Lernzettel - RvS

6. Februar 2018

Dieser Lernzettel wurde von Studenten als Hilfestellung zur RvS Klausurvorbereitung erstellt.

Dabei handelt es sich im Wesentlichen um ergänzendes Material zu den bisherigen Übungen.
Er erhebt keinerlei Anspruch auf Vollständigkeit - im Gegenteil:
Er deckt lediglich die wichtigsten Algorithmen und Rechenaufgaben aus den Übungen ab, nicht aber das Hintergrund- und Detailwissen, das in der Klausur abgefragt werden kann!
Zur Klausurvorbereitung solltet Ihr also in erster Linie Foliensätze und Übungen heranziehen.
Der Lernzettel sollte nicht zu Spekulationen über Klausurinhalte ermuntern, da die Ersteller keine näheren Informationen zu Euren Klausurinhalten kennen und selbige somit auch nicht in den Lernzettel hätten einfließen lassen können.

Inhaltsverzeichnis

1 ARP

ARP → "Address Resolution Protocol"

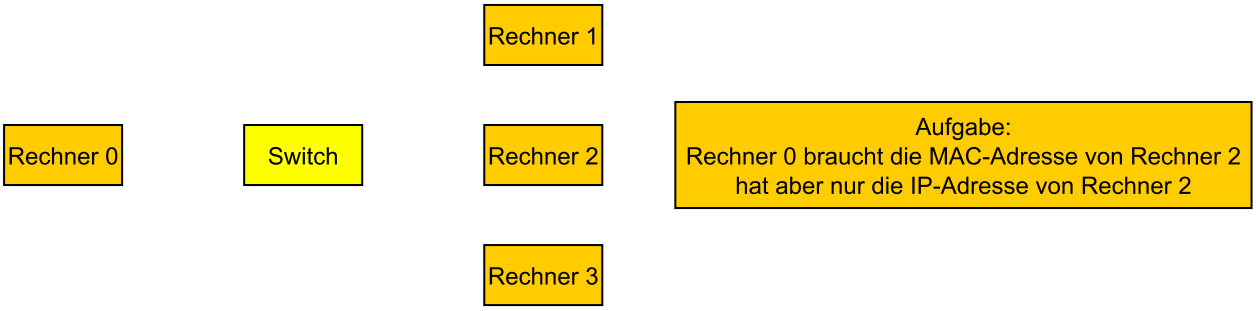


Abbildung 1: Aufbau des Netzes

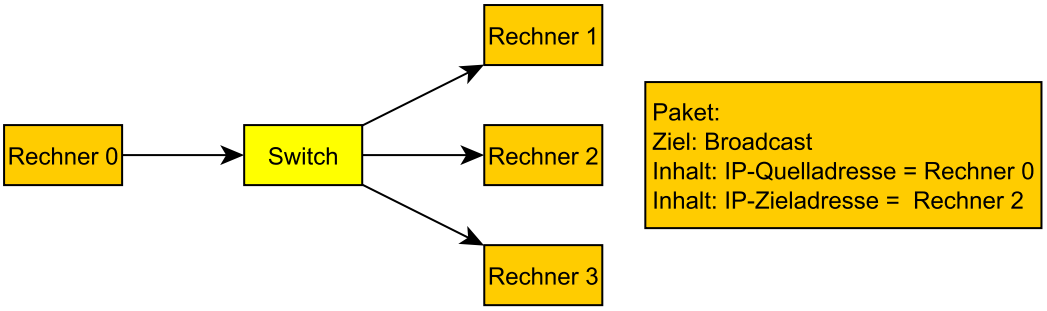


Abbildung 2: Anfrage

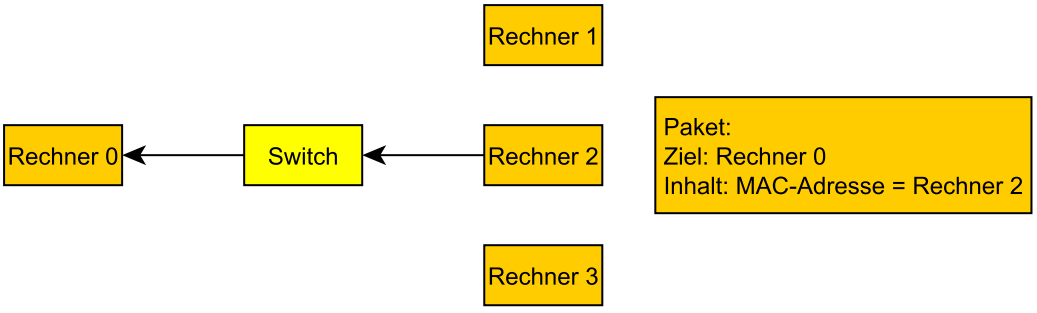


Abbildung 3: Antwort

2 Hamming-Distanz

Die Hamming-Distanz zwischen zwei Binärzahlen ist die Anzahl der sich unterscheidenden Bits.

Hamming-Distanz zwischen "10101011" und "01011001"

XOR	1	0	1	0	1	0	1	1	
	0	1	0	1	1	0	0	1	
$\sum$	1	1	1	1	0	0	1	0	**5**

Hamming-Distanz zwischen "11111111" und "00000000"

XOR	1	1	1	1	1	1	1	1	
	0	0	0	0	0	0	0	0	
$\sum$	1	1	1	1	1	1	1	1	**8**

Hamming-Distanz zwischen "10010100" und "10101011"

XOR	1	0	0	1	0	1	0	0	
	1	0	1	0	1	0	1	1	
$\sum$	0	0	1	1	1	1	1	1	**6**

3 Hamming-Code

Hamming-Code ist sowohl ein FEC als auch ein ECC
FEC → Forward Error Correction → Vorwärtsfehlerkorrektur
ECC → Error-Correcting Code → Fehlerkorrekturverfahren
Dabei können jedoch Fehler nur bis 2-Bit Fehler erkannt und nur 1-Bit Fehler korrigiert werden.

Check-Bits stehen an allen 2^x Potenzen also $2^0, 2^1, 2^2, 2^3, 2^4, 2^5$... aka 1,2,4,8,16,32,64...

Ein Bit wird für einen Check-Bit benutzt, wenn seine Binärcodierung eine "1" an einer Stelle hat, an der die Binärcodierung des Check-Bits auch eine hat.
Mit Binärcodierung ist die binäre Durchnummerierung gemeint.

Um einen Check-Bit zu berechnen müssen alle relevanten Bits XOR-verknüpft werden.
Den Prozess kann man auch relativ leicht abkürzen, indem man einfach die für einen Check-Bit relevanten "1"-Bits zusammen addiert.
Ergebnis gerade → Check-Bit = 0
Ergebnis ungerade → Check-Bit = 1

3.1 Erzeugen

"11000011" in Hamming-Code Codieren:

Position	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	
Original-Wort:	?	?	1	?	1	0	0	?	0	0	1	1	
1. Check-Bit:			1		1		0		0		1		1
2. Check-Bit:			1			0	0			0	1		0
3. Check-Bit:					1	0	0					1	0
4. Check-Bit:									0	0	1	1	0
Code-Wort:	1	0	1	0	1	0	0	0	0	0	1	1	

"11001010" in Hamming-Code Codieren:

Position	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	
Original-Wort:	?	?	1	?	1	0	0	?	1	0	1	0	
1. Check-Bit:			1		1		0		1		1		0
2. Check-Bit:			1			0	0			0	1		0
3. Check-Bit:					1	0	0					0	1
4. Check-Bit:									1	0	1	0	0
Code-Wort:	0	0	1	1	1	0	0	0	1	0	1	0	

"10101100" in Hamming-Code Codieren:

Position	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	
Original-Wort:	?	?	1	?	0	1	0	?	1	1	0	0	
1. Check-Bit:			1		0		0		1		0		0
2. Check-Bit:			1			1	0			1	0		1
3. Check-Bit:					0	1	0					0	1
4. Check-Bit:									1	1	0	0	0
Code-Wort:	0	1	1	1	0	1	0	0	1	1	0	0	

3.2 Überprüfen

Um zu überprüfen, ob ein Codewort korrekt/fehlerfrei übertragen wurde, müssen erneut die Check-Bits berechnet und überprüft werden, ob sie mit den mitgeschickten Check-Bits übereinstimmen.
Wenn alle Check-Bits übereinstimmen war die Übertragung fehlerfrei.
Wenn genau ein Check-Bit nicht übereinstimmt, ist es das Check-Bit selbst, das gekippt ist und kann somit ignoriert werden.
Wenn zwei oder mehr Check-Bits nicht übereinstimmen, kann man das defekte Bit finden, indem man die

Stelligkeit der Check-Bits addiert, wodurch man die Stelligkeit des Fehlers erhält.
Wenn zwei Daten-Bits nicht übereinstimmen ist mehr als nur ein Bit gekippt und dies ist mit Hamming-Code nicht reparierbar.

"101010001011" wurde in Hamming-Code übertragen:

Position(Dec)	1	2	3	4	5	6	7	8	9	10	11	12	
Position(Bin)	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	
Eingabe-Wort:	1	0	1	0	1	0	0	0	1	0	1	1	
1. Check-Bit:			1		1		0		1		1		0
2. Check-Bit:			1			0	0			0	1		0
3. Check-Bit:					1	0	0					1	0
4. Check-Bit:									1	0	1	1	1
Prüf-Wort:	0	0	1	0	1	0	0	1	1	0	1	1	
Ori. Code-Wort:	1	0	1	0	1	0	0	0	0	0	1	1	

Stelle 1 und Stelle 8 sind falsch, also 1+8, somit ist das gekippte Bit das 9te
Korrigiertes-Codewort: 101010000011
Ausgelesenes-Quellwort: 11000011

011110001010 wurde in Hamming-Code übertragen:

Position	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	
Original-Wort:	0	1	1	1	1	0	0	0	1	0	1	0	
1. Check-Bit:			1		1		0		1		1		0
2. Check-Bit:			1			0	0			0	1		0
3. Check-Bit:					1	0	0					0	1
4. Check-Bit:									1	0	1	0	0
Prüf-Wort:	0	0	1	1	1	0	0	0	1	0	1	0	
Ori. Code-Wort:	0	0	1	1	1	0	0	0	1	0	1	0	

Stelle 2 ist kaputt und da es ein Check-Bit ist (und nur ein Check-Bit) können wir es einfach ignorieren (bzw. kippen).
Korrigiertes-Codewort: 001110001010
Ausgelesenes-Quellwort: 11001010

"011101001100" wurde in Hamming-Code übertragen:

Position	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	
Original-Wort:	0	1	1	1	0	1	0	0	1	1	0	0	
1. Check-Bit:			1		0		0		1		0		0
2. Check-Bit:			1			1	0			1	0		1
3. Check-Bit:					0	1	0					0	1
4. Check-Bit:									1	1	0	0	0
Prüf-Wort:	0	1	1	1	0	1	0	0	1	1	0	0	
Ori. Code-Wort:	0	1	1	1	0	1	0	0	1	1	0	0	

Es wurde fehlerfrei übertragen.
Ausgelesenes-Quellwort: 10101100

3.3 Checkbit-Anzahl

n	k	$N = n + k$
Datenbits (Datenwort)	Paritätsbits (Kontrollstellen)	Nachrichtenbits (Gesamtlänge des Codewortes)
1	2	3
4	3	7
11	4	15
26	5	31
57	6	63
120	7	127
247	8	255

4 CRC

CRC → "cyclic redundancy check" aka. "Zyklische Redundanzprüfung"
CRC ist lediglich ein "FEC", also ein Fehler-Erkennender-Code.

4.1 Erzeugen

Das Generatorpolynom kann beliebig gewählt werden. Ist jedoch sowohl in Aufgaben als auch in den Standards, welche in der Realität benutzt werden, meist vorgegeben.
An das Quellwort, das in CRC codiert werden soll, müssen Nullen angehängt werden und zwar Generatorpolynomlänge-1 viele.
Nach der Polynomdivision, dessen Funktionsweise aus den Beispielen recht eindeutig ablesbar sein sollte, muss das Quellwort mit dem Rest verodert werden, um das Codewort zu erhalten.
110011 mit Generatorpolynom 101: (Generatorpolynomlänge = 3, also zwei Nullen anhängen)

11001100 : 101 = BlaBla
101
110
101
111
101
101
101
000 ← Rest

Codewort = 11001100 or 00 = 11001100.

110011 mit Generatorpolynom 110: (Generatorpolynomlänge = 3, also zwei Nullen anhängen)

11001100 : 110 = BlaBla
110
00110
110
00 ← Rest

Codewort = 11001100 or 00 = 11001100.

010101 mit Generatorpolynom 101: (Generatorpolynomlänge = 3, also zwei Nullen anhängen)

01010100 : 101 = BlaBla
101
00100
101
01 ← Rest

Codewort = 01010100 or 01 = 01010101.

4.2 Überprüfen

Um zu überprüfen, ob ein Codewort korrekt übertragen wurde, muss man lediglich mit dem übertragenen Codewort erneut die Polynomdivision durchführen. Wenn kein Rest, also ein Rest von 0, entsteht, war die Übertragung fehlerfrei.

11001100 wurde übertragen mit Generatorpolynom 101:

11001100 : 101 = BlaBla
101
110
101
111
101
101
101
000 ← Rest

Der Rest ist "00", also war die Übertragung fehlerfrei.

01111110 wurde übertragen mit Generatorpolynom 101:

01111110 : 101 = BlaBla
101
101
101
0110
101
11 ← Rest

Der Rest ist "11", also war die Übertragung fehlerhaft.

10011001 wurde übertragen mit Generatorpolynom 110:

10011001 : 110 = BlaBla
110
101
110
111
110
100
110
101
110
11 ← Rest

Der Rest ist "11", also war die Übertragung fehlerhaft.

5 Zugriffsverfahren der Sicherungsschicht

CSMA steht für "Carrier Sense Multiple Access", also gleichzeitiges Schreiben und Horchen in einem Netz mit beliebig vielen Kommunikationspartnern.
CSMA/CD ="Carrier Sense Multiple Access/Collision Detection"

- Gleichzeitiges Senden und Hören
- Sobald eine Collision erkannt wird abbrechen
- Senden eines Jam-Signals abhängig von Paketlänge
- p = Wahrscheinlichkeit jetzt zu senden
- 1-p = Wahrscheinlichkeit mit der noch eine zusätzliche Paketlänge gewartet wird

CSMA/CA ="Carrier Sense Multiple Access/Collision Avoidance"

- Anfragen, ob man senden darf (RTS-Paket = Request to Send")
- Entweder keine Antwort oder eine Erlaubnis zu reden (CTS-Paket = Clear to Send")
- Soziales System, es funktioniert so lange sich alle daran halten
- Geeignet für Funknetzwerk

ALOHA / Unsynchronisiertes ALOHA

- Alle Pakete gleich groß
- Jederzeit senden
- Collisionen werden durch fehlendes ACK erkannt
- Bei Collision zufällig langes Warten bis zur Wiederholung

Slotted ALOHA / Synchronisiertes ALOHA

- Alle Pakete gleich groß
- Senden zu festgelegten Zeitslots
- Wenn Collision, dann vollständige Überlagerung des Zeitslots
- Schwer zu implementieren, da alle Beteiligten eine gemeinsame Zeitquelle benötigen

6 Erweiterte-Mealy-Automat

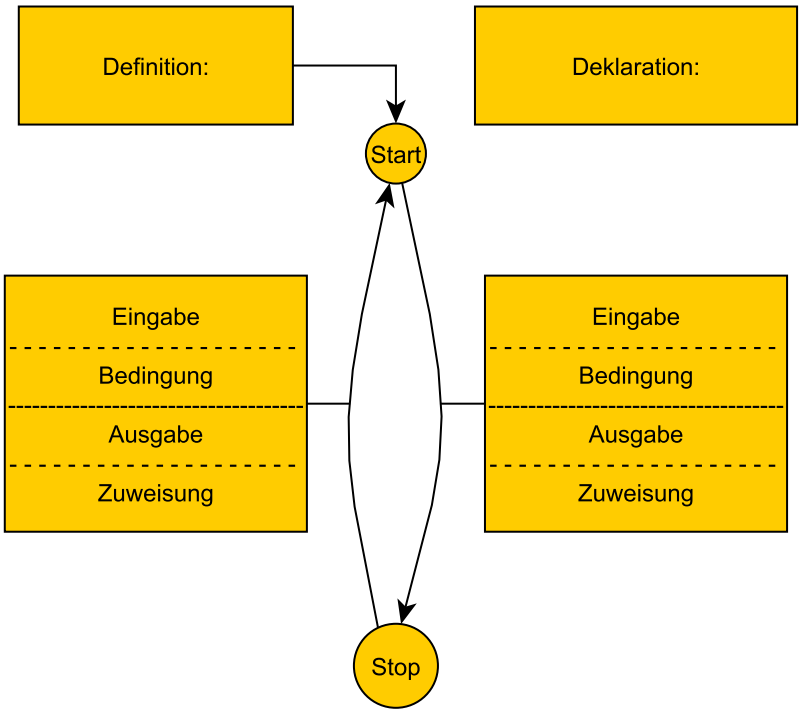
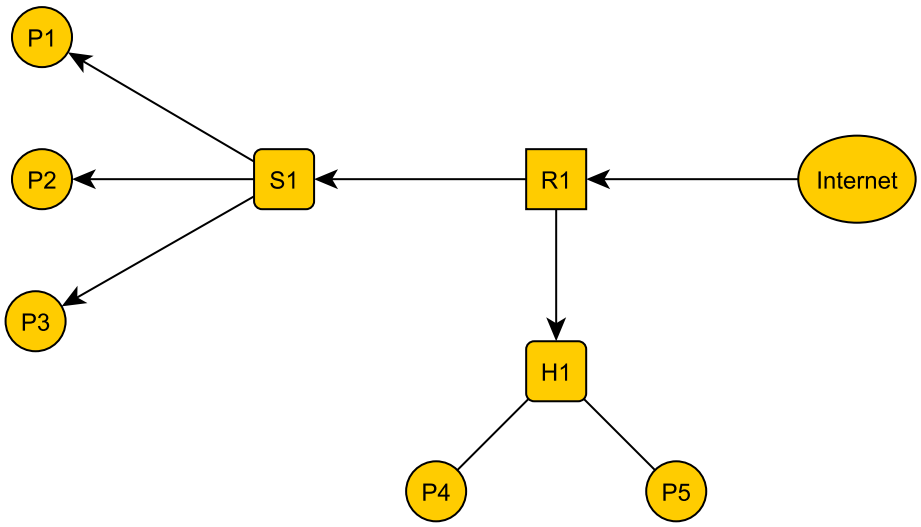


Abbildung 4: Aufbau eines erweiterten Mealy-Automatens

7 Subnetzmasken

CIDR-Notation	Binär	Dezimal
/0	0.0.0.0	00000000.00000000.00000000.00000000
/1	128.0.0.0	10000000.00000000.00000000.00000000
/2	192.0.0.0	11000000.00000000.00000000.00000000
/3	224.0.0.0	11100000.00000000.00000000.00000000
/4	240.0.0.0	11110000.00000000.00000000.00000000
/5	248.0.0.0	11111000.00000000.00000000.00000000
/6	252.0.0.0	11111100.00000000.00000000.00000000
/7	254.0.0.0	11111110.00000000.00000000.00000000
/8	255.0.0.0	11111111.00000000.00000000.00000000
/9	255.128.0.0	11111111.10000000.00000000.00000000
/10	255.192.0.0	11111111.11000000.00000000.00000000
/11	255.224.0.0	11111111.11100000.00000000.00000000
/12	255.240.0.0	11111111.11110000.00000000.00000000
/13	255.248.0.0	11111111.11111000.00000000.00000000
/14	255.252.0.0	11111111.11111100.00000000.00000000
/15	255.254.0.0	11111111.11111110.00000000.00000000
/16	255.255.0.0	11111111.11111111.00000000.00000000
/17	255.255.128.0	11111111.11111111.10000000.00000000
/18	255.255.192.0	11111111.11111111.11000000.00000000
/19	255.255.224.0	11111111.11111111.11100000.00000000
/20	255.255.240.0	11111111.11111111.11110000.00000000
/21	255.255.248.0	11111111.11111111.11111000.00000000
/22	255.255.252.0	11111111.11111111.11111100.00000000
/23	255.255.254.0	11111111.11111111.11111110.00000000
/24	255.255.255.0	11111111.11111111.11111111.00000000
/25	255.255.255.128	11111111.11111111.11111111.10000000
/26	255.255.255.192	11111111.11111111.11111111.11000000
/27	255.255.255.224	11111111.11111111.11111111.11100000
/28	255.255.255.240	11111111.11111111.11111111.11110000
/29	255.255.255.248	11111111.11111111.11111111.11111000
/30	255.255.255.252	11111111.11111111.11111111.11111100
/31	255.255.255.254	11111111.11111111.11111111.11111110
/32	255.255.255.255	11111111.11111111.11111111.11111111

8 Routing/Netzzuteilung



Abkürzung	Benennung
P	PC
R	Router
S	Switch
H	Hub

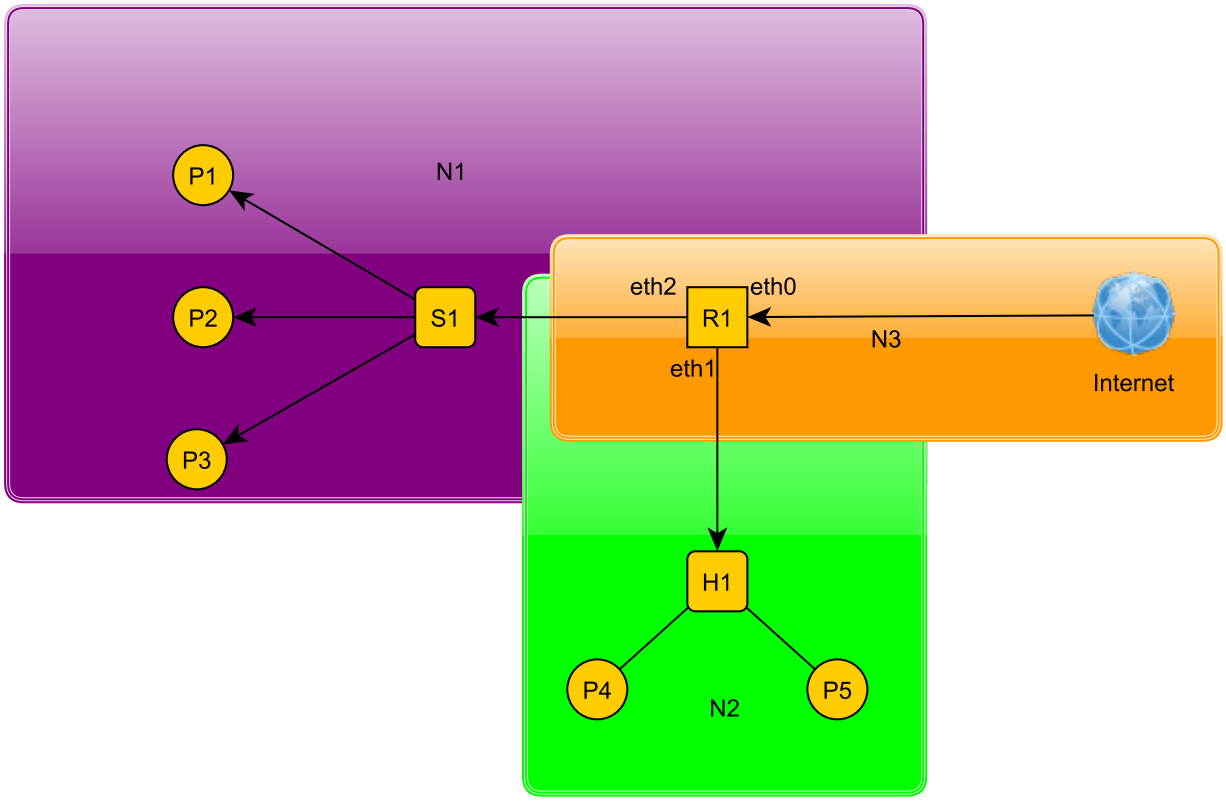


Abbildung 5: Netzzuteilung

Aufgabenstellung: Zuteilen in Subnetze für 193.88.42.128/27

11000001.01011000.00101010.10000000 ← IP-Binär
11111111.11111111.11111111.11100000 ← Subnetzmaske-Binär
Offensichtlich hat man fünf Bits zum Verteilen der Adressen und muss drei Netze zuteilen.
Jedes Subnetz braucht für jedes Gerät eine IP sowie eine Broadcastadresse und eine Netzadresse.
Somit braucht N1 für vier Geräte sechs Adressen, N2 für drei Geräte fünf Adressen und N3 für zwei Geräte vier Adressen.
Oder mit anderen Worten: N1 packen wir in ein achter Netz (Subnetzmaske /29), N2 packen wir in ein weiteres achter Netz (Subnetzmaske /29) und N3 passt in ein vierer Netz (Subnetzmaske /30).
Beim Einteilen kann man an den Bits mitzählen, dass die Netze in die freien Bits passen und sogar noch Platz ist.
Tipp: Wenn man sich die IP binär aufschreibt und markiert, welcher Bereich bei der Vorgegebenen Subnetzmaske noch veränderbar/nutzbar ist, kann man sich diese einfach von links nach rechts zum Durchnummerieren "markieren".
Wir brauchen die ersten x 0-Bits zum Nummerieren der Netze, somit haben wir noch x Bits frei, womit wir wissen, wie groß die einzelnen Netze jeweils sind/sein können.
Jetzt muss man die Netze zuteilen, dabei MUSS man mit dem größten Netz anfangen.

Netzname	**Netzadresse**	**Subnetzmaske (CIDR)**
N1	193.88.42.128	/29
N2	193.88.42.136	/29
N3	192.88.42.144	/30

Dann könnte man noch eine Routing Tabelle für "R1" schreiben:

- Adresse/Netz: Netzadresse der Netzes.
- Subnetzmaske: Subnetzmaske aus der Netzliste abschreiben.
- Gateway: Wenn die Flag "G" gesetzt ist, muss die Adresse des Gerätes, welches als Gateway benutzt wird, angegeben werden.
- Flag: "U"-Up, "G"-Gateway, "H"-Host.
- iFace: Das Interface des Gerätes durch das zu dem Netz geroutet wird.

Das Vorgehen dabei:

1. Was ist der größte Knoten/Verteilungspunkt → Den Weg dahin als 0.0.0.0 / "default"-Route wählen?
2. Welche Knotenpunkte kann ich in meinem eigenen Netz direkt ohne Hop erreichen?
3. Welche Netze kann ich über einen oder mehrere Hops erreichen?

Adresse/Netz	**Subnetzmaske**	**Gateway**	**Flag**	**iFace**
0.0.0.0	0.0.0.0	193.88.42.145	UG	eth0
193.88.42.128	255.255.255.248	0.0.0.0	U	eth2
193.88.42.136	255.255.255.248	0.0.0.0	U	eth1
193.88.42.144	255.255.255.252	0.0.0.0	U	eth0

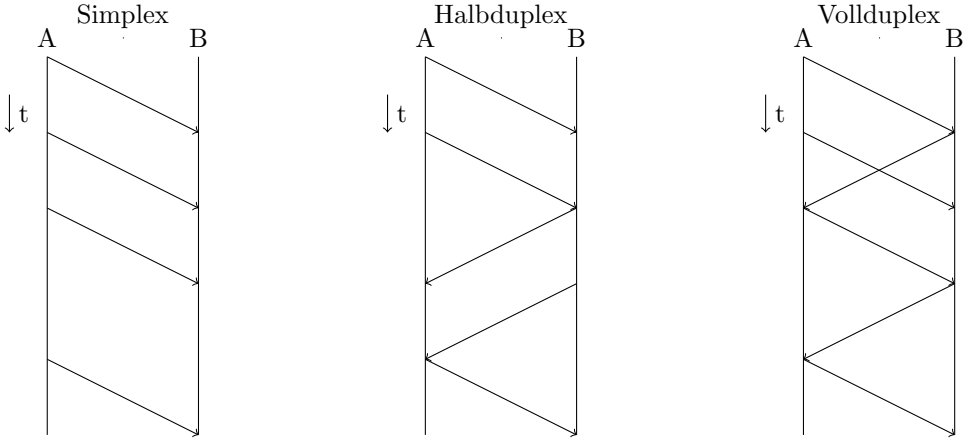
Mögliche Adressverteilung:

Gerät/Typ	Adresse	Binär
N1-Netzadresse	193.88.42.128	11000001.01011000.00101010.10000000
P1-IP-Adresse in N1	193.88.42.129	11000001.01011000.00101010.10000001
P2-IP-Adresse in N1	193.88.42.130	11000001.01011000.00101010.10000010
P3-IP-Adresse in N1	193.88.42.131	11000001.01011000.00101010.10000011
R1-IP-Adresse in N1	193.88.42.132	11000001.01011000.00101010.10000100
Freie-Adresse-1 in N1	193.88.42.133	11000001.01011000.00101010.10000101
Freie-Adresse-2 in N1	193.88.42.134	11000001.01011000.00101010.10000110
N1-Broadcastadresse	193.88.42.135	11000001.01011000.00101010.10000111
N2-Netzadresse	193.88.42.136	11000001.01011000.00101010.10001000
P4-IP-Adresse in N2	193.88.42.137	11000001.01011000.00101010.10001001
P5-IP-Adresse in N2	193.88.42.138	11000001.01011000.00101010.10001010
R1-IP-Adresse in N2	193.88.42.139	11000001.01011000.00101010.10001011
Freie-Adresse-1 in N2	193.88.42.140	11000001.01011000.00101010.10001100
Freie-Adresse-2 in N2	193.88.42.141	11000001.01011000.00101010.10001101
Freie-Adresse-3 in N2	193.88.42.142	11000001.01011000.00101010.10001110
N2-Broadcastadresse	193.88.42.143	11000001.01011000.00101010.10001111
N3-Netzadresse	193.88.42.144	11000001.01011000.00101010.10010000
Internet(Router)-IP in N3	193.88.42.145	11000001.01011000.00101010.10010001
R1-IP-Adresse in N3	193.88.42.146	11000001.01011000.00101010.10010010
N3-Broadcastadresse	193.88.42.147	11000001.01011000.00101010.10010011

9 Sockets

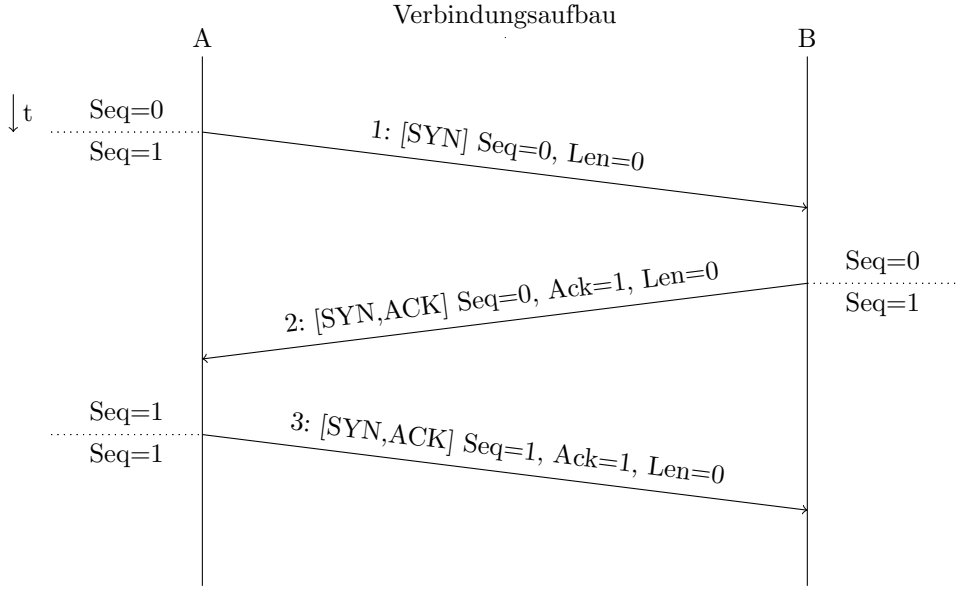
Client	Server
Socket()	ServerSocket()
connect()	accept()
send()	close()
close()	

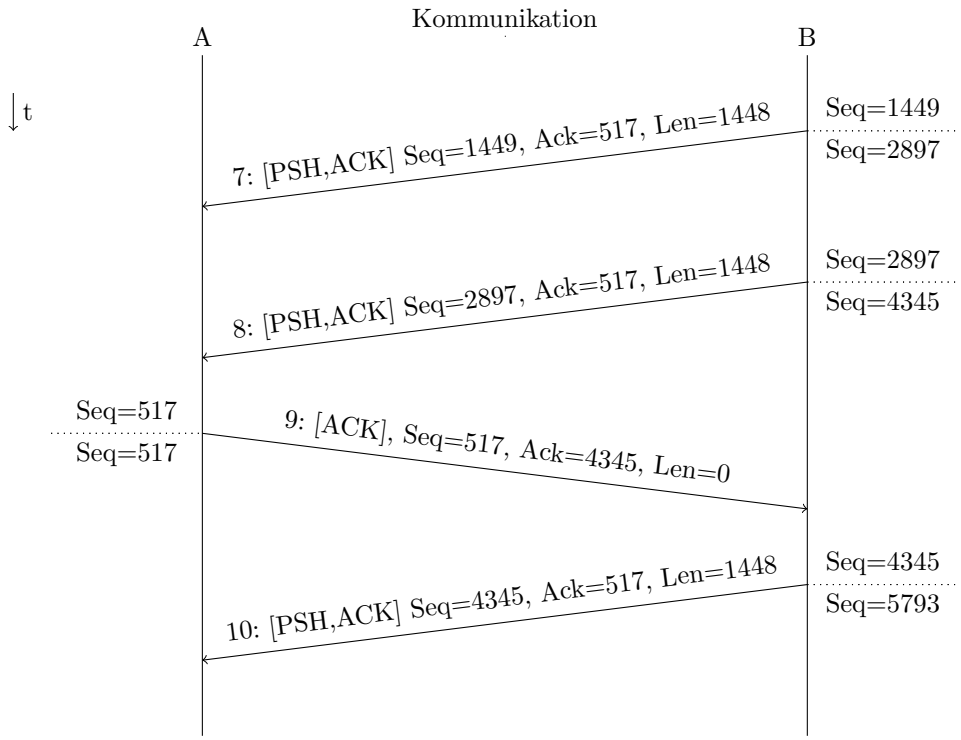
10 Richtungsunabhängigkeit eines Kommunikationskanals

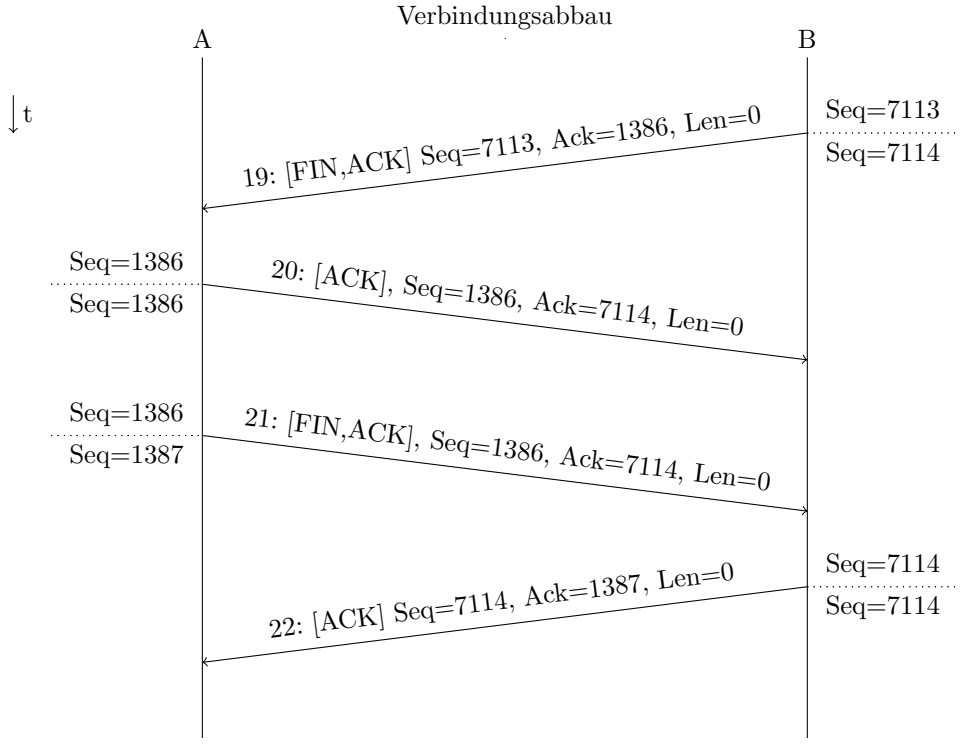


11 Sequenznummer-Diagramm

Flag	Erklärung
SYN	Verbindungsaufbau: Jedes erste gesendete SYN-Flag erhöht die Sequenzummer des Senders um 1.
ACK	Segment enthält einen Acknowledgement
PSH	Überspringt Buffering
FIN	Verbindungsabbau: Jedes erste gesendete FIN-Flag erhöht die Sequenzummer des Senders um 1.






Verbindungsabbau
A
B
t
Seq=7113
19: [FIN,ACK] Seq=7113, Ack=1386, Len=0
Seq=7114
Seq=1386
20: [ACK], Seq=1386, Ack=7114, Len=0
Seq=1386
Seq=1386
21: [FIN,ACK], Seq=1386, Ack=7114, Len=0
Seq=1387
Seq=7114
22: [ACK] Seq=7114, Ack=1387, Len=0
Seq=7114

12 ISO-OSI/TCP-IP

12.1 Geräte ↔ Layer

Gerät	Layer
Bridge	2
Gateway	3 oder 5
Hub	1
Proxy	5
Repeater	1
Router	3
Router Switch	2 bzw. 3
Switch	2

12.2 ISO-OSI

ISO = International Organization for Standardization
OSI = Open Systems Interconnection
Merksatz Englisch: "**P**lease **D**o **N**ot **T**hrow **S**alami **P**izza **A**way" (Von unten nach oben)
Merksatz Deutsch: "**A**lle **d**eutschen **S**chüler **t**rinken **v**erschiedene **S**orten **B**ier" (Von oben nach unten)

Schichten	
Deutsch	Englisch
Anwendungsschicht	**A**pplication Layer
Darstellungsschicht	**P**resentation Layer
Sitzungsschicht	**S**ession Layer
Transportschicht	**T**ransport Layer
Vermittlung-/Paketschicht	**N**etwork Layer
Sicherungsschicht	**D**ata Link Layer
Bitübertragungsschicht	**P**hysical Layer

12.3 TCP-IP

TCP=Transmission Control Protocol
IP=Internet Protocol

Schichten	
Deutsch	Englisch
Anwendungsschicht	Application Layer
Transportschicht	Transport Layer
Vermittlung-/Paketschicht	Network Layer
Sicherungsschicht	Data Link Layer
Bitübertragungsschicht	Physical Layer

13 Quality of Service

Quality of Service umfasst Ansätze, eine gleichbleibende Dienstqualität zu gewährleisten. Unter Dienstqualität wird eine Einhaltung von Parametern wie

- Jitter
- Durchsatz

- RTT
- Verlustrate

verstanden. Dazu gibt es mehrere mögliche Ansätze und Ideen. Diese Ideen beruhen auf den sog. Säulen des QoS,

- Traffic Characterisation (Klassifizierung der Nachrichten)
- Isolation, Scheduling & Policing (Isolation der Traffic-Ströme)
- High Resource Utilisation (möglichst hohe Auslastung der vorhandenen Ressourcen)
- Call Admission (Zulassung oder Nichtzulassung von Verbindungen)

Leaky Bucket Ein bekanntes Verfahren, um eine gegebene Traffic-Klasse zu regeln, ist Leaky Bucket. Dabei wird des ausgehende Strom für eine Traffic-Klasse mithilfe einer Warteschlange durch eine konstante Rate von oben beschränkt, um Bursts zu glätten.

IntServ und DiffServ Es existieren zwei generelle Ansätze, um eine QoS-Architektur in den Routern umzusetzen.

DiffServ Pakete werden am Rand des Netzwerks klassifiziert und innerhalb des Netzwerks anhand von weiteren Regeln je nach Klasse unterschiedlich behandelt.

IntServ Verbindungen werden innerhalb eines Routers klassifiziert und geregelt. Dabei kann der Router einer Verbindung weitere Garantien (etwa Bandbreite) zusichern oder lediglich eine nach Möglichkeit hohe Dienstqualität versprechen.

14 Verteilte Algorithmen

Atome/Ereignisse sind Sequenzen von Anweisungen, die höchstens ein `receive` enthalten und keine Anweisung nach einem `receive` haben.

direkte kausale Abhängigkeit besteht, falls zwei Ereignisse

- nacheinander im gleichen Prozess stattfinden oder
- Senden und Empfang der gleichen Nachricht sind.

indirekte kausale Abhängigkeit besteht, falls es zwischen zwei Ereignissen eine Kette direkter Kausalität gibt (etwa wie: existiert gerichteter Pfad).

Schnappschusskonsistenz besteht, falls die Ereignisse, in denen der Schnappschuss in den einzelnen Prozessen gebildet wird, paarweise kausal unabhängig sind.

15 Segmentierung/Fragmentierung

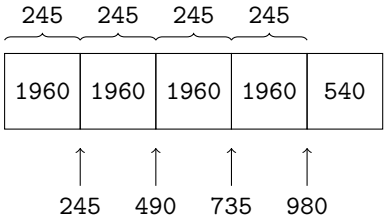
Bei IPV4 gibt die Maximum Transmission Unit (MTU) an wie groß ein Paket maximal sein darf ohne das es auf Schicht 3 Fragmentiert werden muss um erfolgreich über Schicht 2 verschickt zu werden.
Dies ist in IPV6 nicht länger erlaubt.

Es soll ein 8400Byte IP-Datagramm mit der Identifizierungsnummer (ID) 42 übertrage werden soll und es liegt eine MTU von 1980Byte zugrunde.
Dabei haben alle Fragmente die ID des Originalpaketes und können anhand der ID zugeordnet werden.
Der 20 Byte Header muss dabei für jedes Packet beachtet werden.
Das heißt das jedes zusätzliche Segment einen Übertragungsoverhead von 20Byte erzeugt.
Somit können im MTU von 1980Byte insgesamt 1960Byte Daten übertragen werden.

Es werden so lange Fragmente übertragen bis alle Daten des Ursprünglichen Paketes abgearbeitet wurden.
Der Offset rechnet in Oktetts und gibt die Startposition der Daten des Fragmets in der Originaldatei an.
Da das erste Fragment 1960Bytes enthält ergibt sich der Offset für das zweite Fragment 1960Byte / 8 = 245.

Datagramm	Länge	ID	FragFlag	Offset
Original-Datagramm	8400 (20 Byte Header + 8380 Byte Daten)	42	0	0
Fragment 1	1980 (20 Byte Header + 1960 Byte Daten)	42	1	0
Fragment 2	1980 (20 Byte Header + 1960 Byte Daten)	42	1	245
Fragment 3	1980 (20 Byte Header + 1960 Byte Daten)	42	1	490
Fragment 4	1980 (20 Byte Header + 1960 Byte Daten)	42	1	735
Fragment 5	560 (20 Byte Header + 540 Byte Daten)	42	0	980

Zusammensetzung der Daten beim Empfänger:

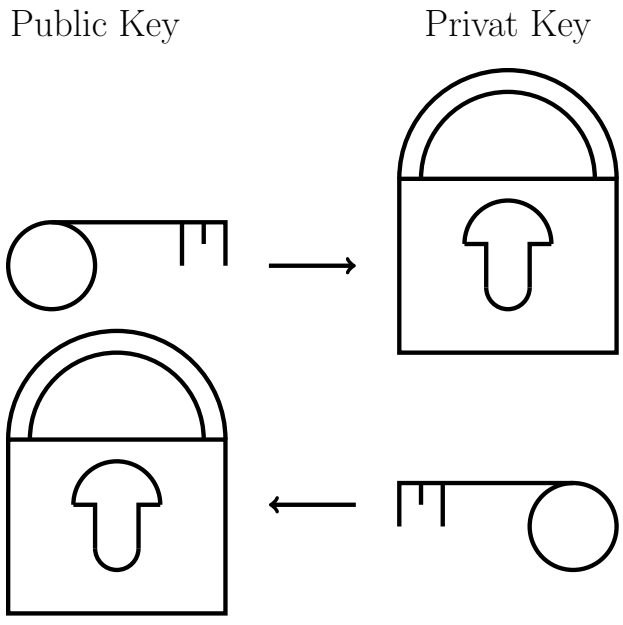


16 Security

16.1 Symmetrische/Asymmetrische Verschlüsselung

Bei einem symmetrischen Verschlüsselungsverfahren gibt es nur einen Key. Daten, welche mit einem Key verschlüsselt wurden, können mit dem selben Key wieder entschlüsselt werden.

Bei einem asymmetrischen Verschlüsselungsverfahren gibt es zwei Keys, den Private Key und den Public Key. Dabei wird der Public Key, wie der Name schon sagt, an andere Nutzer herausgegeben und der Private Key bleibt exklusiv beim erzeugenden Nutzer.
Daten, welche mit dem Private Key eines Nutzers verschlüsselt wurden, können nur mit dem Public Key des Nutzers entschlüsselt werden.
Daten, welche mit dem Public Key eines Nutzers verschlüsselt wurden, können nur mit dem Private Key des Nutzers entschlüsselt werden.



16.2 Garantien

Da asymmetrische Verschlüsselung verhältnismäßig rechenintensiv ist, ist es nicht immer effizient große Nachrichten asymmetrisch zu verschlüsseln.
$K_A^-(M)$ ist die Nachricht M verschlüsselt mit dem Privat Key von A.
$K_A^+(M)$ ist die Nachricht M verschlüsselt mit dem Public Key von A.
$K_S(M)$ ist die Nachricht M verschlüsselt mit einem symmetrischen Session Key.

Sicherzustellendes Kriterium	Kleine Datenmenge	Große Datenmenge
Authentizität	$K_A^-(M)$	$M, K_A^-(\#(M))$
Unverfälschtheit im Sinne von safety	$M, \#(M)$	$M, \#(M)$
Unverfälschtheit im Sinne von security	$M, K_A^-(\#(M))$	$M, K_A^-\#((M))$
Übertragungssicherheit (Herkunfts- und Zielgarantie)	$K_B^+(K_A^-(M))$	$M, K_B^+(K_A^-(\#(M)))$
+ Abhörsicherheit	$K_B^+(K_A^-(M))$	$K_S(M), K_B^+(K_A^-(S, \#(M)))$

Die Lösungsvariante für kleine Datenmengen funktionieren natürlich auch für große Datenmengen und umgekehrt. Bei der Differenzierung geht es lediglich um die Effizienz der Lösung.

17 TODO

- Beispiele für Mealy Automaten
- Snapshot Beispiele
- Sequenznummerrechenkreise zur besseren Visualisierung
- Zweidimensinale Parität