

## 9. Foliensatz

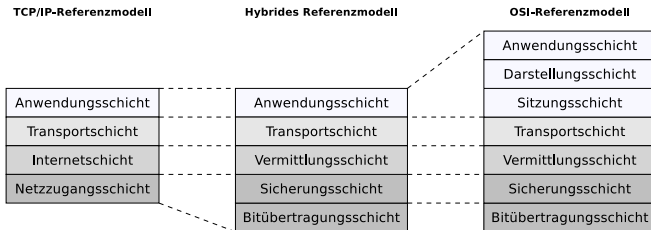
# Betriebssysteme und Rechnernetze

Prof. Dr. Christian Baun

Frankfurt University of Applied Sciences  
(1971–2014: Fachhochschule Frankfurt am Main)  
Fachbereich Informatik und Ingenieurwissenschaften  
[christianbaun@fb2.fra-uas.de](mailto:christianbaun@fb2.fra-uas.de)

# Sicherungsschicht

- Aufgaben der Sicherungsschicht (Data Link Layer):
  - Sender: Pakete der Vermittlungsschicht in Rahmen (Frames) verpacken
  - Empfänger: Rahmen im Bitstrom der Bitübertragungsschicht erkennen
  - Korrekte Übertragung der Rahmen innerhalb eines physischen Netzes gewährleisten durch Fehlererkennung mit Prüfsummen
  - Physische Adressen (MAC-Adressen) bereitstellen
  - Zugriff auf das Übertragungsmedium regeln



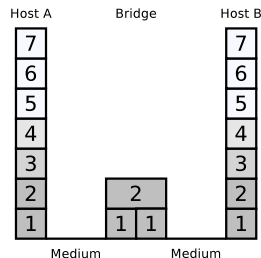
- Geräte: Bridge, Layer-2-Switch (Multiport-Bridge), Modem
- Protokolle: Ethernet, Token Ring, WLAN, Bluetooth, PPP

# Sinnvolle Themen zur Sicherungsschicht...

- ... und was aus Zeitgründen davon übrig bleibt...
- Geräte der Sicherungsschicht
  - Lernende Bridges
  - Kreise auf der Sicherungsschicht
  - ~~Spanning Tree Protocol~~
  - ~~Auswirkungen auf die Kollisionsdomäne~~
- Adressierung in der Sicherungsschicht
  - MAC-Adressen (Format, ~~Eindeutigkeit~~, ~~Sicherheit~~)
- ~~Rahmen abgrenzen (Längenangabe, Zeichen-/Bitstopfen, Regelverstöße)~~
- Rahmenformate aktueller Computernetze (Ethernet, ~~WLAN~~)
- Fehlererkennung
  - ~~Zweidimensionale Parität~~
  - Zyklische Redundanzprüfung
- ~~Fehlerkorrektur (Hamming Code)~~
- ~~Flusskontrolle (Stop-and-Wait-Protokoll, Schiebefensterprotokoll)~~
- ~~Medienzugriffsverfahren bei Ethernet und bei WLAN~~
- Adressauflösung mit ARP

## Geräte der Sicherungsschicht: Bridges

- Geräte der Bitübertragungsschicht verlängern physische Netze
  - Sollen aber Rahmen von einem physischen Netz in andere weitergeleitet werden, sind **Bridges** nötig
- Eine Bridge hat nur 2 Schnittstellen
- Einfaches Bridges leiten alle eintreffenden Rahmen weiter



- Bridges mit  $> 2$  Schnittstellen heißen **Multiport-Bridge** oder **Layer-2-Switch**
  - Sie haben typischerweise zwischen 4 und 48 Schnittstellen

# Arbeitsweise von Bridges und Layer-2-Switches

- Bridges und Switches untersuchen die Rahmen mit **Prüfsummen** auf Korrektheit
- Zum Filtern und Weiterleiten der Rahmen brauchen sie **keine Adresse**, da sie selbst nicht aktiv an der Kommunikation teilnehmen
  - Sie arbeiten wie die Geräte der Bitübertragungsschicht transparent
    - Grund: Sie kommunizieren nicht auf einer höheren Protokollschicht als der Sicherungsschicht

## Beispiel für Bridges im Alltag – WLAN-Bridge

- Ermöglicht die Integration von Geräten mit RJ45-Netzwerkanschluss (z.B. Netzwerkdrucker, Desktops, Spielkonsolen, . . . ) in ein lokales Funknetz (WLAN)
- Verbindet ein kabelgebundenes Netzwerk mit einem Funknetz



Bildquelle: Google Bildersuche

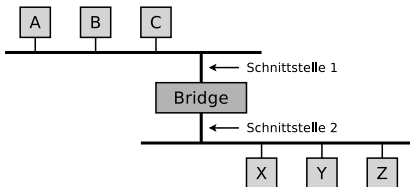
## Beispiel für Bridges im Alltag – Laser-Bridges

- Verbinden via Laserstrahl 2 Gebäude
  - Auf jedem Gebäude steht eine Laser-Sende-/Empfangseinheit
  - Die Bridges leiten Datenrahmen auf Schicht 2 (Sicherungsschicht) anhand der MAC-Adressen der Netzwerkgeräte weiter
  - Beide Bridges liegen im selben logischen Netz auf Schicht 3 (Vermittlungsschicht)



Bildquelle: <http://www.made-in-zelenograd.com>

# Lernende Bridges (1/2)



- Optimierung: **Lernende Bridges**
- Die Abbildung zeigt, dass es nicht sinnvoll ist, wenn eine Bridge alle Rahmen weiterleitet

- Kommt zum Beispiel ein Rahmen von Teilnehmer B für Teilnehmer A an Schnittstelle 1 der Bridge an, ist es nicht nötig, dass die Bridge diesen Rahmen über Schnittstelle 2 weiterleitet

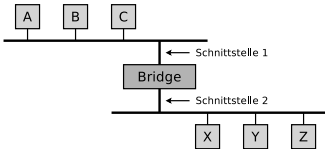
Gerät	Schnittstelle
A	1
B	1
C	1
X	2
Y	2
Z	2

- Bridges müssen lernen, welche Netzwerkgeräte über welchen Schnittstelle erreichbar sind
- Administratoren könnten die Tabellen in den Bridges pflegen
  - Das wäre sehr aufwändig
- Manuelle Eingriffe sind nicht nötig, da die Bridges ihre **Weiterleitungstabellen** selbst pflegen



# Lernende Bridges (2/2)

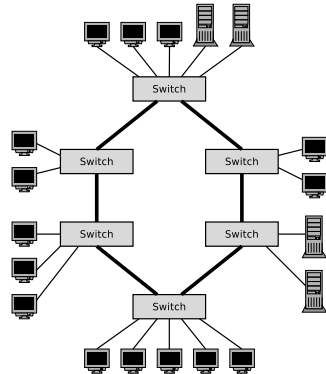
- Vorgehensweise:
  - **Bridges speichern die Quellen der Rahmen, die sie erreichen**
    - Wenn Gerät A ein Rahmen an einen anderen Host sendet, merkt sich die Bridge, dass der Rahmen von Host A an Schnittstelle 1 einging
  - So füllt sich die Weiterleitungstabelle mit der Zeit mit Einträge, welche Netzwerkgeräte sich in den verbundenen physischen Netzen befinden



- Beim Hochfahren einer Bridge ist ihre Weiterleitungstabelle leer
  - Die Einträge werden im Laufe der Zeit erfasst
  - Jeder Eintrag hat ein **Verfallsdatum** (Time to Live – TTL)
    - Sie sind nur eine bestimmte Zeit gültig
- Die Weiterleitungstabelle ist nicht unbedingt vollständig
  - Das ist aber kein Problem, da sie zur **Optimierung** dient
    - Existiert für ein Netzwerkgerät kein Eintrag in der Weiterleitungstabelle, leitet die Bridge den Rahmen in jedem Fall weiter

# Kreise auf der Sicherungsschicht

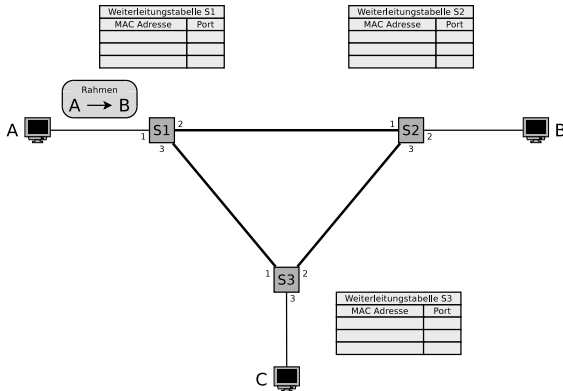
- Ein potentielles Problem sind **Kreise**
  - Computernetze sollten auf der Sicherungsschicht zu jedem möglichen Ziel immer nur einen Pfad haben
    - Das soll vermeiden, dass Rahmen dupliziert werden und mehrfach am Ziel eintreffen
  - Kreise können die Leistung des Netzes vermindern oder sogar zum Totalausfall führen
    - Andererseits dienen redundante Netzpfade als Backup für den Ausfall einer Leitung



## Gründe für das Entstehen von Kreisen auf der Sicherungsschicht

- Unachtsame Administratoren
- Absicht zum Ausgleich gestörter Verbindungen (Redundante Leitung)

# Beispiel für Kreise in einem LAN (1/6)

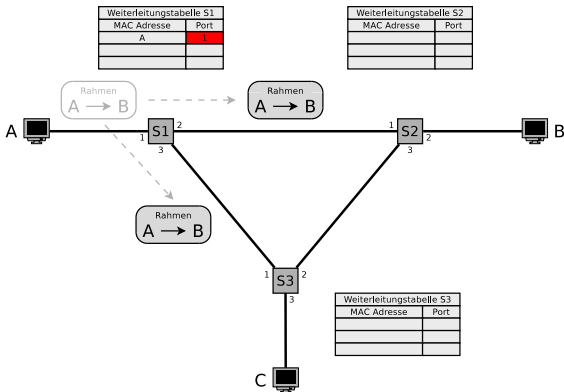


- Ein lokales Netz (LAN) hat Kreise auf der Sicherungsschicht
- Die Weiterleitungstabellen der Switches sind leer
- Im Beispiel will Knoten A einen Rahmen an Knoten B senden

Quellen für ähnliche Beispiele:

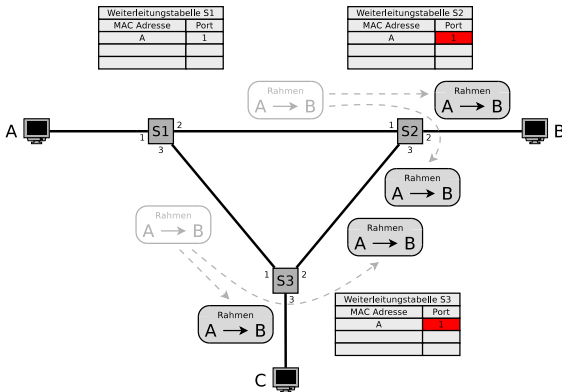
- Olivier Bonaventure. <http://cnp3book.info.ucl.ac.be/2nd/html/protocols/lan.html>
- Rüdiger Schreiner. Computernetzwerke. Hanser (2009)

# Beispiel für Kreise in einem LAN (2/6)



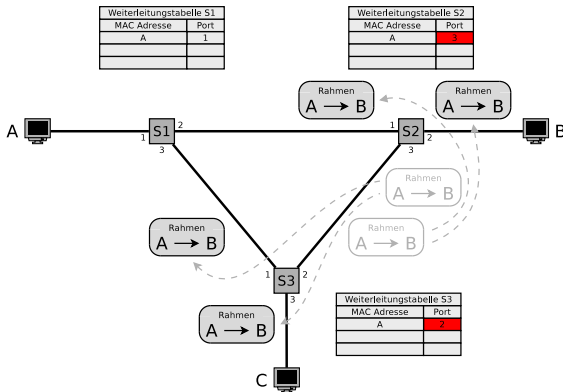
- Der Rahmen passiert Switch 1
- Switch 1 trägt den Port zu Knoten A in seine Tabelle ein
- Switch 1 kennt den Port zu Knoten C nicht
  - Darum sendet er Kopien des Rahmens über alle Ports (außer Port 1)

## Beispiel für Kreise in einem LAN (3/6)



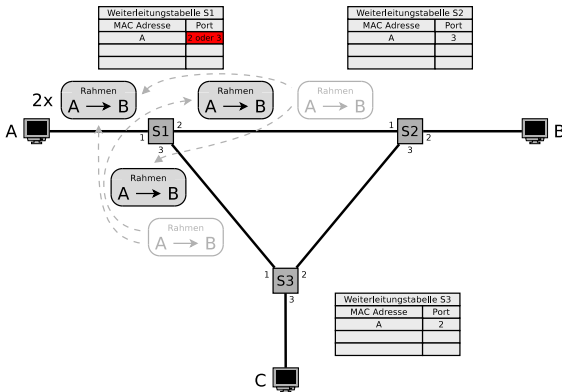
- Der Rahmen passiert Switch 2 und 3
- Switch 2 und 3 tragen den Port zu Knoten A in ihre Tabellen ein
- Switch 2 und 3 kennen den Port zu Knoten C nicht
  - Darum leiten Switch 2 und 3 Kopien des Rahmens über alle Ports weiter, außer über die Ports, an denen der Rahmen Switch 2 und 3 erreicht hat

## Beispiel für Kreise in einem LAN (4/6)



- Kopien des Rahmens passieren erneut Switch 2 und 3
- Switch 2 und 3 aktualisieren ihre Tabellen

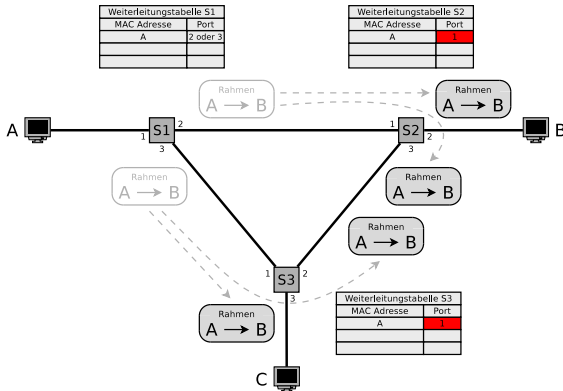
# Beispiel für Kreise in einem LAN (5/6)



- 2 Kopien des Rahmens erreichen Switch 1  
⇒ **Schleife!**
- Switch 1 sendet Kopien des Rahmens, die er über...
  - Port 2 empfangen hat an Port 1 und 3
  - Port 3 empfangen hat an Port 1 und 2
- Switch 1 aktualisiert seine Tabelle

- Die Reihenfolge, in der die Rahmen Switch 1 erreichen, kann nicht vorhergesagt werden

# Beispiel für Kreise in einem LAN (6/6)



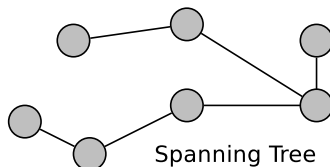
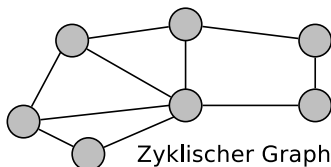
- Jeder Rahmen von Knoten A verursacht 2 Kopien, die endlos im Netz kreisen
  - Das Senden weiterer Rahmen durch Knoten A flutet das Netz und lässt es irgendwann zusammenbrechen

- Kopien des Rahmens passieren erneut Switch 2 und 3
- Switch 2 und 3 aktualisieren ihre Tabellen
- **Ethernet definiert keine TTL oder ein HopLimit**
  - Darum besteht die Schleife so lange, bis die Tabellen der Switche einen Eintrag für Knoten B enthalten



# Kreise im LAN handhaben

- Bridges müssen in der Lage sein, Kreise zu handhaben
- Lösung: **Spanning Tree Algorithmus**



- Ein Computernetz, das aus mehreren physischen Netzen besteht, ist ein Graph, der möglicherweise Kreise enthält
  - Der Spannbaum (Spanning Tree) ist ein Teilgraph des Graphen, der alle Knoten abdeckt, aber kreisfrei ist, weil Kanten entfernt wurden
  - Die Implementierung des Algorithmus ist das **Spanning Tree Protocol (STP)**

Eine detaillierte Beschreibung des STP wäre nun sinnvoll, aber leider fehlt im Rahmen der Vorlesung BSRN dazu die Zeit

# Adressierung auf der Sicherungsschicht

- Die Protokolle der Sicherungsschicht definieren das Format der physischen Adressen
- **Endgeräte (Hosts), Router und Layer-3-Switches** benötigen zwingend physische Adressen
  - Diese Geräte müssen auf der Sicherungsschicht adressierbar sein, um Dienste auf höheren Schichten anzubieten
- **Bridges und Layer-2-Switches** nehmen nicht aktiv an der Kommunikation teil
  - Darum brauchen sie für ihre Basisfunktionalität, also das Filtern und Weiterleiten der Rahmen, keine physischen Adressen
  - Bridges und Switches benötigen dann physische Adressen, wenn sie das STP zur Vermeidung von Kreisen anwenden, oder Dienste aus einer höheren Schicht anbieten
    - z.B. Monitoring-Dienste zur Überwachung oder grafische Weboberflächen zur Administration
- **Repeater und Hubs**, die nur auf der Bitübertragungsschicht arbeiten, haben keine Adressen

# MAC-Adressen (1/2)

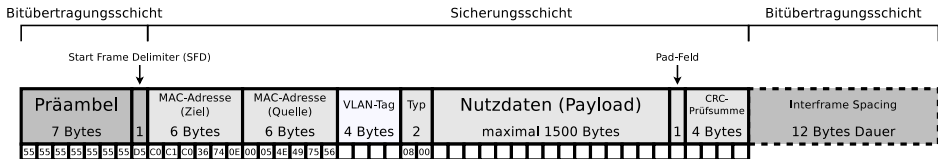
- **Physische Adressen** heißen **MAC-Adressen** (Media Access Control)
  - Sie sind unabhängig von den logischen Adressen der Vermittlungsschicht
- Ethernet verwendet das **Address Resolution Protocol** (ARP) um die logischen Adressen der Vermittlungsschicht (IPv4-Adressen) in MAC-Adressen aufzulösen
  - Bei IPv6 wird das **Neighbor Discovery Protocol** (NDP) verwendet, dessen Funktionalität identisch ist und das ähnlich arbeitet
- MAC-Adressen sind 48 Bits (6 Bytes) lang
  - Damit sind insgesamt  $2^{48}$  Adressen möglich
- Für eine kompakte und gut lesbare Darstellung sind MAC-Adressen meist in hexadezimaler Schreibweise geschrieben
  - Zudem sind die einzelnen Bytes durch Bindestriche oder Doppelpunkte voneinander getrennt
- Beispiel für diese Schreibweise: 00-16-41-52-DF-D7

## MAC-Adressen (2/2)

- Jede MAC-Adresse soll dauerhaft einem Netzwerkgerät zugewiesen und eindeutig sein
  - Es ist aber auch meist möglich, MAC-Adressen softwaremäßig zu ändern
    - Allerdings gilt die Änderung nur bis zum nächsten Neustart des Rechners
- **MAC-Broadcast-Adresse**
  - Will ein Netzwerkgerät einen Rahmen an alle anderen Geräte im gleichen physischen Netz senden, fügt es im Rahmen in das Feld der Zieladresse die Broadcast-Adresse ein
  - Bei dieser MAC-Adresse haben alle 48 Bits den Wert 1
  - Hexadezimale Schreibweise: FF-FF-FF-FF-FF-FF
  - Rahmen, die im Zielfeld die Broadcast-Adresse tragen, werden von Bridges und Switches nicht in andere physische Netze übertragen

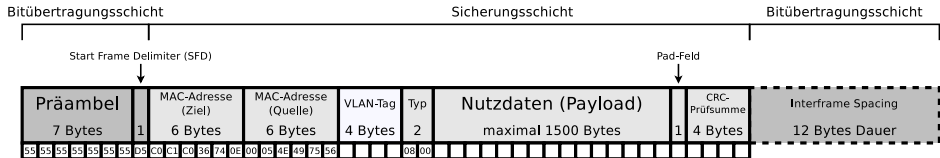
# Rahmenformate aktueller Computernetze (1/4) – Ethernet

- Um im Bitstrom der Bitübertragungsschicht die Rahmen zu erkennen und die Daten der Vermittlungsschicht in Rahmen zu verpacken, muss der Anfang jedes Rahmens markiert sein
  - Das ist nötig, damit der Empfänger die Rahmengrenzen erkennt



- Die Präambel besteht aus der 7 Bytes langen Bitfolge 101010...1010
  - Synchronisiert bei Bus-Topologien den Empfänger auf die Bit-Abstände
  - Darauf folgt der 1 Byte große SFD mit der Bitfolge 10101011

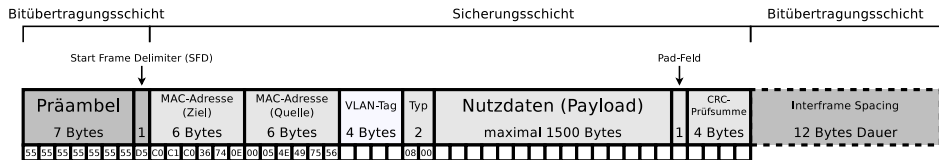
# Rahmenformate aktueller Computernetze (2/4) – Ethernet



- Die Datenfelder für die physischen Adressen (MAC-Adressen) von Sender und Ziel sind jeweils 6 Bytes lang
- Der 4 Bytes lange optionale VLAN-Tag enthält unter anderem...
  - eine 12 Bits lange VLAN-ID
  - und ein 3 Bits großes Feld zur Priorisierung
- Das Datenfeld Typ enthält das verwendete Protokoll der nächsthöheren Schicht

- Bei IPv4 hat das Datenfeld Typ den Wert 0x0800
- Bei IPv6 hat das Datenfeld Typ den Wert 0x86DD
- Enthalten die Nutzdaten eine ARP-Nachricht, hat das Datenfeld Typ den Wert 0x0806

# Rahmenformate aktueller Computernetze (3/4) – Ethernet



- Mindestgröße eines Ethernet-Rahmens: 72 Bytes
  - Maximale Größe (inkl. Präambel und SFD): 1526 Bytes
  - Der VLAN-Tag vergrößert die maximale Größe um 4 Bytes
- Jeder Rahmen kann maximal 1500 Bytes Nutzdaten enthalten
  - Mit dem Datenfeld Pad werden Rahmen bei Bedarf auf die erforderliche Mindestgröße von 72 Bytes gebracht
    - Das ist nötig, damit die Kollisionserkennung via CSMA/CD funktioniert
- Abschließend folgt eine 32 Bits lange Prüfsumme, die aber nicht die Präambel und den SFD einschließt

# Rahmenformate aktueller Computernetze (4/4) – Ethernet



- Das **Interframe Spacing** bzw. **Interframe Gap** ist der minimale zeitliche Abstand zwischen 2 gesendeten Rahmen auf dem Übertragungsmedium
- Die minimale Wartezeit entspricht 96 Bitzeiten (12 Bytes)
  - Bei 10 MBit/s sind es 9,6 Mikrosekunden
  - Bei 100 MBit/s sind es 0,96 Mikrosekunden
  - Bei 1 GBit/s sind es 96 Nanosekunden
- Einige Netzwerkgeräte erlauben es den Interframe Spacing zu verkürzen
  - Vorteil: Höherer Datendurchsatz ist möglich
  - Nachteil: Eventuell werden die Grenzen von Rahmen nicht korrekt erkannt ( $\implies$  Kollisionen nehmen zu)



# Fehlerursachen

- Bei der Übertragung von Bitfolgen kann es zu Fehlern kommen
- Typische Fehlerursachen sind
  - **Signalverformung**
    - Dämpfungseffekt des Übertragungsmediums
  - **Rauschen**
    - Thermisches und elektronisches Rauschen
  - **Nebensprechen**
    - Unerwünschter Einfluss benachbarter Kanäle (z.B. kapazitive Kopplung)
    - Kapazitive Kopplung nimmt mit wachsender Frequenz zu
  - **Kurzzeitstörungen**
    - Mehrere Bits hintereinander fehlerhaft (Burstfehler)
    - Kosmische Strahlung
    - Mangelhafte bzw. nicht ausreichende Isolierung
- Fehler müssen mindestens erkannt werden
  - Je nach Anwendungsfall kann es auch nötig sein, das Fehler korrigiert werden

# Fehlererkennung

- Zur Fehlererkennung wird jedem Rahmen vom Sender eine **Prüfsumme** angefügt
  - So werden fehlerhafte Rahmen vom Empfänger erkannt und verworfen

Das erneute Anfordern verworfener Rahmen durch den Empfänger sehen die Protokolle der Sicherungsschicht nicht vor

Diese Funktionalität erbringt das Transportprotokoll TCP ( $\implies$  Foliensatz 11)

- Möglichkeiten der Fehlererkennung:
  - ~~Zweidimensionale Parität~~ (wegen Zeitmangel gestrichen)
  - Zyklische Redundanzprüfung – **Cyclic Redundancy Check** (CRC)

# Zyklische Redundanzprüfung

- Zyklische Redundanzprüfung – Cyclic Redundancy Check (CRC) basiert darauf, dass man Bitfolgen als Polynome mit den Koeffizienten 0 und 1 schreiben kann
- Ein Rahmen mit  $k$  Bits wird als Polynom vom Grad  $k - 1$  betrachtet
  - Das werthöchste Bit ist der Koeffizient von  $x^{k-1}$
  - Das nächste Bit ist der Koeffizient für  $x^{k-2}$
  - ...

- Beispiel: Die Bitfolge 10011010 entspricht also dem Polynom:

$$\begin{aligned} M(x) &= 1 * x^7 + 0 * x^6 + 0 * x^5 + 1 * x^4 + 1 * x^3 + 0 * x^2 + 1 * x^1 + 0 * x^0 \\ &= x^7 + x^4 + x^3 + x^1 \end{aligned}$$

- Das Senden und Empfangen von Nachrichten kann man sich als **Austausch von Polynomen** vorstellen

## Polynome in der Mathematik

Ein Polynom ist die Summe von Vielfachen von Potenzen mit natürlichzahligen Exponenten einer Variablen

# Zyklische Redundanzprüfung (Vorgehensweise)

- Das Protokoll der Sicherungsschicht legt ein **Generatorpolynom** bzw. **Divisor-Polynom**  $C(x)$  fest
  - Die Auswahl des Generatorpolynoms legt fest, welche Fehlerarten erkannt werden
- $C(x)$  ist ein Polynom vom Grad  $k$ 
  - Sei z.B.  $C(x) = x^3 + x^2 + x^0 = 1101$ , dann ist  $k = 3$ 
    - Das Generatorpolynom ist also vom Grad 3

Der Grad des Generatorpolynoms entspricht der Anzahl der Bits minus eins

- Soll für einen Rahmen die CRC-Prüfsumme berechnet werden, werden  $n$  Nullen an den Rahmen angehängt
  - $n$  entspricht dem Grad des Generatorpolynoms

## Auswahl gängiger Generatorpolynome

Name	$C(x)$	Anwendung
CRC-5	$x^5 + x^2 + x^0$	USB
CRC-8	$x^8 + x^2 + x^1 + x^0$	ISDN
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + x^0$	Ethernet

# Beispiel zur zyklischen Redundanzprüfung (1/4)

Generatorpolynom:	100110
-------------------	--------

- Das Generatorpolynom hat 6 Stellen
  - Also werden 5 Nullen ergänzt

Rahmen (Nutzdaten):	10101
Rahmen mit Anhang:	1010100000

- Der Rahmen mit Anhang wird von links nur mit XOR durch das Generatorpolynom dividiert
  - $1 \text{ XOR } 1 = 0$ ,  $1 \text{ XOR } 0 = 1$ ,  $0 \text{ XOR } 1 = 1$ ,  $0 \text{ XOR } 0 = 0$
  - Dabei fängt man immer mit der ersten gemeinsamen 1 an
  - Der Rest (Restpolynom) ist die **Prüfsumme**

Der Sender  
berechnet  
die Prüfsumme

```

1010100000
100110||||
-----vv||
  110000||
  100110||
  -----v|
    101100|
    100110|
    -----v
      10100 = Rest
  
```

## Beispiel zur zyklischen Redundanzprüfung (2/4)

- Die Prüfsumme wird an die Nutzdaten angehängt
  - Der Rest muss aus  $n$  Bits bestehen
  - $n$  ist der Grad des Generatorpolynoms
- Ergebnis: 10100 wird an den Rahmen angehängt
- Übertragener Rahmen inkl. Prüfsumme (Codepolynom): 1010110100

Generatorpolynom:	100110
Rahmen (Nutzdaten):	10101
Rahmen mit Anhang:	1010100000
Rest (Restpolynom):	10100
Übertragener Rahmen (Codepolynom):	1010110100

### Den Rest auffüllen

- Der Rest muss aus  $n$  Bits bestehen und  $n$  ist der Grad des Generatorpolynoms
- Wenn der Rest kürzer ist als  $n$  muss mit Nullen *aufgefüllt* werden
- Beispiel: Der Rest ist 101 und  $n$  ist 5, dann ist die anzuhängende Prüfsumme 00101

## Beispiel zur zyklischen Redundanzprüfung (3/4)

Übertragener Rahmen (Codepolynom):	1010110100
Generatorpolynom:	100110

Der Empfänger  
testet, ob die  
Übertragung  
korrekt war

- Der Empfänger des Rahmens kann prüfen, ob dieser korrekt angekommen ist
- Mit einer Division (ausschließlich via XOR) durch das Generatorpolynom werden fehlerhafte Übertragungen erkannt
  - Bei der Division mit XOR immer mit der ersten gemeinsamen 1 anfangen!
- Ist der Rest der Division gleich null, war die Übertragung fehlerfrei

```

1010110100
100110||||
-----vv||
      110101||
      100110||
      -----v|
      100110|
      100110|
      -----v
              0
  
```

# Beispiel zur zyklischen Redundanzprüfung (4/4)

Übertragener Rahmen (Codepolynom):	1110110100
Generatorpolynom:	100110
<b>Korrekte Übertragung:</b>	1010110100

- Ist im übertragenen Rahmen 1 Bit fehlerhaft, ist der Rest der Division mit XOR durch das Generatorpolynom ungleich null
- Das Verfahren erkennt neben allen Einbitfehlern auch jede ungerade Anzahl fehlerhafter Bits und alle Bündelfehler der Länge  $n$ 
  - $n$  ist der Grad des Generatorpolynoms

```

1110110100
100110|||
-----v|||
111010|||
100110|||
-----v||
111001||
100110||
-----v|
111110|
100110|
-----v
110000
100110
-----
10110

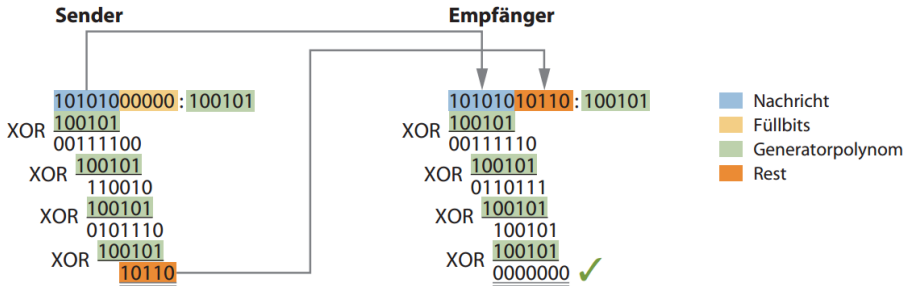
```

## Quellen

- Hompel, Büchter, Franzke. *Identifikationssysteme und Automatisierung*. Springer. 2008. S.219-221
- Meinel, Sack *Digitale Kommunikation: Vernetzen, Multimedia, Sicherheit*. Springer. 2009. S.122-124



# Zusammenfassung zur CRC-Vorgehensweise



Der CRC-Prüfwert ist der Rest der Division der Nachricht durch das Generatorpolynom. Dieselbe Rechnung mit der Nachricht plus angehängtem CRC ergibt beim Empfänger 0, wenn kein Übertragungsfehler aufgetreten ist.

Quelle: CRCs berechnen mit Intrinsic. Oliver Lau. c't 9/2013. S.184-185

## Arbeitsweise von ARP (1/2)

- Das **Address Resolution Protocol** (ARP) übersetzt IP-Adressen der Vermittlungsschicht in MAC-Adressen der Sicherungsschicht
- Will ein Netzwerkgerät Daten an einen Empfänger senden, gibt es auf der Vermittlungsschicht die IP-Adresse des Empfängers an
- Auf der Sicherungsschicht ist aber die MAC-Adresse nötig
  - Darum muss in der Sicherungsschicht die **Adressauflösung** erfolgen
  - Um die MAC-Adresse eines Geräts im LAN zu erfahren, sendet ARP einen Rahmen mit der MAC-Broadcast-Adresse FF-FF-FF-FF-FF-FF als Zieladresse
    - Diesen Rahmen nimmt jedes Netzwerkgerät entgegen und wertet ihn aus
    - Der Rahmen enthält die IP-Adresse des gesuchten Netzwerkgeräts
  - Fühlt sich ein Gerät mit dieser IP-Adresse angesprochen, schickt es eine ARP-Antwort an den Sender
    - Die gemeldete MAC-Adresse speichert der Sender im lokalen ARP-Cache

## Arbeitsweise von ARP (2/2)

- Der **ARP-Cache** dient zur Beschleunigung der Adressauflösung
  - Er enthält eine Tabelle mit folgenden Informationen für jeden Eintrag:
    - Protokolltyp (IP)
    - Protokolladresse des Senders (IP-Adresse)
    - Hardware-Adresse des Sender (MAC-Adresse)
    - Ablaufzeit – Time To Live (TTL)
  - Die TTL legt das Betriebssystem fest
  - Wird ein Eintrag in der Tabelle verwendet, verlängert sich die TTL
- Aktuelle Linux-Distributionen verwerfen Einträge nach  $\approx 5$  Minuten

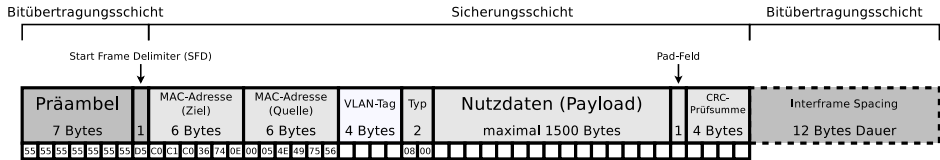
ARP-Cache ausgeben: `arp -n` oder alternativ `ip neighbour`

```
# arp -n
Address HWtype HWaddress      Flags Mask    Iface
192.168.178.1 ether 9c:c7:a6:b9:32:aa C          wlan0
192.168.178.24 ether d4:85:64:3b:9f:65 C          wlan0
192.168.178.41 ether ec:1f:72:70:08:25 C          wlan0
192.168.178.25 ether cc:3a:61:d3:b3:bc C          wlan0
```

Mit `arping` kann man manuell Anforderungen zur Adressauflösung versenden

# Aufbau von ARP-Nachrichten

- ARP-Nachrichten werden als Nutzdaten in Ethernet-Rahmen übertragen (Typ = 0x0806 für das ARP-Protokoll)



- H-Länge = Länge der HW-Adressen (MAC-Adressen) in Bytes
  - Bei Ethernet: 6 Bytes
- P-Länge = Länge der IP-Adressen in Bytes
  - Bei IPv4: 4 Bytes

Bei ARP-Anfragen ist der Inhalt des Felds MAC-Adresse (Ziel) egal

