MeshCore Deep Dive

Die Technik „unter der Haube": Flooding und Routing im Detail

Sebastian Muszynski (DO2KSM)

Notfunkübung im Distrikt G

21. Februar 2026

14:00 – 17:00 Uhr

Großraum Aachen – Köln – Bonn

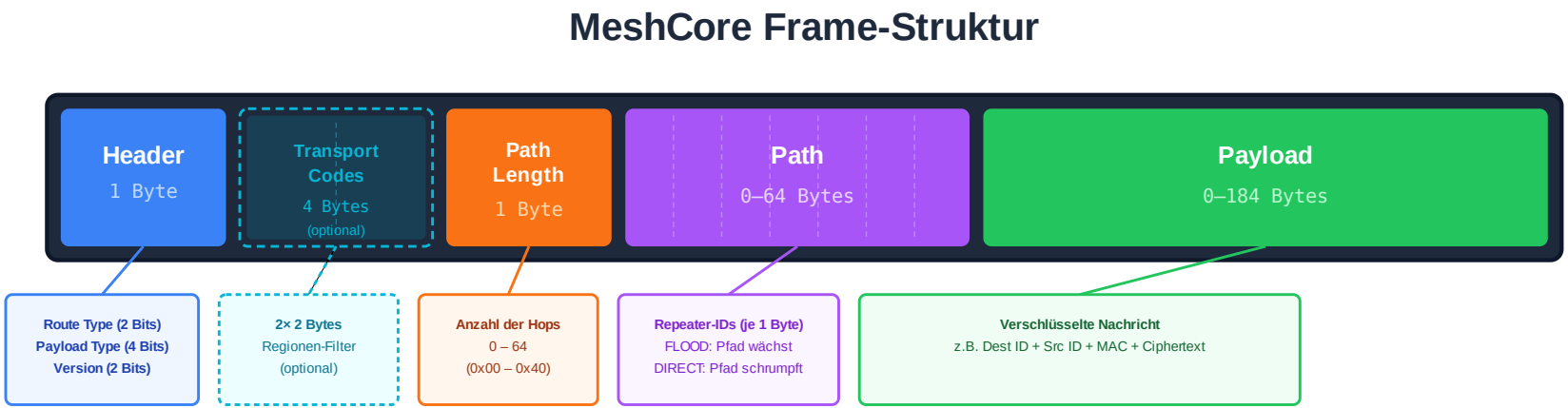
- Amateurfunk (primär 2m FM)
- CB-Funk
- PMR

Großraum Bonn (zusätzlich)

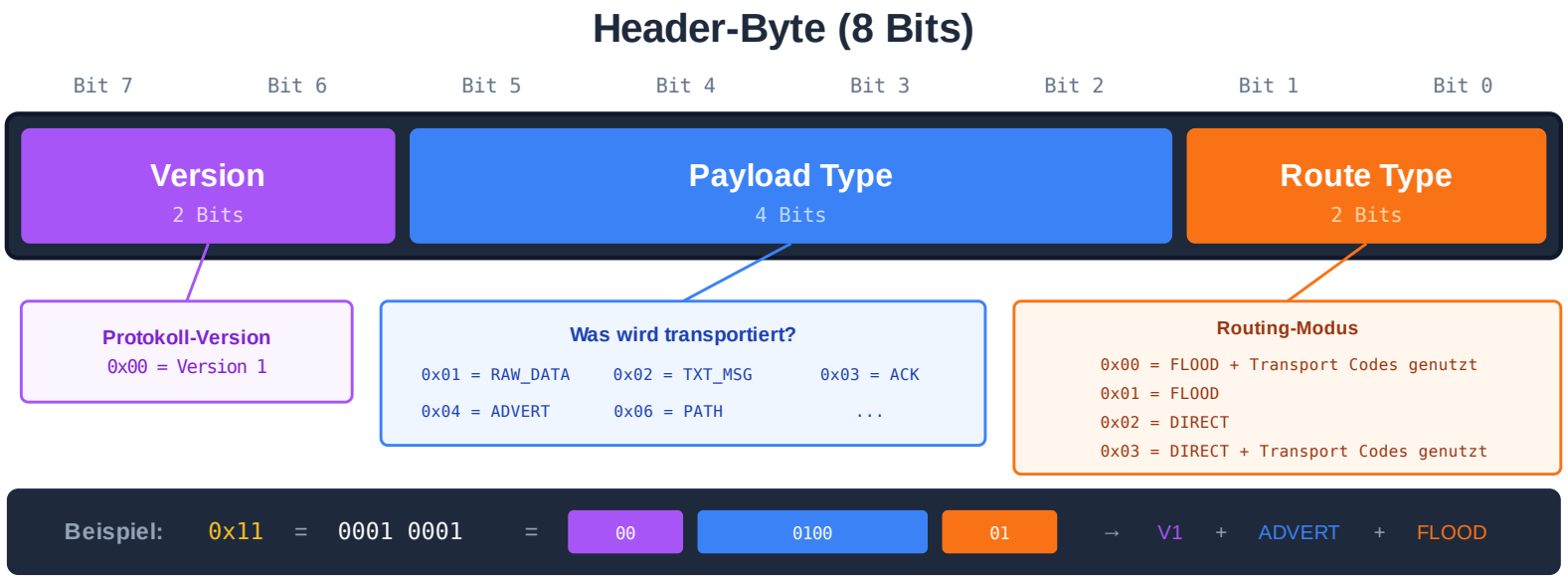
- Meshtastic
- MeshCore 🎉 #emergency

Notfunkrelais DB0DBN wird bespielt

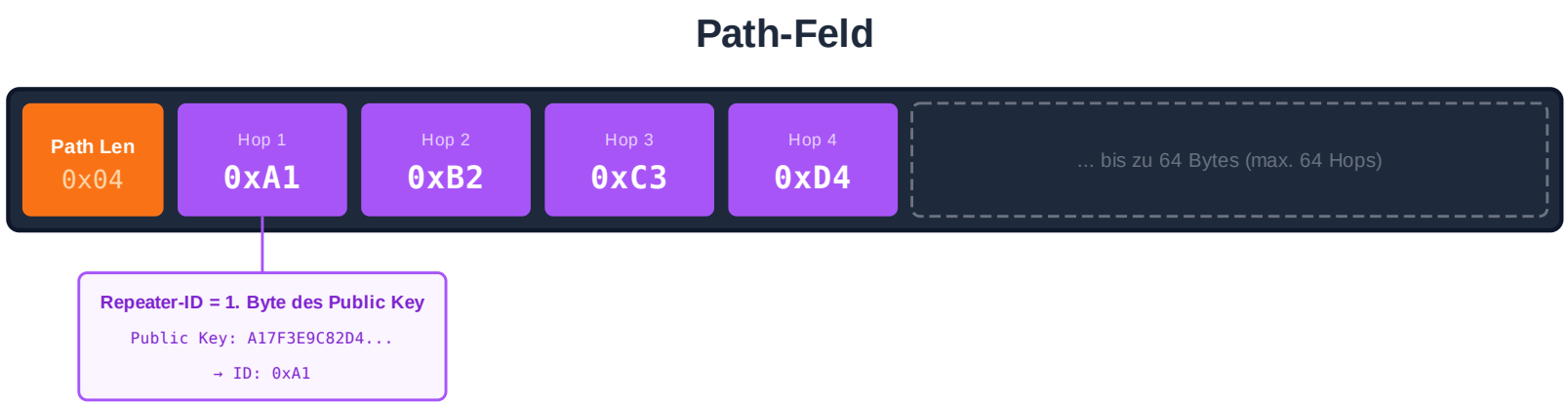
Frame-Struktur



Header-Byte


Header-Byte (8 Bits)
Bit 7
Bit 6
Bit 5
Bit 4
Bit 3
Bit 2
Bit 1
Bit 0
Version
2 Bits
Payload Type
4 Bits
Route Type
2 Bits
Protokoll-Version
0x00 = Version 1
Was wird transportiert?
0x01 = RAW_DATA
0x02 = TXT_MSG
0x03 = ACK
0x04 = ADVERT
0x06 = PATH
...
Routing-Modus
0x00 = FLOOD + Transport Codes genutzt
0x01 = FLOOD
0x02 = DIRECT
0x03 = DIRECT + Transport Codes genutzt
Beispiel:
0x11
=
0001 0001
=
00
0100
01
→
V1
+
ADVERT
+
FLOOD

Path-Feld



FLOOD: Path wächst

Sender: PathLen=0 []
↓ +A1
Nach Repeater A1: PathLen=1 [A1]
↓ +B2
Nach Repeater B2: PathLen=2 [A1, B2]
↓ +C3
Nach Repeater C3: PathLen=3 [A1, B2, C3]

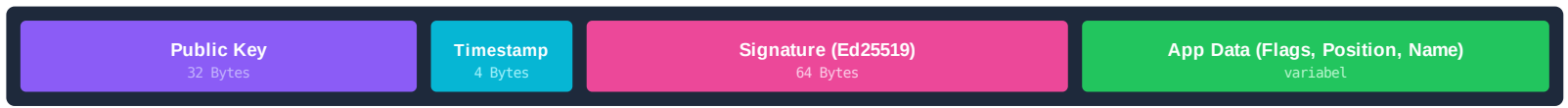
DIRECT: Path schrumpft

Sender: PathLen=3 [A1, B2, C3]
↓ -A1
Nach Repeater A1: PathLen=2 [B2, C3]
↓ -B2
Nach Repeater B2: PathLen=1 [C3]
↓ -C3
Nach Repeater C3: PathLen=0 [] → Ziel!

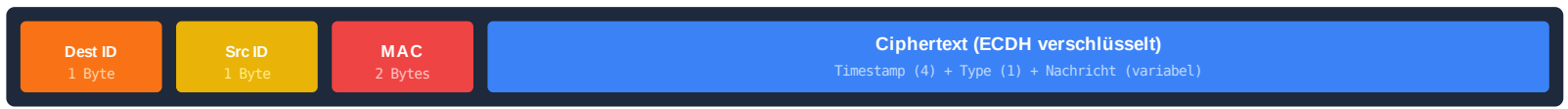
Payload-Formate

Payload-Format nach Typ

ADVERT (0x04) – Broadcast, unverschlüsselt

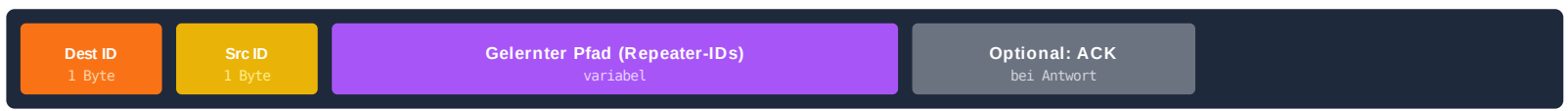


TXT_MSG (0x02) – Private Nachricht, verschlüsselt



ACK (0x03) – Empfangsbestätigung

PATH (0x06) – Pfad-Lernen / Bestätigung

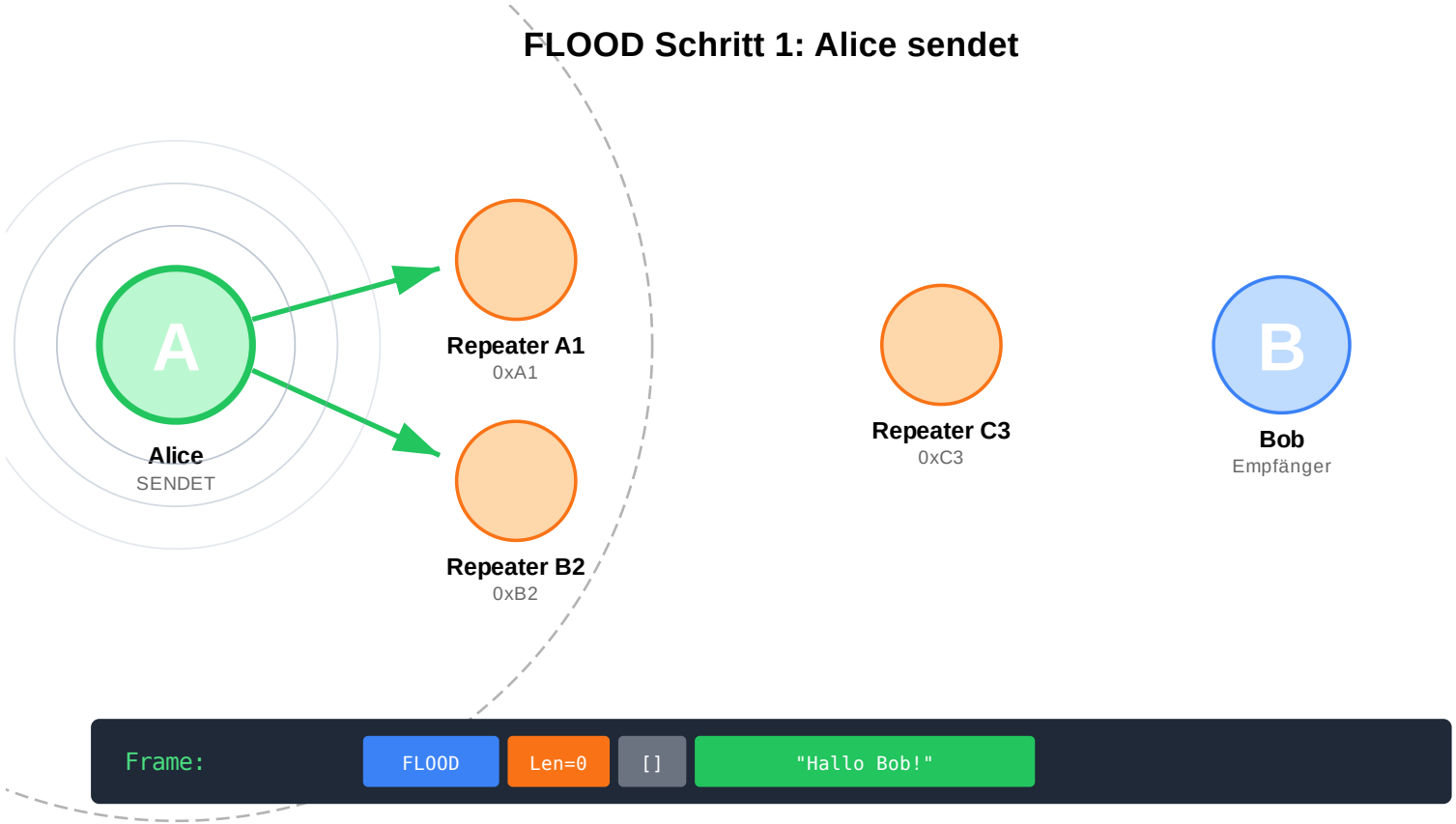


Dest/Src ID: Erstes Byte des Public Key – identifiziert Empfänger/Absender

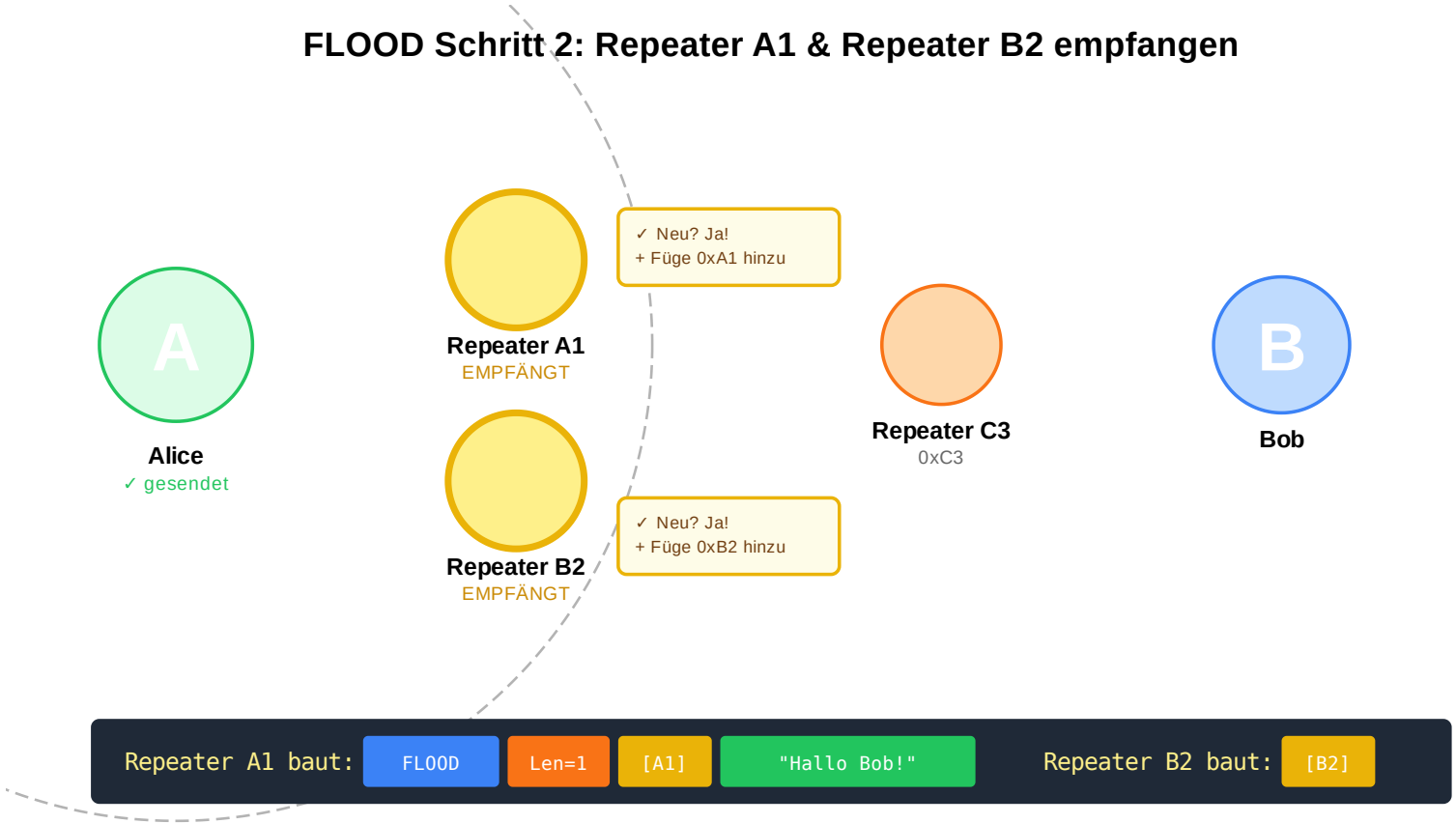
Teil 1: FLOODING

"Eine Nachricht spült sich durchs Land"

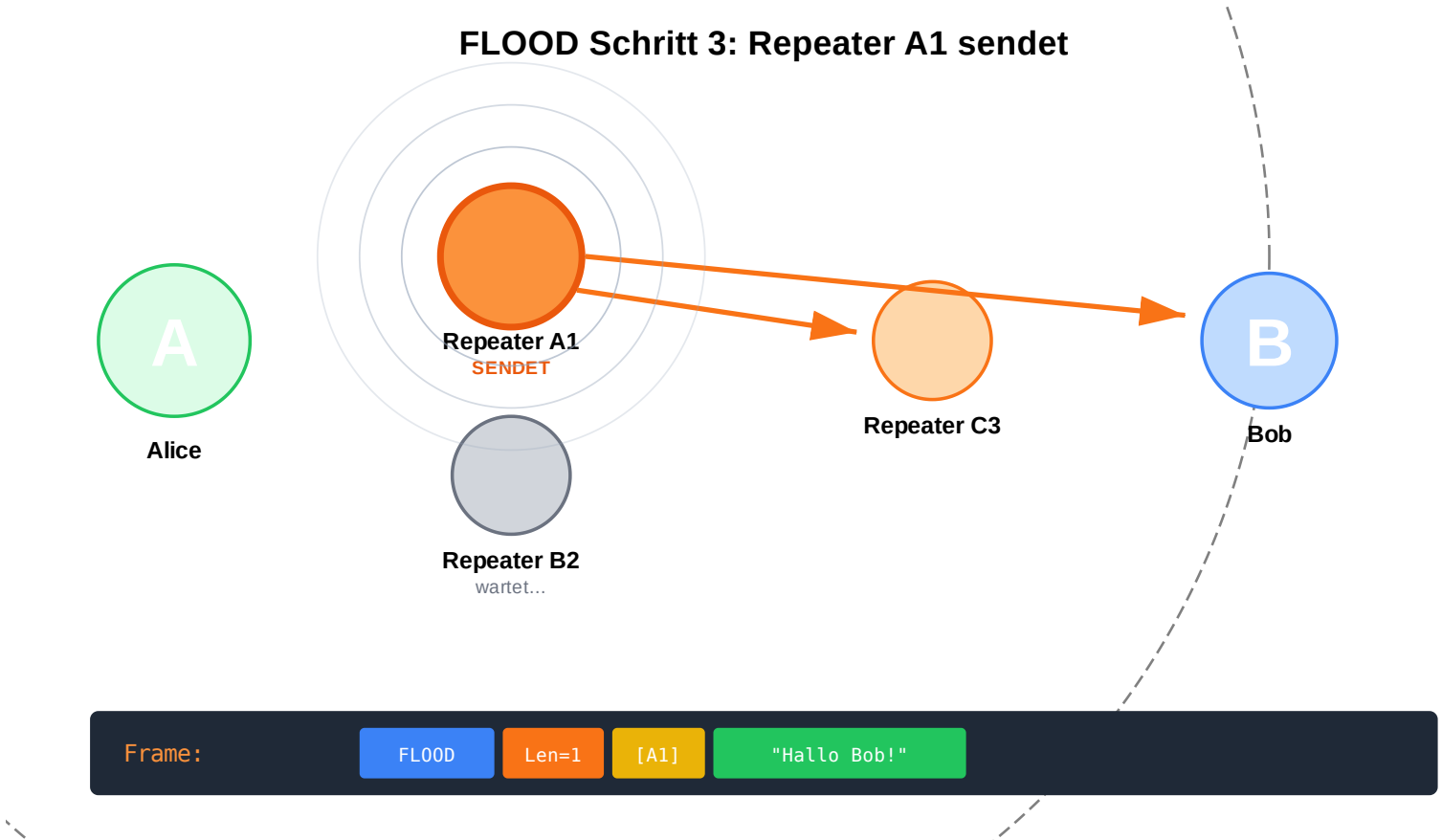
FLOOD Schritt 1: Alice sendet



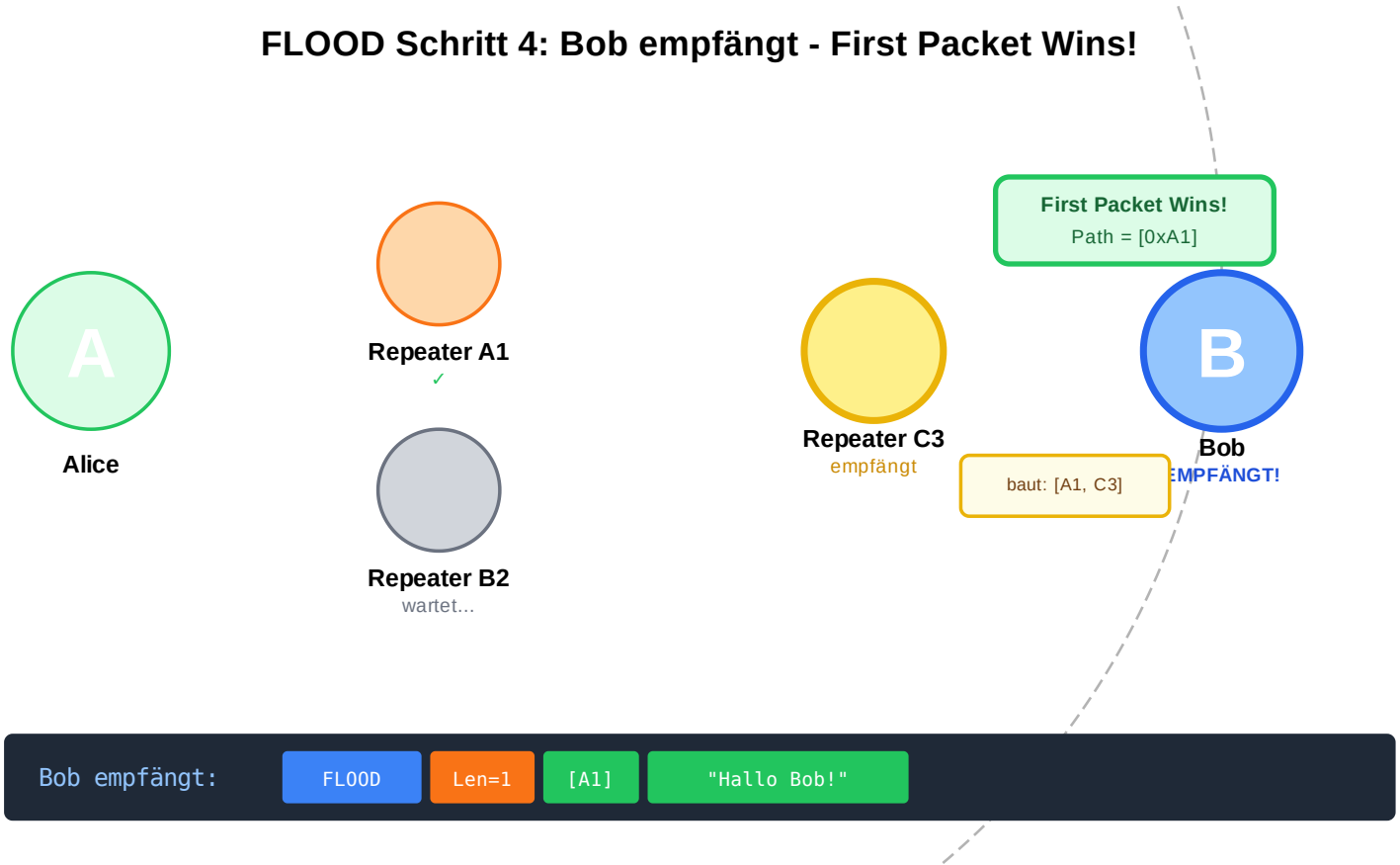
FLOOD Schritt 2: R1 & R2 empfangen



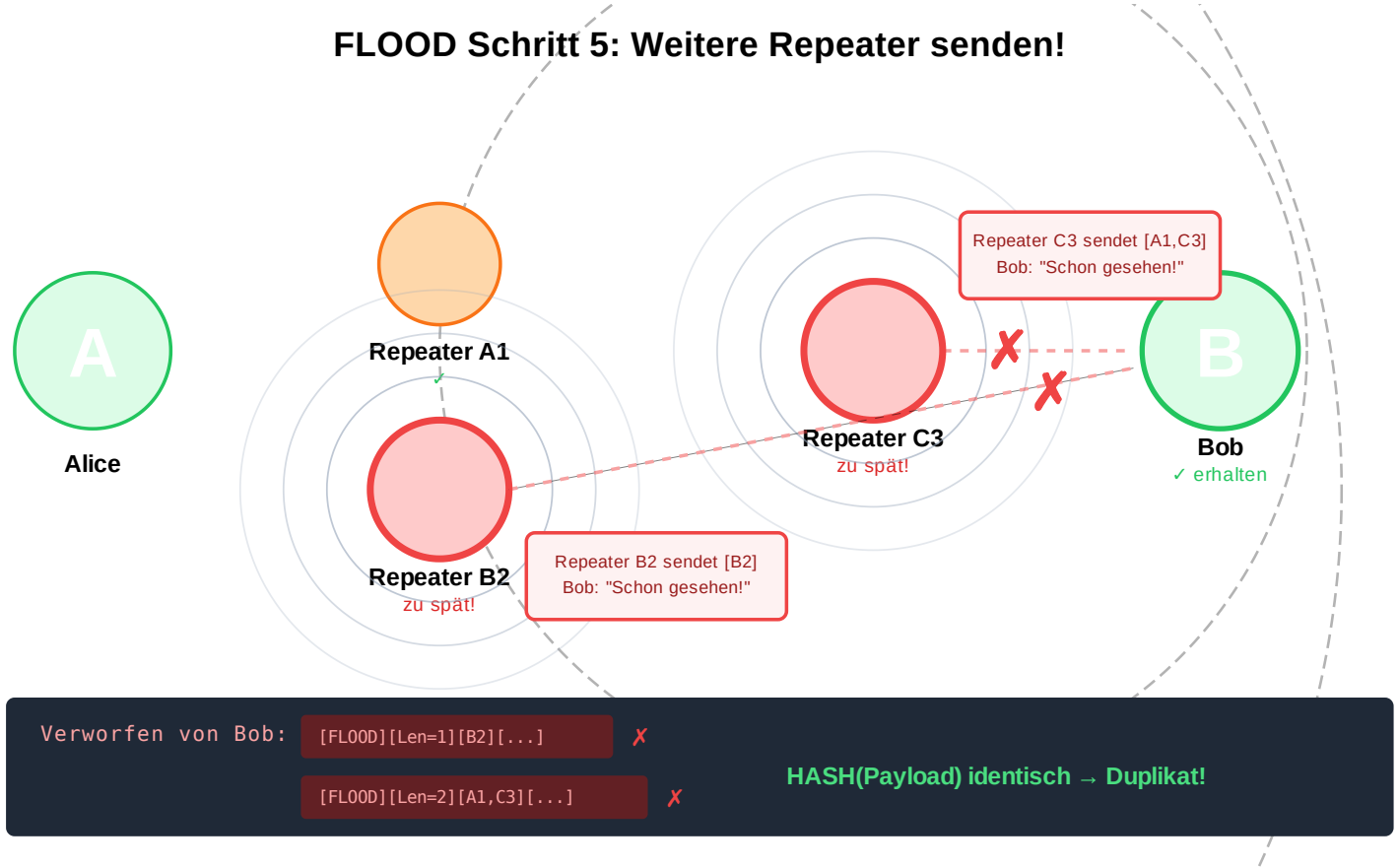
FLOOD Schritt 3: R1 sendet zuerst



FLOOD Schritt 4: Bob empfängt



FLOOD Schritt 5: Duplikate verwerfen



Frame-Wachstum bei FLOOD

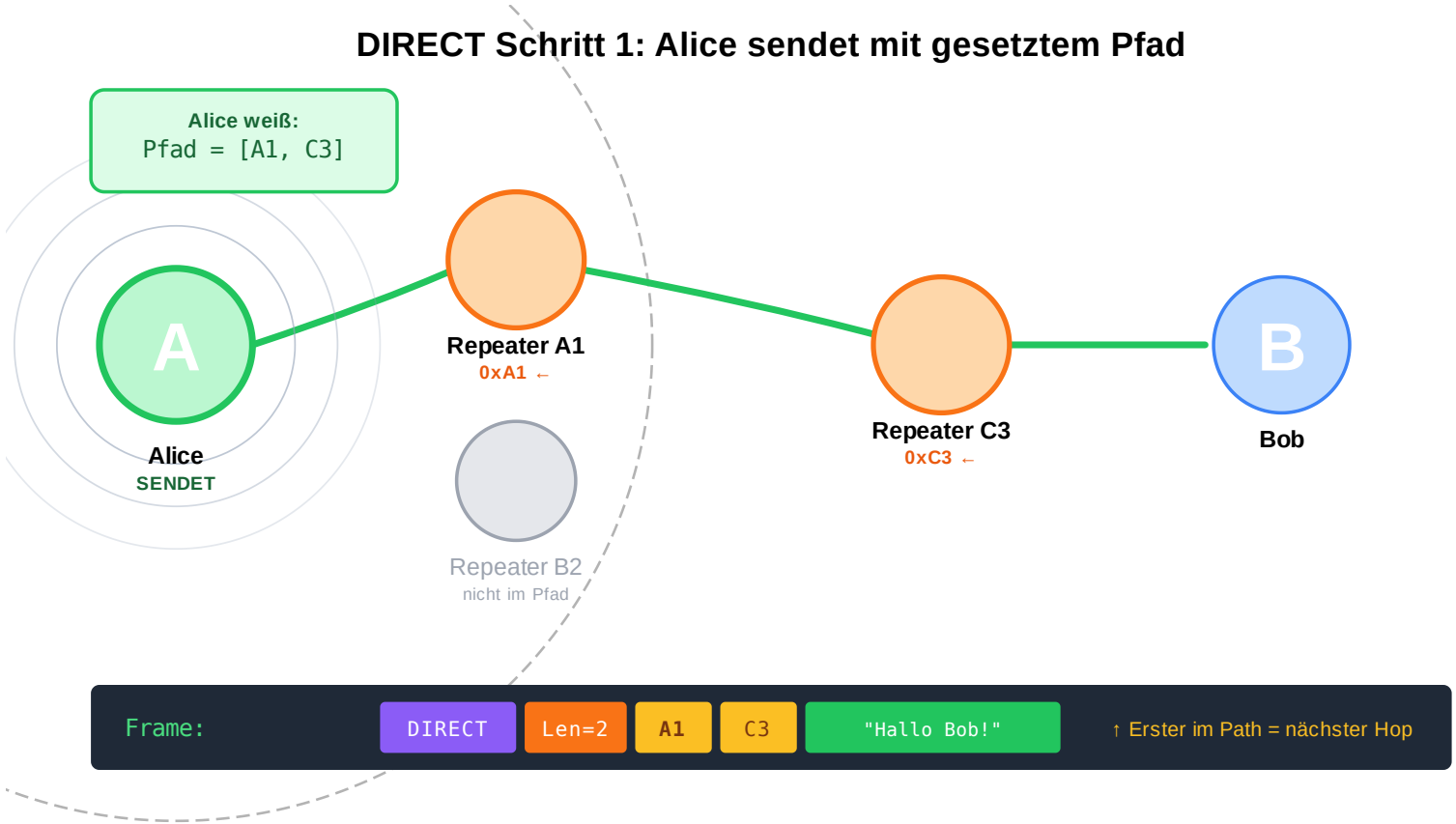
```
Alice:     [H][0][Payload]              → 2 + Payload Bytes
                              ↓
R1:        [H][1][A1][Payload]          → 3 + Payload Bytes
                              ↓
R3:        [H][2][A1][C3][Payload]      → 4 + Payload Bytes
```

PathLen wächst mit jedem Hop!

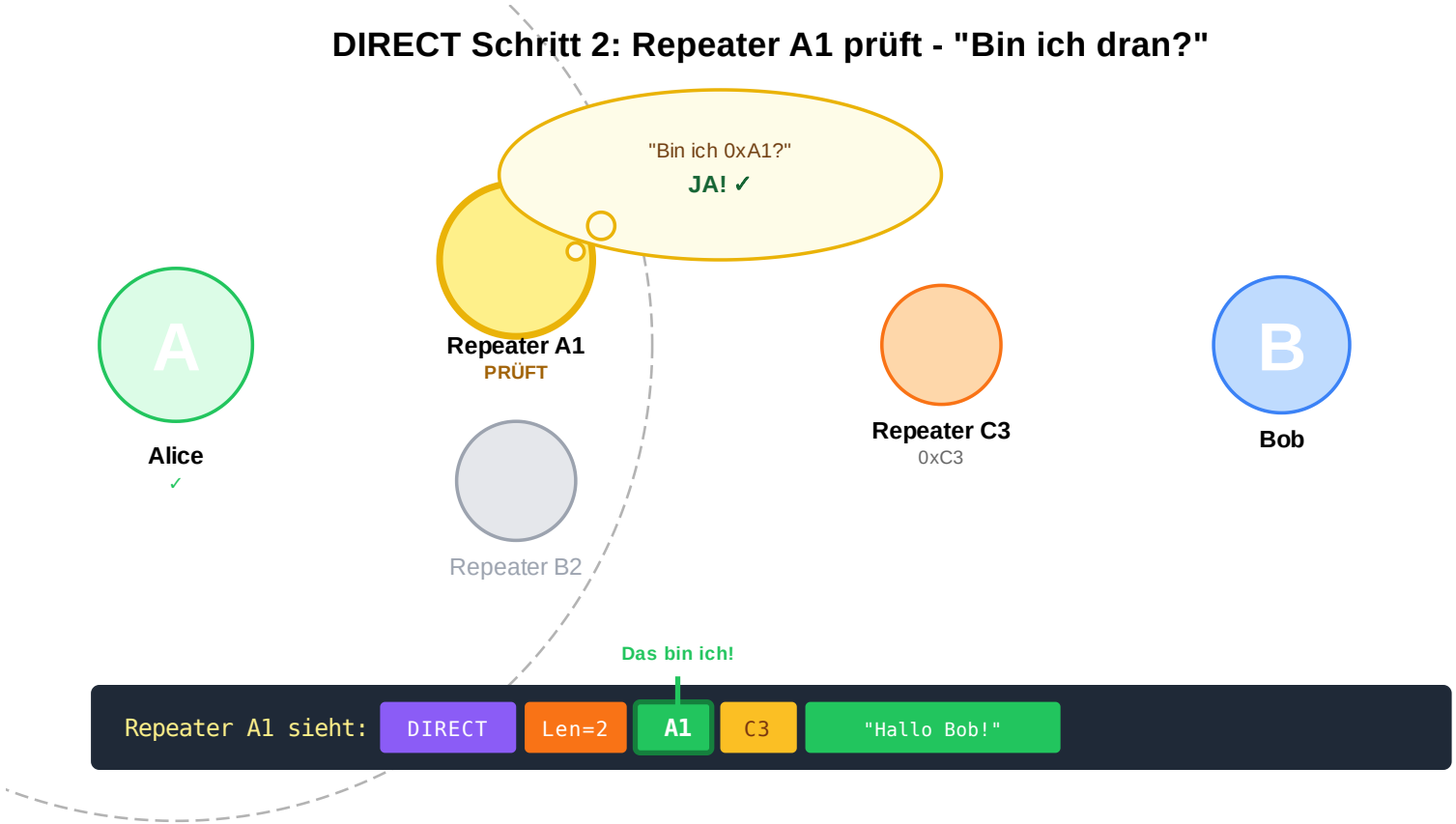
Teil 2: DIRECT Routing

"Der Pfad wird eingebettet und abgebaut"

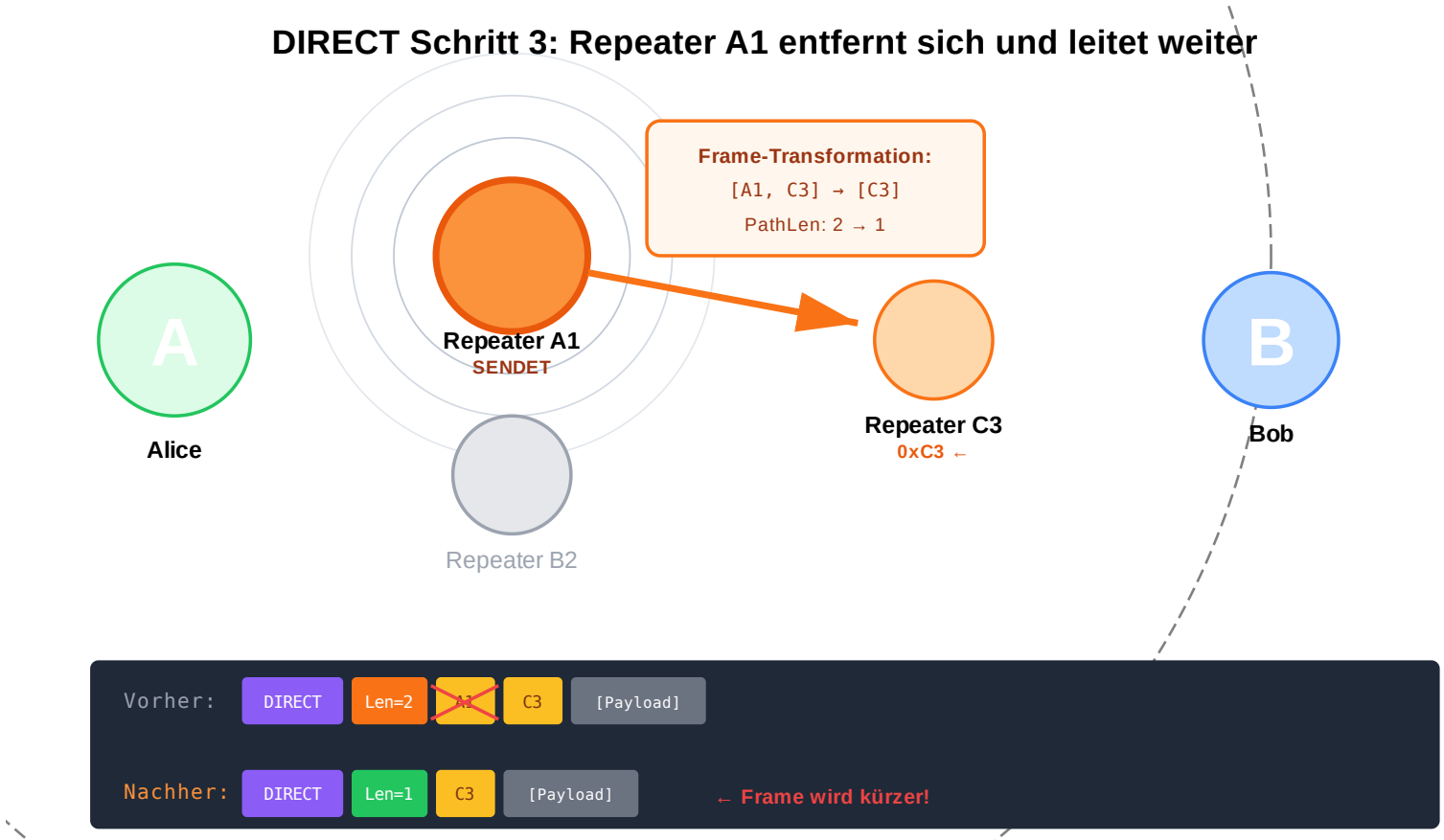
DIRECT Schritt 1: Alice sendet mit Pfad



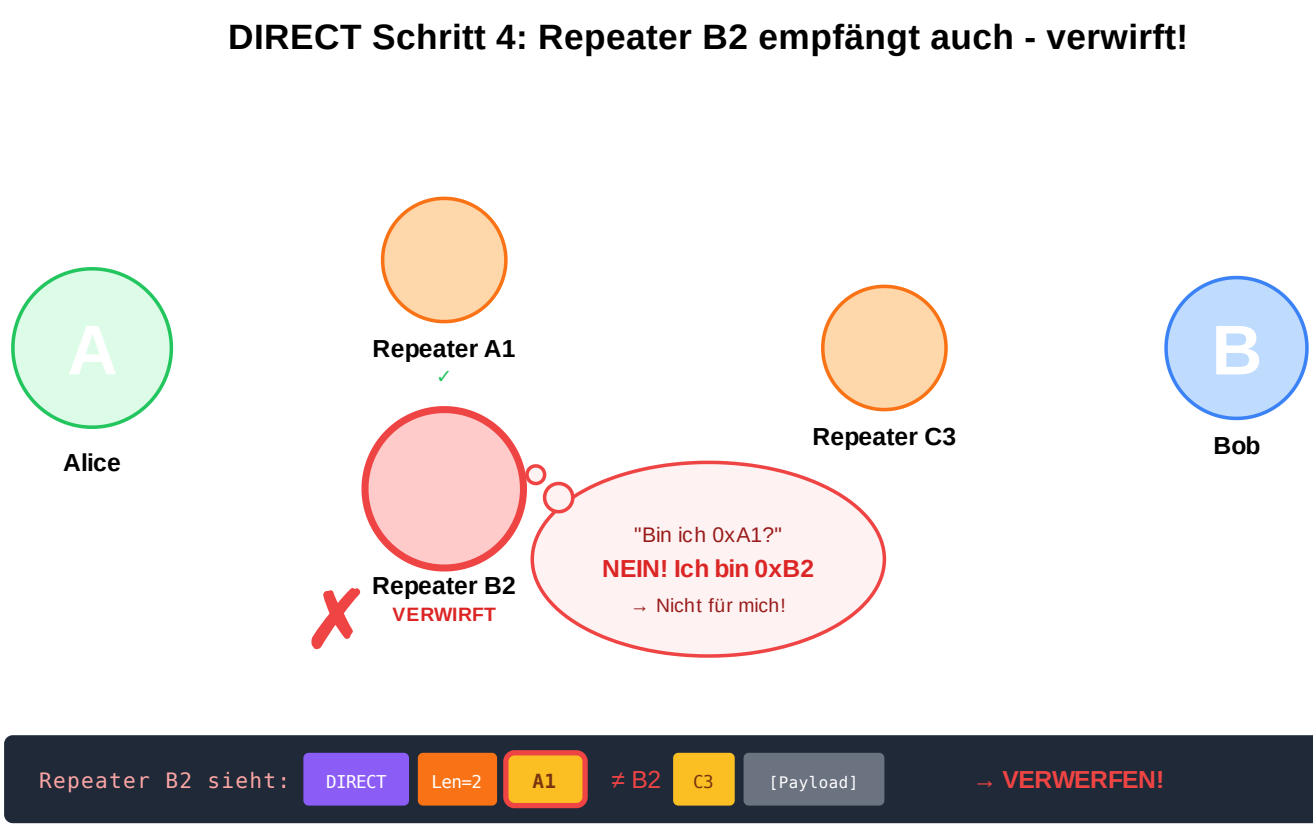
DIRECT Schritt 2: R1 prüft



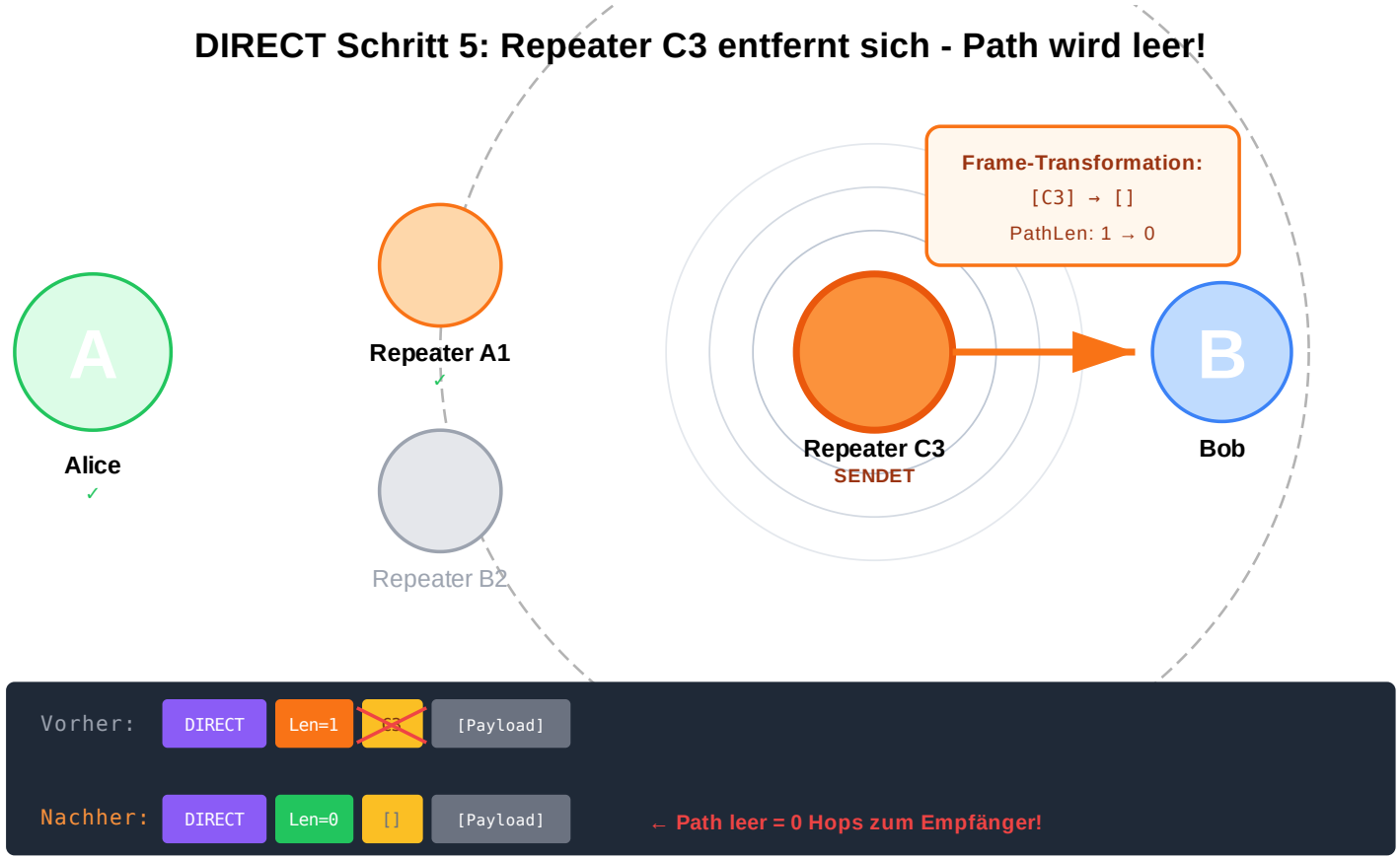
DIRECT Schritt 3: R1 kürzt und leitet weiter



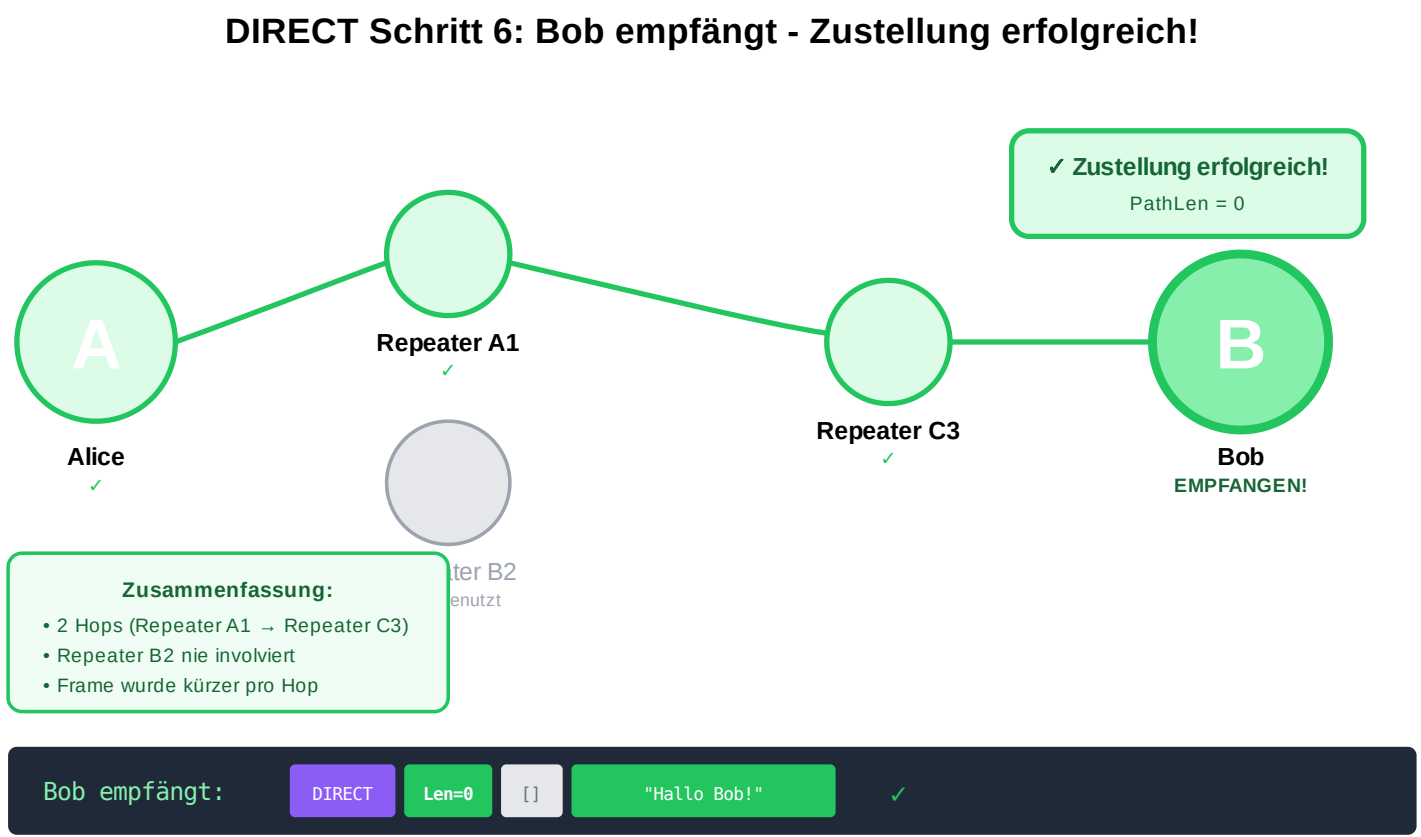
DIRECT Schritt 4: R2 verwirft



DIRECT Schritt 5: R3 kürzt - Path wird leer



DIRECT Schritt 6: Bob empfängt



Frame-Schrumpfung bei DIRECT

```
Alice:    [H][2][A1][C3][Payload]     → 4 + Payload Bytes
                                ↓
R1:       [H][1][C3][Payload]         → 3 + Payload Bytes
                                ↓
R3:       [H][0][Payload]             → 2 + Payload Bytes
                                ↓
                  Bob:      empfängt!
```

PathLen schrumpft mit jedem Hop!

Teil 3: Repeater-IDs

Das erste Byte des Public Keys

Wie entsteht eine Repeater-ID?

Public Key (32 Bytes)

```
FE5616140E71B9E01E5DA751
03F56550FFFD78C7DE35CEB3
0161401CD3A15599 ...
```

Repeater-ID (1 Byte)

```
0×FE
```

Das erste Byte!

254 mögliche IDs (0x00-0xFF, minus reservierte)

Kollisionen sind wahrscheinlich und eingeplant!

Warum nur 1 Byte?

1-Byte ID

- 64 Hops × 1 Byte = **64 Bytes**
- Max Payload: **184 Bytes**
- Airtime: optimal

2-Byte ID (hypothetisch)

- 64 Hops × 2 Bytes = **128 Bytes**
- Max Payload: **~120 Bytes**
- Airtime: +50% länger

> "Airtime ist die wertvollste Ressource im Mesh."

ID-Kollisionen: Nervig aber unkritisch

Nervig

- App zeigt: "2 known repeaters"
- Welcher Repeater war’s?
- User ist verwirrt

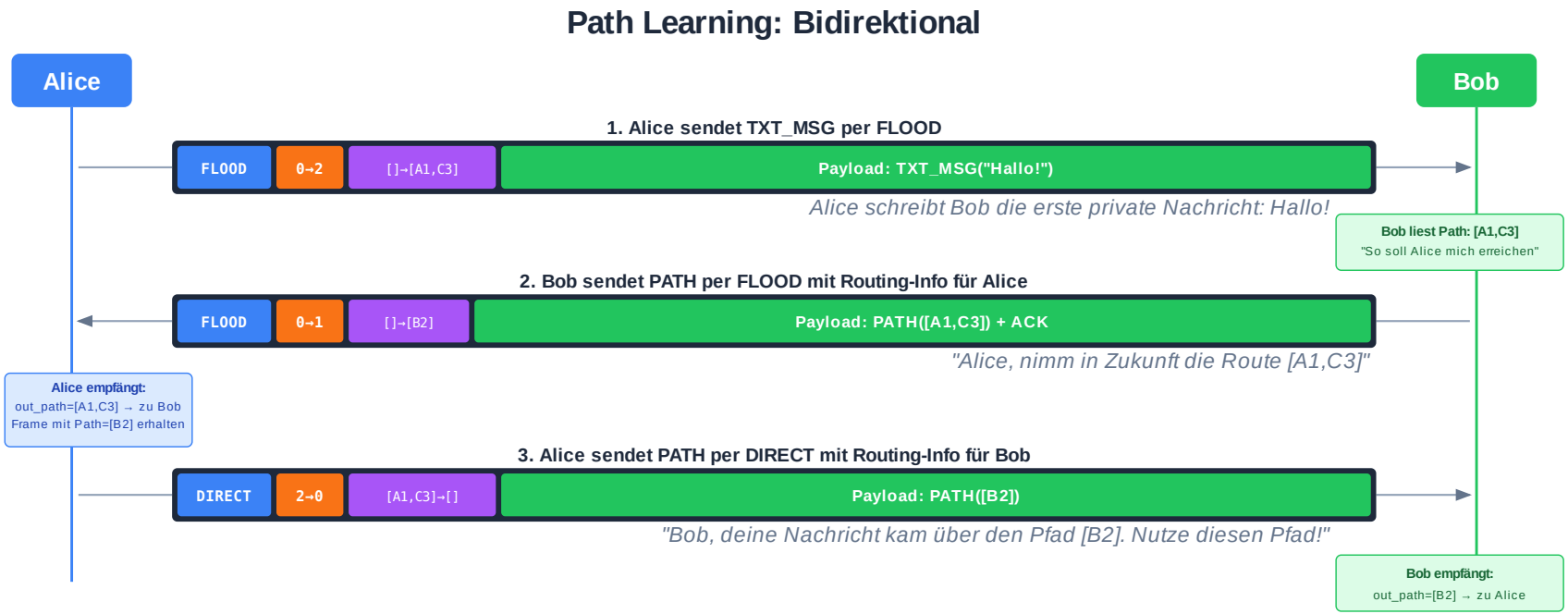
Unkritisch

- Routing funktioniert!
- FLOOD: Beide leiten weiter
- DIRECT: Einer oder Beide leiten weiter
- Krypto nutzt vollen Key

Teil 4: Path Learning

Wie lernen Clients den Weg?

Path Learning: Bidirektional



Zusammenfassung

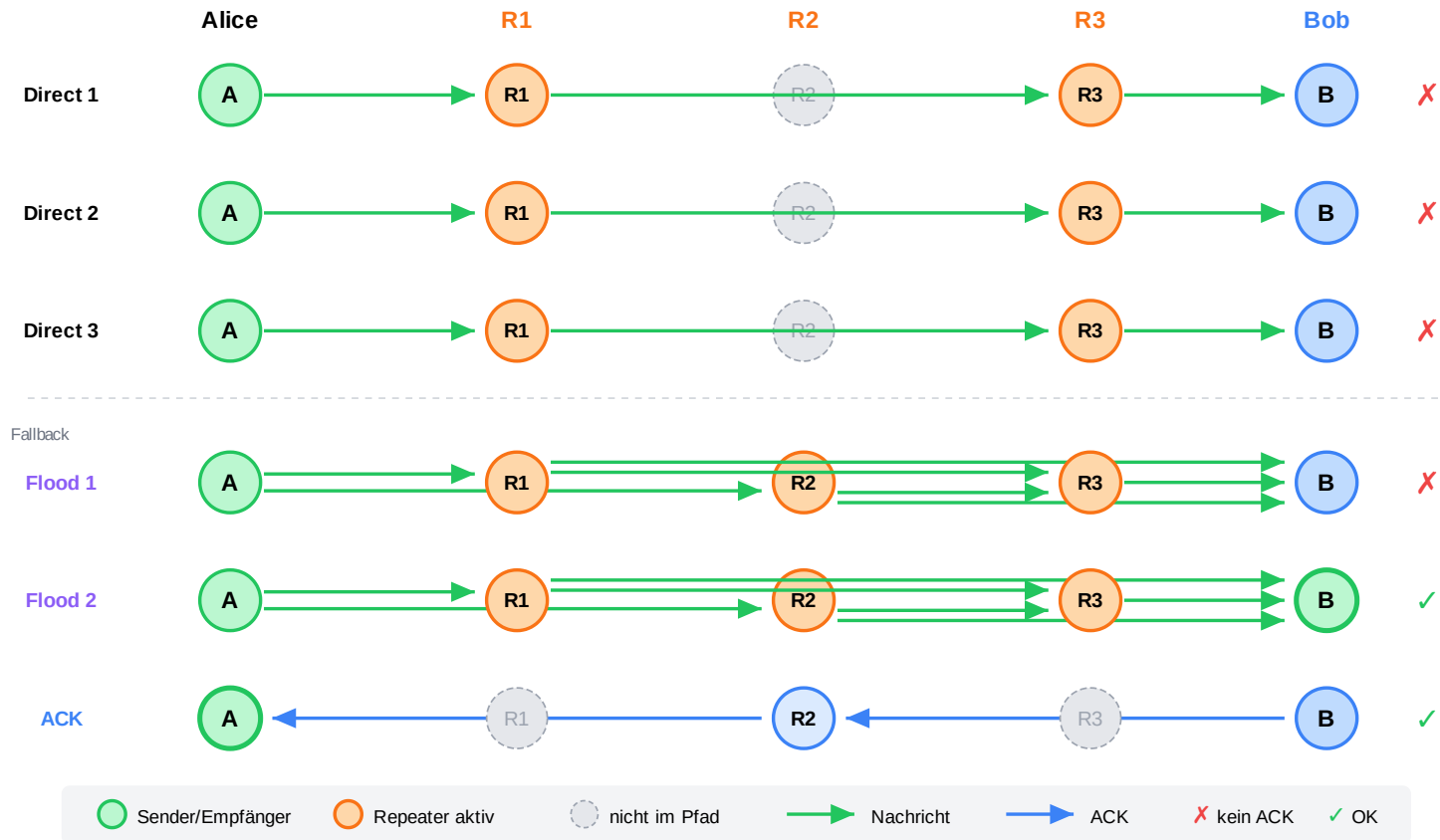
	FLOOD	DIRECT
Path	wächst	schrumpft
Wer leitet?	alle Repeater	nur Repeater im Path
Delay	(RX +) TX	nur TX (minimal)
Duplikate	viele	keine
Wann?	erste Nachricht	Folgenachrichten

Fragen?

Sebastian Muszynski (DO2KSM)
basti@linkt.de

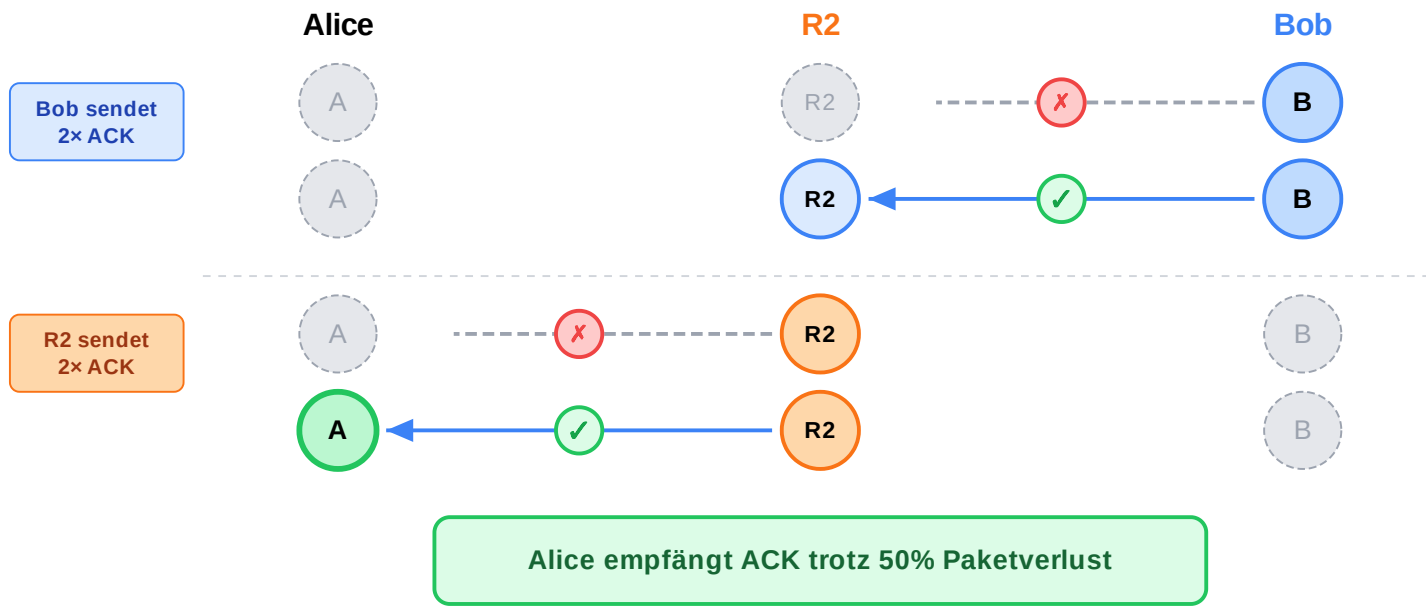
Bonus-Folien 

Retries und ACKs

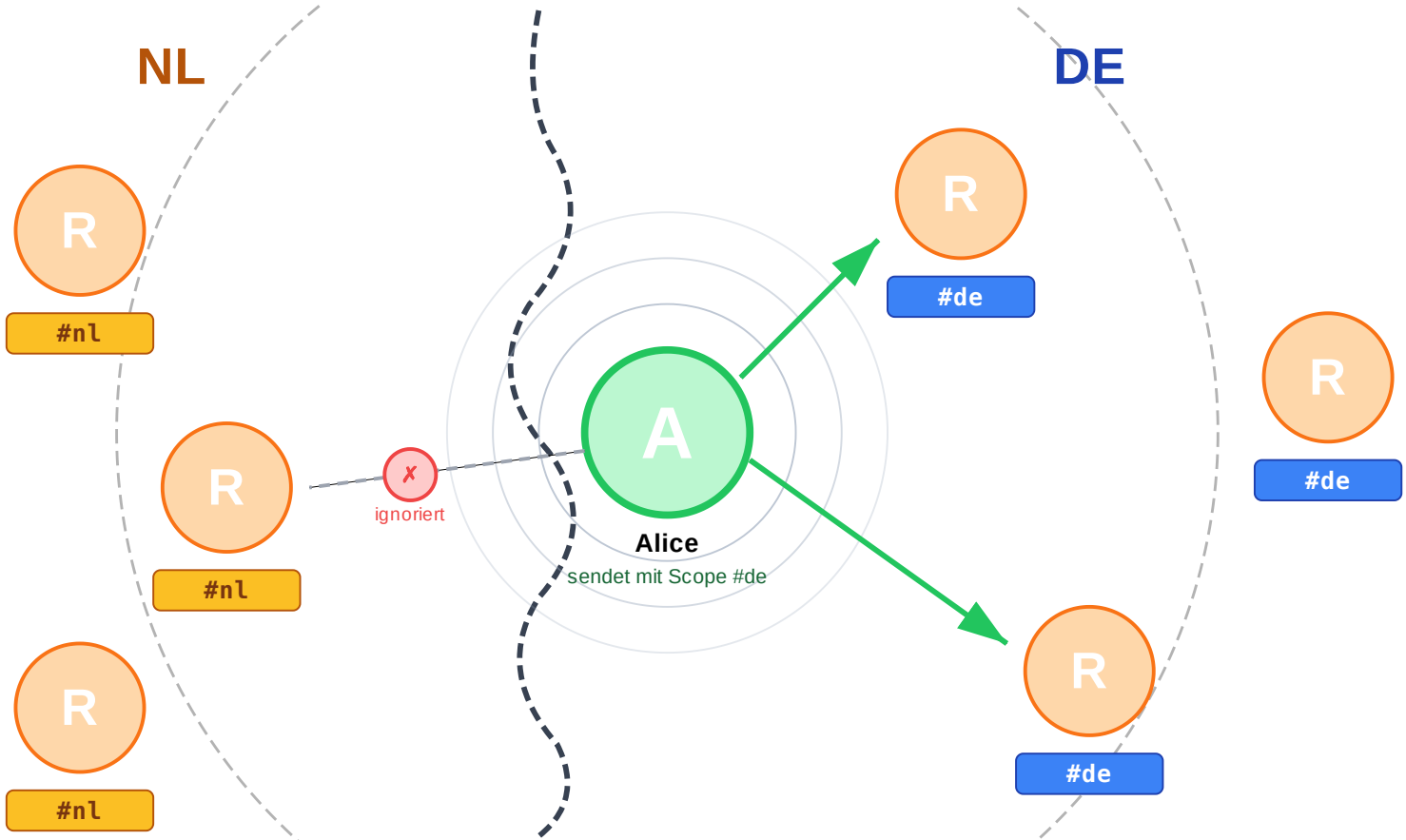


multi.acks EIN

Verdopplung auf jeder Strecke: Selbst bei 50% Paketverlust kommt ein ACK durch



Region Scopes



LoRa Airtime

Szenario: 254 Bytes Nachricht
Parameter: BW 62.5 kHz, SF 8

Berechnung

Symbol-Zeit:

```
T_symbol = 2^SF / BW = 256 / 62500 = 4.096 ms
```

Preamble (konstant):

```
T_preamble = (8 + 4.25) × 4.096 ms = 50.2 ms
```

Coding Rate 8 vs. 5

Payload-Symbole

CR5 (4/5 - 20% Redundanz):

```
n_symbols = 8 + ⌈(2032-32+44)/32⌉ × 5
          = 8 + 64 × 5 = 328
T_payload = 328 × 4.096 = 1343 ms
```

CR8 (4/8 - 50% Redundanz):

```
n_symbols = 8 + ⌈(2032-32+44)/32⌉ × 8
          = 8 + 64 × 8 = 520
T_payload = 520 × 4.096 = 2130 ms
```

Ergebnis: CR5 vs CR8

CR5

1.39s

✓ Schneller

20% Redundanz
Weniger Fehlerkorrektur

CR8

2.18s

✓ Robuster

50% Redundanz
Mehr Fehlerkorrektur

⚡ **CR5 ist 36% schneller (-790 ms)**

Trade-off: Geschwindigkeit ⚡ vs. Zuverlässigkeit 🛡️

Raum-Server