsschicht

- **Vermittlungsverfahren**
 - ▶ Leitungsvermittlung, Speicher-/Paketvermittlung
 - ▶ Beispiel ATM
- **Die Vermittlungsschicht im Internet – IP**
 - ▶ IPv4: Adressen, Subnetze, CIDR, NAT
 - ▶ IPv4-Header
 - ▶ Hilfsprotokolle: ARP, DHCP, ICMP
 - ▶ IPv4 vs. IPv6
- **Wegewahlverfahren (Routing)**
 - ▶ Hierarchien, einfache Verfahren
 - ▶ Distance Vector
 - ▶ Link State



Routing

• Paketweiterleitung

- ▶ Verbindungslos: unabhängige Wegewahlentscheidung für jedes Paket (z.B. IP)
- ▶ Verbindungsorientiert: Wegewahlentscheidung nur bei Verbindungsherstellung (z.B. ATM, Telefonnetz)
 - Alle innerhalb einer Verbindung ausgetauschten Pakete folgen dem bei Verbindungsherstellung festgelegten Weg

• Routing

- ▶ Ermittlung von Wegen für das Forwarding
- ▶ Beteiligte Komponenten beim Routing
 - *Routing-Protokoll*: Protokoll zum Austausch von Routing-Informationen
 - *Routing-Algorithmen*: Ermittlung von günstigsten Wegen im Netz, Erzeugung von Einträgen für Routing-Tabelle
 - *Routing-Tabellen*: Halten der Wegdaten



Die Paketweiterleitung selbst wird als “Forwarding” bezeichnet. “Routing” ist der Prozess der Wegeermittlung, so dass es Regeln für die Paketweiterleitung gibt.

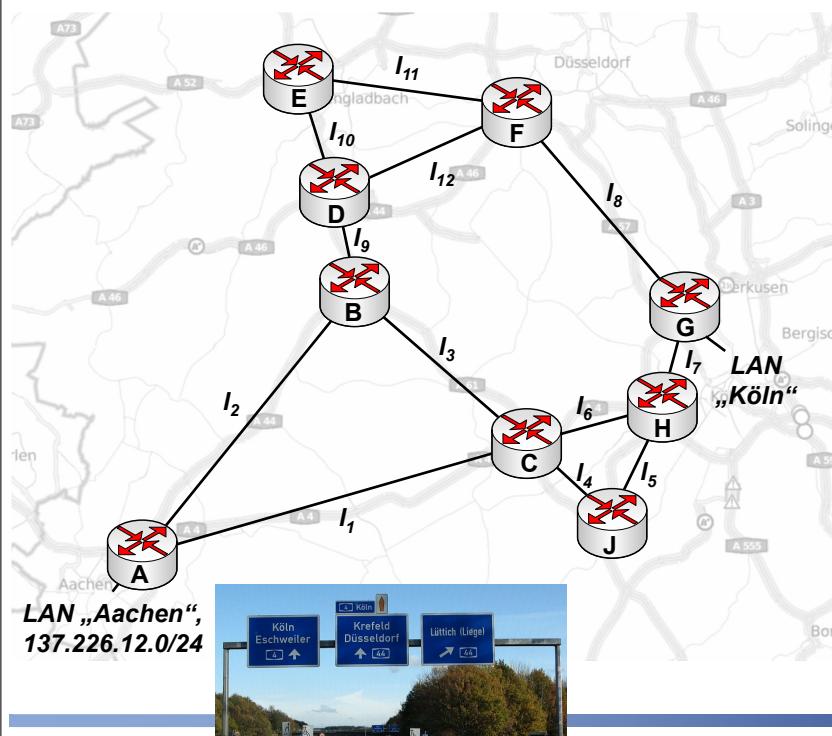
Umgangssprachlich wird auch schon mal der Weiterleitungsprozess als Routing bezeichnet, aber konzeptionell sollte man zwischen den Begriffen Routing und Weiterleitung sauber trennen.

Um Daten weiterleiten zu können, muss ein Rechner oder ein Router über eine Routing-Tabelle verfügen, die die notwendigen Weiterleitungsinformationen enthält. Diese kann statisch konfiguriert werden, was z.B. auf den Endrechnern und auf vielen Routern, die LANs anschließen, passiert. Endrechner sind üblicherweise nur durch einen einzigen Router mit dem Rest des Internets gekoppelt, daher gibt es nur einen einzigen Weg, den Daten an entfernte Rechner nehmen können. Dieser ändert sich nie, so dass die entsprechenden Informationen statisch eingepflegt werden können. Gleicher gilt für viele Router, die lokale Netze anschließen – auch hier gibt es oft nur zwei Anschlüsse: den ins lokalen Netz und einen hin zu einem Router auf höherer Ebene.

Sobald man den lokalen Bereich allerdings verlässt, existieren meist mehrere mögliche Pfade hin zum Ziel – hier benötigt man nun Routing-Protokolle, über die sich Router über den aktuellen Netzzustand austauschen können, um auf Basis der so ermittelten Informationen durch einen Routing-Algorithmus kürzeste Wege durchs Netz zu berechnen und daraus Einträge für die Routing-Tabelle zu berechnen. Durch regelmäßigen Informationsaustausch mit anderen Routern kann so auch dynamisch auf Änderungen im Netz reagiert werden.

Routing-Protokolle und -Algorithmen werden im Folgenden zusammen auch als Routing-Verfahren bezeichnet.

Prinzip einer Routing-Tabelle



Routing-Tabelle für C:

nach	Next Hop	Link
A	A	I ₁
B	B	I ₃
D	B	I ₃
E	B	I ₃
F	B	I ₃
G	H	I ₆
H	H	I ₆
J	J	I ₄

„nach A“ in tatsächlicher
Routing-Tabelle z.B.
„137.226.12.0/24“

Eine Routing-Tabelle haben wir bereits oben gesehen – sie enthält eine Menge von Weiterleitungsregeln der Form (X,Y): „Pakete an einen Rechner in Zielnetz X müssen an Router Y weitergeleitet werden“. Ist der Zielrechner im gleichen Netz wie der sendende Knoten, hat man einen Eintrag der Form (Z,*): „Der Sender sitzt selbst in Zielnetz Z. Mittels z.B. ARP muss die MAC-Adresse bestimmt werden, damit die Daten über Schicht 2 zugestellt werden können.“

Im lokalen Bereich kann man diese Regeln manuell konfigurieren. In einem einfachen lokalen Netz braucht man nur zwei Regeln – eine der Form (Z,*), damit alle Rechner im lokalen Bereich zugestellt werden können und eine der Form (X, Z_Router), so dass alle Pakete für Rechner außerhalb des eigenen Netzes an den lokalen Router weitergeleitet werden.

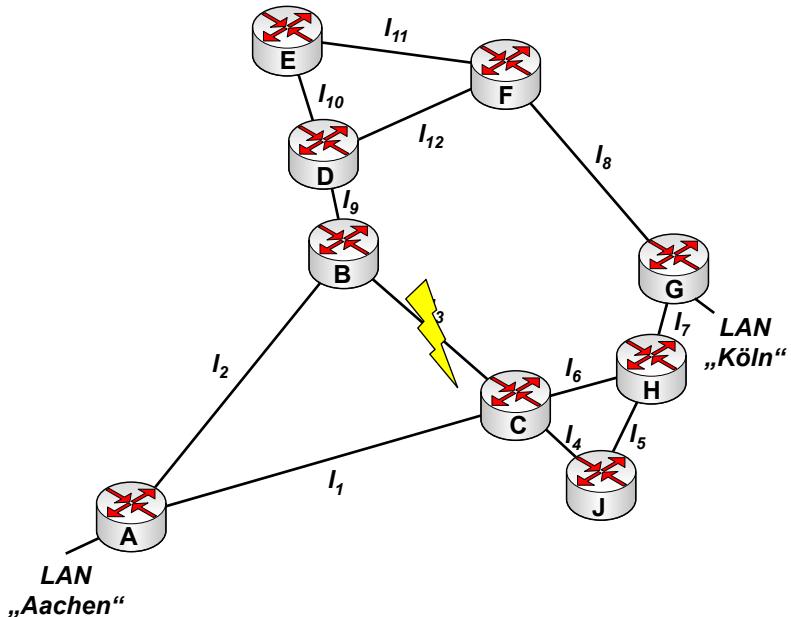
Ein Kernrouter in einem Backbone benötigt eine umfangreichere Tabelle – er muss im Extremfall einen Eintrag für jedes Subnetz im Internet besitzen. Auch wenn viele Einträge mittels CIDR zusammengefasst werden können, hat man immer noch sehr umfangreiche Routingtabellen.

Als Analogie kann man ein Autobahnkreuz als Router betrachten und die Beschilderung als Routing-Tabelle, die den Paketen (Autos) den Weg weist.

Dynamische Anpassung der Tabelle: Ausfall eines Links

Routing-Tabelle für C:

nach	Next Hop	Link
A	A	I_1
B	A	I_1
D	A	I_1
E	A	I_1
F	H	I_6
G	H	I_6
H	H	I_6
J	J	I_4



In lokalen Netzen sind Routen normalerweise statisch konfiguriert, da es kaum alternative Pfade gibt. In Backbones hingegen müssen Einträge in Routing-Tabellen dynamisch und automatisiert konfigurierbar sein, um schnell auf Änderungen im Netzzustand reagieren zu können. Dazu werden Routing-Verfahren benötigt.

Routing: Kosten

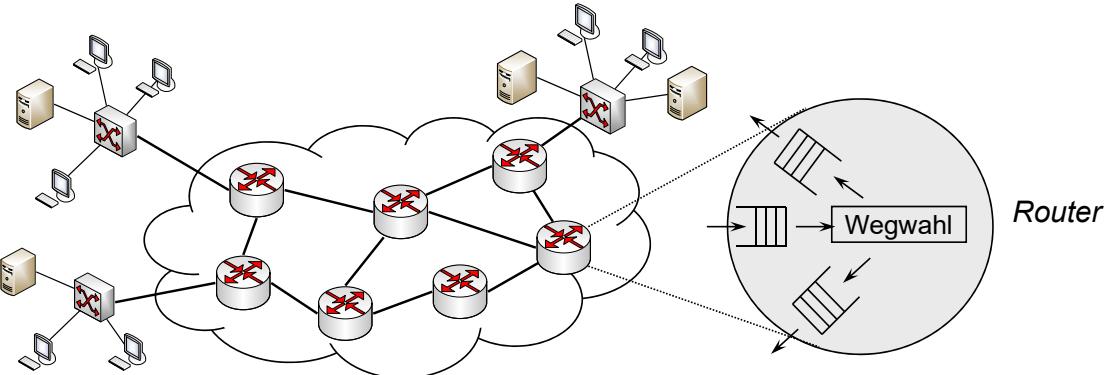
- **Kosten** beim Routing:

► Wegwahlentscheidung für ein eingehendes Paket, orientiert an:

- Niedriger mittlerer Paketverzögerung
- Hohem Netzdurchsatz
- ...



► Generell: ermittle Weg mit geringsten "Kosten"

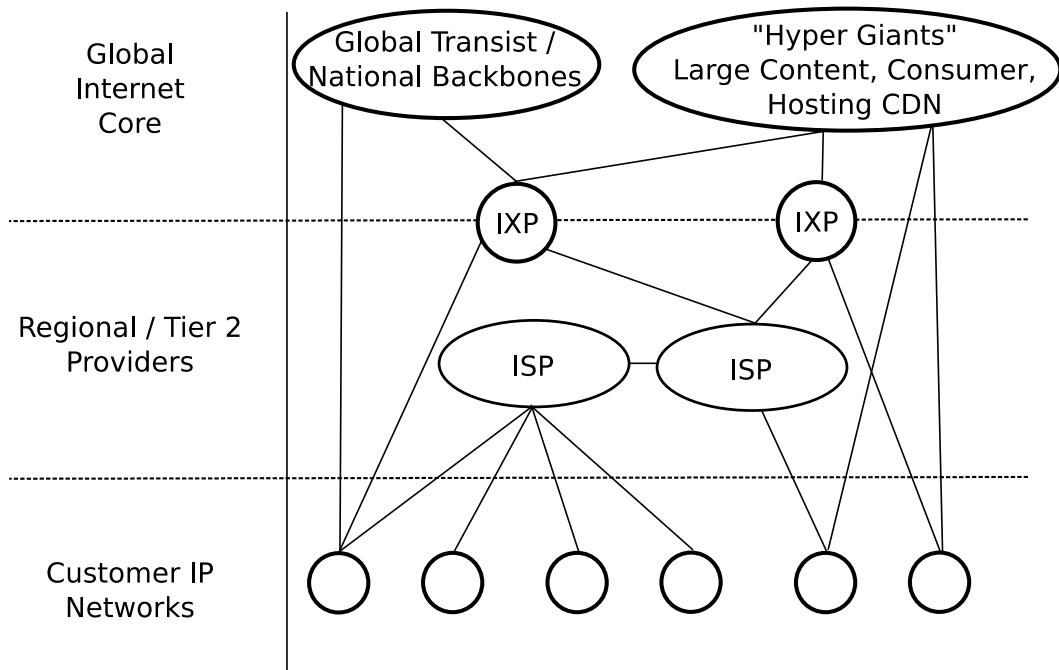


Generell wird beim Erstellen der Einträge der Routing-Tabelle eine Kostenminimierung angestrebt: gibt es mehrere Wege hin zum Ziel, wird der günstigste genommen.

Was genau "Kosten" sind, ist nicht festgelegt. Es können tatsächliche monetäre Kosten für die Verwendung einer Leitung sein, aber auch die Anzahl der zu passierenden Router bis zum Ziel (Verarbeitungskosten), die erwartete Übertragungsverzögerung über einen Weg, der erzielbare Durchsatz, die Fehlerrate, die Anzahl der aktuell auf Übertragung wartenden Pakete ... - hier bleibt Freiraum, eine passende Metrik für die Kosten zu wählen.

Aufbau des Internet

• Hierarchische Struktur



Das Internet hat eine hierarchische Struktur mit traditionell drei Ebenen: Tier-1 bis Tier-3.

Tier-1 sind nationale Backbones (z.B. deutschlandweiter Telekom-Backbone), globale Transit-Netze (die die Backbones verbinden) und sogenannte "Hyper Giants", z.B. die Netze großer Content Distribution Networks (CDNs). Diese können sowohl untereinander verbunden sein, als auch mit Tier-2-Netzen. Die Verbindungsstellen werden Internet Exchange Point (IXP) genannt.

Tier-2 sind Netze regionaler Anbieter. (Internet Service Provider, ISP)

Tier-3 sind Netze lokaler Anbieter, z.B. NetAachen, Campus, Firma.

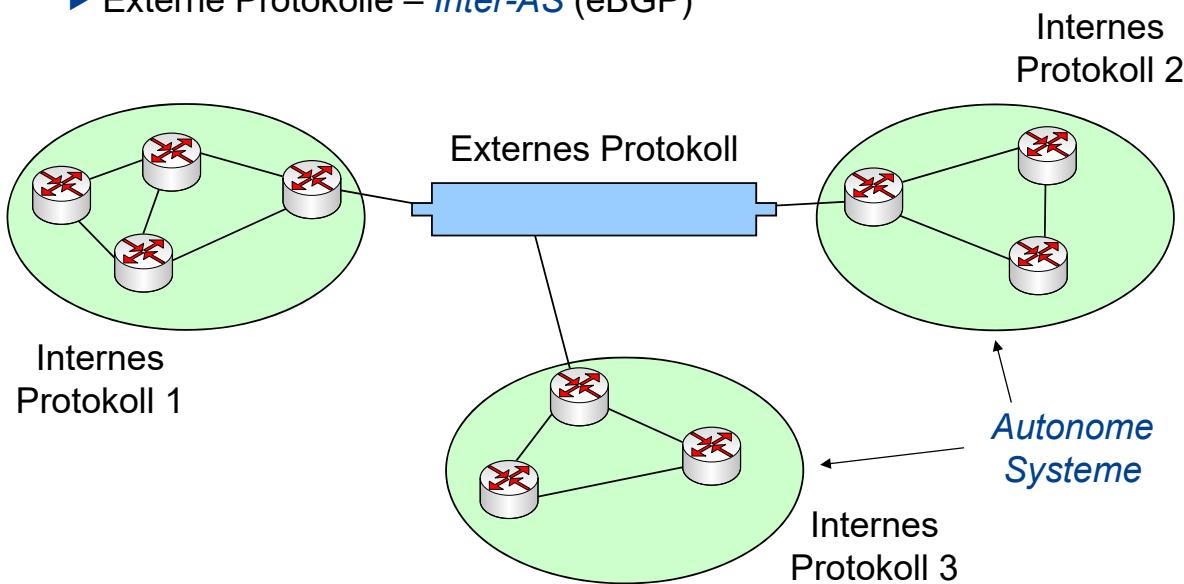
Während lange Zeit eine klare Trennung der Ebenen und festgelegte Verbindungsstrukturen bestanden, hat in den letzten (ca. 10) Jahren ein Wandel stattgefunden. Die klare Hierarchie wurde etwas aufgeweicht, um den neuen Strukturen des Internets gerecht werden zu können.

Das Internet besteht somit aus unterschiedlichen Bereichen, die unter lokaler Kontrolle einer Organisation stehen. Um zu vermeiden, dass alle Router in all diesen Bereichen das gleiche Routing-Protokoll verwenden und mit allen anderen Routern weltweit Informationen austauschen müssen (Skalierbarkeit!), wurde das Prinzip der *autonomen Systeme* eingeführt – ein AS ist ein administrativer Bereich des Internets, der unter der Kontrolle einer Organisation steht und innerhalb dessen alle Router das gleiche Routing-Protokoll (und den gleichen Routing-Algorithmus) verwenden. Diese Aufteilung reduziert auch den Kommunikationsaufwand der Router untereinander: Router innerhalb eines AS müssen nur mit anderen Routern des gleichen AS Informationen austauschen; nur die Router, die das eigene AS mit anderen ASs verbinden, müssen auch mit diesen Informationen austauschen.

Routing im Internet

- **Zwei Klassen von Routing-Protokollen**

- ▶ Interne Protokolle – *Intra-AS* (OSPF, RIP, iBGP)
- ▶ Externe Protokolle – *Inter-AS* (eBGP)



Man unterscheidet generell zwischen internen Protokollen (die innerhalb autonomer Systeme eingesetzt werden) und externen Protokollen (die autonome Systeme koppeln).

An Intra-AS- und Inter-AS-Protokolle und -Algorithmen werden unterschiedliche Anforderungen gestellt. Innerhalb eines AS können die genauen Anforderungen dabei noch von der Größe und dem Aufbau des AS abhängen. Daher gibt es eine Vielzahl unterschiedlicher interner Protokolle, während es nur ein externes Protokoll gibt (da alle ASs auf einheitliche Art gekoppelt werden müssen).

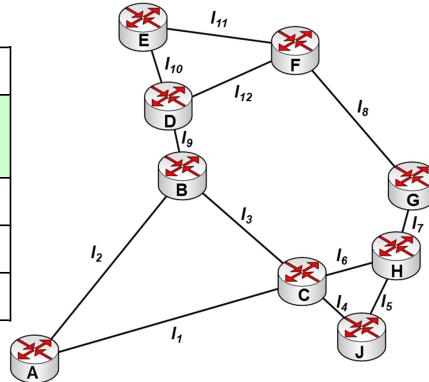
Bei der Entwicklung eines Routing-Verfahrens (sowohl intern als auch extern) müssen unterschiedliche Aspekte berücksichtigt werden:

- Dynamik: soll sich die Wegwahl adaptive an den Netzzustand anpassen? Wenn ja, wie schnell kann/muss dies geschehen?
- Zentralisation: wird die Wegwahl in einem dynamischen Autonomen System zentral getroffen oder erfolgt die Ermittlung günstiger Wege verteilt durch die Router selbst?
- Interaktion: wird die Wegwahl in einem dynamischen, dezentral organisierten Autonomen System isoliert durch jeden Router getroffen oder interagieren die Router zur Ermittlung von Wegen?

Statisches Routing (Static Routing, Directory Routing)

- Nicht adaptiv, einfach, viel benutzt
 - ▶ Routing-Tabelle enthält eine Zeile für jeden Zielknoten
 - ▶ Jede Zeile: n Einträge (beste, zweitbeste, etc. Übertragungsleitung zum Ziel) zusammen mit einer relativen Gewichtung
 - ▶ Vor Weiterleitung: ziehe Zufallszahl und wähle entsprechenden Eintrag
- Tabelle in Knoten C:

Ziel	1. Wahl		2. Wahl		3. Wahl	
	Next Hop	Gew.	Next Hop	Gew.	Next Hop	Gew.
D	B	0,6	A	0,3	H	0,1
G	H	0,7	J	0,25	B	0,05
...



Ablauf des statischen Routings: jeder Knoten unterhält eine Tabelle mit einer Zeile für jeden möglichen Zielknoten. Der komplette Inhalt dieser Tabellen wird statisch in jedem Rechner konfiguriert. Eine Zeile enthält n Einträge, welche die beste, zweitbeste, etc. Übertragungsleitung für dieses Ziel angeben (zusammen mit einer relativen Gewichtung, welche die Eignung der Leitung zur Übertragung angibt). Vor der Weiterleitung eines Pakets wird eine Zufallszahl gezogen und eine der Alternativen anhand der Gewichtung ausgewählt.

Statisches Routing in seiner einfachsten Form wird, wie bereits erwähnt, auch in IP-Netzen verwendet: Hierbei wird in jedem Rechner statisch eine einzelne Route konfiguriert, über die ein Ziel (Endsystem oder Netzwerk) erreicht werden kann. Existiert nur eine Route, benötigt man natürlich keine Gewichtung – dieses Vorgehen kann dann verwendet werden, wenn es mehrere mögliche Wege gibt und man eine Lastverteilung vornehmen will.

Bitte beachten: ist hier und im Folgenden die Rede davon, dass „ein Routing-Eintrag hin zu einem Knoten“ existiert, ist dies eine Abstraktion – in der Realität ist solch ein „Knoten“ der Zugangsrouter zu einem LAN, also einem ganzen Netz von Knoten. Mit einem bestimmten IP-Adressbereich.

Backward Learning

- **Dynamisch, jeder Router trifft seine Wegwahlentscheidungen *isoliert* für sich**
 - ▶ Lernen von Pfaden aus übertragenen Paketen (siehe Switch)
 - Verwende einen Zähler, der mit jedem Hop um 1 erhöht wird
 - Oder nutze TTL, die mit jedem Hop verringert wird
 - Verwende als Weg hin zu einem Knoten denjenigen, über den Pakete vom Knoten mit kleinstem Zählerwert empfangen wurden
 - Bzw mit höchster TTL (= geringster Zahl an passierten Zwischenknoten)
 - Aber:
 - Wie kann die Verschlechterung eines Pfads wahrgenommen werden?
 - Und wie der Ausfall einer Leitung?
 - Lange Lernphase, bis ein Router alle Knoten kennt
 - Initiale Phase? Broadcast von Paketen durchs ganze Netz?



Backward-Learning realisiert eine sehr einfache und intuitive Idee – im Prinzip erfolgt das Lernen auf die gleiche Art wie bei Switches. Aufgrund der komplexeren Topologie ergeben sich allerdings im Vergleich zu Switches in LANs einige Nachteile und Probleme.

Nachteile

- Nur Änderungen zum Besseren werden zur Kenntnis genommen. Eine Erhöhung der Kosten ist nicht vorgesehen.
- Ausfälle oder Überlastung von Übertragungsleitungen werden nicht weitergemeldet. Ein Switch wird daher nach einer gewissen Zeit einen Eintrag löschen, falls einige Zeit keine Rahmen mehr von einer Adresse empfangen wurden. In ähnlicher Weise müsste ein Router periodisch sein Wissen verwerfen.

Probleme

- Während der Lernperiode ist das Routing nicht optimal.
- Bei häufigem Verwerfen des gesammelten Wissens nehmen viele Pakete Wege unbekannter Qualität. Bei seltenem Verwerfen ergibt sich ein schlechtes Reaktionsverhalten in Fehlerfällen.

Dynamische Routing-Protokolle

- **Verbesserte Wegwahl**

- ▶ Interaktion zwischen Routern
 - Austausch von Wissen über den Netzzustand
- ▶ Erweiterte Kostenwerte
 - Kosten orientiert an Entfernung, Latenz, Datenrate, ...
 - ... oder sind einfach immer 1 (wie bei Backward Learning)

- **Zwei Klassen von Protokollen zum Austausch von Routing-Informationen**

- ▶ *Distance Vector*
- ▶ *Link State*

Heutige Routing-Verfahren basieren meist auf dem Informationsaustausch zwischen Routern, damit Routing-Entscheidungen anhand des globalen Zustands eines Netzes getroffen werden können. Die beiden Klassen von Verfahren, die eingesetzt werden, sind Link State Routing und Distance Vector Routing. Beide betrachten ein Netzwerk als einen Graphen, in dem es kürzeste Wege zu finden gilt.

Die Kanten sind dabei mit Kosten für ihre Nutzung beschriftet. Hier können verschiedene Kostenmaße eingesetzt werden: aktuelle Auslastung der Ausgangsleitung, Latenz auf der Übertragungsstrecke, Datenrate auf der Übertragungsstrecke, Fehlerrate auf der Übertragungsstrecke...

Oft wird statt „Kosten“ auch der Begriff „Metrik“ verwendet – denn bei der Berechnung günstiger Pfade interessieren uns die Gesamtkosten für die Nutzung eines bestimmten Pfades; die Einzelkosten der Teilstrecken müssen also zu einem Gesamtwert für die Güte eines Pfades aggregiert werden. Dazu verwendet man eine Metrik – ein Abstandsmaß, welches die Einzelwerte der Teilstrecken zu einem Gesamtwert aggregiert, der die Güte eines Pfades angibt.

In den Folien sowie auch bei Übungsaufgaben (sofern nicht explizit angegeben) werden additive Metriken verwendet, d.h. die Kosten eines Pfades ergeben sich aus der Summe der Kosten der Teilstrecken. Ein kleinerer Wert ist dabei besser.

Werden andere Kostenmaße verwendet, kann es allerdings auch andere Berechnungsvorschriften geben und es könnte auch ein größerer Wert besser sein.

Dynamische Routing-Protokolle

- Basis: Graphentheorie

Graph: $G = (N, E)$

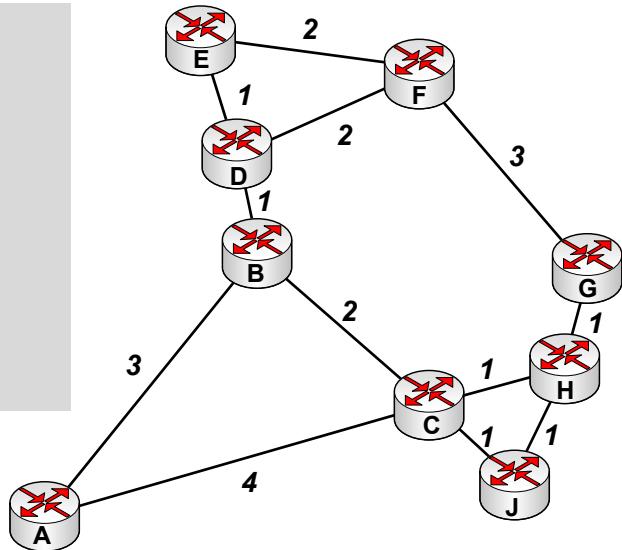
N = Menge von Routern

$$= \{ A, B, C, D, E, F, G, H, J \}$$

E = Menge von Verbindungen

$$= \{(A,B), (A,C), (B,C), (B,D), (D,E), (E,F), (D,F), (F,G), (G,H), (C,H), (C,J), (H,J)\}$$

Kostenfunktion c , z.B. $c(A,C) = 4$



Distance Vector Routing

- Erstes im Internet verwendete Verfahren

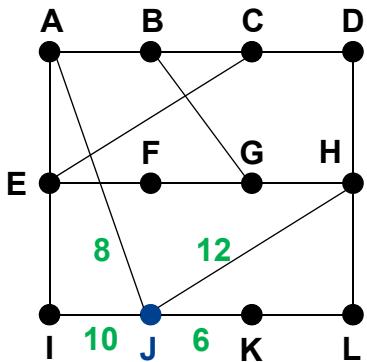
- ▶ Implementiert als *Routing Information Protocol, RIP*
- ▶ Jeder Router hat Routing-Tabelle mit Einträgen
(Ziel, Next Hop, Kosten)
- ▶ Lokaler Austausch globalen Wissens
 - Periodische Generierung von *Abstandsvektoren* (Ziel, Kosten) aus allen Einträgen und Versendung an benachbarte Router
 - Vektor von C: ((A, 4), (B, 2), (C, 0), (D, 3), (E, 4), ...)
 - Notation mit $D_x(y)$ = Kosten von x zu y:
[$D_x(y) : y \in N$]
 - Bei Empfang eines Abstandsvektors eines Nachbarn v aktualisiere eigenen Abstandsvektor und Routing-Tabelle:
 - $D_x(y) = \min \{c(x, v) + D_v(y)\}$ for each node $y \in N$

Einfaches Prinzip: tausche mit den Nachbarn Informationen über die eigene Routing-Tabelle aus, indem periodisch Abstandsvektoren aus der Routing-Tabelle generiert werden. Die nötigen Informationen, die anderen Routern zugestellt werden sollten, sind lediglich die Informationen, welche Ziele erreicht werden können und mit welchen Kosten. Auf diese Art wird aus der Routing-Tabelle eine Menge an Tupeln (bekanntes Ziel, Kosten zum Ziel) errechnet – ein Abstandsvektor.

Ein Router, der solch einen Vektor erhält, muss nur wissen, welche Kosten durch die Übertragungsleitung zwischen dem sendenden Router A und ihm selbst entstehen, um zu wissen, welche Kosten entstehen würden, um andere Ziele über A zu erreichen – addiere auf die im Abstandsvektor enthaltenen Kosten jeweils die Übertragungskosten hin zu A auf. Entsteht hierbei ein Wert, der kleiner ist als der, der aktuell in der Routing-Tabelle zu einem Ziel eingetragen ist (oder existiert zu einem Ziel noch gar keine Information), dann speichere den neuen Routingeintrag (Ziel, Next Hop, Kosten) als: (Ziel, A, von A empfangene Kosten + Übertragungskosten hin zu A).

Bei der Implementierung RIP sind die Kosten sehr einfach gehalten: sie betragen für jede Leitung 1, man versucht also nur die Anzahl zu passierender Router zu minimieren.

Distance Vector Routing – Beispiel



Grün: Kosten der Leitungen von J zu anderen Knoten

J erhält folgende Abstandsvektoren:

	von A nach...		von I nach...		von H nach...		von K nach...	
	A	0	I	24	H	20	K	21
A	A	0	I	24	H	20	K	21
B	12	B	36	B	31	B	28	
C	25	C	18	C	19	C	36	
D	40	D	27	D	8	D	24	
E	14	E	7	E	30	E	22	
F	23	F	20	F	19	F	40	
G	18	G	31	G	6	G	31	
H	17	H	20	H	0	H	19	
I	21	I	0	I	14	I	22	
J	9	J	11	J	7	J	10	
K	24	K	22	K	22	K	0	
L	29	L	33	L	9	L	9	

Neue Routing-Tabelle von J:

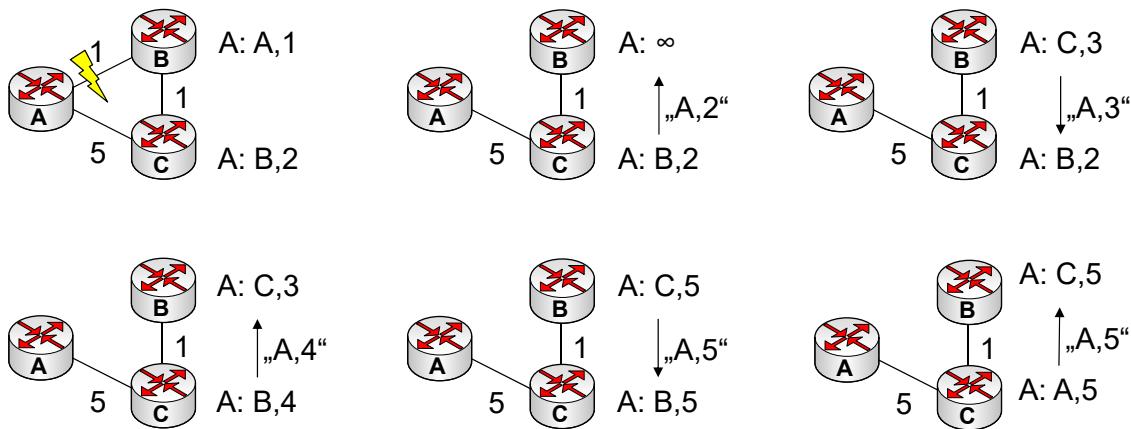
Ziel	Next Hop	Kosten
A	A	8
B	A	20
C	I	28
D	H	20
E	I	17
F	I	30
G	H	18
H	H	12
I	I	10
J	-	0
K	K	6
L	K	15

Quelle: Tanenbaum IV-93

Distance Vector Routing – Probleme

• Probleme bei Distance Vector Routing

- Bei größeren Netzen *langsame Konvergenz* zu einem konsistenten Zustand (langsame Informationsausbreitung)
 - Bouncing Effect* – langsames Hochschaukeln von Einträgen benachbarter Router



Das Distance Vector Routing war das ursprüngliche Routing-Verfahren, das im Arpanet unter dem Namen RIP implementiert wurde. Ein Problem ist die langsame Konvergenz im Falle eines Systemausfalls oder einer Änderung der Topologie: Änderungen der Informationen eines Routers werden nur an benachbarte Router weitergereicht. Diese verarbeiten die empfangenen Informationen und reichen entsprechende Änderungen wieder an ihre Nachbarn weiter. Durch diese lokale Weitergabe der Änderungen verbreitet sich eine Topologie-Änderung nur langsam – besonders bei Implementierungen wie RIP, bei denen nur alle 30 Sekunden eine Weitergabe von Änderungen erfolgt.

Darüber hinaus treten durch die lokale Weitergabe aber noch weitere Probleme auf: der sogenannte Bouncing-Effect, bei dem die Kosten in Routing-Einträgen sich aufgrund der Weitergabe veralteten Wissens zwischen zwei Routern langsam hochschaukeln, und Count-to-Infinity als Extremfall, wenn keine alternative Route gefunden werden kann.

Die Problematiken wurden zwar in einer Reihe von Erweiterungen des Algorithmus‘ zu beheben versucht, jedoch konnte keine entscheidende Verbesserung erzielt werden. Deshalb wurde das Verfahren schließlich breitflächig vom sogenannten Link State Routing abgelöst.

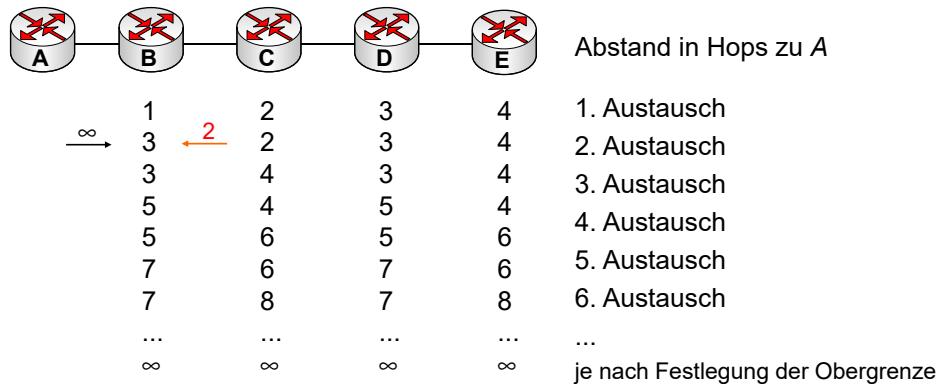
Distance Vector Routing – Probleme

• Probleme bei Distance Vector Routing

- ▶ Bei größeren Netzen *langsame Konvergenz* zu einem konsistenten Zustand (langsame Informationsausbreitung)

- *Count-to-Infinity* – Extremfall des Bouncing Effect

- Beispielszenario: Router A fällt aus:



Implementierung: RIP

- **Erstes Intra-AS-Protokoll im Internet**
 - ▶ Abstandsvektoren werden alle 30 Sekunden ausgetauscht
 - Konvergenz im Minutenbereich!
 - ▶ 4 Bit für Kosten – maximaler Abstand zu Zielen ist 14 Hops
 - 15 wird als „unendlich“ = unerreichbar angesehen
 - ▶ Maximal 25 Abstandswerte können in einer Nachricht versendet werden
- **Erweiterungen: RIPv2, RIPv3**
 - ▶ Authentifizierung bei Informationsaustausch
 - ▶ Multicasting von Abstandsvektoren für schnellere Konvergenz
 - ▶ IPv6-Unterstützung

Trotz der Probleme wurde RIP weiterentwickelt und kann auch heute noch innerhalb von Autonomen Systemen eingesetzt werden – es bleibt jedem AS-Betreiber überlassen, ein eigenes Protokoll auszuwählen.

Implementierung: BGP

- ***Border Gateway Protocol***

- ▶ Aktuell: BGPv4
- ▶ *Einziges Inter-AS-Protokoll*
- ▶ Distance Vector mit kleinen Modifikationen
 - Tausche keine Abstandsvektoren aus, sondern komplette Pfade hin zu den Zielen
 - Damit möglich:
 - Berücksichtigung von Policies (z.B. Vermeidung der Paketweiterleitung durch bestimmte ASs)
 - Kein Bouncing-Effekt, kein Count-to-Infinity
- ▶ Verfügbar als eBGP (exterior BGP) und iBGP (interior BGP)
- ▶ Auch Unterstützung von IPv6

Bei BGP werden keine Abstandsvektoren ausgetauscht, sondern komplettete Pfade. Dadurch können der Bouncing-Effect und Count-to-Infinity vermieden werden; zudem ist es für Routerbetreiber möglich, eigene Policies durchzusetzen, z.B. die Vermeidung bestimmter AS bei der Weiterleitung von Paketen.

BGP ist das Protokoll, das zum Routing zwischen autonomen Systemen eingesetzt wird (eBGP). Zusätzlich gibt es auch eine Variante zum Einsatz innerhalb autonomer Systeme (iBGP).

Link State Routing

- **Algorithmus:**
 1. Erkennung der Nachbarn (Hello-Pakete)
 2. Kostenschätzung/-messung (Echo-Pakete)
 3. Erzeuge Link-State-Paket (*Link State Advertisement, LSA*)
 - Inhalt: Absender, Sequenznummer, Alter, Liste mit Nachbarn + Kosten
 4. Verteile Link-State-Paket auf angeschlossenen Netzen
 - Periodisches oder ereignisgesteuertes *Flooding* mit Sequenznummern
 - Vernichten von Duplikaten und veralteten Link-State-Informationen.
 - Bestätigung von erhaltenen Link-State-Paketen
 5. Berechnung von Routing-Einträgen aus empfangenen Informationen
 - Z.B. Dijkstra-Algorithmus
- **Implementierung als *OSPF (Open Shortest Path First)***

Anfang der 80er Jahre wurde das Distance Vector Routing von einem effizienteren Verfahren mit Link State Routing abgelöst (die derzeit im Internet eingesetzten Routing-Protokolle OSPF und IS-IS basieren auf diesem Verfahren). Die wichtigsten Vorteile stellen dabei die schnellere Reaktionsfähigkeit (Konvergenz) beim Ausfall einzelner Knoten/Verbindungen sowie die Unterstützung mehrerer Wege zum Zielsystem dar.

Das Prinzip von Link-State-Algorithmen ist die globale Verteilung von lokalem Wissen – jeder Router ermittelt für sich seine Nachbarn sowie die Kosten der Leitungen hin zu den Nachbarn und sendet diese Information per Flooding ins Netz. Um das Flooding effizient zu gestalten, werden zwei Einträge in den Routing-Paketen verwendet:

- Sequenznummer zur Elimination von Duplikaten
- Altersangabe – Zeitangabe seit Generierung des LSAs. Nach Ablauf, d.h. bei Überalterung des Pakets werden die enthaltenen Angaben nicht mehr zur Erstellung von Routing-Tabellen berücksichtigt, damit man Netzinkonsistenzen aufgrund veralteter Link-Informationen vermeidet.

Da sichergestellt werden muss, dass alle Router alle Nachrichten erhalten, damit jeder seine Routing-Tabelle auf Basis der gleichen Informationen berechnet, wird der Erhalt von Routing-Paketen zwischen benachbarten Routern bestätigt.

Hat ein Router die Informationen aller anderen Router erhalten, kennt er die genaue Topologie des Netzes und kann damit beginnen, kürzeste Wege hin zu allen Routern (und damit zu den an sie angeschlossenen Teilnetzen) zu berechnen. Dies kann z.B. durch Anwendung des Dijkstra-Algorithmus' erfolgen.

Flooding

- Ähnlich wie Broadcasting:

- ▶ Jedes eingehende Paket wird auf *jeder* Übertragungsleitung weitergeleitet, außer auf derjenigen, auf der es eintraf
 - Aber: Paketflut, Zirkulieren von Paketen

- Maßnahmen zur Eindämmung der Flut

- ▶ Erkennung von Duplikaten durch Nummerierung der Pakete
- ▶ Kontrolle der Lebensdauer eines Pakets durch Zählen der zurückgelegten Teilstrecken (wie TTL)
- ▶ Optimierung: *selektives Flooding* – Weiterleitung nicht auf allen, sondern nur auf ausgewählten Leitungen
 - Nicht in OSPF, verwendet z.B. in drahtlosen Mesh-Netzen

Flooding ist zwar das einfachst denkbare Verfahren und kommt ohne Algorithmen und Tabellen aus, führt aber zu enormen Problemen: wenn jeder Router jede Nachricht an jeden anderen weiterleitet, werden massig Duplikate erzeugt, die auch weitergeleitet werden, was im schlimmsten Fall dazu führt, dass Pakete auf ewig im Netz zirkulieren und die Netzlast immer weiter steigt.

Daher muss man noch Zusatzinformationen in den Paketen mitschicken: zum einen kann durch die TTL die Anzahl der Weiterleitungsschritte begrenzt werden, zum anderen sollte ein Router jedes Paket nur einmal weiterleiten und nicht später bei ihm über andere Wege eintreffende Duplikate noch einmal. Daher werden Sequenznummern in den Paketen verwendet, und ein Router merkt sich, welche Pakete er bereits empfangen hat und wird später eintreffende Duplikate verwerfen.

Dijkstra-Algorithmus

- **Algorithmus von Dijkstra (1959)**

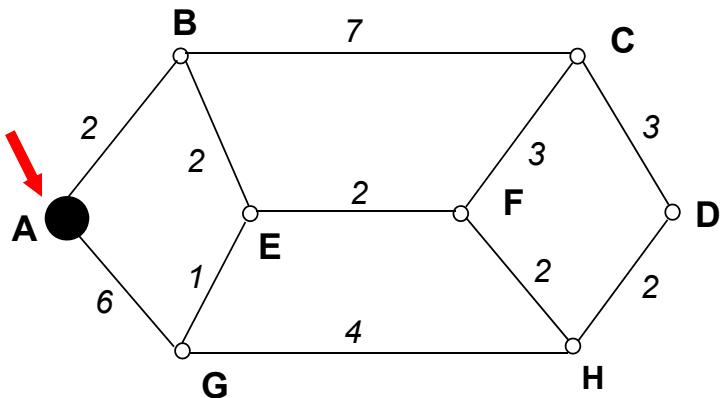
1. Mache den Ausgangsknoten zum “Arbeitsknoten” und markiere ihn als “permanent” (d.h. die Entfernung zu ihm wird sich nicht mehr ändern)
2. Betrachte alle Nachbarknoten des Arbeitsknotens und berechne aufgrund der Kosteninformationen an den Kanten ihre Distanz zum Ausgangsknoten.
Packe alle Nachbarn mit ihren Distanzen in eine “Arbeitsmenge”, falls sie sich nicht schon mit geringeren Kosten darin befinden oder bereits als permanent markiert sind
3. Wähle aus der Arbeitsmenge den Knoten mit geringster Distanz zur Quelle.
Mache ihn zum Arbeitsknoten und markiere ihn als permanent.
Gehe zurück zu 2.
4. Ein Abbruch erfolgt, sobald die Arbeitsmenge leer ist.

Bitte beachten: der Dijkstra-Algorithmus wurde in der Vorlesung nicht behandelt, da er aus anderen Veranstaltungen bekannt sein sollte. Wir haben trotzdem ein Beispiel in den Folien, falls nicht ganz klar sein sollte, wie man mithilfe von Dijkstra seine Routing-Tabellen berechnen kann. Das Beispiel ist allerdings nicht relevant als Prüfungsstoff.

Dijkstra-Algorithmus – Beispiel

- **Beispiel: Berechnung des kürzesten Pfades von A nach D**

1. Markiere A als permanent
2. Betrachte die Nachbarn von A (B, G)



Permanent: {A}

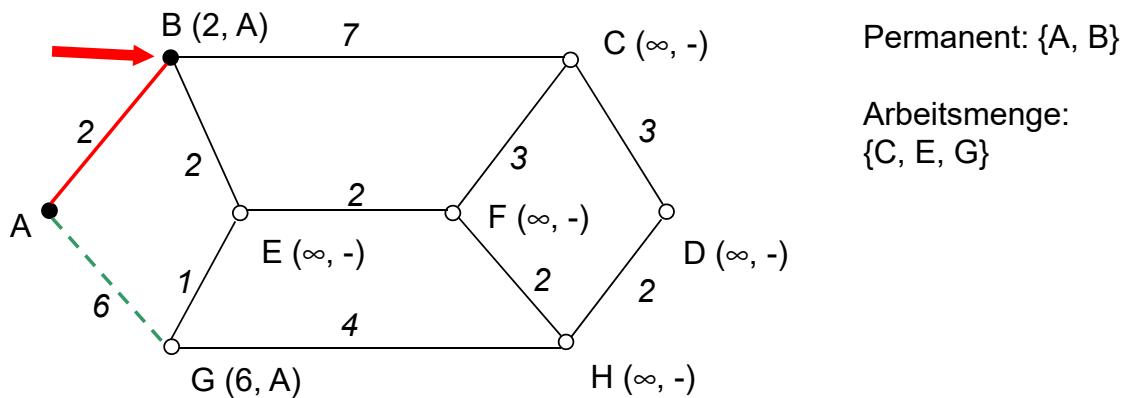
Arbeitsmenge:
{B, G}

Dijkstra-Algorithmus – Beispiel

- **Beispiel: Berechnung des kürzesten Pfades von A nach D**

Um später den Pfad auslesen zu können, speichern wir jeweils die Information, wie wir einen Knoten erreicht haben

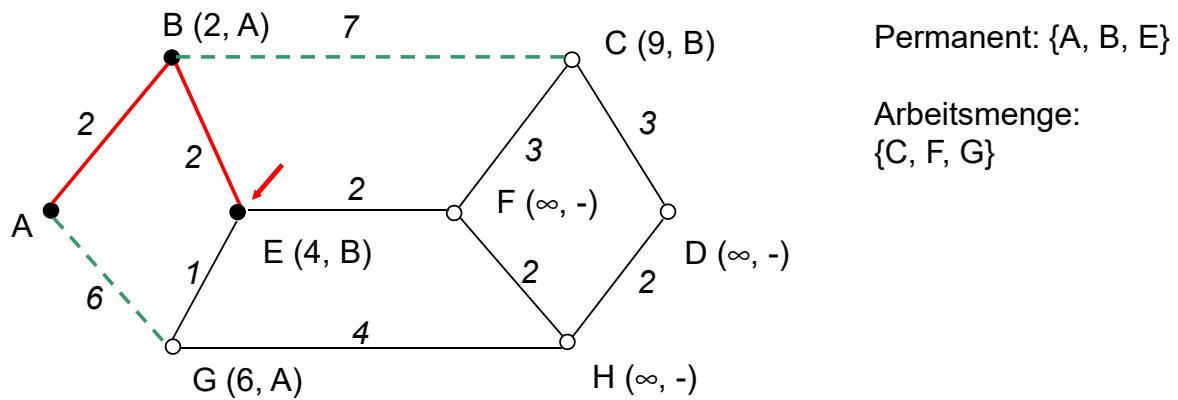
3. B wird Arbeitsknoten (permanent), da er die kürzere Distanz zu A hat
4. Betrachte die Nachbarn von B



Dijkstra-Algorithmus – Beispiel

- **Beispiel: Berechnung des kürzesten Pfades von A nach D**

5. E wird Arbeitsknoten (permanent), da er die kürzeste Distanz zu A hat
6. Betrachte die Nachbarn von E

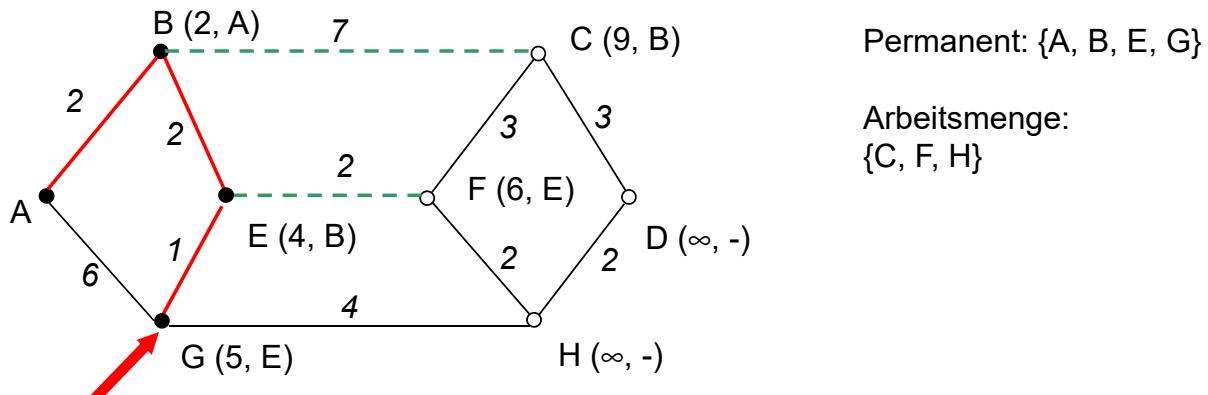


Dijkstra-Algorithmus – Beispiel

- **Beispiel: Berechnung des kürzesten Pfades von A nach D**

Vorheriges Label von G wird überschrieben, da ein kürzerer Weg hin zu ihm gefunden wurde

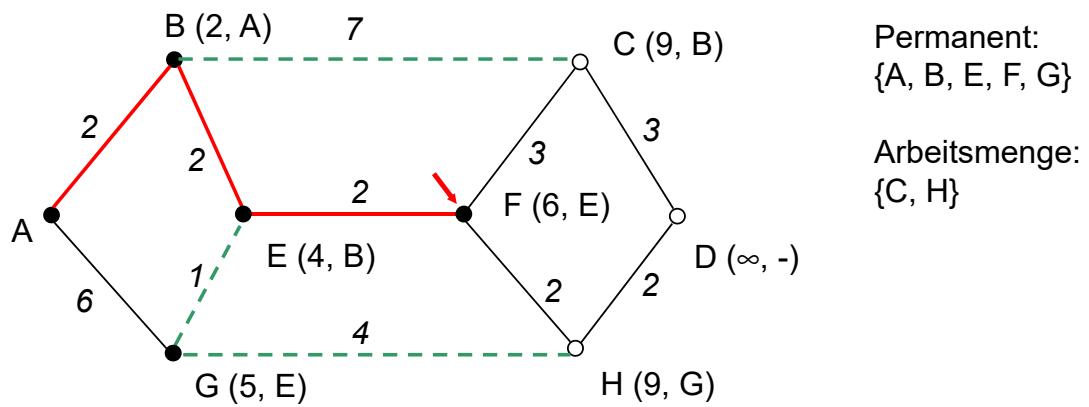
7. G wird Arbeitsknoten (permanent), da er die kürzeste Distanz zu A hat
8. Betrachte die Nachbarn von G



Dijkstra-Algorithmus – Beispiel

- **Beispiel: Berechnung des kürzesten Pfades von A nach D**

9. F wird Arbeitsknoten (permanent), da er die kürzeste Distanz zu A hat
10. Betrachte die Nachbarn von F

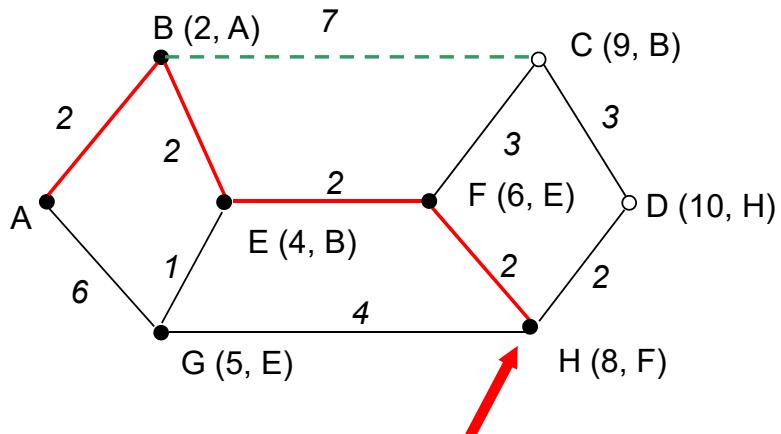


Dijkstra-Algorithmus – Beispiel

- **Beispiel: Berechnung des kürzesten Pfades von A nach D**

11. H wird Arbeitsknoten (permanent), da er die kürzeste Distanz zu A hat

12. Betrachte die Nachbarn von H



Permanent:
 $\{A, B, E, F, G, H\}$

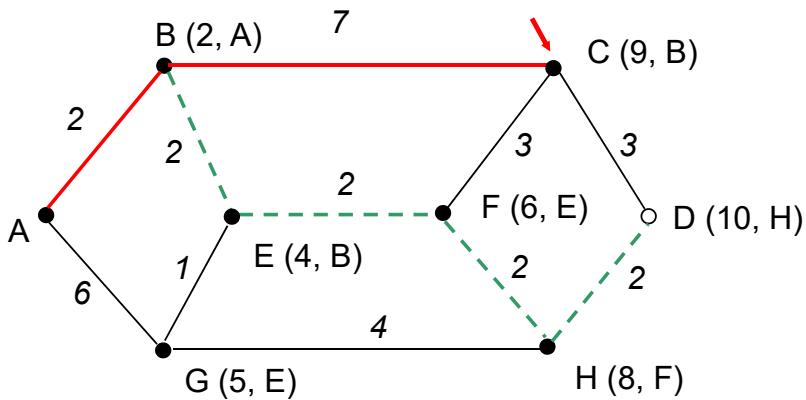
Arbeitsmenge:
 $\{C, D\}$

Dijkstra-Algorithmus – Beispiel

- **Beispiel: Berechnung des kürzesten Pfades von A nach D**

13. C wird Arbeitsknoten (permanent), da er die kürzeste Distanz zu A hat

14. Betrachte die Nachbarn von C



Permanent:
 $\{A, B, C, E, F, G, H\}$

Arbeitsmenge:
 $\{D\}$

Dijkstra-Algorithmus – Beispiel

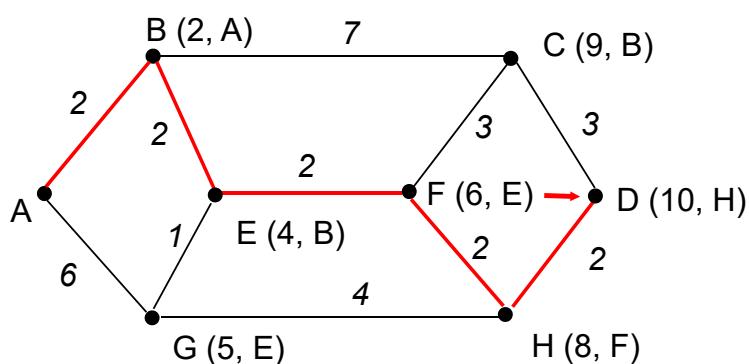
- **Beispiel: Berechnung des kürzesten Pfades von A nach D**

15. D wird Arbeitsknoten (permanent), da er die kürzeste Distanz zu A hat

Da D permanent ist, kann kein kürzerer Pfad hin zum Ziel mehr existieren; der kürzeste Weg ist gefunden und kann nun aus den Labeln vom Ziel zur Quelle ausgelesen werden

In der Routing-Tabelle von A wird gespeichert

„Ziel D, Next Hop B, Kosten: 10“



Permanent:
{A, B, C, D, E, F, G, H}

Arbeitsmenge:
{}
}

Errechnete Routing-Tabelle

- **Dijkstra-Algorithmus aus vorherigem Beispiel**
 - ▶ Ermittlung des kürzesten Weges zu D
 - ▶ Aber: kürzeste Wege zu allen Knoten wurden gleichzeitig ermittelt
 - ▶ Daher vollständige Generierung der Routing-Tabelle durch einen Lauf:

nach	Next Hop	Kosten
B	B	2
C	B	9
D	B	10
E	B	4
F	B	6
G	B	5
H	B	8

Open Shortest Path First (OSPF)

- **Weit verbreitete Implementierung: OSPF**

- ▶ 1990 standardisiert durch IETF (RFC 1247)
- ▶ OSPFv3: Unterstützung von IPv6
- ▶ Unterstützt viele verschiedene Metriken als Kosten
- ▶ Erzeuge LSAs je nach Konfiguration alle 10 bis 30 Sekunden
- ▶ Flooding von LSAs mittels Multicast an 224.0.0.5
- ▶ Lastverteilung möglich – Speicherung alternativer Wege, probabilistische Verteilung von Daten über alternative Wege
- ▶ Authentifizierung beim Austausch von LSAs

Zusammenfassung

- **IP als zentrales Protokoll zur Kopplung von Netzen**
 - ▶ Verbindungslos, unzuverlässig
 - ▶ Definiert Weiterleitungsregeln, Adressschema
 - ▶ Network Address Translation zum Umgang mit Adressknappheit
 - ▶ IPv4 vs. IPv6
- **Hilfsprotokolle**
 - ▶ ARP zur Ermittlung des Zielknotens auf dem letzten Hop
 - ▶ DHCP (Schicht-7-Protokoll) zur automatischen Konfiguration
 - ▶ ICMP zum Austausch von Kontrollnachrichten
- **Routingprotokolle**
 - ▶ Distance Vector vs. Link State
 - ▶ In autonomen Systemen können Netzprovider bevorzugte Protokolle wählen