# Kausaler Multicast mittels Vektoruhren

Referent: Belal Karimzai 2417580
Bachelor Angewandte Informatik – Verteile Systeme
Betreuer: Prof. Dr. Klauck
HAW Hamburg Sommersemester 2021

# Gliederung

- Theoretische Grundlage
  - Logische Uhren und Happens-Before-Relation
  - Gruppenkommunikation : Zuverlässigkeit und Ordnung
- Entwurf
  - Bausteinsicht & Datenstrukturen
  - Laufzeitsicht – Ablauf der Sende- und Empfang-Ereignisse
- Vorteile & Nachteile von Vektoruhren
- Anwendungsszenarien

# Motivation für logische Uhren

- Verteiltes Verständnis von Zeit
- Problem der Uhrensynchronisation – perfekte Uhr de facto in verteilten Anwendungen unmöglich

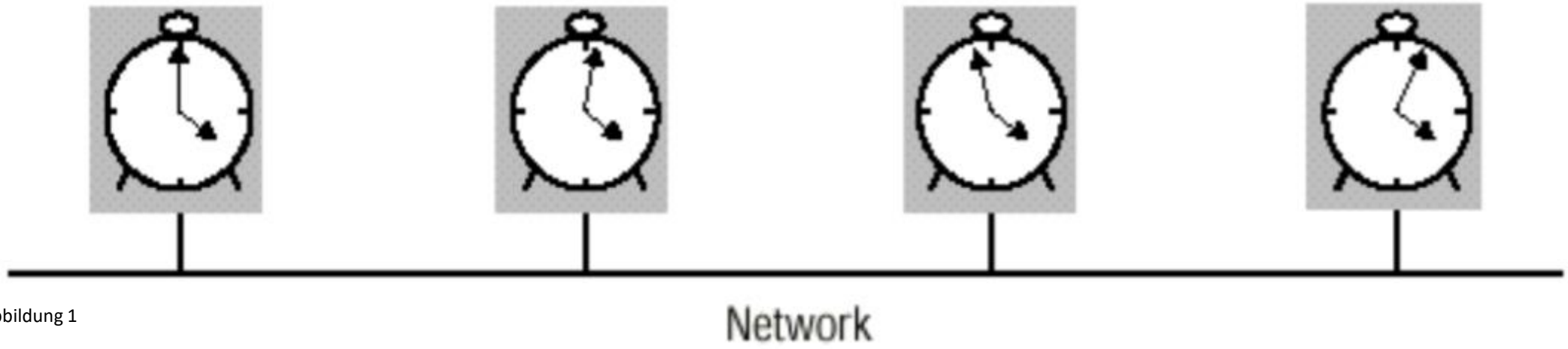
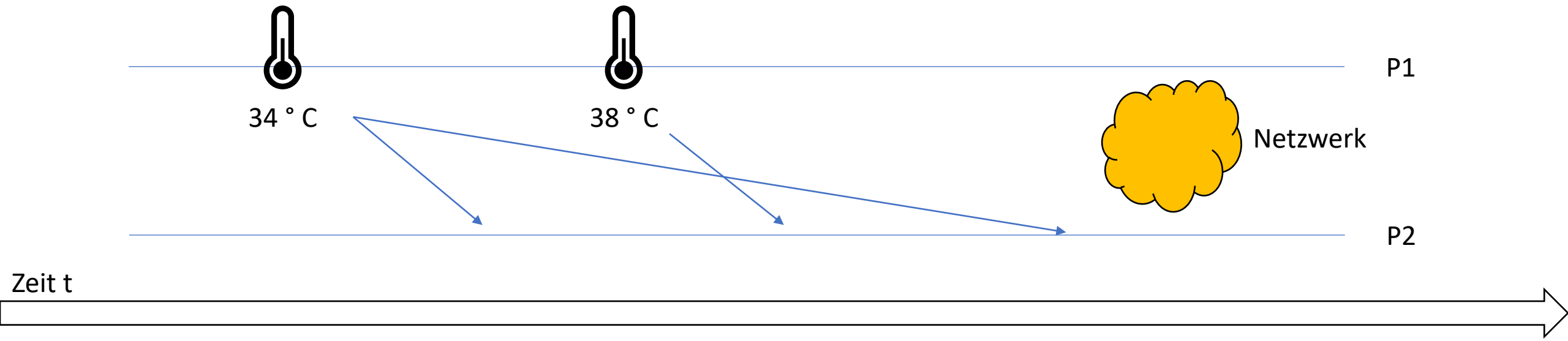



Abbildung 1

# Motivation für logische Uhren

- Zeit als Kriterium für Reihenfolge – Network Delay




- Aussagekraft von Timestamps – kausale Beziehung
- Synchronisation über paketorientierte Netzwerkkommunikation (NTP)

# Konzept der Logischen Uhren

- Ereignisse (Zeitpunkte) haben eine Beziehung zu einander
- Happens-Before Relation (e1 -> e2)
- Ereignis durch einen Zähler identifizierbar
- Unterscheidung von Ereignissen:
  - Origination
  - Transmission
  - Reception
  - Delivery

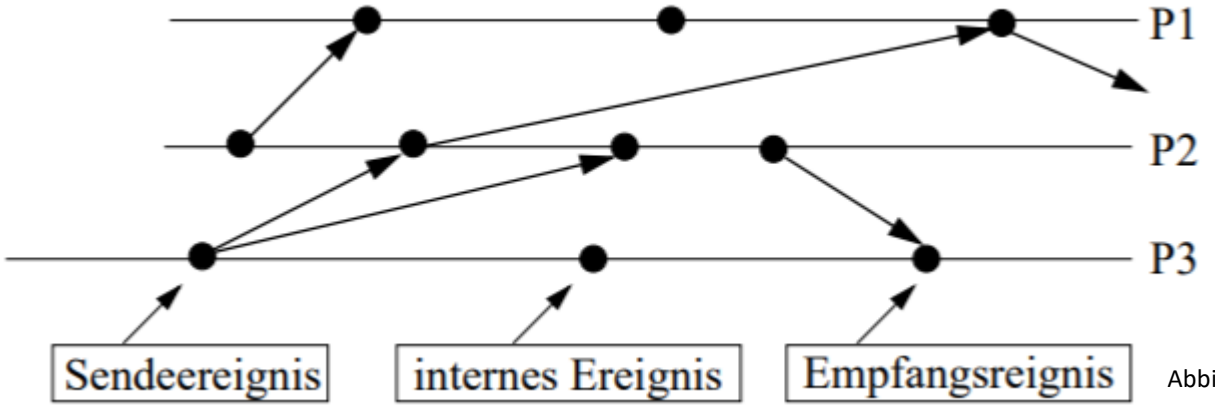
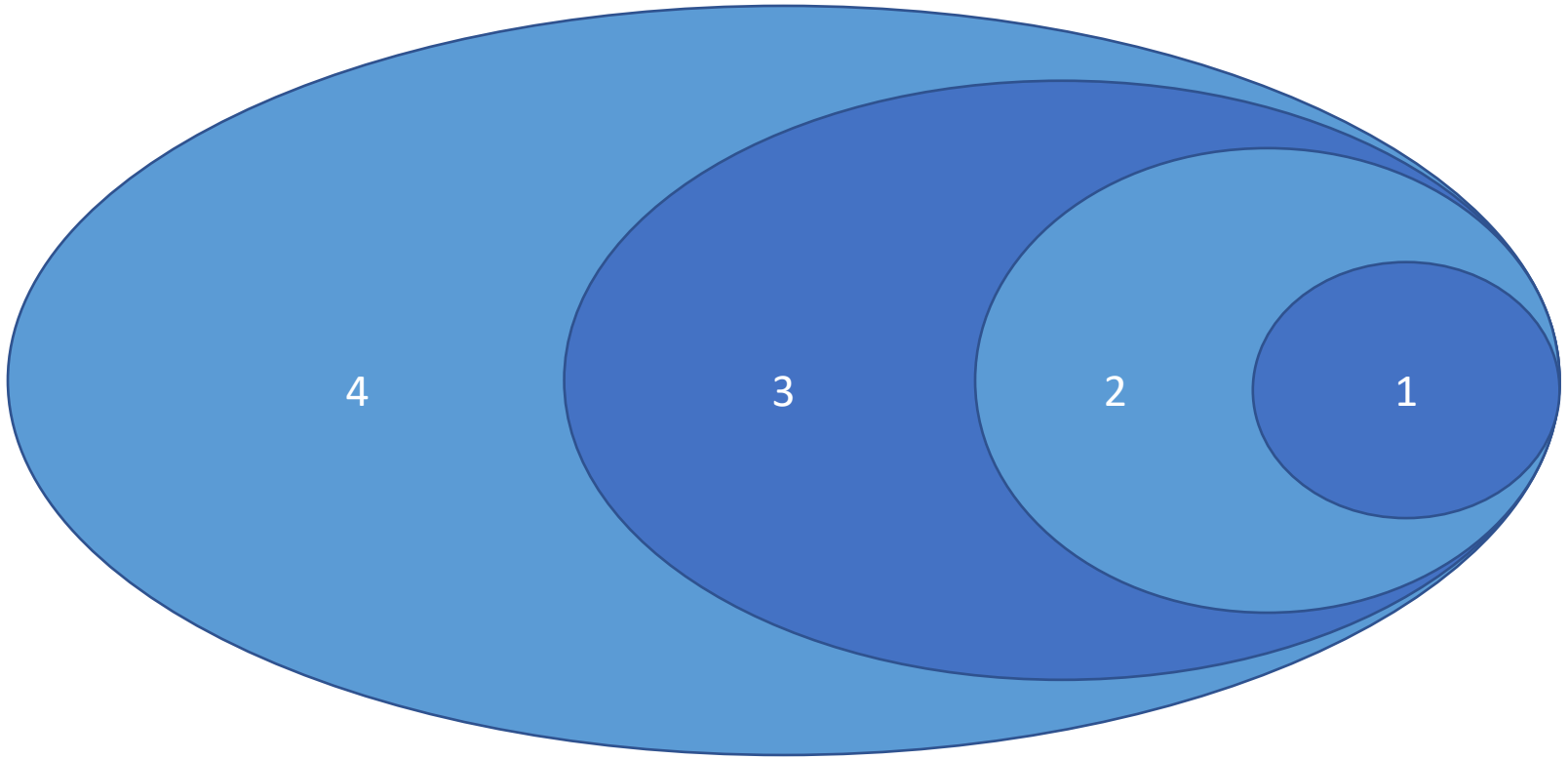



Abbildung 2

# Was ist Gruppenkommunikation? (Multicast)

Abbildung 3

# Gruppenkommunikation - Ordnung & Zuverlässigkeit




1 Total
2 Kausal
3 FIFO
4 Ungeordnet

# 1. Entwurf – Bausteinsicht

Abbildung 4

# Aufbau der Middleware-Anwendung

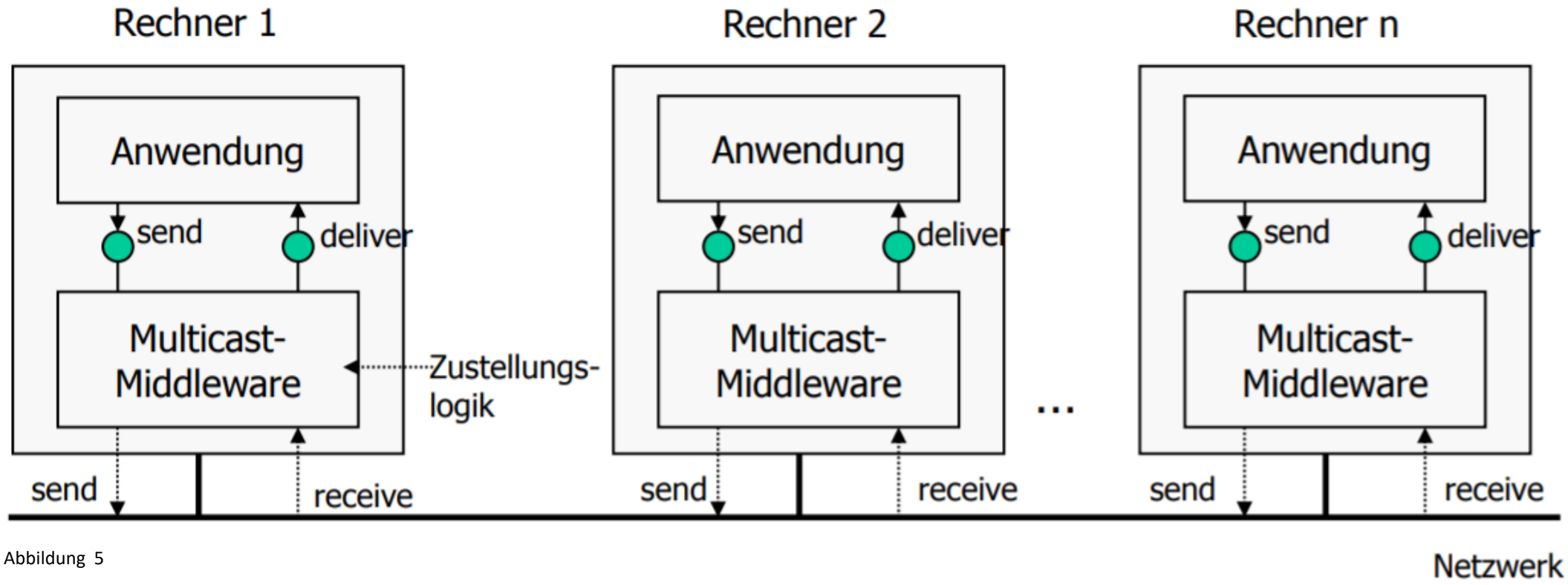
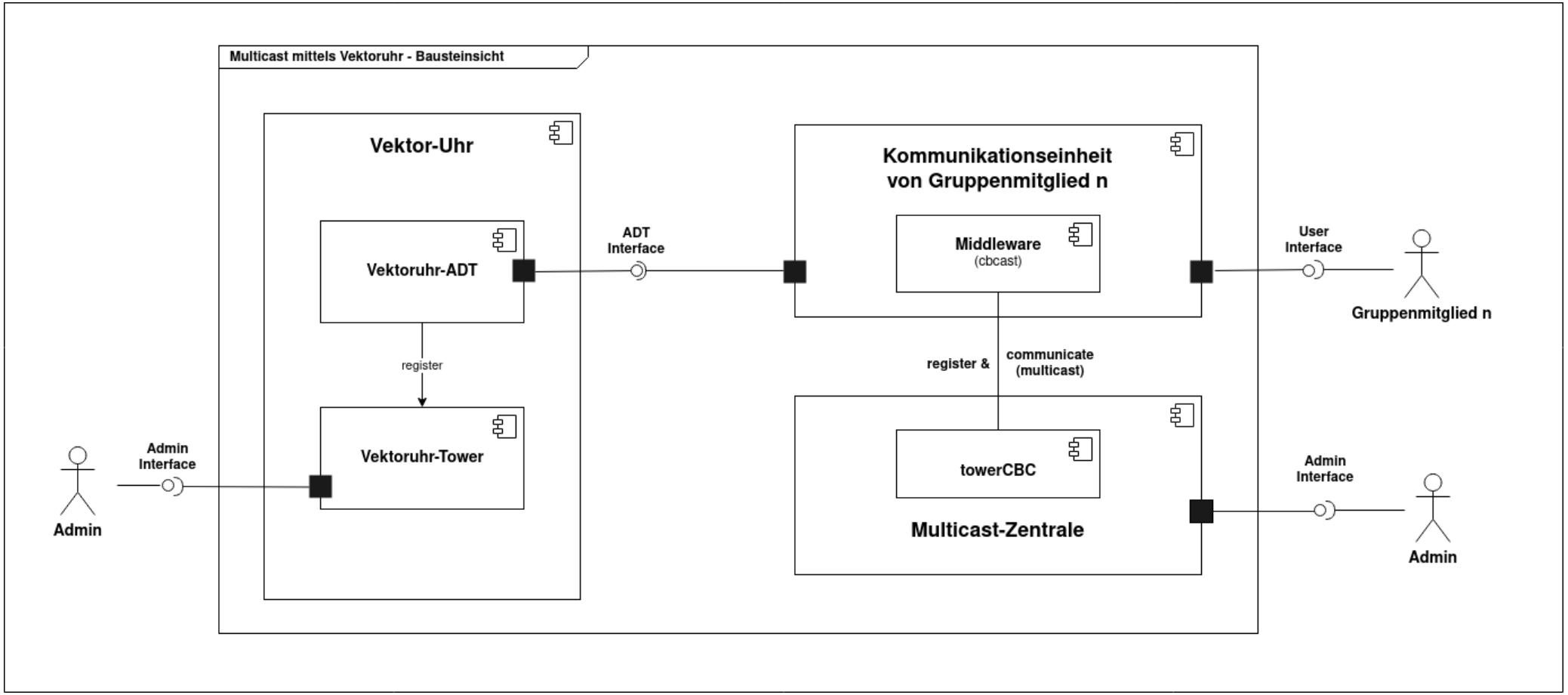



Abbildung 5

# Komponenten der Anwendung



# Datenstrukturen

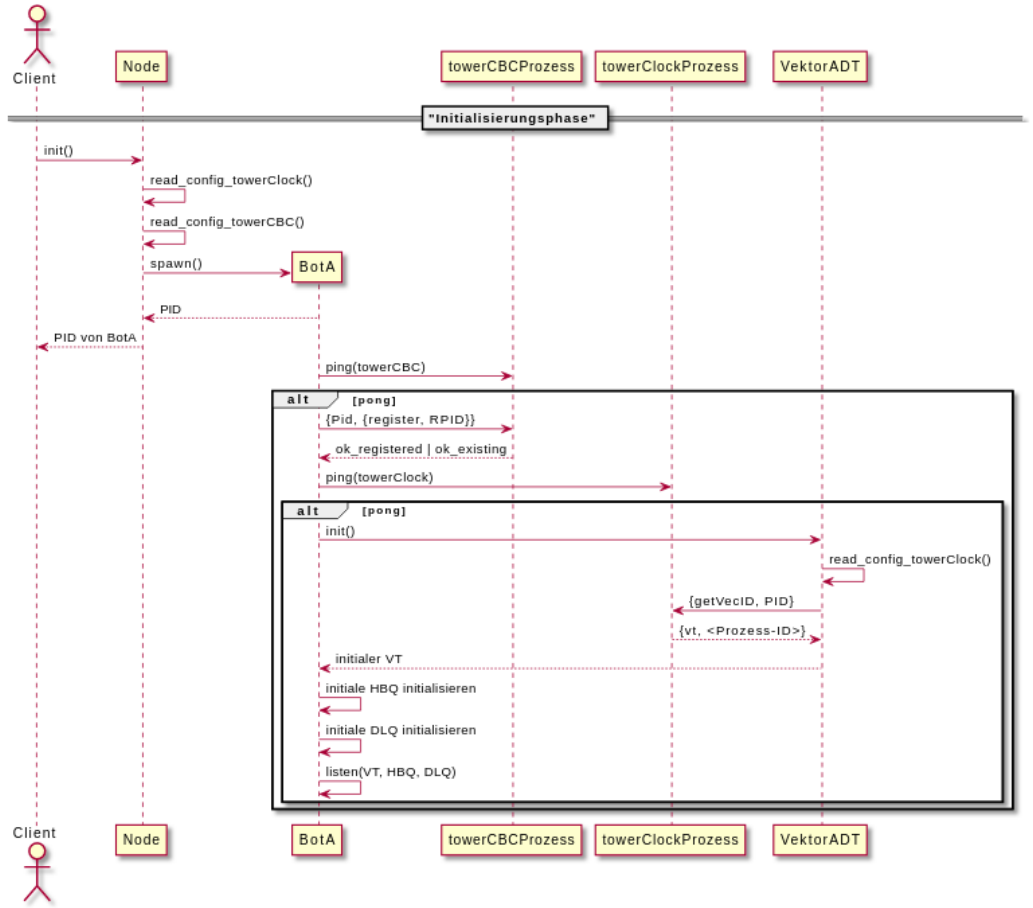
- VT besteht aus Pnum und VC : {Pnum, VC}
- HBQ : [ $\{Message, VT\}_1$, … , $\{Message, VT\}_n$ ]
- DLQ: [ $\{Message, VT\}_1$, … , $\{Message, VT\}_n$ ]

- Vorschlag für HBQ (Hash)
- [$\{Pnum_a$, [$\{Message, VT\}_1$, … , $\{Message, VT\}_n$ ]}, …,
  $\{Pnum_c$, [$\{Message, VT\}_1$, … , $\{Message, VT\}_n$ ]} ]

# 2. Entwurf – Laufzeitsicht

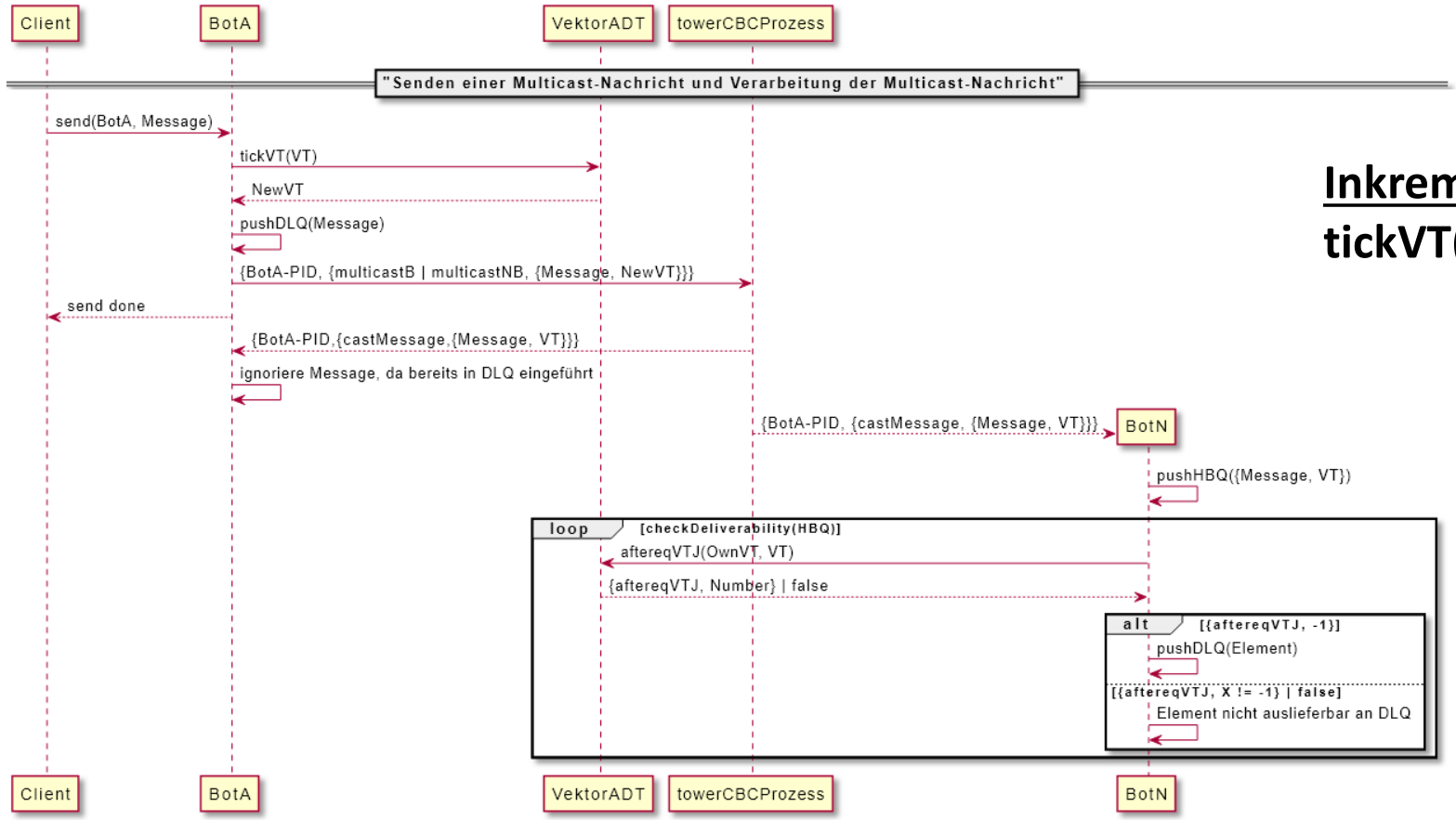

Abbildung 4

# Initialisierungsphase




**VT = {Pnum, [0]}**

# Sendeereignis

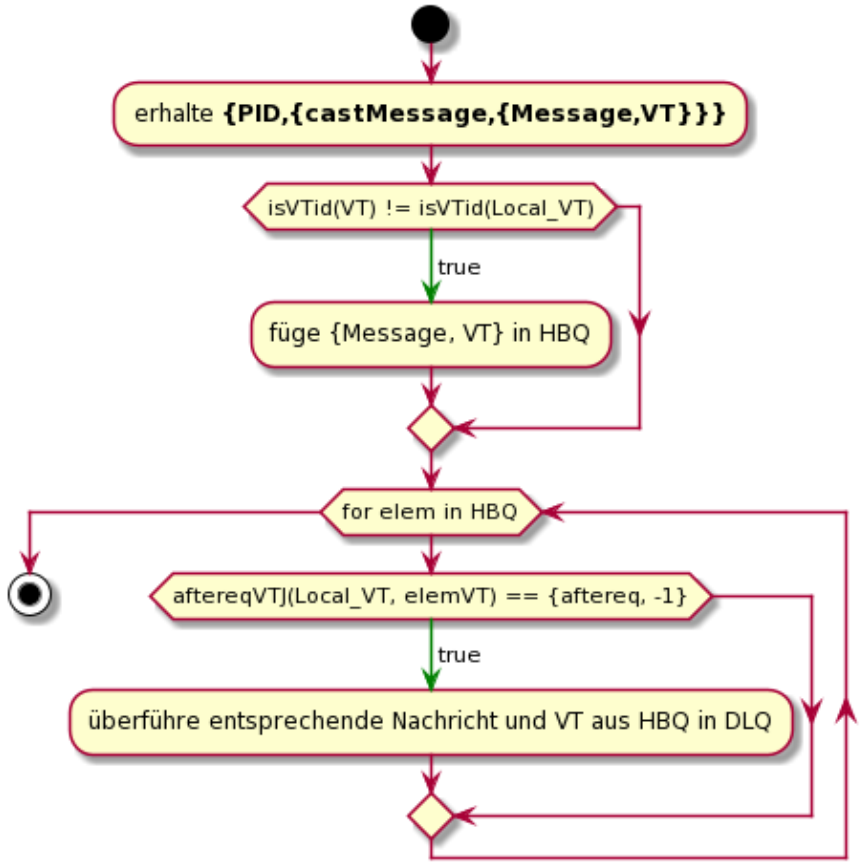



**<u>Inkrementiere Ereigniszähler:</u>**
**tickVT(VT)**

**VT = {Pnum, [0]}**

**VT = {Pnum, [1]}**

# Empfang-Ereignis an der Middleware



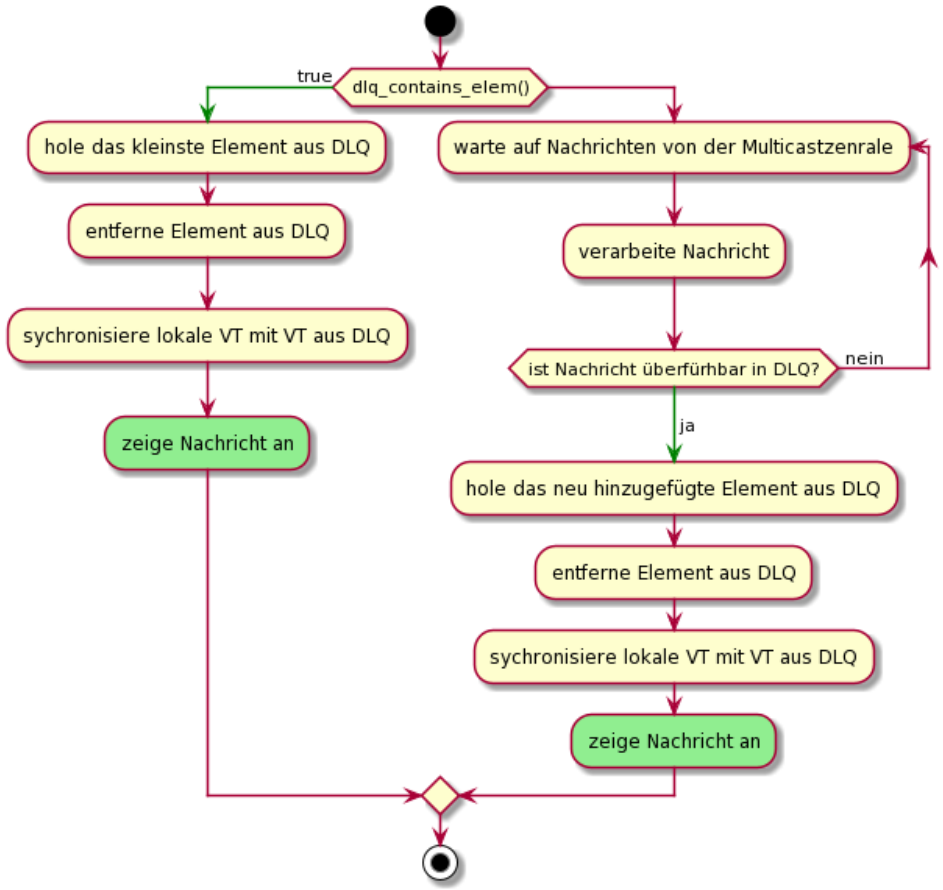

**Vergleichsmethode der Vektoruhr compVT(VT1, VT2):**

1. **VGL [0,1] > [0,0] -- aftertVT**
2. **VGL [0,1] < [0,2] -- beforeVT**
3. **VGL [1,0] || [0,1] -- concurrentVT**

**Bedingung für Auslieferung**

1. **VT[i] < vt[i]**
2. **VTi[k] ≥ vt[k] (für k ≠ j)**

# Auslieferung an anfragenden Client




**<u>Synchronisierung der Vektoruhren:</u>**
**syncVT(VT1, VT2)**

**NewVT = MAX(VT1, VT2)**

# Vorteile und Nachteile von Vektoruhren

- Vorteile:
  - Kausale Relation in beiden Richtungen (e1 -> e2) ⇔ L(e1) < L(e2)
  - Historie der Ereignisse abgespeichert

- Nachteil:
  - Overhead aufgrund der Zeitstempel mit steigender Anzahl an Teilnehmer
  - Skalierbarkeit und Performance leiden darunter

# Anwendungsszenario

- Transaktionen über Ethereum
- Ziel: Transaktionen Nonces zu ersetzen (eindeutige Nummer)
- Probleme: Nonce schützt vor Transaktionsduplizierungen
  - Zusammenhänge von Transaktionen sind nicht gegeben
  - Auftragsabhängigkeit bei der Verarbeitung von Transaktionen
- Lösung:
  - Vektoruhren
    - Problem: Skalierbarkeit und Performance
    - Lösung: Andere Ansätze, wie Binary Vector Clocks

# Bildquellen (letzter Zugriff 02.07.2021, 9.Uhr)

- Abb. 1: https://www.ibr.cs.tu-bs.de/courses/ss11/vs/VS-Kap05-Synchronisation-Single.pdf (Seite 8)
- Abb. 2: https://www.vs.inf.ethz.ch/edu/WS0203/VA/slides/Vorl.VertAlgWS0203_14.pdf (Seite 6)
- Abb. 3: https://unsplash.com/photos/2FPjlAyMQTA
- Abb. 4: https://unsplash.com/photos/fteR0e2BzKo
- Abb. 5: https://www.wirtschaftsinformatik-muenchen.de/wp-content/uploads/Peter%20Mandl/Lehrveranstaltungen/WiSe%2014-15/Verteilte%20Systeme/09-Gruppenkommunikation.pdf (Seite 40)