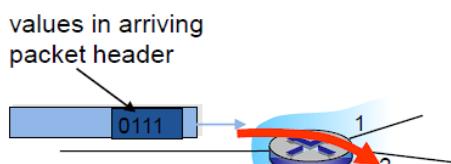


Netzwerkschicht - Kapitel 5

Datenebene, Data plane

- ▶ Anwenderebene (user plane), Forwarding-Ebene, Datenschicht oder Trägerschicht
- ▶ Teil eines Netzes, der Daten transportiert
- ▶ lokale, per-Router Funktion
- ▶ bestimmt, wie Paket auf Eingangsleitung des Routers weitergeleitet wird auf Ausgangsleitung: **Weiterleitung**



Data Plane: Teil des Netze, der den Datenverkehr transportiert

- ▶ Überblick Netzwerkschicht
 - Datenebene (*data plane*)
 - Steuerebene (*control plane*)
- ▶ Das Innere eines Routers
- ▶ IP: Internet Protocol
 - Paketformat
 - Fragmentierung
 - IPv4 Adressierung
 - Network Address Translation
 - IPv6
- ▶ Router verwendet die Longest-Prefix-Matching-Regel (längstes übereinstimmendes Präfix): findet den längsten übereinstimmenden Eintrag in der Tabelle und leitet das Paket an die Schnittstelle weiter, die zu diesem Präfix gehört.

Kontrollebene, Control plane

- ▶ Teil eines Netzes, der Signalisierungsverkehr trägt
- ▶ Netzwerk-weite Logik
- ▶ bestimmt, wie Pakete zwischen Routern entlang des Ende-zu-Ende Pfades von Quelle zu Ziel geleitet werden: **Routing**
- ▶ Zwei Control Plane Ansätze
 - herkömmliche Routing Algorithmen: in Router implementiert
 - **Software defined networking (SDN):** implementiert in (entfernten) Controllern, Servern

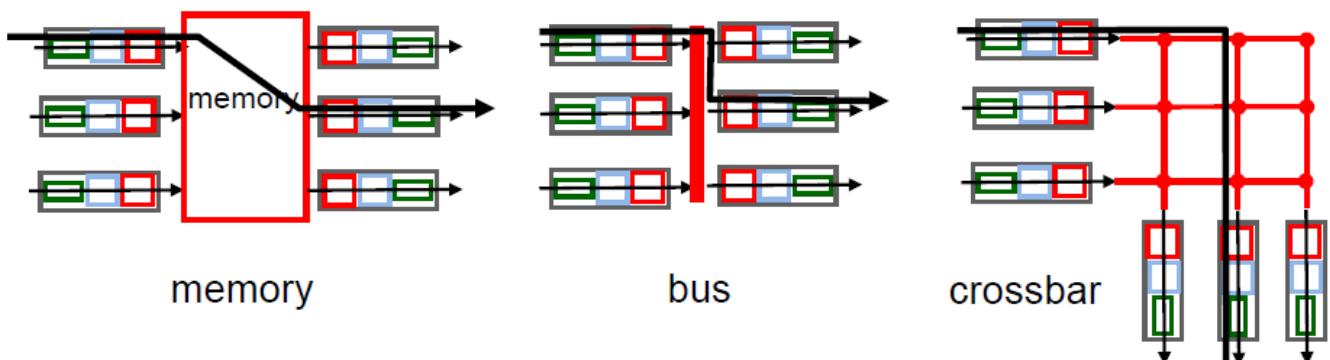
Control Plane: Steuert und dient der Datenebene

- ▶ Routing
 - Link State Routing
 - Distanzvektor-Routing
- ▶ Intra-AS Routing im Internet: OSPF
- ▶ Routing zwischen ISPs: BGP
- ▶ Die SDN Steuerebene
- ▶ ICMP: Internet Control Message Protocol
- ▶ Netzwerkmanagement und SNMP

Destination Address Range	Link interface
11001000 00010111 00010*** *****	0
11001000 00010111 00011000 *****	1
11001000 00010111 00011*** *****	2
otherwise	3

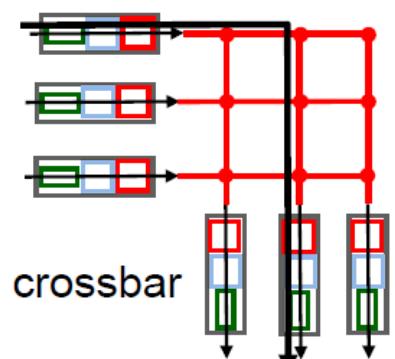
Switching Fabrics

- ▶ Verbindet Eingangsports mit Ausgangsport (*eigenes Netz im Router!*)
- ▶ Transportiere Paket von Eingangspuffer zum richtigen Ausgangspuffer
- ▶ Switching Rate: Rate, mit der Pakete von Input zu Output transportiert
 - oft als Vielfaches der Input/Output Leitungsrate angegeben
 - N Eingänge: gewünschte Switching Rate = N-fache Leitungsrate
- ▶ Drei Arten von Switching Fabrics

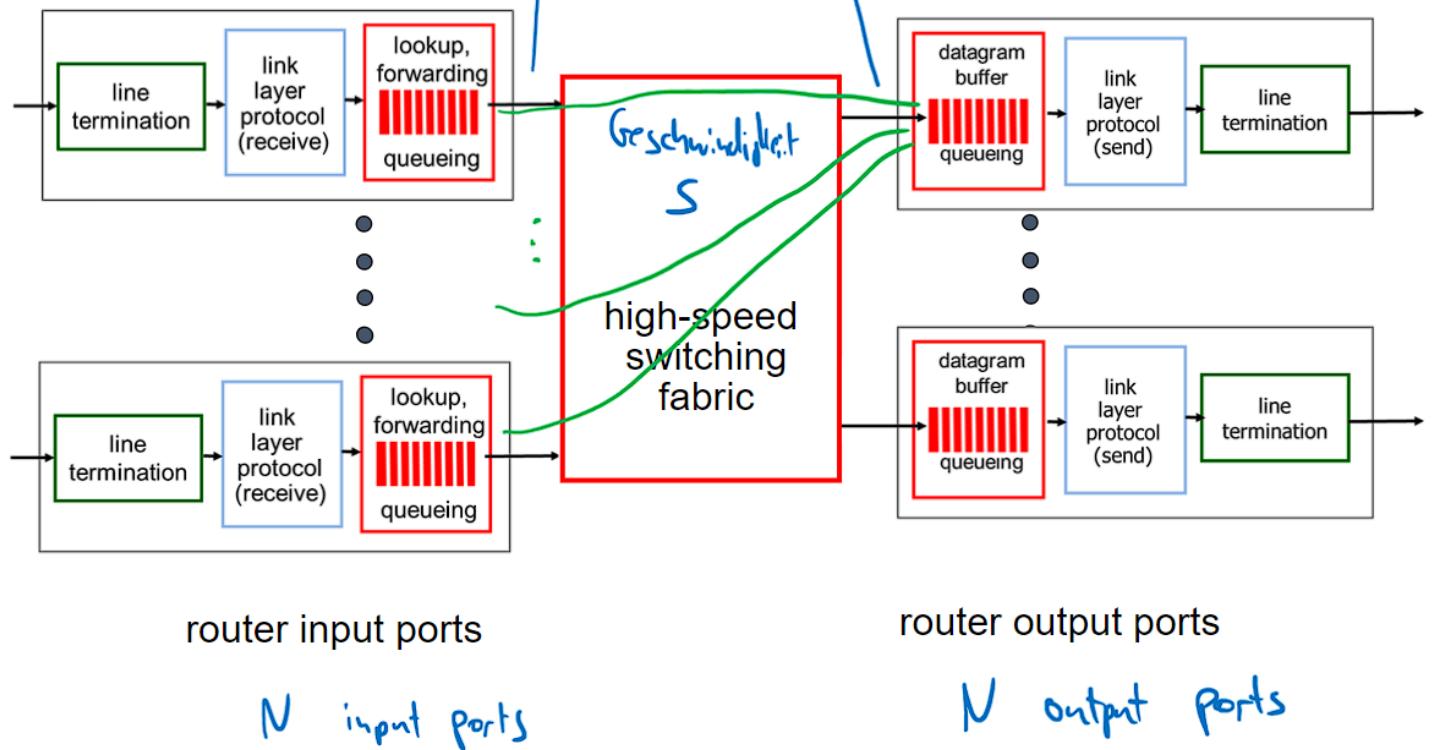


Switching mittels Verbindungsnetz

- ▶ Überwindet Bandbreitenbeschränkung eines gemeinsamen Buses
- ▶ Crossbar (s. Raummultiplex, Koppelnetze)
 - Verbindungsnetzwerk, das aus $2N$ Bussen besteht, die N Eingangsports mit N Ausgangsport verbindet
 - alle N Eingangsports mit allen N Ausgangsports verbinden: quadratischer Schaltaufwand für Kreuzungspunkte
 - parallele Verarbeitung möglich



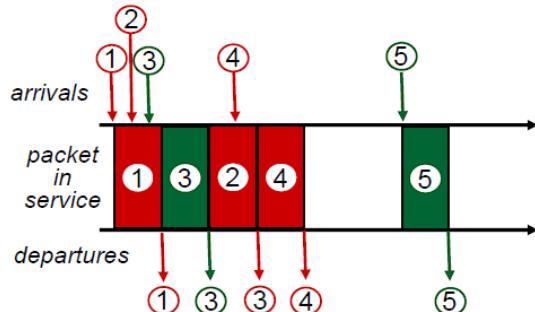
Falls $S > N \cdot R$,
können Warteschlangen
vernachlässigt werden



- ▶ Queuing erforderlich, falls die Ankunftsrate größer als die Übertragungsgeschwindigkeit des ausgehenden Links ist
- ▶ Auftritt von Warteverzögerungen und Verlusten durch Überlauf des Buffers am Ausgangsport
- ▶ RFC 3439: Durchschnittliche Buffergröße gleich der “charakteristischen” RTT (bspw. 250 ms) multipliziert mit der Linkkapazität C
 - $C = 10 \text{ Gpbs}$
 - $RTT = 250 \text{ ms}$
 - $\rightarrow 2.5 \text{ Gbit Buffer}$
- ▶ Aktuellere Empfehlung: Berücksichtigung der Anzahl der Flows
 - N Flows, Buffergröße gleich $\frac{RTT \cdot C}{\sqrt{N}}$

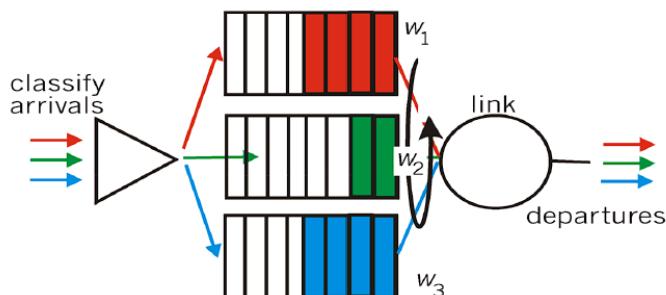
Round Robin (RR) Scheduling

- ▶ Zuordnung der Pakete zu verschiedenen Klassen
- ▶ Zyklische Abarbeitung der Warteschlangen der Klassen
- ▶ Falls vorhanden, wird jeweils ein wartendes Paket pro Warteschlange gesendet

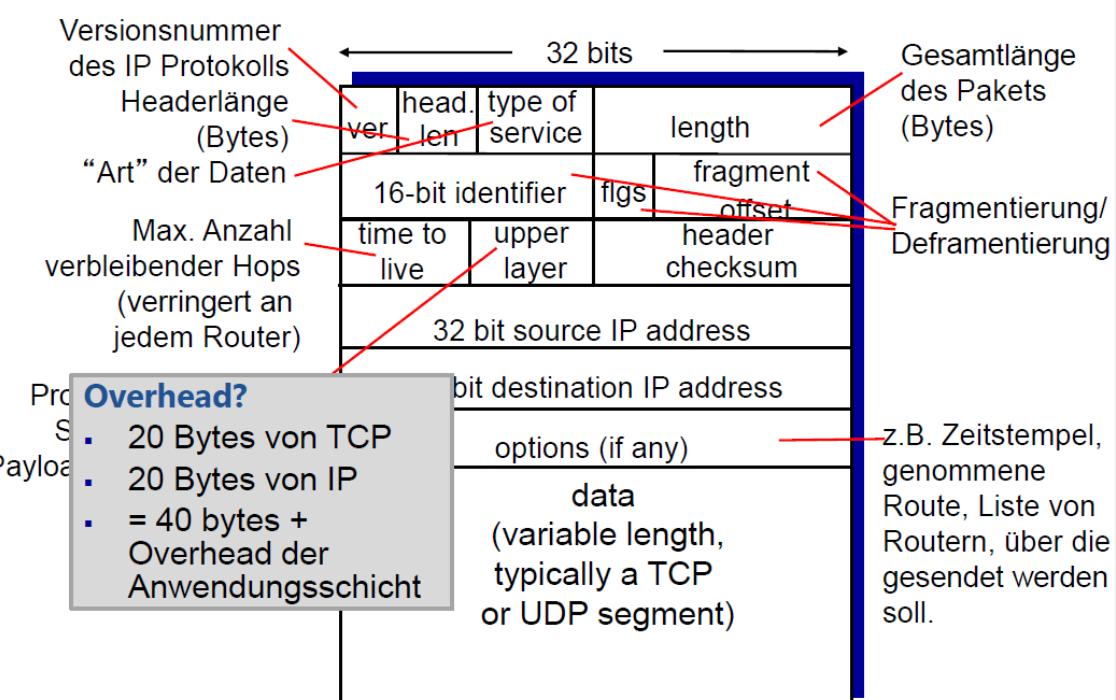


Weighted Fair Queuing (WFQ) Scheduling

- ▶ Verallgemeinerte Form des Round Robin Scheduling
- ▶ Jede Klasse erhält eine Gewichtung
- ▶ Gewichtung bestimmt die Servicezeit (z.B. Anzahl der versendeten Bytes) pro Durchlauf



IP Paket Format

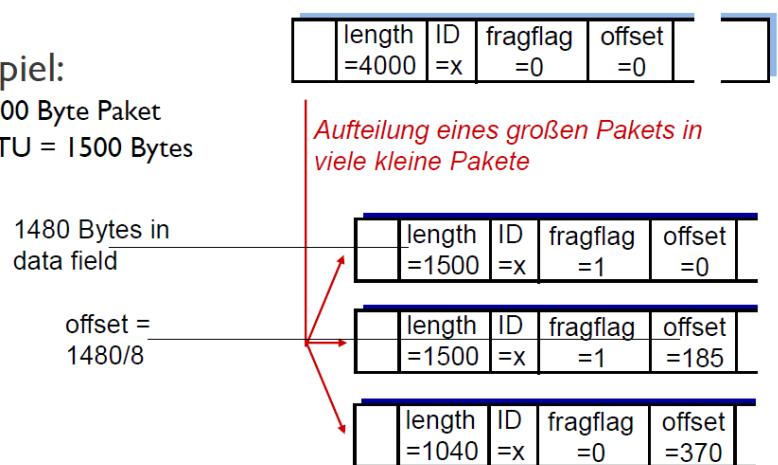


- Netzwerklinks haben eine max. erlaubte Paketgröße bzw. Framegröße MTU (Maximum Transmission Unit)
- MTU abhängig vom Linktyp
- Zu große IP Pakete werden aufgeteilt (fragmentiert)

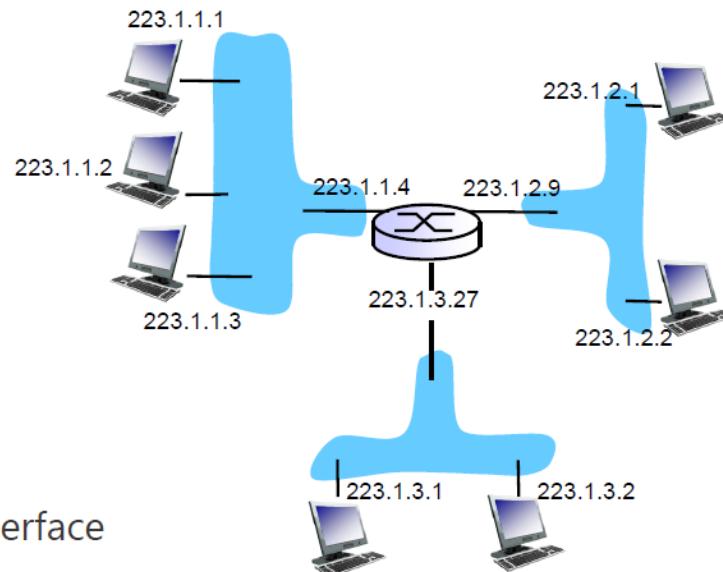
- Fragmentierung eines Paketes in viele Pakete
- Wiederzusammenbau erfolgt am Zielhost
- Identifizierung und Reihenfolge zusammengehöriger Fragmente anhand des IP Headers

Beispiel:

- 4000 Byte Paket
- MTU = 1500 Bytes



- IP Adresse: 32-bit Kennung für Host- und Routerinterfaces
- Interface: Verbindung zwischen Host/Router und physikalischem Link
 - Router haben meist viele Interfaces
 - Hosts haben meist ein oder zwei Interfaces (z.B. wired Ethernet, wireless 802.11)
- Jede IP Adresse ist einem Interface zugeordnet

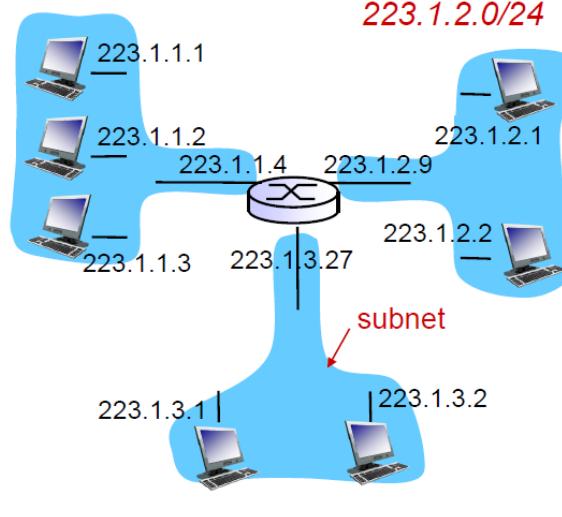


$223.1.1.1 = \underline{11011111} \underline{00000001} \underline{00000001} \underline{00000001}$

223 1 1 1

223.1.1.0/24

223.1.2.0/24



subnet mask: /24

DHCP: Dynamic Host Configuration Protocol

- ▶ Ziel: Host erhält dynamisch seine IP Adresse vom Netzwerkserver, wenn er einem Netzwerk beitritt
 - Kann Adresszuordnung erneuern
 - Ermöglicht Wiederverwendung von Adressen
 - Unterstützung von mobilen Nutzern, die Netzwerk beitreten möchten
- ▶ DHCP kann zusätzliche Informationen liefern wie Adresse des first-hop Routers, Netzwerkmaske, Name und Adresse des DNS Servers
- ▶ DHCP-Nachrichten Übersicht
 - Host sendet "DHCP discover" Nachricht (optional)
 - DHCP Server antwortet mit "DHCP offer" Nachricht (optional)
 - Host fragt IP Adresse an mit "DHCP request" an
 - DHCP Server sendet die IP Adresse mittels "DHCP ack" Nachricht

DHCP server: 223.1.2.5

DHCP discover



src : 0.0.0.0, 68
dest.: 255.255.255.255,67
yiaddr: 0.0.0.0
transaction ID: 654

arriving client



DHCP offer

src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddr: 223.1.2.4
transaction ID: 654
lifetime: 3600 secs

DHCP request

src: 0.0.0.0, 68
dest:: 255.255.255.255, 67
yiaddr: 223.1.2.4
transaction ID: 655
lifetime: 3600 secs

DHCP ACK

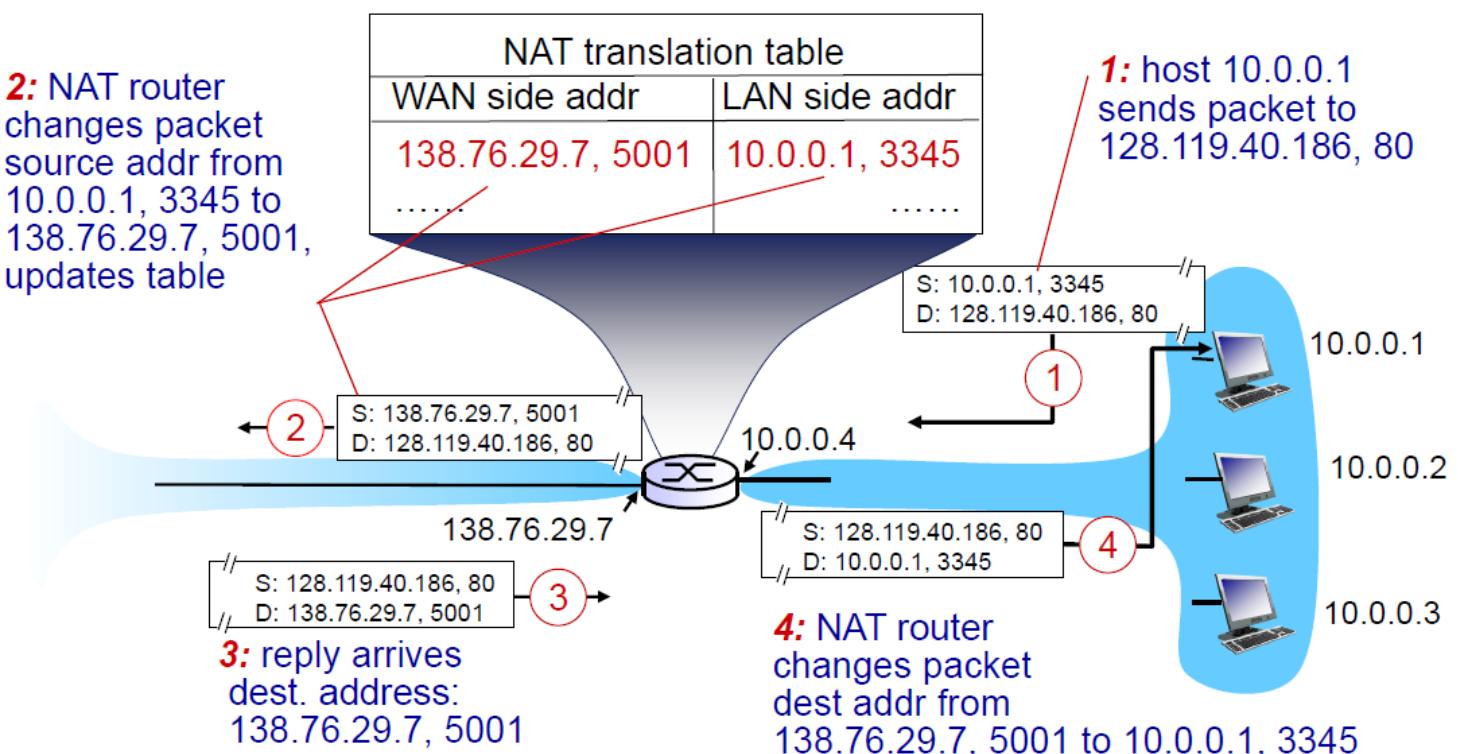
src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddr: 223.1.2.4
transaction ID: 655
lifetime: 3600 secs

NAT: Network Address Translation

Motivation: lokale Netze nutzen nach "außen" nur eine IP Adresse

- ▶ Änderung von (Source IP Adresse, Portnummer) bei jedem ausgehenden Paket zu (NAT IP Adresse, neue Portnummer) am NAT-Router
- ▶ Clients/Server außerhalb des lokalen Netzes verwenden (NAT IP Adresse, neue Portnummer) als Zieladresse
- ▶ Übersetzung von (Source IP Adresse, Portnummer) zu (NAT IP Adresse, neue Portnummer) wird in der NAT Translation Table des Routers gespeichert
- ▶ NAT-Router ersetzt (NAT IP Adresse, neue Portnummer) in jedem eingehenden Paket durch die zugehörige (Source IP Adresse, Portnummer)
- ▶ Port Address Translation (PAT) als spezielle Form des NAT

NAT: Network Address Translation



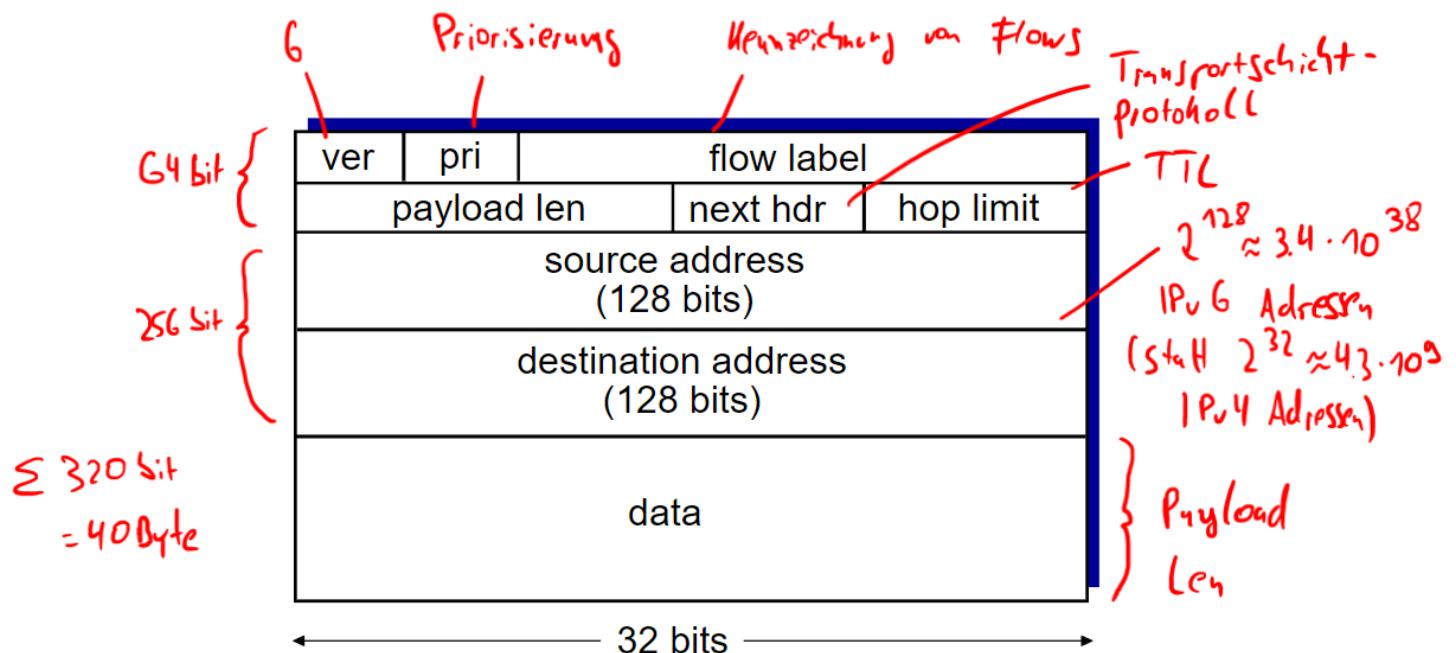
IPv6: Motivation

- ▶ Ursprüngliche Motivation: Adressmangel, da 32-bit Adressraum bald vollständig zugewiesen ist
- ▶ Weitere Motivation
 - Header-Format beschleunigt die Verarbeitung und Weiterleitung der Pakete
 - Änderungen am Header ermöglichen QoS
- ▶ IPv6 Paket Format
 - Feste Hänge von 40 Bytes
 - Keine Fragmentierung erlaubt
- ▶ **checksum:** vollständig entfernt, um Verarbeitungszeit bei jedem Hop zu verringern
- ▶ **options:** sind erlaubt, aber außerhalb des Headers; Next Header Feld kann auf diese verweisen
- ▶ **ICMPv6:** neue Version von ICMP (s. Abschnitt hinten)
 - Zusätzliche Nachrichtentypen, z.B. "Paket ist zu groß"
 - Multicast group management Funktionen

priority: Priorisierung von Datagrammen im Flows

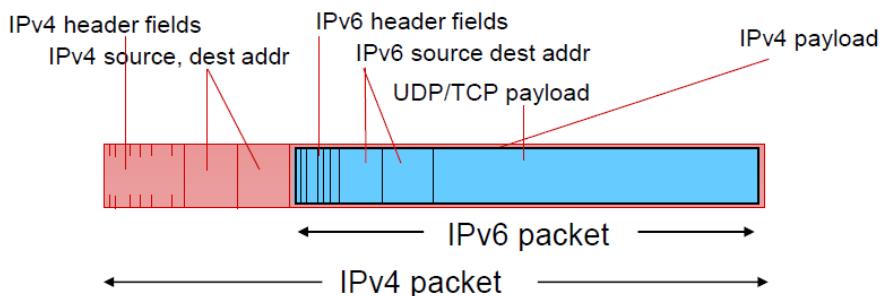
flow Label: identifiziert Datagramme des selben Flows

next header: definiert das Protokoll der darüber liegenden Schicht

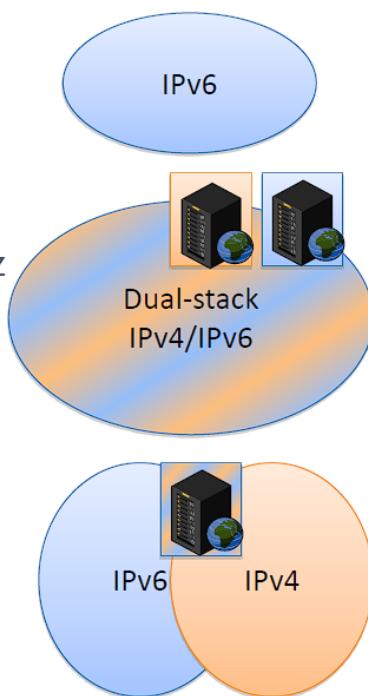


Übergang von IPv4 zu IPv6

- ▶ Simultaner Upgrade aller Router nicht möglich
- ▶ Übergangslösung durch Tunneling
- ▶ Tunneling: IPv6 Pakete werden als Payload in IPv4 Pakete gekapselt



- ▶ Reines IPv6
 - Kompletter Wechsel auf IPv6
 - Einsatz schwierig bei laufenden IT Infrastrukturen
- ▶ Dual-stack
 - Geringe Kosten für Umstellung & Einsatz
 - Nachteile
 - Störung des parallelen IPv4 Betriebs
 - Übernahme von "legacy" Strukturen
- ▶ Paralleler Einsatz
 - Geringe Gefahr für IPv4 Betrieb
 - Legacy IPv4 Strukturen können vermieden werden
 - Höhere Kosten Umstellung und Einsatz



Vorteile von IPv6

- ▶ Vergrößerung des Adressraums
- ▶ Vereinfachung und Verbesserung des Protokollrahmens (Kopfdaten); dies entlastet Router von Rechenaufwand.
- ▶ Zustandslose automatische Konfiguration von IPv6-Adressen
- ▶ Mobile IP und Multihoming
- ▶ Implementierung von IPsec innerhalb des IPv6-Standards.
- ▶ Unterstützung von Netztechniken wie Quality of Service und Multicast

Aufgaben von Routing Protokollen

- ▶ Routing Protokolle definieren, wie Routen gefunden werden
 - Routing-Algorithmus:
 - **Distance-Vector** oder **Link State**
 - Routing Metrik:
 - beschreibt, wie Link-Kosten definiert sind und wie aus mehreren Link-Kosten die Pfad-Kosten für eine Route bestimmt werden

Global

- ▶ Alle Router besitzen Informationen über die gesamte Topologie und Linkkosten
- ▶ Route lokal berechnet
- ▶ Link-State-Algorithmen

Dezentral

- ▶ Jeder Router kennt nur seine direkten Nachbarn und die Kosten dieser Links
- ▶ Iterative Berechnung der Pfade und Austausch von Informationen mit den Nachbarn
- ▶ Distanzvektoralgorithmen

Statisch

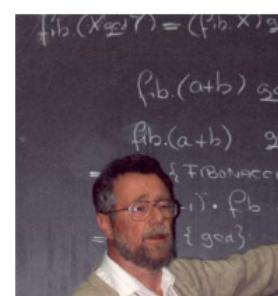
- ▶ Routen ändern sich langsam im Laufe der Zeit

Dynamisch

- ▶ Routen ändern sich schneller
 - Durch periodische Updates
 - Als Reaktion auf Änderungen der Linkkosten

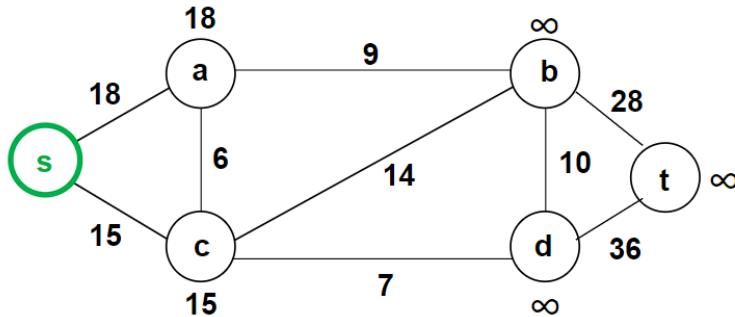
Link-State Routing

- ▶ Jeder Router teilt **allen** anderen Routern regelmäßig seine lokalen Verbindungen mit
 - Flooding: die Routing Information eines Routers wird von den anderen Routern an alle Nachbarn weitergeleitet
- ▶ Jeder Router kennt die **gesamte Netz-Topologie**
- ▶ Jeder Router berechnet die **beste Route** zu einem Ziel lokal
 - alle Router berechnen die gleichen Routen
 - Dijkstra-Algorithmus
- ▶ Wichtigstes Link State Routing Protokoll
 - OSPF (Open Shortest Path First)



Beispiel: Dijkstra (1)

- Nächster Schritt: $D(a) = 18, D(c) = 15$
Knoten s wird gefärbt, d.h. zu N' hinzugefügt
 $N' = \{s\}$
- Nach Färbung von Knoten s



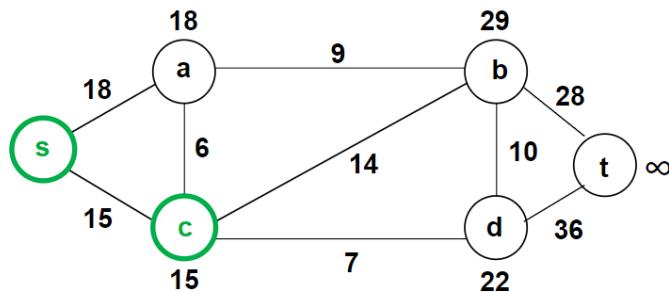
Schleife

- Wähle aus der Menge der ungefärbten Knoten den Knoten mit minimaler Markierung, hier Knoten c mit $D(c) = 15$.

Betrachte Knoten c

- Wir berechnen die Markierung für die adjazenten Knoten a, b, d $\notin N'$
 $D'(a) = D(c) + 6 = 21$
 $D'(b) = D(c) + 14 = 29$
 $D'(d) = D(c) + 7 = 22$ und c wird gefärbt

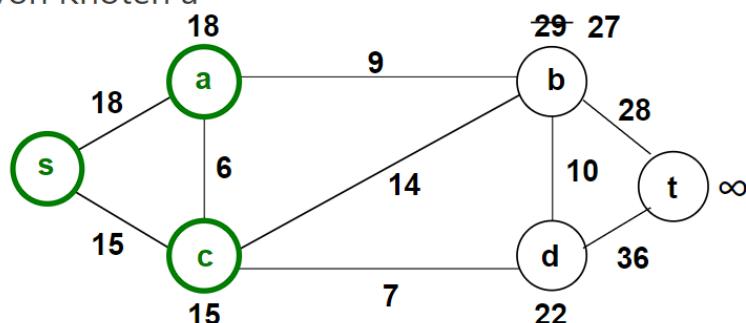
ungefärbte Nachbar-Knoten von c
- Nach Färbung von Knoten c



- Wähle wieder die Knoten mit minimaler Markierung, hier a mit $D(a) = 18$.

Betrachte Knoten a

- Berechne $D(b) = D(a) + 9 = 27$. Das ist kürzer als 29! D.h. Knoten a wird gefärbt und b erhält neue Markierung.
- Nach Färbung von Knoten a



- (1) Jeder Router kennt seine Nachbarn und die Kosten zu seinen Nachbarn
- (2) Jeder Router sendet diese Information zu allen seinen Nachbarn
- (3) Die Nachbarn leiten die Information weiter, so dass alle Router diese Information erhalten (Flooding)
- (4) Jeder Router kennt alle anderen Router inklusive deren Nachbarschaft
- (5) **Berechnung der kürzesten Pfade nach dem Dijkstra-Algorithmus**

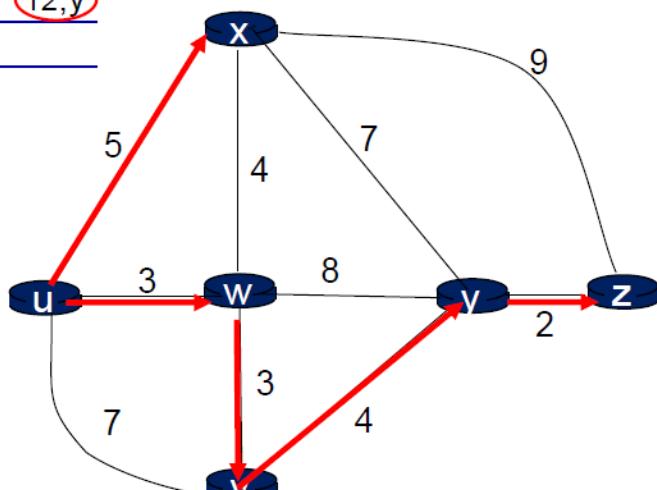
Weiteres Beispiel (Kurose & Ross)

Step	N'	D(v)	D(w)	D(x)	D(y)	D(z)
		p(v)	p(w)	p(x)	p(y)	p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w		11,w	14,x	
3	uwxv			10,v	14,x	
4	uwxvy				12,y	
5	uwxvyz					

c(x,y): Linkskosten
D(v): Pfadkosten
N': Menge aller entschiedener Knoten
p(v): Vorheriger Knoten im Pfad vom Sender zu Knoten v

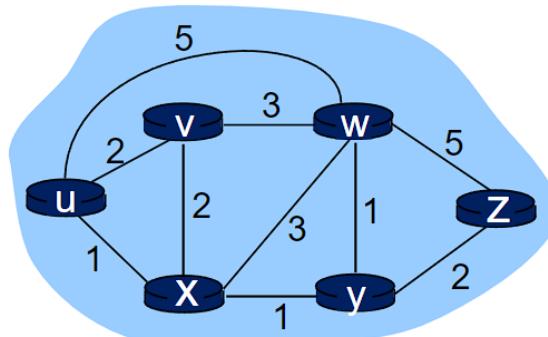
Algorithmus terminiert:

- ▶ Für jeden Knoten ist sein Vorgänger entlang des kostengünstigsten Pfads vom Quellknoten bekannt
- ▶ Ganzer Pfad von der Quelle zu allen Zielen konstruierbar



Julius-Maximilians-

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



Distance-Vector Routing

- ▶ Jeder Router kennt nur seine **lokale Umgebung**
- ▶ Jeder Router tauscht nur mit seinen direkten **Nachbarn** Informationen aus
 - Distanzvektor: enthält für alle bekannten Ziele die Pfadkosten aus Sicht des Absenders
- ▶ Jeder Router berechnet den **besten nächsten Hop** für ein Ziel, ohne die gesamte Route zu kennen
 - Kosten einer Route ergeben sich aus den Distanzvektoren der Nachbarn plus den Kosten zum jeweiligen Nachbarn
 - beste Route geht über den Nachbarn mit den geringsten Kosten
 - **Bellman-Ford Algorithmus**
- ▶ Wichtigster Vertreter:
 - EIGRP (Enhanced Interior Gateway Routing Protocol, CISCO)

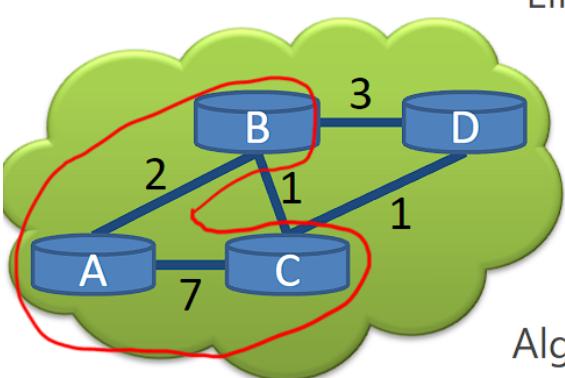
Bellman-Ford Gleichung

- Sei
 $d_x(y) :=$ Kosten des kostengünstigsten Pfades von x zu y

- Dann

$$d_x(y) = \min \{ c(x, v) + d_v(y) \}$$

v
 |
 cost from neighbor v to destination y
 |
 cost to neighbor v
 min taken over all neighbors v of x



Ein Knoten x speichert folgendes

- Kosten zu jedem Nachbar v: $c(x, v)$
- Distanzvektor: $D_x = [D_x(y): y \in N]$
- Distanzvektoren für alle Nachbarn v:
 $D_v = [D_v(y): y \in N]$

Algorithmus berechnet und aktualisiert

- Distanzen für alle Ziele y: D_x
- Next Hop für alle Ziele y: H_x

Sicht des Knoten A

$$\begin{array}{c}
 \begin{array}{cccc} A & B & C & D \end{array} \\
 D_A \quad - \quad 2 \quad 7 \quad \infty \\
 H_A \quad - \quad B \quad C \quad / \\
 D_B \quad 2 \quad - \quad 1 \quad 3
 \end{array}$$

Knoten B sendet D_B

$$\begin{array}{c}
 \begin{array}{cccc} A & B & C & D \end{array} \\
 D_B \quad 2 \quad - \quad 1 \quad 3
 \end{array}$$

Sicht des Knoten A

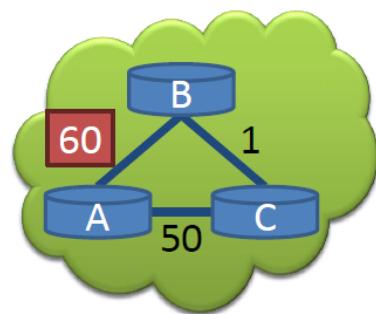
$$\begin{array}{c}
 \begin{array}{cccc} A & B & C & D \end{array} \\
 D_A \quad - \quad 2 \quad 7 \quad \infty \\
 H_A \quad - \quad B \quad C \quad / \\
 D_B \quad 2 \quad - \quad 1 \quad 3
 \end{array}
 \Rightarrow
 \begin{array}{c}
 \begin{array}{cccc} A & B & C & D \end{array} \\
 D_A \quad - \quad 2 \quad 3 \quad 5 \\
 H_A \quad - \quad B \quad B \quad B \\
 D_B \quad 2 \quad - \quad 1 \quad 3
 \end{array}$$

Neuberechnung

$$\begin{aligned}
 & \min \{ D_A(C), \\ & D_A(B) + D_B(C) \} = \min \{ 7, 2+1 \} \\ & \min \{ D_A(D), \\ & D_A(B) + D_B(D) \} = \min \{ \infty, 2+3 \}
 \end{aligned}$$

Anderes Beispiel: Count to Infinity (1)

- Die Kosten des Links A-B steigen von 4 auf 60.
- Knoten B erkennt diese Veränderung, hat die Information, dass C eine Route nach A mit Kosten 5 kennt und trägt in seine Tabelle eine Route mit Kosten $5+1=6$ über B ein
 - Knoten C weiß nicht, dass die von C gemeldete Route über ihn selbst läuft



	D	C	N		D	C	N		D	C	N		D	C	N
Node B	A	4	A		A	6	C		A	4	A		A	6	C
	C	1	B		C	1	B		C	1	B		C	1	B
	D	C	N		D	C	N		D	C	N		D	C	N
Node C	A	5	B		A	5	B		A	5	B		A	7	B
	B	1	B		B	1	B		B	1	B		B	1	B

Initial Maximilians

- Das Spiel geht weiter:
 - in jeder Update-Runde erhöhen sich die Kosten der Route nach A um 1, bis die Route über den Link A-C gefunden wird
 - im Falle eines ausgefallenen Links ohne Alternativroute gibt es kein Abbruchkriterium
 - **Count-to-Infinity**
- Generell gilt: Bessere Routen verbreiten sich schnell im Netz.
 - „**Good news travel fast.**“
- Generell gilt: Informationen über höhere Link-Kosten verbreiten sich langsam im Netz
 - „**Bad news travel slowly**“
- Lösung für das **Count-to-Infinity** Problem
 - **Poisonous Reverse**

Poisonous Reverse

- Wenn C über B nach A routet
 - dann sendet C an B, dass $D(C, A) = \infty$
 - so dass B nicht über C nach A routet

Vergleich der Routing Algorithmen

Link-State Routing

- ▶ zentrales Verfahren
- ▶ bei n Knoten Komplexität des Dijkstra-Verfahrens: $O(n^2)$
- ▶ effiziente Implementierungen schaffen $O(n \log n)$
- ▶ beschränkt Skalierbarkeit
- ▶ Nachrichtenaustausch: $O(ne)$ bei e Kanten
- ▶ Robustheit: Router können nur fehlerhafte Verbindungs-information weitergeben

Distance Vector Routing

- ▶ verteilter Algorithmus
- ▶ Konvergenzprobleme bei Zyklen
- ▶ beschränkt Skalierbarkeit
- ▶ Robustheit: Router können fehlerhafte Pfade weitergeben, Fehlerfortpflanzung möglich
- ▶ Fehlfunktion eines Routers wirkt sich auf andere aus

Warum Aufteilung in Autonome Systeme?

- ▶ Geringere Komplexität, geringerer Management-Overhead
 - ▶ Höhere Flexibilität, größere Individualität
-
- ▶ Strukturierte Zuweisung von Internet-Adressen
 - effizient möglich in einem AS
 - ▶ Routing Algorithmen nicht effizient ausführbar für das ganze Internet
 - Abstraktionsebenen reduzieren die Komplexität
 - ▶ Unterschiedliche Routing-Metriken innerhalb eines AS und zwischen AS
 - Intra AS: hohe Bandbreite, niedriger Delay
 - Inter AS: geringe Kosten
 - ▶ Organisationen verbergen ihre interne Struktur
 - ▶ Am häufigsten verwendete Intra-AS-Routing-Protokolle:
 - RIP: Routing Information Protocol
 - EIGRP: Enhanced Interior Gateway Routing Protocol (ehemals proprietäres Cisco Routing Protokoll, nun offener Standard)
 - OSPF: Open Shortest Path First
 - IS-IS: Intermediate System to Intermediate System (im wesentlich das gleiche wie OSPF)

Distance
Vector

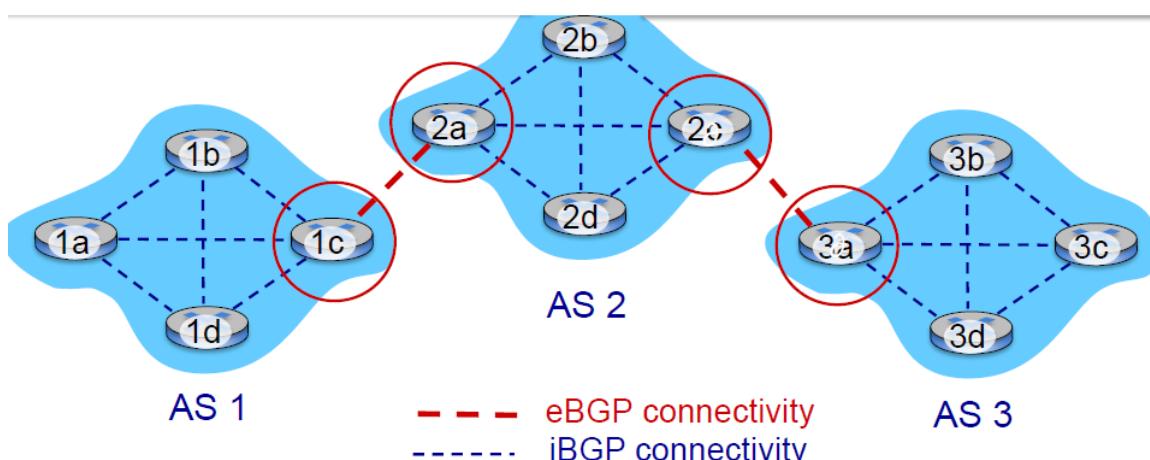
Link
State

Weitere Features von OSPF

- ▶ **Sicherheit:** Informationsaustausch zwischen OSPF-Routern kann authentifiziert werden, d.h. nur vertrauenswürdige Router
- ▶ **Mehrere Pfade mit gleichen Kosten:** OSPF erlaubt die parallele Nutzung mehrerer Pfade (im Gegensatz zu RIP; nur 1 Pfad)
- ▶ **ToS-aware Routing:** Für jeden Link, **unterschiedliche Routing-Metriken** in Abhängigkeit von ToS möglich: z.B. Kosten einer Satellitenverbindung niedrig für best effort ToS; hoch für real-time ToS
- ▶ Integrierte Unterstützung für Unicast- und Multicast Routing: **Multicast OSPF (MOSPF)** nutzt gleichen Topologie-Informationen wie OSPF
- ▶ Unterstützung einer **Hierarchie** innerhalb einzelner Routing-Domain

Internet Inter-AS Routing: BGP

- ▶ **BGP (Border Gateway Protocol)**
 - de Facto Inter-Domain Protokoll im Internet
 - Routing Protokoll basierend auf "Policies"
 - basiert auf einem Bellman-Ford Pfad-Vektor-Algorithmus
- ▶ BGP bietet für jedes AS folgende Mittel:
 - **eBGP:** erhalte Informationen über Erreichbarkeit von Subnetzen von benachbarten AS
 - **iBGP:** propage diese Informationen über Erreichbarkeit an alle Router innerhalb des AS
 - bestimme "gute" Routen zu anderen Netzen anhand von Erreichbarkeit und Routing-Policy
- ▶ ermöglicht jedem Subnetz, seine Existenz dem Rest des Internets bekanntzumachen

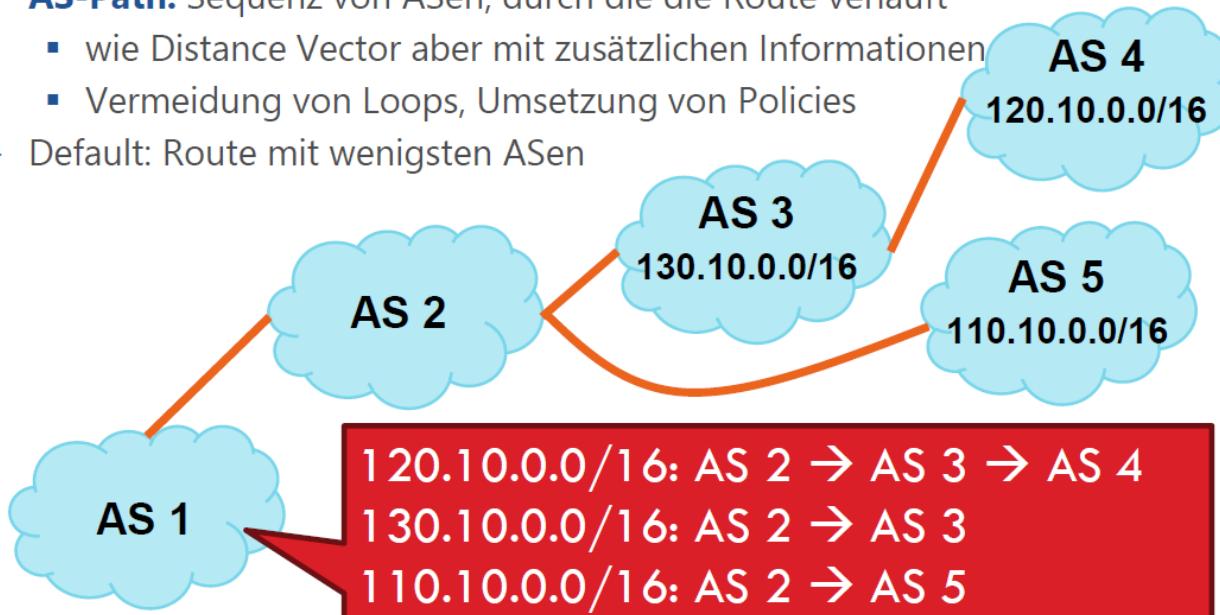


- ▶ Wenn AS3 Gateway Router 3a den Pfad „AS3,X“ dem AS2 Gateway Router 2c bekanntmacht:
 - AS3 **verspricht** AS2, dass Pakete nach X weitergeleitet werden

- ▶ **AS-Path:** Sequenz von ASen, durch die die Route verläuft

- wie Distance Vector aber mit zusätzlichen Informationen
- Vermeidung von Loops, Umsetzung von Policies

- ▶ Default: Route mit wenigsten ASen

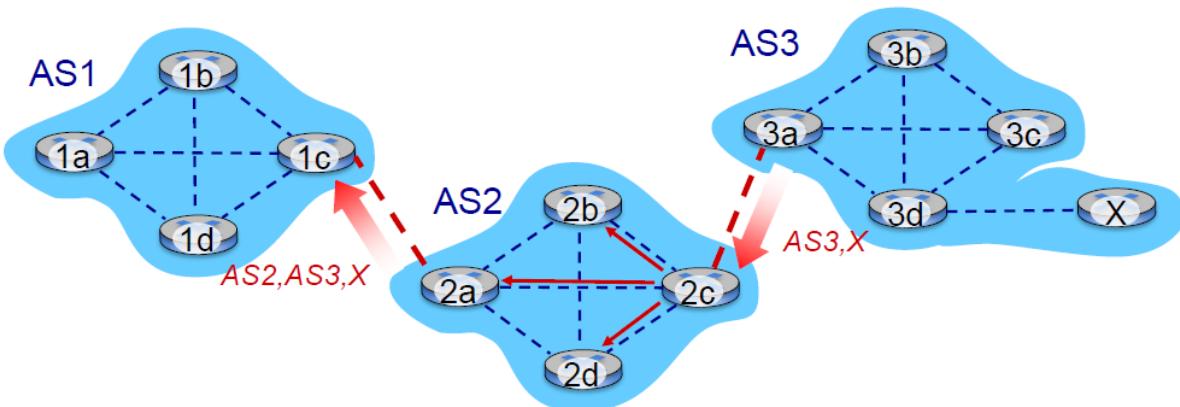


- ▶ Zwei wichtige Attribute

- **AS-PATH:** Liste der AS, durch die das Advertisement des Präfix geläufen ist
- **NEXT-HOP:** ist die Routerschnittstelle, bei der der AS-PATH beginnt

- ▶ Ein **Pfad** besteht aus

- Liste von Netzwerken (Adresse+Präfix, CIDR) in erreichbaren AS
- Sequenz von AS-Nummern zu diesem AS
- IP-Adresse des sendenden Gateways

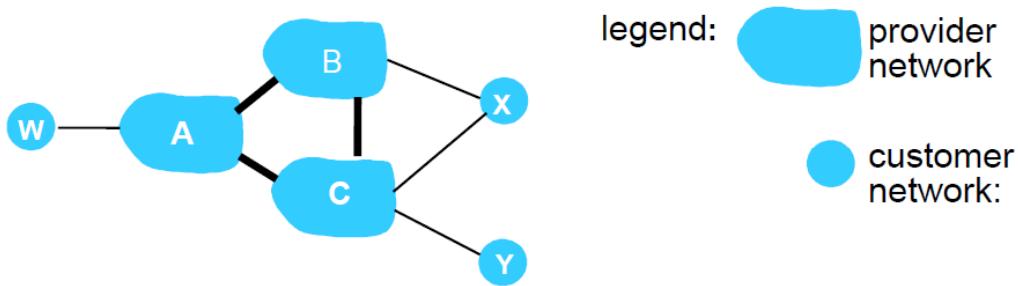


- ▶ AS2 Router 2c empfängt Advertisement AS3,X (via eBGP) von AS3 3a
- ▶ AS2 Policy: AS2 Router 2c akzeptiert Pfad AS3,X und propagiert (via iBGP) zu allen anderen AS2 Router
- ▶ AS2 Policy: AS2 Router 2a macht Pfad AS2, AS3, X zu AS1 Router 1c (via eBGP) bekannt

Gateway Router kann **mehrere Pfade** zum Ziel lernen

- ▶ AS1 Gateway Router 1c lernt Pfad AS2,AS3,X von 2a
- ▶ AS1 Gateway Router 1c lernt Pfad AS3,X von 3a
- ▶ Policy: AS1 Gateway Router 1c wählt Pfad AS3,X, und macht Pfad innerhalb von AS1 via iBGP bekannt

BGP: Routing Policy



Angenommen ein ISP möchte Verkehr zur zu/von Kundennetzen weiterleiten, aber keinen Transitverkehr zu anderen ISPs

- ▶ A macht Pfad Aw bekannt für B und C
- ▶ B macht Pfad BAw **nicht bekannt** für C (*no advertisement*):
 - B bekommt keinen "Revenue" für Weiterleitung CBAw, da C, A, w nicht Kunden von B sind
 - C erfährt nichts über Pfad CBAw
- ▶ C nutzt CAw (ohne B), um zu w zu gelangen

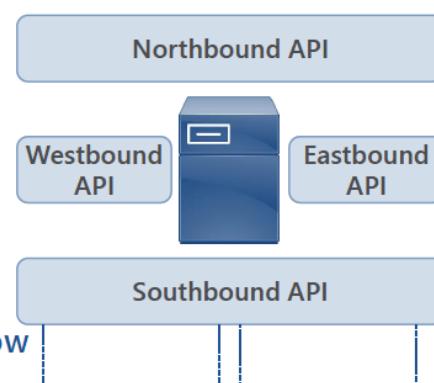
Warum unterschiedliche Intra- und Inter-AS Routing ?

- ▶ **Policy**
 - inter-AS: Administrator möchte kontrollieren, wie Verkehr vermittelt wird, welcher Verkehr durch Netz geht (politische, strategische Zielsetzungen)
 - intra-AS: einzelner Administrator, alles unter derselben Verwaltung, keine Policy nötig
- ▶ **Skalierbarkeit**
 - inter-AS: hierarchisches Routing reduziert Größe der Routing Tabellen, reduziert Update Verkehr
 - intra-AS: weniger wichtig, kann in mehrere AS aufgeteilt werden
- ▶ **Performanz**
 - intra-AS: fokussiert auf Leistung und Kosten des Routing Algorithmus
 - inter-AS: Policy dominiert über Leistung

Software-Defined Networking: Charakteristik

SDN Controller

- ▶ Entscheidet, **wohin** Daten geschickt werden
- ▶ **Softwareinstanz** auf Standardhardware
- ▶ Steuert Switches mittels Protokollen wie **OpenFlow**



Charakteristiken von SDN

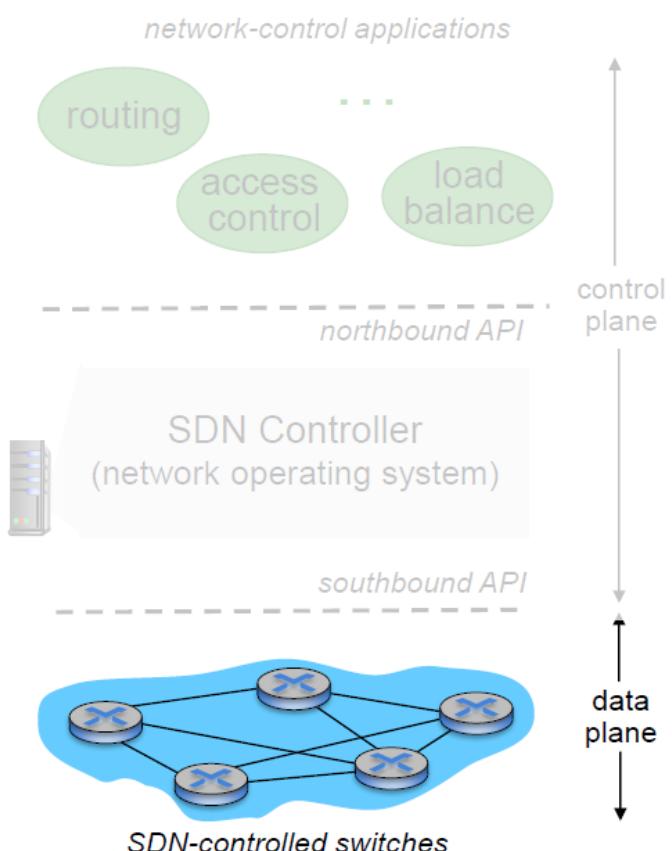
- ▶ **Entkopplung** von Control und Data Plane
- ▶ **Logisch zentralisierte** Control Plane
- ▶ **Offene Schnittstellen**
- ▶ **Programmierbarkeit**

Gründe für logisch zentralisierte Control Plane

- ▶ Forwarding mittels Tabellen erlaubt "**Programmierung**" von Routern
 - zentralisierte "Programmierung" einfacher: berechne Tabellen zentral und verteile diese
 - verteilte "Programmierung" schwieriger: berechne Tabelle als Ergebnis von verteilten Algorithmen (Routing Protokoll), in jedem Router implementiert
- ▶ **Offen:** standardisierte und einheitliche Schnittstellen der Control Plane, um Geräte zu programmieren
 - **Schnelle Innovation** durch offene Schnittstellen (z.B. OpenFlow)
 - Entkopplung von Control & Data Plane → unabhängige Entwicklung
- ▶ **einfacheres Netzwerkmanagement**
 - vermeide Router Fehlkonfigurationen
 - größere Flexibilität, Berücksichtigung von Ressourcen
 - automatisiertes Netzwerkmanagement

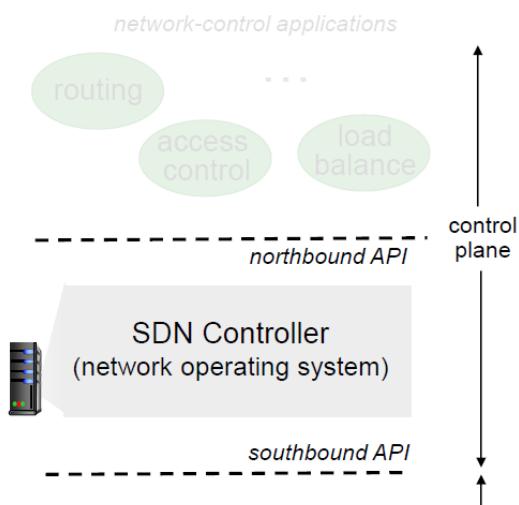
SDN Perspektive: Data Plane Switches

- ▶ schnelle, einfache Commodity Switches, die allgemeine Data Plane Weiterleitung in Hardware implementieren
- ▶ Switch Flow Table berechnet, vom Controller installiert
- ▶ API für Tabellen basierte Kontrolle des Switches (z.B. OpenFlow)
 - definiert, was kontrolliert werden kann und was nicht
- ▶ Protokoll zur Kommunikation mit Controller (z.B. OpenFlow)



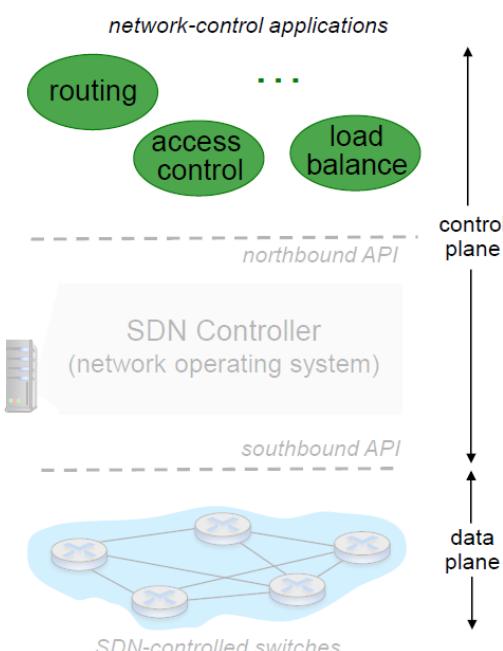
SDN Perspective: SDN Controller

- ▶ verwaltet Statusinformationen des Netzes
- ▶ interagiert mit Anwendungen zur Netzwerksteuerung über die "Northbound" API
- ▶ interagiert mit Network Switches über „Southbound“ API
- ▶ implementiert als verteiltes System zwecks Leistung, Skalierbarkeit, Fehlertoleranz, Robustheit
- ▶ **SDN controller: network OS**



SDN Perspektive: Network Control Applications

- ▶ „Intelligenz“ der Kontrolle: implementieren Steuerfunktionen mit Hilfe untergeordneter Diensten, API bereitgestellt vom SDN-Controller
- ▶ Kann von Drittanbietern bereitgestellt werden: unterschiedlich zum Routinganbieter oder SDN-Controller
- ▶ Beispiele: Network Monitoring, Intrusion Detection, Load Balancing, Routing, Cloud integration, Ressource Management für bessere Dienstgüte, ...



ICMP: Internet Control Message Protocol

- ▶ ICMP von Hosts und Routern verwendet, um Netzwerkschicht Informationen und Fehlermeldungen über IP auszutauschen
 - Fehlermeldungen: unreachable host, network, port, protocol
 - *ping*: echo request/reply
 - *traceroute*
- ▶ ICMP erkennt einige Fehlerzustände
- ▶ ICMP ist „oberhalb“ von IP
 - ICMP Nachrichten in IP Paketen übertragen
 - Aber auf Schicht 3, da zur Sicherung der Funktionalität der Netzwerkschicht
- ▶ Router und Hosts sprechen ICMP im Internet

Wird IP dadurch zuverlässig?

"Netzwerkmanagement beinhaltet Einsatz, Integration und Koordination von Hardware, Software und menschlichen Beteiligten, um das Netzwerk und die Ressourcen, aus denen es besteht, zu überwachen, zu testen, abzufragen, zu konfigurieren, zu analysieren, auszuwerten und zu steuern, um so die Leistung während des Betriebs in Echtzeit sowie die Dienstgüteanforderungen bei vernünftigen Kosten zu gewährleisten."

- ▶ Selbst für einfaches Netzwerk gibt es viele Szenarien, in denen ein Administrator geeignete Werkzeuge für Netzwerkmanagement benötigt
 - Entdecken von Fehlern
 - Host-Überwachung
 - Überwachung des Verkehrs zur besseren Planung der Ressourcen
 - Überwachung von Service Level Agreements (SLAs)
 - Erkennen von Eindringlingen
- ▶ Bereiche des Network Management
 - Fehlermanagement
 - Konfigurationsmanagement
 - Abrechnungsmanagement
 - Leistungsmanagement
 - Sicherheitsmanagement

*Fault Management,
Configuration Management,
Accounting Management,
Performance Management,
Security Management*

SNMP Protokoll: Nachrichtentypen

<u>Message type</u>	<u>Function</u>
GetRequest GetNextRequest GetBulkRequest	manager-to-agent: "get me data" (data instance, next data in list, block of data)
InformRequest	manager-to-manager: here's MIB value
SetRequest	manager-to-agent: set MIB value
Response	Agent-to-manager: value, response to Request
Trap	Agent-to-manager: inform manager of exceptional event

Sicherungsschicht - Kapitel 6

Dienste der Sicherungsschicht

► Flusskontrolle

- Anpassung der Senderate und des Kanalzugriffs zwischen benachbarten Knoten, etwa durch CSMA/CD

► Fehlererkennung

- Fehler verursacht durch Signaldämpfung, Rauschen
- Empfänger erkennt Fehler: signalisiert Sender eine Wiederholungsaufforderung oder verwirft Rahmen

► Fehlerkorrektur

- Empfänger identifiziert und korrigiert Bitfehler ohne Wiederholungsaufforderung

► Halbduplex und vollduplex Zugriff auf Medium

- halbduplex: Knoten auf beiden Enden den Links können übertragen, aber nicht gleichzeitig

Beispiel zur Fehlererkennung und -Korrektur

Codewort

$\underbrace{x_1 \ x_2 \ x_3 \ x_4}_X \quad \underbrace{y_5 \ y_6 \ y_7}_Y$

Codierungsvorschrift

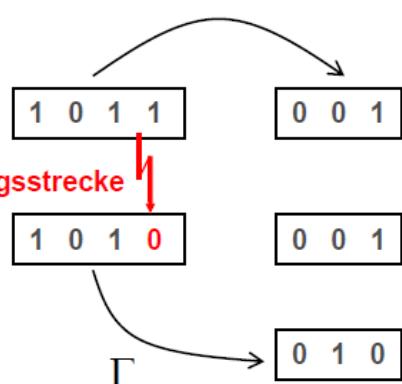
$$Y = \Gamma[X] \quad \begin{cases} y_5 = x_1 \oplus x_2 \oplus x_3 \\ y_6 = x_1 \oplus x_2 \oplus x_4 \\ y_7 = x_1 \oplus x_3 \oplus x_4 \end{cases}$$

($\oplus \cong \text{EXOR}$)

Matrixdarstellung

$$\begin{array}{cccc|ccc} x_1 & x_2 & x_3 & x_4 & y_5 & y_6 & y_7 \\ \hline 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{array}$$

gesendetes
Codewort
Störung der Übertragungsstrecke
empfangenes
Codewort



Fehlererkennung durch stellenweise
Addition von Y^* und Y'

$$Y^* \quad Y' \quad \text{Fehlersyndrom}$$
$$\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \oplus \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

→ Spalte 4, 4. Stelle gestört !

Paritätsprüfung

Einfaches Paritätsbit:

- erkennt einfache Bitfehler
 - gerade parität
 - Paritätsbit so wählen, dass 1er auch 1er bleiben
 - ungerade parität
 - Paritätsbit so wählen, dass 1er zu 0ern werden
- ← d data bits →
 parity
 bit

0111000110101011 0

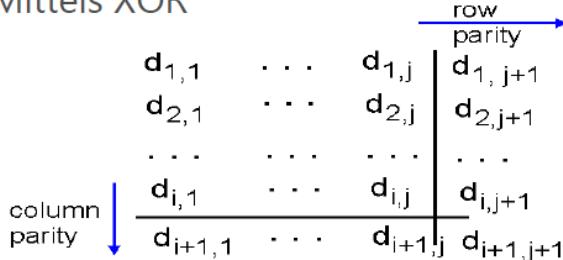
x_1	x_2	y_3
0	0	0
0	1	1
1	0	1
1	1	0
0	0	1
0	1	0
1	0	0
1	1	1

benutzte CW (even parity)

unbenutzte CW (odd parity)

Zweidimensionale Parität

- erkennt und korrigiert einfache Bitfehler
- Mittels XOR



101011
111100
011101

001010

no errors

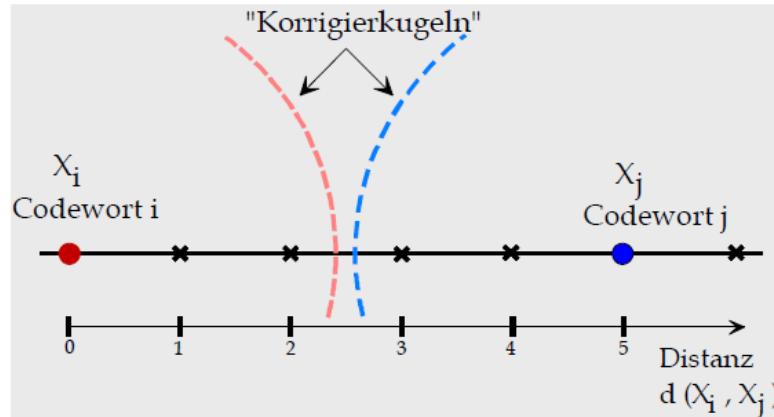
101011
101100
011101

001010

parity error
parity error
correctable single bit error

14

- ▶ Code mit $n=m+k$ Stellen $\Rightarrow 2^m$ Nutzwörter, 2^{m+k} Codewörter



- ▶ **Distanz** zwischen CW X_i und CW X_j (X_i, X_j sind benutzte CW)

$d(X_i, X_j)$: Anzahl unterschiedlicher Binärstellen

Hamming-Distanz

$$h = \min_{\forall i,j} \{d(X_i, X_j)\}$$

- ▶ Anzahl sicher erkennbarer Fehler

$$\bar{\eta} = h - 1$$

- ▶ Anzahl sicher korrigierbarer Fehler

$$\eta = \frac{h-2}{2} \quad \text{für gerade } h$$

$$\eta = \frac{h-1}{2} \quad \text{für ungerade } h$$

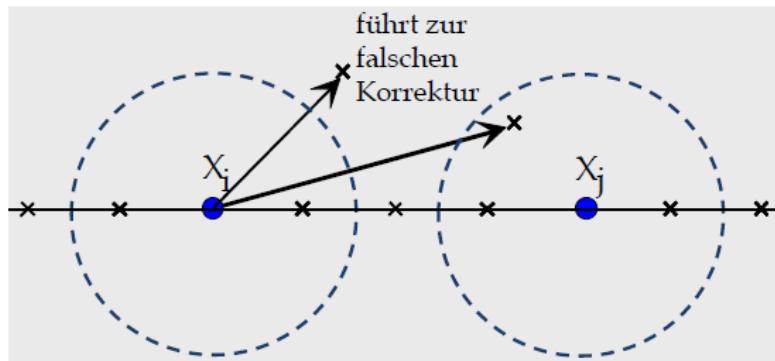
► **Satz:**

Bei einem linearen systematischen Code mit der Hamming-Distanz h sind $(h - 1)$ beliebig herausgewählte Prüfspalten voneinander linear unabhängig.

► **Spezialfall: Hamming-Distanz $h=3$**

- alle Prüfspalten müssen von 0 (Null-Spalte) verschieden sein
- alle Paare von Prüfspalten müssen linear unabhängig sein, d.h. alle Prüfspalten verschieden
- Mit $n = m + k$ Prüfspalten à k Elemente gilt

$$2^k - 1 \geq n = m + k$$



Restfehler

- Verfälschung von x_i in ein CW außerhalb des Korrigierbereichs von x_i bei Bitfehlerwahrs' p_b

$$P_{RK} = P\{\geq \eta \text{ Fehler}\} = \sum_{i=\eta+1}^n p_i = 1 - \sum_{i=0}^{\eta} p_i = 1 - \sum_{i=0}^{\eta} \binom{n}{i} p_b^i (1-p_b)^{n-i}$$

$$p_b \ll 1, \eta \ll n$$

$$P_{RK} \approx 1 - \sum_{i=0}^n \frac{(n \cdot p_b)^i}{i!} e^{-n \cdot p_b} \approx 1 - e^{-n \cdot p_b}$$

Restfehler

- Verfälschung von x_i in ein anderes, benutztes CW x_j

- Näherungsformel bei großer mittlerer Fehlerzahl

$$P_{RE} \approx \frac{\text{Anzahl anderer benutzter CW}}{\text{Anzahl anderer CW}} = \frac{2^m - 1}{2^n - 1} \approx 2^{-k}$$

- Anmerkung: Restfehler bezieht sich jeweils nur auf einen Übertragungsversuch

Cyclic Redundancy Check (CRC)

- ▶ Zyklische Redundanzprüfung, auch als Polynomcodes bekannt
- ▶ Bitfolgen werden als Koeffizienten eines binären Polynoms interpretiert:
 - $(d_{n-1}, d_{n-2}, \dots, d_1, d_0)$ entspricht $d_{n-1}x^{n-1} + d_{n-2}x^{n-2} + \dots + d_1x^1 + d_0x^0$
 - Beispiel: (10011001) entspricht $x^7+x^4+x^3+x^0$
- ▶ Operationen
 - Addition = Subtraktion = XOR (jeweils stellenweise, ohne Übertrag)
 - Multiplikation mit Potenz x^n entspricht Multiplikation der Binärzahl mit 2^n bzw. einer Schiebeoperation um n Bits
 - Division wie im 10-er System (euklidische Division): $(1011) \div (11)$
- ▶ Nutzdaten **D** mit d Bits, Prüfdaten **R** mit r Bits, Senden von **(D,R)**
- ▶ Wähle Generatorpolynom **G**, d.h. r+1 Bits (Bitmuster)
- ▶ Sender wählt R so, dass (D,R) ohne Rest durch G teilbar ist:
 - R ist Rest bei $D \cdot 2^r \div G$, Schreibweise: $\text{rem}(D \cdot 2^r \div G) = R$
 - (D,R) exakt teilbar durch G (modulo 2)
 - (D,R) entspricht $D \cdot 2^r + R$ und ist ohne Rest durch G teilbar
- ▶ Empfänger kennt G
 - teilt (D,R) durch G, kein Fehler falls Rest = 0

Prinzip bei CRC



Generatorpolynom G
Vom Grd r

$$D = (101110), \quad G = (1001), \quad r = 3$$

$$D \cdot 2^r = (101110000)$$

$$D \cdot 2^r \div G$$

$$R = (011)$$

$$(D, R) = (101110011)$$

1	101110000	
1	1001	

0	1010000	
0	0000	

1	1010000	
1	1001	

0	11000	
0	0000	

1	11000	
1	1001	

1	1010	
1	1001	

	11 << remainder	
->	101011 << quotient	

- ▶ Eigenschaften bei der Fehlererkennung
 - alle Einzelbitfehler, wenn Koeffizienten von x^r und x^0 gleich Eins
 - alle Doppelbitfehler, falls G unzerlegbaren Faktor mit mindestens 3 Termen enthält
 - jede ungerade Bitfehlerzahl, falls G den Faktor $(x+1)$ enthält
 - jeder Fehlerburst, der kürzer als r Bits ist (die meisten längeren Fehlerbursts können ebenfalls erkannt werden)

Möglichkeiten für den Mehrfachzugriff

▶ Feste Kanalaufteilung (channel allocation)

- durch geeignete Multiplexverfahren (s. Kapitel 2): Medium wird in feste Kanäle für exklusiven Zugriff aufgeteilt
- Beispiel: Frequenzmultiplex (FDMA), Zeitmultiplex (TDMA)
- Nachteil: ineffiziente Nutzung des Mediums bei geringer Auslastung



▶ Zyklische Zuteilung (taking turns)

- bessere Auslastung durch Zuteilung nach Sendebedarf
- zentralisiert: Polling durch zentralen Knoten
- verteilt: Sendeerlaubnis durch Token

▶ Random access (dynamic)

- Stationen greifen zufällig auf Medium zu, eventuelle gleichzeitige Übertragungen (Kollisionen) müssen beachtet werden
- Urform: ALOHA, abgeleitet: MAC bei Ethernet und WLAN, Kabelnetze

Polling

- ▶ Sendeerlaubnis wird den Knoten sukzessive zugewiesen, durch
 - zentralen Knoten
 - zufällig ausgewählten Knoten
 - ein verteiltes Protokoll
- ▶ Reihenfolge zyklisch oder anders (z.B. prioritätsgesteuert)
- ▶ Zykluszeit: Zeit, bis Sendeerlaubnis zu Knoten zurückkommt
- ▶ Nachteile
 - Overhead
 - Latenz durch Zykluszeit
 - zentraler Knoten ist „Single-Point-of-Failure“

Token Passing

- ▶ Token wird von Knoten zu Knoten weitergereicht
- ▶ Nachteile
 - Token Overhead
 - Latenz
 - Single point of failure (token)

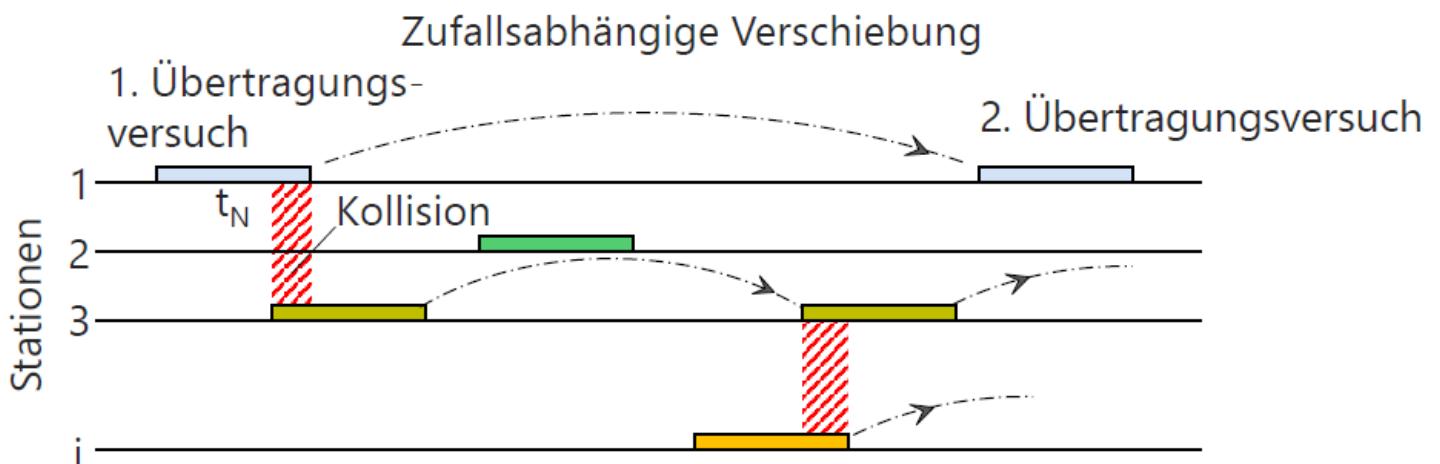
Zufälliger Medienzugriff (Random Access)

- ▶ Prinzipielles Verhalten von Random Access Protokollen
- ▶ Wenn Knoten Rahmen zum Senden hat
 - sendet mit der vollen Bitrate R des Mediums
 - keine a priori Koordination der Knoten
- ▶ Wenn Knoten gleichzeitig senden, überlagern sich die Signale auf dem Medium und zerstören sich gegenseitig: **Kollision**
 - Kollision kann durch Sendewiederholung behoben werden
 - Grundidee: bei schwacher Last ist dies selten
- ▶ unterschiedliche Verfahren zur Vermeidung und Erkennung von Kollisionen
 - ALOHA, slotted ALOHA
 - Carrier Sense Multiple Access (CSMA)
 - mit Collision Detection: CSMA/CD (in Ethernet über Bus)
 - mit Collision Avoidance: CSMA/CA (in WLANs)



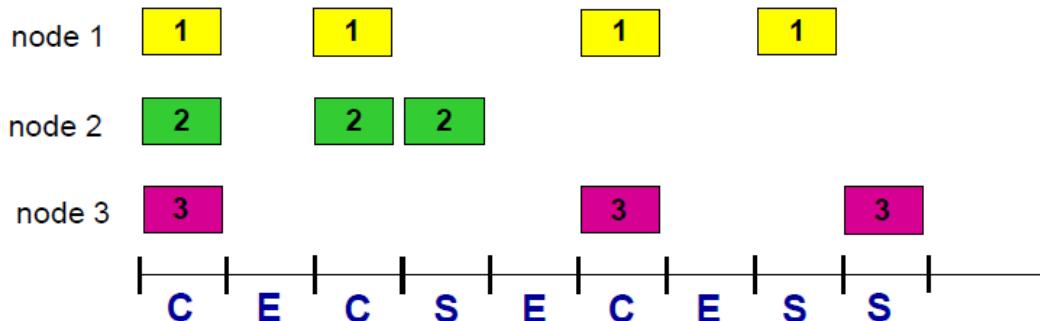
ALOHA

- ▶ Verfahren:
 - wenn die MAC-Schicht eines Knotens von der Netzwerkschicht ein Datagramm erhält, wird der Rahmen sofort gesendet
 - wenn der Empfänger ihn fehlerlos erhält, sendet er eine positive Bestätigung (**ACK**) zurück
 - wenn nach einem Timeout kein ACK zurückkommt, wartet der Sender eine zufällige Wartezeit (**Backoff**) und wiederholt dann das Senden



Slotted ALOHA

- ▶ Alle Rahmen haben gleiche Größe
- ▶ Zeit in **Slots** gleicher Größe eingeteilt (zur Übertragung eines Rahmens)
- ▶ Alle Knoten **synchronisieren** ihre Slots (z.B. durch zentrales Zeitsignal)
- ▶ Sendebeginn nur zu Beginn eines Slots



- ▶ Wenn Knoten einen neuen Rahmen zum Senden hat, wird dieser im nächstmöglichen Slot übertragen
 - Falls keine Kollision: Knoten sendet neuen Rahmen in nächsten Slot
 - Kollision: Knoten wiederholt Rahmen in darauffolgenden Slots mit Wahrscheinlichkeit p , bis erfolgreich
- ▶ **Effizienz:** Anteil erfolgreicher Slots, betrachtet über langen Zeitraum und viele Knoten, die alle zu senden haben
- ▶ Annahme
 - N Knoten, die alle Frames senden möchten
 - Jeder Knoten überträgt in einem Time Slot mit Wahrscheinlichkeit p
- ▶ Wahrscheinlichkeit, dass ein bestimmter Knoten in einem Time Slot erfolgreich ist $\rho \cdot (1-\rho)^{N-1}$
- ▶ Wahrscheinlichkeit, dass irgendeiner der N Knoten erfolgreich ist $N \cdot \rho \cdot (1-\rho)^{N-1}$
- ▶ Wahrs', dass irgendein Knoten erfolgreich überträgt $\underbrace{N \cdot \rho \cdot (1-\rho)^{N-1}}_{u} = f(\rho) \quad \underbrace{v}_{\text{V}}$

$$F(p) = 0 \Rightarrow p = 1 \vee N - N^2 p = 0$$

$$\Rightarrow p = 1 \vee p = \frac{1}{N}$$

Also $p^* = \frac{1}{N}$

$$\lim_{N \rightarrow \infty} F(p^*) = \lim_{N \rightarrow \infty} F\left(\frac{1}{N}\right)$$

$$= \exp(-1) = \frac{1}{e} \approx 37\%$$

- ▶ Vorteil
 - löst MAC Problem
 - aktiver Knoten nutzt komplett Übertragungsrate aus
 - dezentral: keine Koordination notwendig (bis auf Synchronisation der Slots)
 - einfach
- ▶ Nachteil
 - Kollisionen, verschwendete Slots
 - unbenutzte Slots
 - Knoten könnten Kollisionen schneller erkennen: nicht ausgenutzt
 - Uhrensynchronisation
 - Backoffzeit unabhängig von Last

- ▶ Idee:
 - Backoff an aktuelle Last anpassen (statt festes p)
 - **niedrige Last:** vermutlich sind nur wenige Knoten an der Kollision beteiligt, Auswahl von K aus wenigen Möglichkeiten reicht
 - **höhere Last:** mehr kollidierende Knoten, Auswahl von K aus mehr Möglichkeiten, größere mittlere Backoffzeit

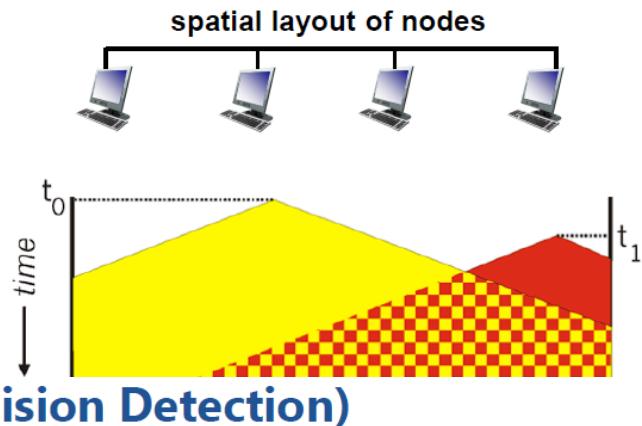
- ▶ Konkret
 - erste Kollision: gleichverteilte Auswahl von K aus {0,1}
 - zweite Kollision: gleichverteilte Auswahl von K aus {0,1,2,3}
 - ...
 - m-te Kollision: gleichverteilte Auswahl von K aus {0,1,2,3,4,..., 2^m-1}
 - nach einer maximalen Zahl M von Kollisionen (z.B. M = 10), bricht die MAC-Schicht ab und meldet einen Fehler an die Netzwerkschicht

- ▶ Wir betrachten Kollisionen zwischen zwei Knoten

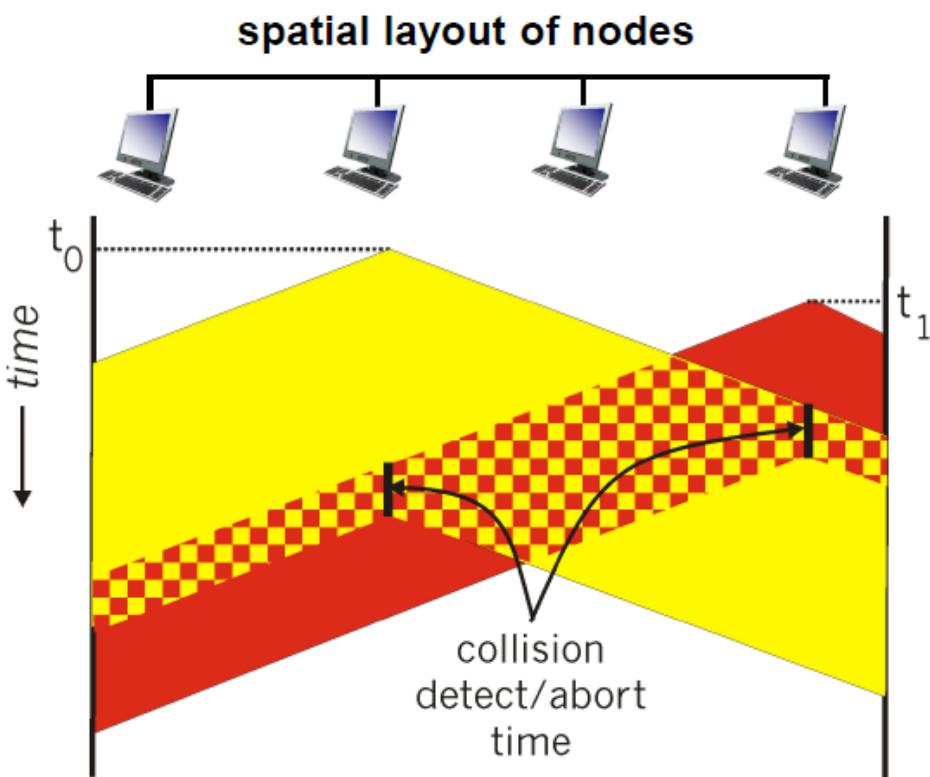
- ▶ Nach m-ter Kollision
 - Erfolgswahrscheinlichkeit: $1 - 2^{-m}$
 - Durchschnittliche Verzögerung (in Slots): $\frac{1}{2} + 2^{m-1}$

CSMA (carrier sense multiple access)

- ▶ Knoten prüfen vor dem Senden, ob Medium belegt
(listen before talking)
 - reduziert Kollisionen
 - Voraussetzung: Ausbreitungsverzögerung < Rahmensendezeit
(sonst sinnlos)
- ▶ Kollisionen immer noch möglich:
wenn anderer Knoten startet,
bevor sich das Signal auf dem
Medium zu ihm ausgebreitet hat
- ▶ Ausbreitungsverzögerung
bedeutet, dass zwei Knoten sich
nicht gegenseitig hören



- ▶ CSMA/CD: carrier sensing wie bei CSMA
 - Knoten besitzen Hardware, um während des Sendens Kollision zu erkennen (**listen while talking**)
 - nach Kollisionserkennung
 - Senden wird abgebrochen (weniger Verschwendung),
 - Jamming-Signal wird gesendet, damit alle Knoten Kollision sicher erkennen



CSMA Varianten

► 1-persistent

- wenn das Medium belegt ist, wartet der Knoten, bis es frei ist, und sendet dann sofort
- geringe Wartezeit, aber mögliche neue Kollision, wenn mehrere Knoten auf freies Medium warten

► nicht-persistent

- wenn das Medium belegt ist, geht der Knoten in Backoff
- weniger Kollisionen, aber längere Wartezeit

► p-persistent

- wenn das Medium belegt war und wieder frei ist, sendet der Knoten jeweils mit Wahrscheinlichkeit p (oder wartet noch einen Slot mit Wahrscheinlichkeit $1-p$)
- Kompromiss

► MAC Adresse (oder LAN oder physikalische oder Ethernet) Adressse

- Funktion: lokal genutzt, um Frame von einem Interface zu einem anderen physikalisch verbundenen Interface weiterzuleiten (d.h. im gleichen Netzwerk im Sinne der IP Adressierung)
- meist: 48 bit, also 6 Bytes, 12 Hexadezimalziffern MAC Adresse (für die meisten LANs), z.B: 1A-2F-BB-76-09-AD
- in ROM des Adapters eingebrannt
- verwaltet durch IEEE, global eindeutig, keine Strukturierung

► Verwaltung und Vergabe von MAC Adressen durch IEEE

- kann von Herstellern gekauft werden
- global eindeutig
- keine Strukturierung

► Analogie

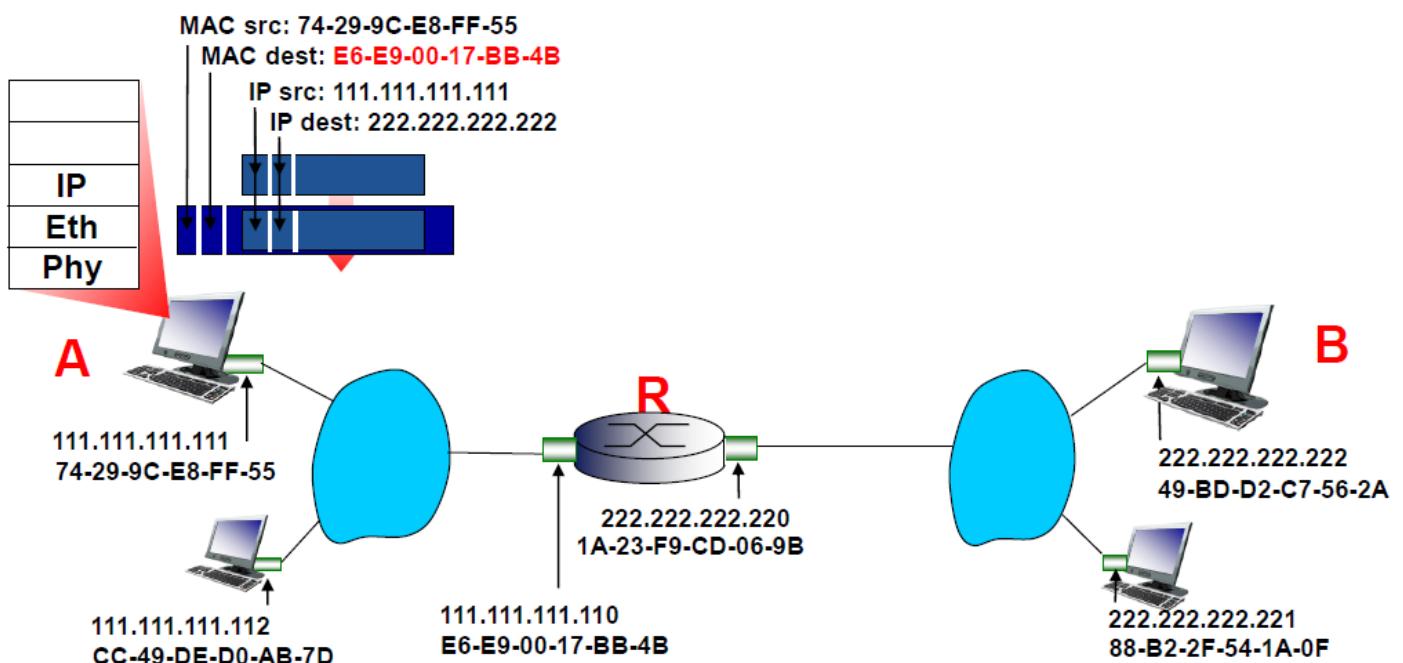
- MAC Adresse: Ausweisnummer eines Menschen
- IP Adresse: Postanschrift, Änderung bei Umzug in andere Stadt

Address Resolution Protocol (ARP)

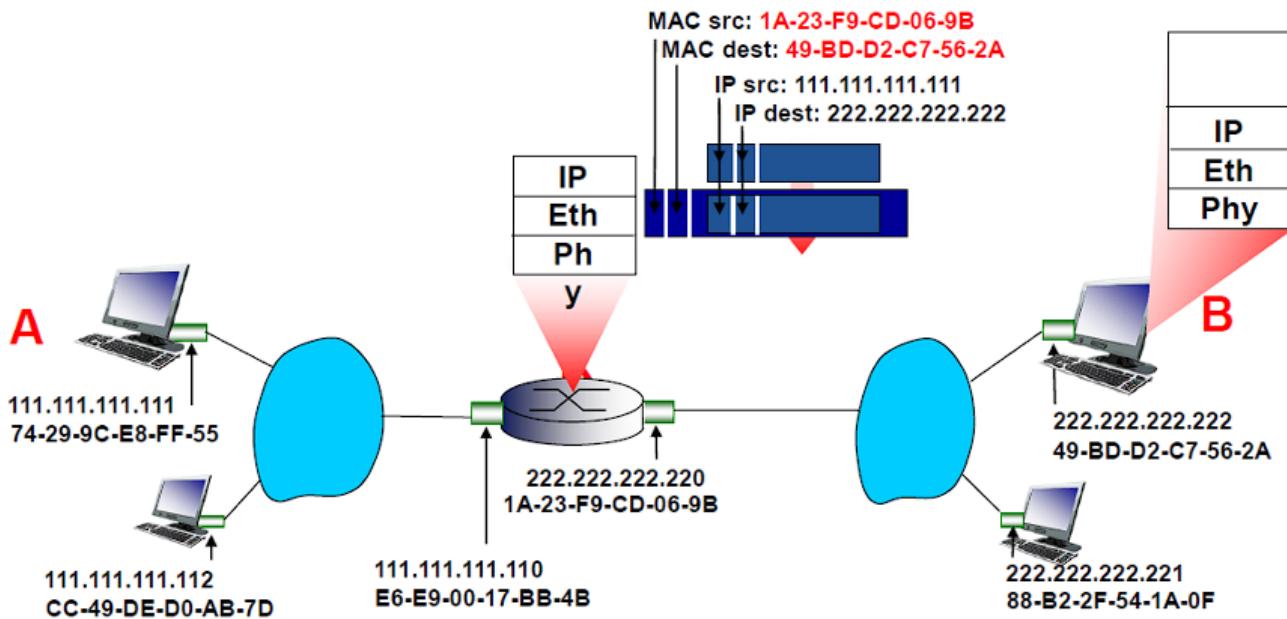
- ▶ IP-Adresse eines Zielknotens bekannt, wie wird physikalische Adresse gefunden? → ARP
- ▶ Jeder Knoten (Host, Router) im LAN besitzt ARP-Tabelle
- ▶ “**plug-and-play**”: ARP Tabellen werden automatisch gefüllt ohne Sys-Admin.
- ▶ Knoten A möchte an B Rahmen schicken, kennt IP-Adresse, aber nicht physikalische Adresse von B
- ▶ A sendet **ARP-Anfrage** als Broadcast (Adresse FF-FF-FF-FF-FF) mit seiner physikalischen Adresse und der IP-Adresse von B
- ▶ Alle Knoten im LAN erhalten Anfrage
- ▶ B erkennt sich als Ziel an IP-Adresse in der ARP-Anfrage und sendet in **ARP-Antwort** seine physikalische Adresse an die physikalische Adresse von A (unicast)
 - A speichert Adressen von B in seiner ARP-Tabelle
 - „Soft State“ mittels TTL

Adressierung: Routing in ein anderes LAN

- ▶ A erzeugt IP Datagramm mit IP Quelladresse A, IP Ziel B
- ▶ A erzeugt Rahmen mit MAC Adresse von R als Zieladresse, Rahmen beinhaltet IP Datagramm von A-zu-B



- der Adapter von A sendet den Rahmen auf LAN1
- R's Adapter in LAN1 empfängt den Rahmen und packt das Datagramm aus, gibt an IP Schicht weiter
- R findet in Routingtabelle heraus, dass B in LAN2 ist: Weiterleitung
- R erzeugt Rahmen mit seinem Adapter in LAN2 als physikalischer Quelladresse und B als Ziel MAC Adresse
- IP Quelladresse bleibt A, Rahmen enthält IP Datagramm A-zu-B

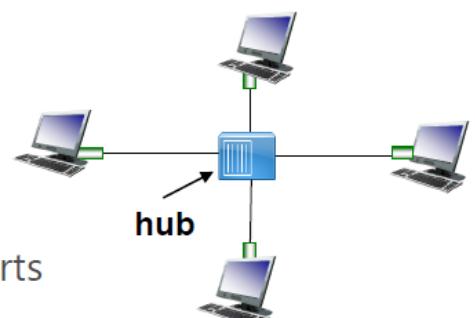


► Repeater

- Bitübertragungsschicht
- Signalregeneration auf Bit-Ebene (Leitungscodierung)

► Hub

- Paketweiterleitung, keine Filterung von Paketen
- Daten werden an alle Ports weitergeschickt, jedoch nicht an Ursprungsport
- Alle Ports teilen sich die verfügbare Bandbreite, d.h. Collision Domain ist die Gesamtheit aller Ports



► Switches

- wie bei Hubs, Segmentierung jedoch erlaubt
- jeder Port bildet somit eigene Collision Domain, jedem Port steht somit die gesamte Bandbreite zu Verfügung
- Steuerung über MAC-Table (Switching Table)
- Steuerungsmöglichkeiten: Selective Forwarding, Filtering, Flooding

- Sender Adapter verkapselt IP Datagramm (oder Packet eines anderen Netzwerkprotokolls) in Ethernet Rahmen



- Rahmenformat gleich in allen Ethernet Varianten (seit 1980er)

- **Präambel** (8 Bytes)

- 7 Bytes mit Muster 10101010, zur Synchronisation der Taktfrequenz (Ausgleich der Drift) vom Empfänger mit dem Sender
- 1 Byte mit Muster 10101011 zur Anzeige des Beginns der Zieladresse

- **Quell- und Zieladresse** (6 Bytes): physikalische Adresse

- Wenn Adapter Frame mit passender Zieladresse empfängt oder mit Broadcast Adresse (z.B. ARP) → Daten werden an Netzwerkschichtprotokoll (Type Feld) weitergegeben
- Ansonsten wird Frame verworfen

- **Type** (2 Bytes): Nummer für Protokoll der Nutzdaten (IP, Novell IPX, ...)

- **CRC** (4 Bytes): CCITT-32

- Wenn Fehler festgestellt wird, wird Rahmen verworfen

- **Nutzdaten** (46-1500 Bytes)

- Gesamtgröße: minimal 64 Bytes

Warum keine
Sendewiederholung bei
Ethernet im Fehlerfall?

- **unzuverlässig**: CRC Prüfung wird durchgeführt, aber empfangende NIC sendet keine Bestätigungen (ACKs oder NACKs)

- dadurch ist Ethernet einfach und kostengünstig zu implementieren
- Datagrammströme, die an Netzwerkschicht weitergeleitet werden, können lückenhaft sein
- verlorene Rahmen müssen von höherer Schichten behandelt werden (z.B. TCP)

- **Ethernet's MAC Protokoll**

- 1-persistentes CSMA/CD (unslotted)
- binärer exponentieller Backoff (s. oben)

- ▶ Woher weiß Switch, dass A' über Interface 4, B' über 5 erreichbar ist?

- ▶ Jeder Switch besitzt eine **Switch Table**, Weiterleitungstabelle mit folgenden Einträgen

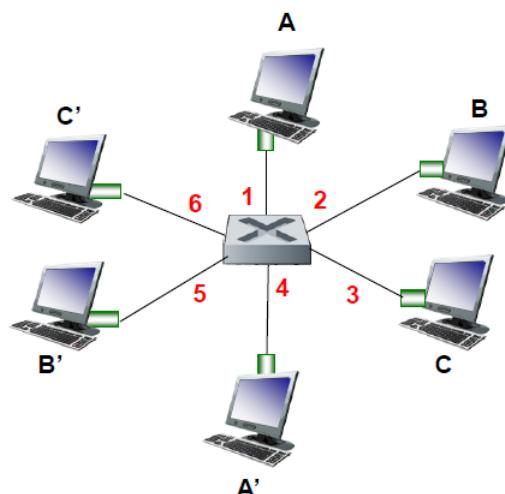
- MAC Adresse des Hosts, Interface, Zeitstempel
- Zeitstempel: Verfallszeit, danach wird Eintrag gelöscht
- (sieht aus wie Routing Tabelle)

- ▶ Wie werden die Einträge erstellt und gepflegt in der Tabelle?

- ähnlich wie Routing Protokoll?

- ▶ Switch lernt, welche Hosts über welche Interface erreicht werden können

- wenn Rahmen empfangen wird, lernt Switch den Ort des Senders: eingehendes LAN Segment
- für einen eingehenden Rahmen wird die Zuordnung von physikalischer Adresse und Portnummer in Tabelle gespeichert

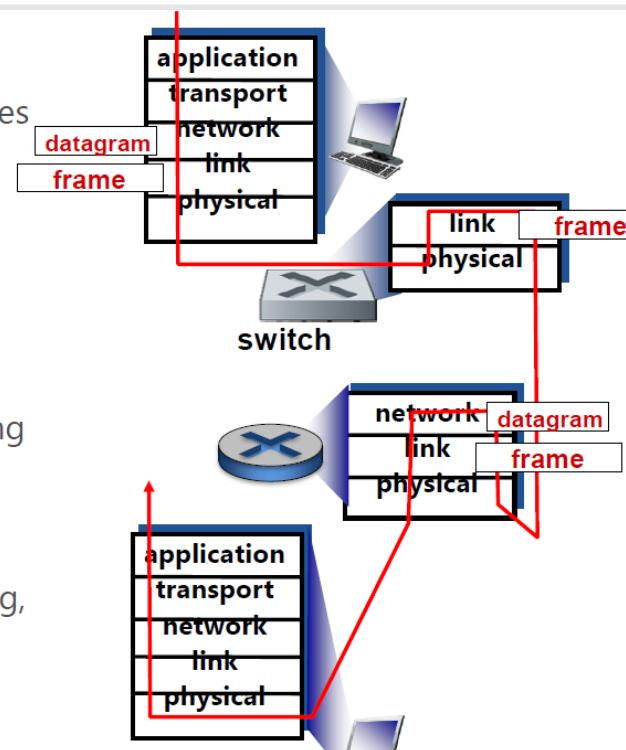


*switch with six interfaces
(1,2,3,4,5,6)*

Switches vs. Routers

▶ store-and-forward

- routers: network-layer devices (examine network-layer headers)
- switches: link-layer devices (examine link-layer headers)

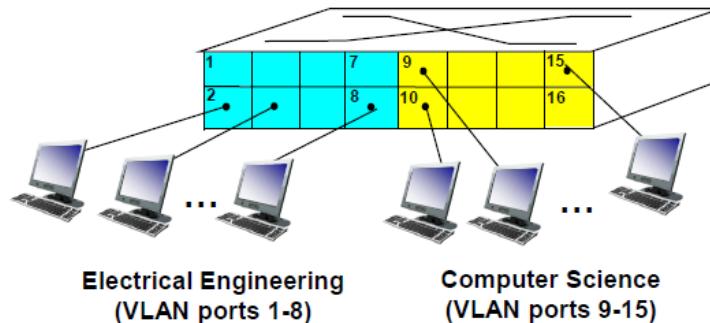


▶ forwarding tables

- routers: compute tables using routing algorithms, IP addresses
- switches: learn forwarding table using flooding, learning, MAC addresses

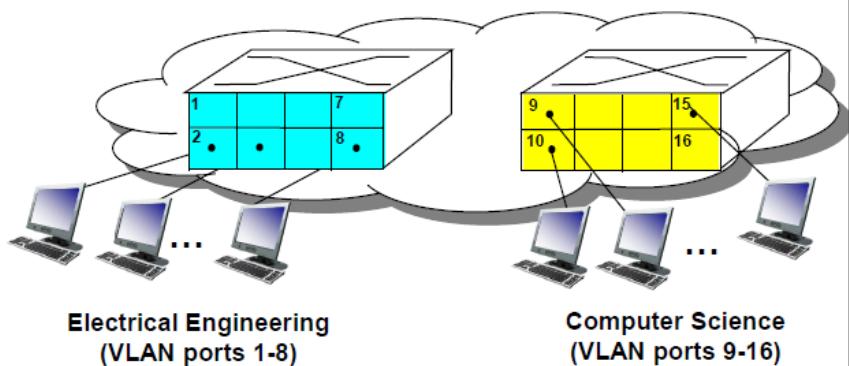
Port-basierte VLANs

- ▶ VLAN-fähige Switche können konfiguriert werden, um mehrere virtuelle LANs über physikalische Infrastruktur zu definieren



- ▶ Switch Ports zusammengefasst (Switch management software), so dass einzelner physikalischer Switch ...

... als mehrere virtuelle Switches agiert



Endgeräte an bestimmten Ports bilden VLAN

- ▶ **Traffic Isolation:** Rahmen werden von Switch nur innerhalb VLAN weitergeleitet
 - z.B. Rahmen auf Port 1-8 erreichen nur Ports 1-8

- ▶ Ports können dynamisch zu VLANs zugewiesen werden
- ▶ VLAN kann auch über MAC Adressen der Endpunkte definiert werden
- ▶ Inter-VLAN-Verkehr geht über Router
- ▶ Hersteller bieten oft Switch und Router in einem Gerät an