A Simple Chord Implementation for Distributed Context Management

Michael Karrenbauer, Jörg Schneider und Hans Schotten

19. ITG Fachtagung
Mobilkommunikation Osnabrück

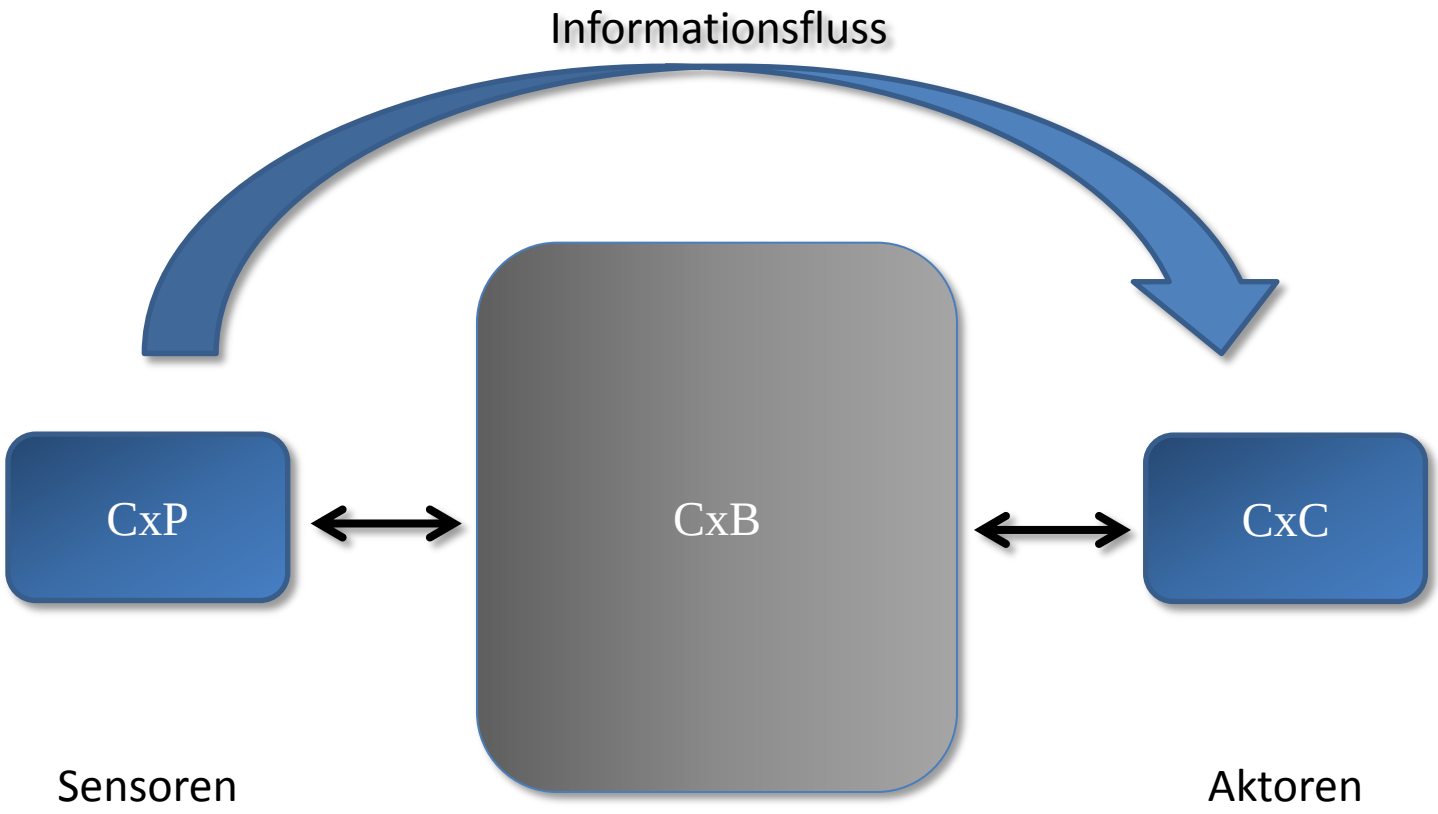
Inhalt

1. Motivation
2. Zentralisierte Kontextmanagementsysteme
3. Verteilte Kontextmanagementsysteme
4. Lightweight Chord-Implementierung
5. Messergebnisse
6. Zusammenfassung

Motivation

- Hohe Bedeutung kontextsensitiver Systeme in zukünftiger Kommunikationstechnik
- Verschiedene Ansätze für Kontextmanagementsysteme in der Literatur bekannt
- Man unterscheidet:
  - Kontextmanagementsysteme mit zentralisierter Architektur
  - Verteilte Kontextmanagementsysteme mit dezentralisierter Architektur

Zentralisierte Kontextmanagementsysteme

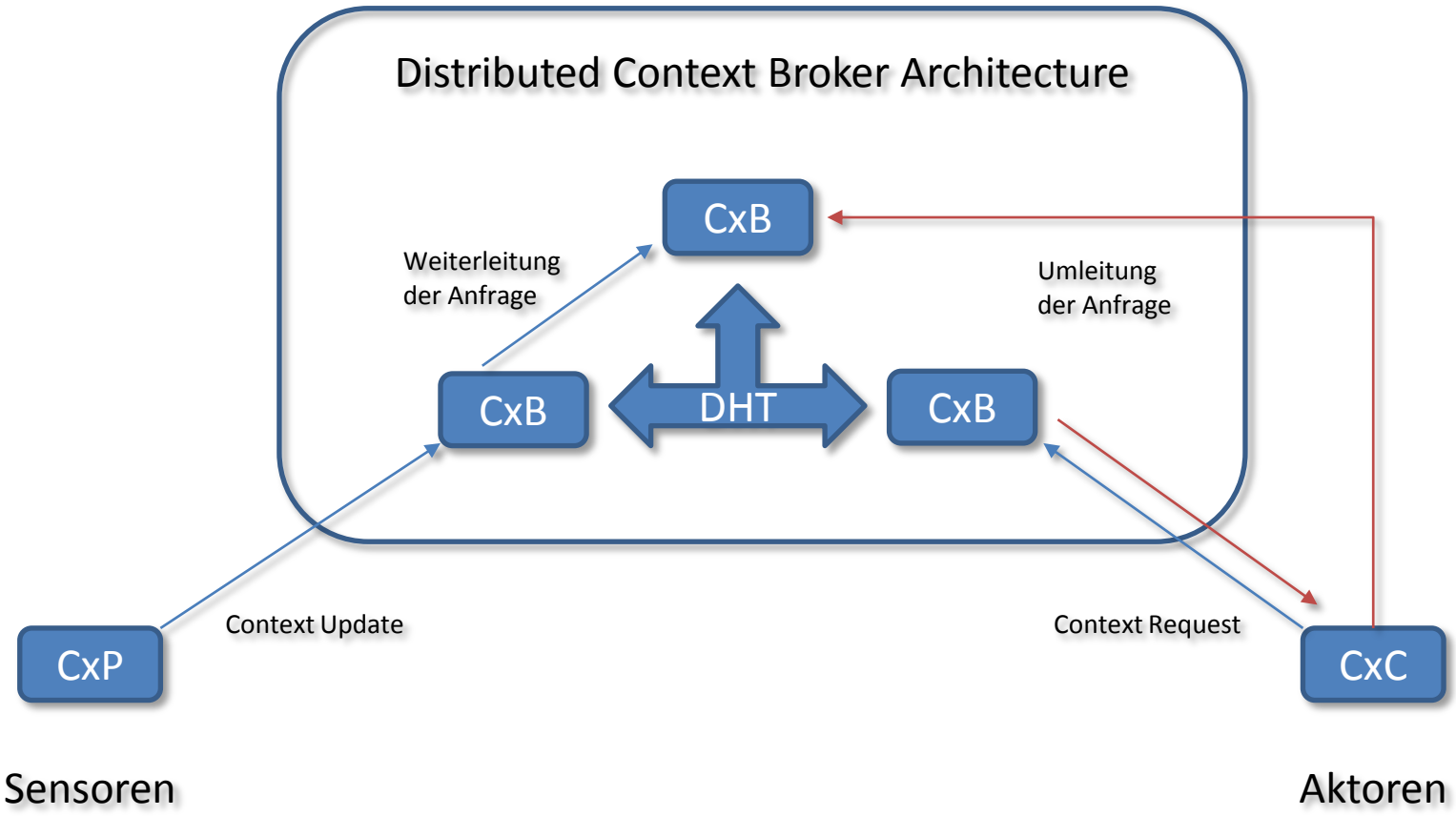


Zentralisierte Kontextmanagementsysteme

- Zentralisierte Kontextmanagementsysteme sind mit Einschränkungen behaftet bezüglich:
  - Robustheit (Ausfallsicherheit)
  - Skalierbarkeit
  - Datensicherheit (Redundanz)
  - Lastverteilung
- Verteilte Kontextmanagementsysteme überwinden diese Nachteile durch die Verwendung von Distributed Hash Tables (DHT) [1] , sind aber bisweilen auf leistungsstarke Hardware angewiesen

[1] “A distributed context service architecture for heterogeneous radio Environments“, Proceedings of the 15th ”ITG Fachtagung Mobilkommunikation”, Osnabrück, Germany, May 2010

Verteilte Kontextmanagementsysteme



Verteilte Kontextmanagementsysteme

- Das vorliegende Kontextmanagementsystem basiert auf dem Peer-to-Peer-Netzwerk „Chord“, welches von Ion Stoica et al. am MIT entwickelt wurde [2]
- Chord als zugrundeliegendes Peer-to-Peer-Netzwerk wurde ausgewählt wegen:
  - seiner einfachen und dadurch gut auf die Bedürfnisse von leistungsschwacher Hardware abzubildenden Topologie
  - der Möglichkeit effizient nach abgespeicherten Datensätzen zu suchen (Komplexität $\mathcal{O}(\log(n))$)

[2] “Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications“, Stoica, Ion et al., MIT Laboratory for Computer Science, online verfügbar: http://pdos.csail.mit.edu/papers/chord:sigcomm01/chord sigcomm.pdf, letzter Zugriff: 19.05.2014

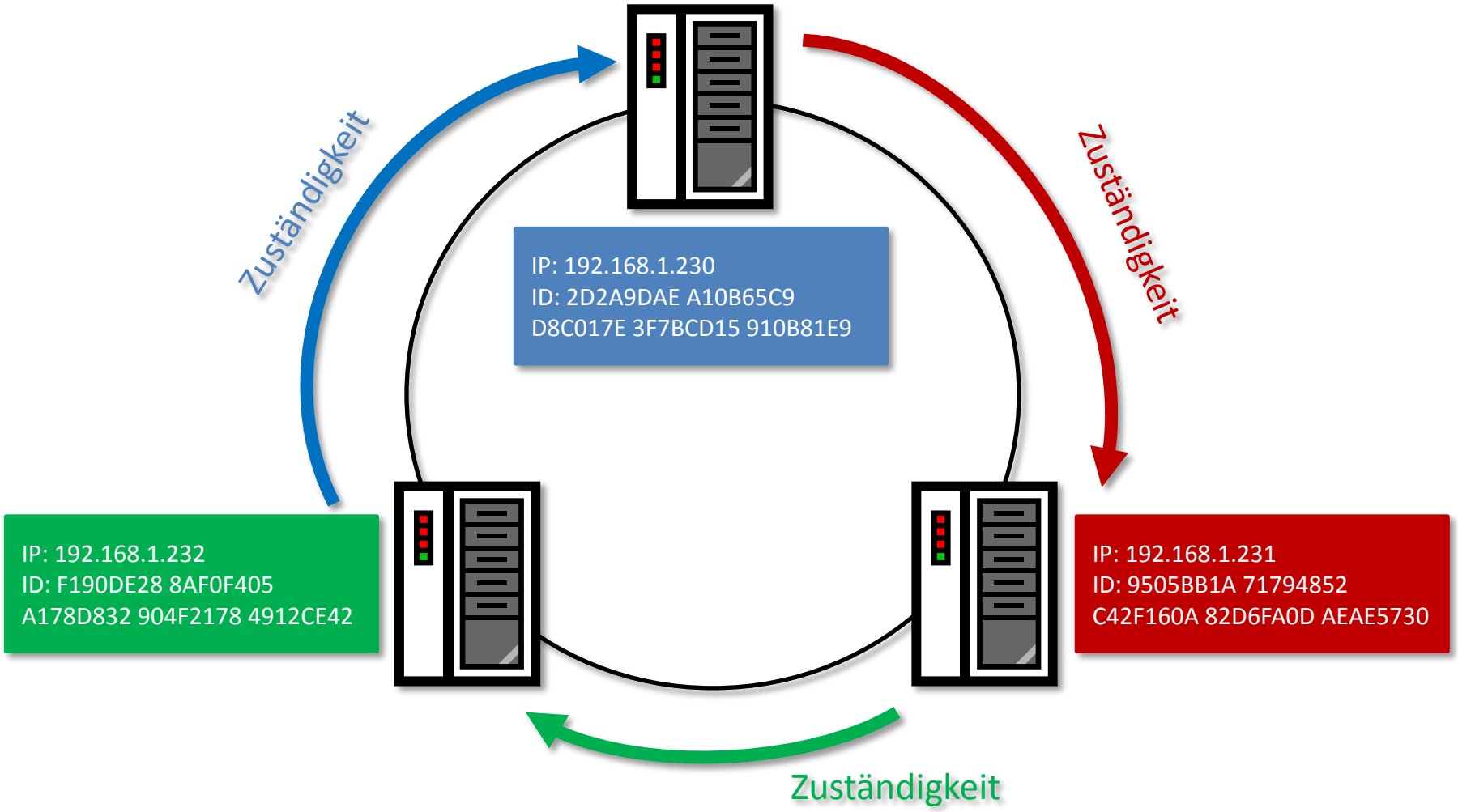
Verteilte Kontextmanagementsysteme

- Teilnehmer sind in einer Ring-Topologie angeordnet
- ID der Teilnehmer ergibt sich durch Hashing ihrer IP-Adresse
- Datensätze werden ebenfalls durch Hashing auf Ring-Topologie angeordnet
- Als Basis-Hashfunktion kommt SHA-1 [3] mit einer Breite von 160 Bit zum Einsatz
- Die maximale Teilnehmeranzahl ergibt sich damit zu:

$$N = 2^{160} = 1{,}46 \cdot 10^{48}$$

[3] " US Secure Hash Algorithm 1 (SHA1)",
Network Working Group, RFC 3174,
Online verfügbar: http://tools.ietf.org/html/rfc3174,
letzter Zugriff: 19.05.2014

Lightweight Chord-Implementierung



Lightweight Chord-Implementierung

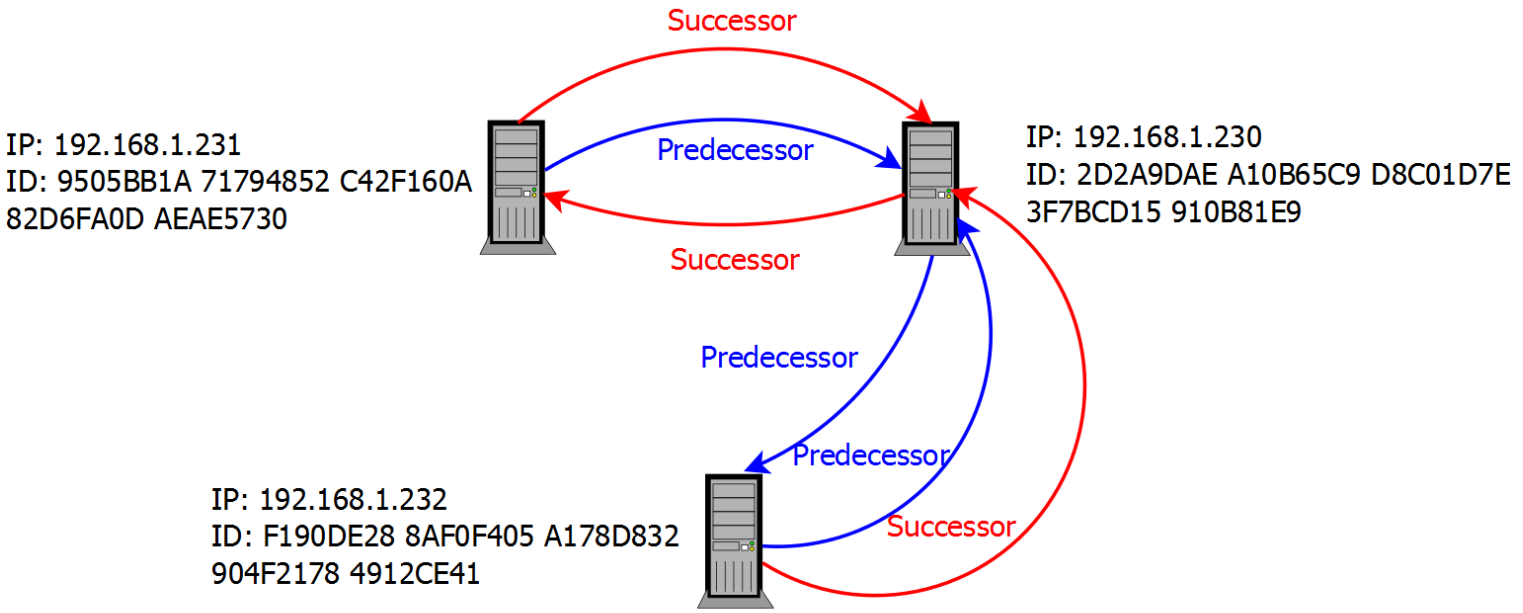
Stabilisierungs-Task:

a. Fix Predecessor Task:
   - Stellt sicher, dass Verweis auf Vorgänger jederzeit korrekt ist

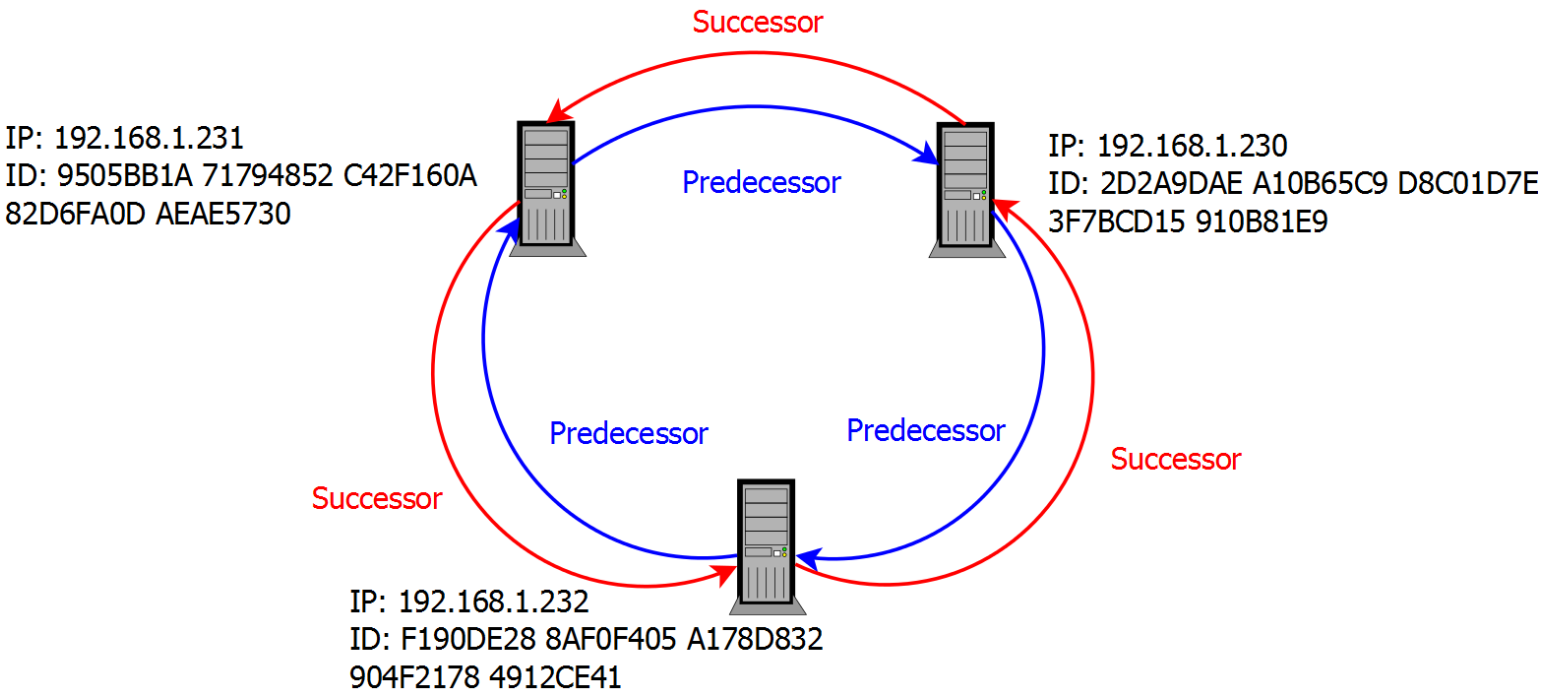
b. Fix Successor Task:
   - Stellt sicher, dass Verweis auf Nachfolger jederzeit korrekt ist

c. Fix Fingertable Task:
   - Stellt durch periodische Verbindungsversuche sicher, dass Verweise innerhalb der Fingertable noch gültig sind

Lightweight Chord-Implementierung



Lightweight Chord-Implementierung



Lightweight Chord-Implementierung

Nachrichtentypen der schlanken Chord-Implementierung:

a) *bootstrapNode:* Dient dem Hinzufügen eines neuen Knotens zu einem bestehenden Chord-Netzwerk. Ihm folgt die IP des zu übermittelnden Bootstrap-Nodes.

b) *join:* Wird von einem hinzukommenden Knoten an den Bootstrap-Node gesendet, gefolgt von dessen IP-Adresse.

c) *getPredecessor:* Wird an den hinterlegten Nachfolger des Knotens geschickt, gefolgt von dessen IP-Adresse, zum Zwecke der Stabilisierung.

d) *getSuccessor:* Wird in analoger Weise an den hinterlegten Vorgänger des Knotens geschickt.

e) *SUCC:* Antwort auf ankommende *getSuccessor*-Nachricht.

f) *PRED:* Antwort auf ankommende *getPredecessor*-Nachricht.

Messergebnisse

Antwortende Knoten	**1**	**2**	**3**
Anzahl der Anfragen	3.000	3.000	3.000
Größe einer Anfrage / Bytes	598	598	598
Anfragen pro Sekunde (Durchschnitt)	7,4	8,0	10,2
Minimale Antwortzeit / ms	37	18	49
Durchschnittliche Antwortzeit / ms	134	124	97
Standardabweichung / ms	52,57	43,48	51,02
Maximale Antwortzeit / ms	280	285	354

Zusammenfassung

- Eine ressourceneffiziente Chord-Implementierung zum Kontextmanagement unter Verwendung von leistungsschwacher Hardware (bspw. Mikrocontrollern) wurde entwickelt
- Ein Proof-of-concept Demoaufbau bestehend aus drei Mikrocontrollerboards wurde erstellt
- Erste Messergebnisse wurden ermittelt

Vielen Dank für Ihre Aufmerksamkeit!