Untersuchungen zur Komplexität komponierter Netzwerkarchitekturen im Future Internet

Nora Berg, Sebastian Meiling,
Thomas C. Schmidt, Matthias Wählisch

6. September 2013

Inhalt

Motivation

- Jetziges Internet hat einige Probleme
  - Middleboxes (NATs, Firewalls)
  - Skalierbarkeit des Routings
  - Fehlende Services: z. B. Multicast
  - ...
- Future-Internet-Frameworks beheben einzelne Probleme

Fragestellungen

- Was passiert wenn man FI-Frameworks kombiniert, um mehrere Probleme auf einmal zu lösen?
- Wie komplex werden diese kombinierten Netzwerkstacks?
- Was ist Komplexität in Netzwerk-Stacks und wo entsteht sie?

→ Die Komplexität wird anhand der Komposition von H∀Mcast und Ariba/MCPO untersucht.

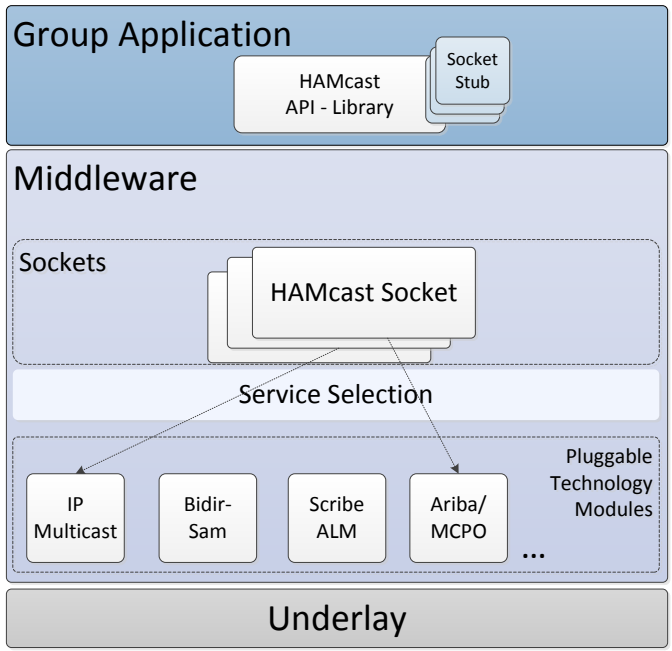
HΛMcast

Konzept

- Vernetzung verschiedener Multicast-Domänen
- Abstrahiert von unterliegenden Multicast-Technologien
- Einheitliche Schnittstelle für Gruppenkommunikation

Architektur

- Common API als Schnittstelle für Programme
- Middleware nimmt Funktionsaufrufe entgegen
- Module implementieren die einzelnen Multicast-Technologien



`http://www.realmv6.org/hamcast.html`

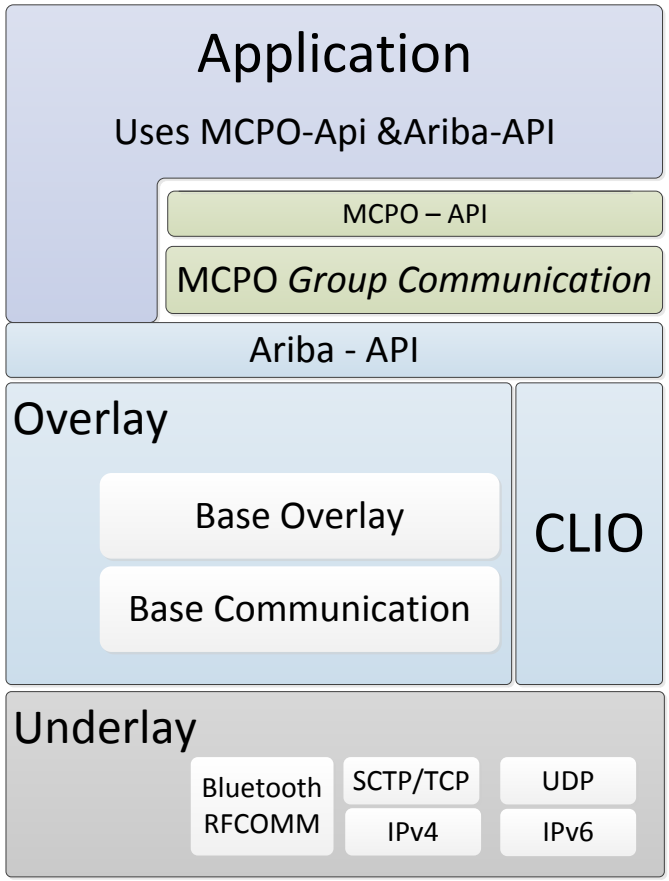
Ariba/MCPO

Ariba

- Unicast-Overlay
- Überwindung von Middleboxes und Protokollheterogenität
- CLIO [optional] - Informationen aus unteren Ebenen

MCPO

- Gruppenkommunikation über Ariba
- MCPO erweitert die Ariba-Schnittstelle um Multicast-Funktionalität
- Datenverteilung ähnlich des NICE-Protokolls



`http://ariba-underlay.org/`

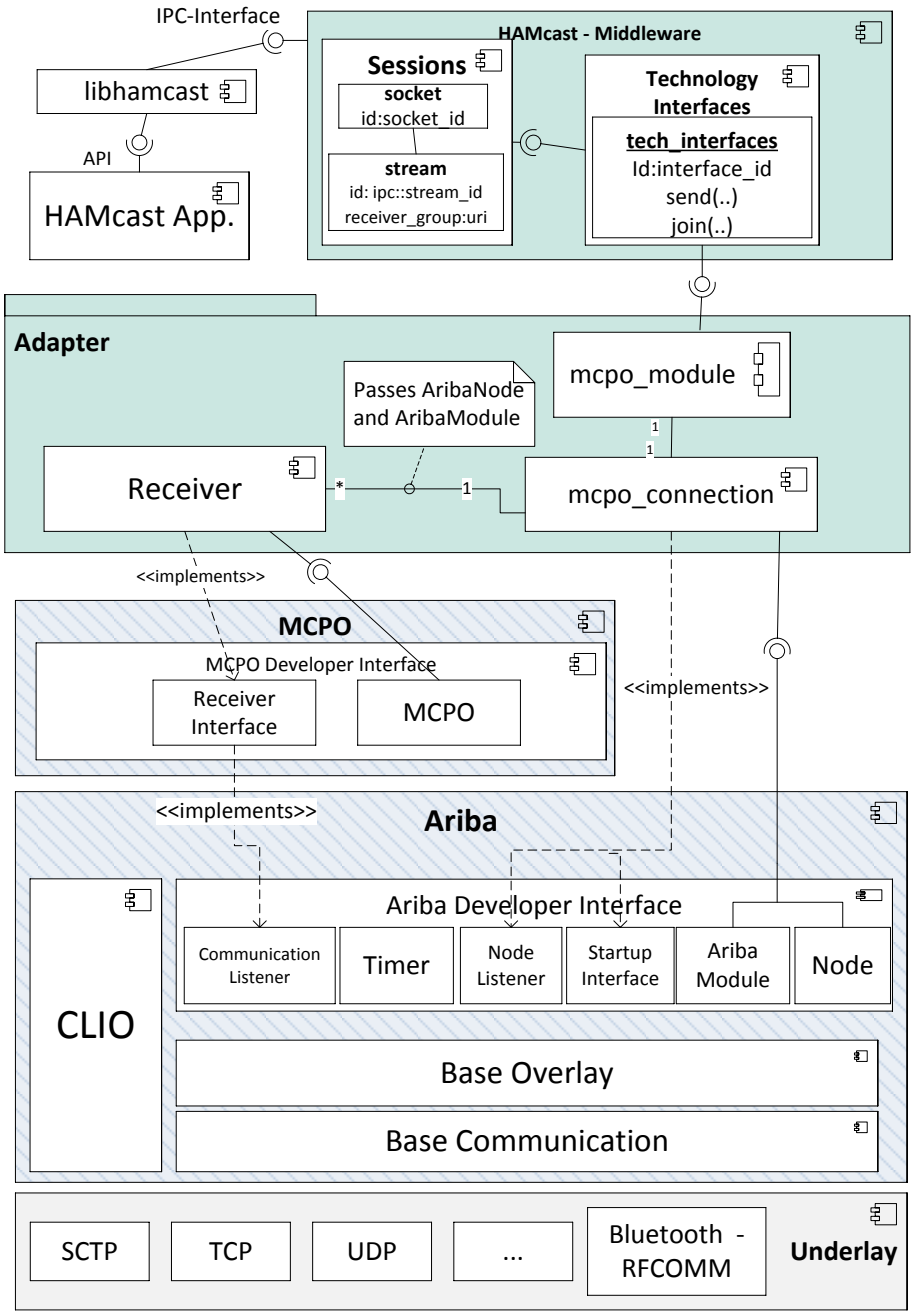
Adaptermodul

Kompositon

- Ariba/MCPO als Multicast-Technologie für H∀Mcast
- Adapter-Modul als Wrapper für Ariba/MCPO

Ziele

- Vermittlung zwischen den Schnittstellen
- effiziente Abbildung der verschiedenen FI-Konzepte



Komplexität

Was ist Komplexität?

Network complexity is proportional to state, dependencies between components, and rate of change in a network. Too much complexity can cause unpredictable, non-linear behaviour.

M.Behringer

Quelle: IRTF/NCRG - `http://networkcomplexity.org`

Komplexität

Vorgehen

- Frage: Wie entsteht Komplexität?
- Suche nach Merkmalen, die Komplexität vermehren

Untersuchung

- des Kommunikationsablauf:
  - Anzahl der Netzwerkschichten
  - Serialisierungen der Daten
- der Steuerung:
  - Verwaltung der Ebenen
  - Durchlässigkeit der Ebenen
  - Funktionsvielfalt
  - Verknüpfung der Architekturen und Schnittstellen

Funktionalität

Funktionalität

- Breite des Aufgabenbereiches
- mehr Möglichkeiten das Verhalten der unteren Ebene zu beeinflussen

Parameter

- Verwaltung zusätzlicher Abstraktionsebenen
  - benötigt zusätzliche Konfigurationsparameter
- Art der Parameterfindung beeinflusst Komplexität
- möglichst viele Parameter abbilden (und dabei Konzepte erhalten)
  - Beispiel: Adressierung der Multicast-Gruppen

Adressierung

Adressierung in H∀Mcast:

```
namespace :// group @ instantiaton : port / credentials
```

Adressierung in Ariba/MCPO

- SpoVNet-Name
- Knotenname zur Identifizierung eines Knotens im SpoVNet
- ServiceID zur Identifizierung einer Anwendung in Ariba
- ServiceID zur Identifizierung einer MCPO-Gruppe

Problemstellung:

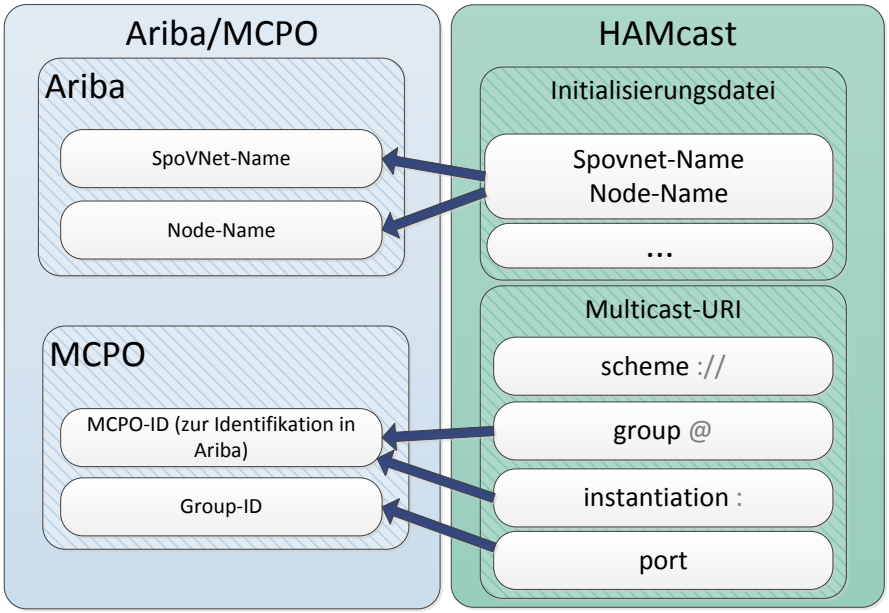
- Abbildung dieser Konfigurationsdaten
- Konzepte erhalten
- möglichst wenig neue Komplexität

Adressierung

Abbildung

Vorteile

- Nutzen der Funktionalitäts-Breite
- Skalierbarkeit von Ariba/MCPO unter H∀Mcast
  - mehrere MCPO-Instanzen
- Erhalten der Overlay-Konzepte von H∀Mcast und Ariba/MCPO

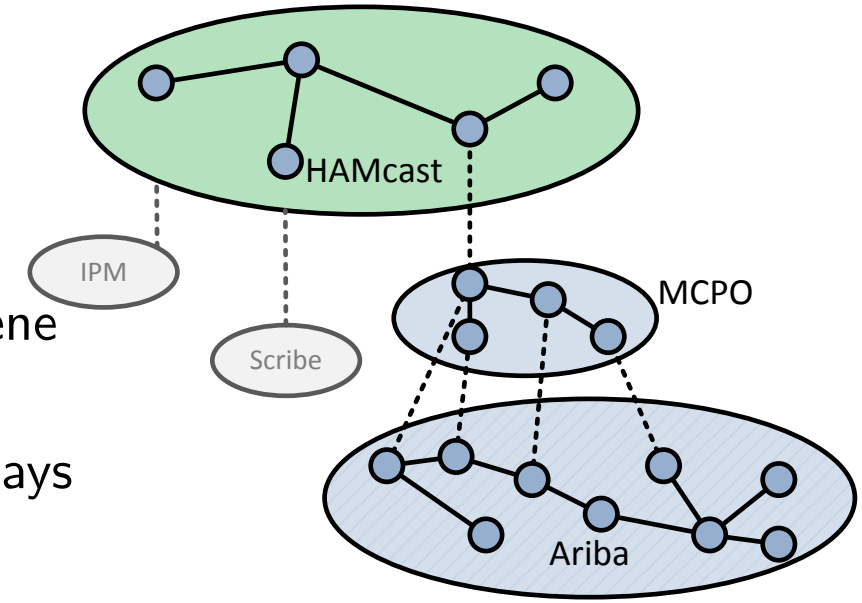


Nachteile

- Cross-Layer-Information durch SpoVNet- und Node-Name
- Zusatzkonzept SSM in Ariba/MCPO nur ineffizient abbildbar

Geschichtete Overlays

- Overlay-Netze erzeugen Verwaltungsaufwand
- Für jede Multicast-Gruppe auf H∀Mcast-Ebene ein neues MCPO-Objekt
- Der Initialisierungsaufwand des MCPO-Overlays wird zum konstanten Aufwand bei *join*

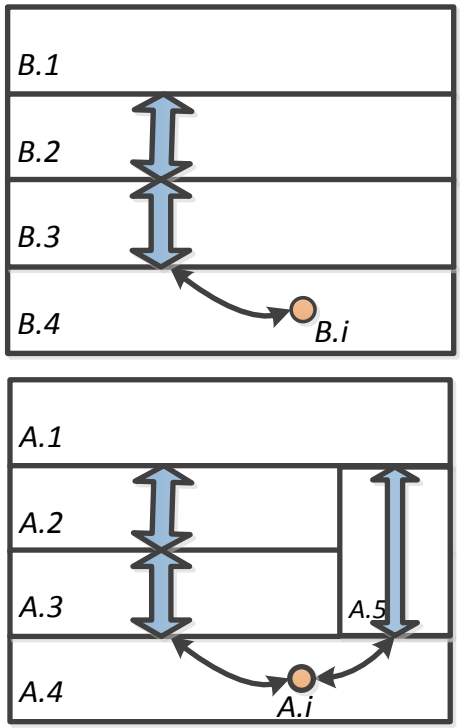


Problemstellung:

- Zeitspanne für den Gruppenbeitrittsprozess (*join*) mit dem Ariba/MCPO-Adaptermodul

Cross-Layer-Elemente

- Klassischer Informationsfluss über 2 Ebenen
- Cross Layer: direkter Informationsfluss über mehr als 2 Ebenen
- Einfluss verschiedener Komponenten auf einzelne Information *(A.i)*
- Abhängigkeit der Komponenten (*A.3*, *A.4*, *A.5*) von dieser Information
- *A.1* benötigt Informationen über Komponenten *(A.5, A.4, A.3)*
  - Was ist *A.i*?
  - Wie wirkt sich *A.i* in *A.3*, *A.4* und *A.5* aus?



Problemstellung

- Cross-Layer-Infos für neue/verbesserte Funktionalität im F.I.
- Konflikt: erhöhte Komplexität vs. Mehraufwand durch Neuimplementierung

Konflikte I

Ende-zu-Ende-Prinzip

- Funktion soll auf der Ebene sichergestellt werden, auf der sie benötigt wird.
- unterliegende Schichten haben kein Informationen über Anwendung (sollen abstrakt sein)

$\rightarrow$ Schichtenprinzip kann zu Leistungseinbußen führen

Eigenschaften an der falschen Stelle

- wird Eigenschaft sichergestellt, aber nicht benötigt $\rightarrow$ zusätzlicher Aufwand
- „Vergessen“ von Eigenschaften $\rightarrow$ Neuimplementierung und noch mehr Aufwand

Problemstellung

- es kann nicht frei festgelegt werden, *wo* im Stack, Funktionen bereitgestellt werden
- Abwägung zwischen dem E2E-Prinzip und dem Transparenzprinzip

Konflikte II

Verbindungsorientierung vs. Verbindungslosigkeit

- Multicast verbindunglos
- unterliegene Ebenen verbindungsorientiert
- konstanter zusätzlicher Aufwand für jede Nachricht die verschickt wird

zuverlässiger vs. unzuverlässiger Multicast

- Multicast traditionell unzuverlässig
- MCPO und H∀Mcast ignorieren die Zuverlässigkeit Aribas
- hoher Aufwand, falls Anwendung wieder Zuverlässigkeit benötigt

Problemstellung:

- Aufwand beim Versenden einer Nachricht?
- Zuverlässigkeit in Multicast erhalten, um das unterliegende Konzept weiterzuführen - sinnvoll?

Einfluss auf Komplexität

Merkmale für Komplexität

- Komplexität der Control-Operationen:
  - Anzahl der Cross-Layer-Elemente
  - Anzahl der nicht abgebildeten Konfigurationsparameter
- Komplexität der Data-Forwarding-Ebene:
  - Namensgebung und Adressierung innerhalb einer Ebene
  - Garantierte Eigenschaften der einzelnen Ebenen (z.B. „Zuverlässigkeit“)
- Komplexität der Anwendungsschnittstelle
  - Anzahl der verschiedenen Funktionalitäten einer Schnittstelle
  - Anzahl der Cross-Layer-Elemente, die auf Anwendungsebene ausgelagert werden

Tests
Übersicht

Durchsatz und CPU-Auslastung

- maximaler Nachrichten-Durchsatz
- Aufwand beim Versenden und Empfangen einer Nachricht
- 7 Paketgrößen
- 2 direkt verbundene Rechner

Gruppenbeitrittsverzögerung

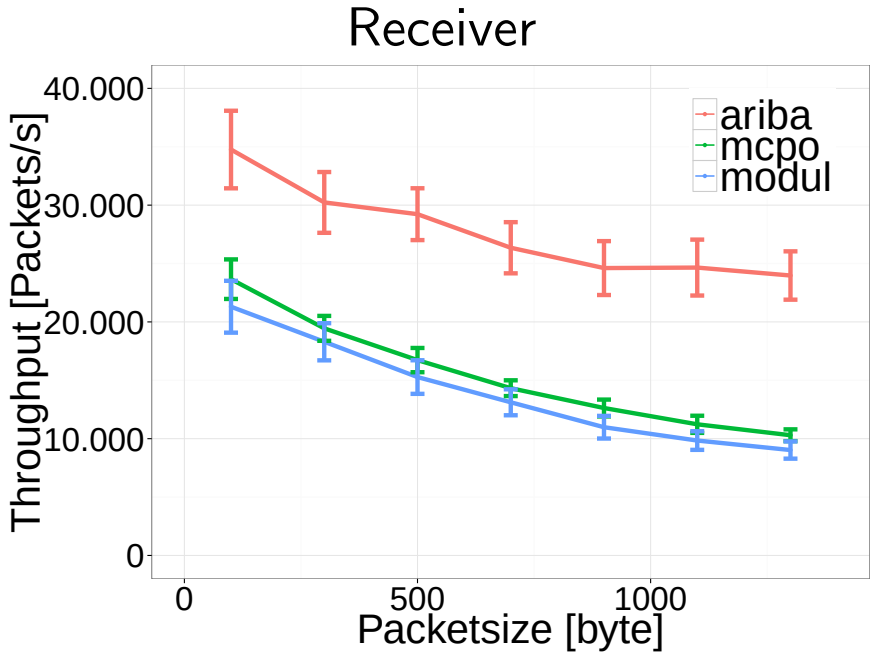
- Abbildung der Konzepte führt zu erhöhtem Aufwand bei *join*
- 21 Knoten im Planetlab
- Messung der Zeit vom *join* bis zur ersten Nachricht

Test: Durchsatz

Testergebnisse

- Maximale Obergrenze des Durchsatzes
- Beeinflussung durch:
  - Overlay-Verwaltungen
  - Kopieroperation in H∀Mcast
- weitere Module im Vergleich:
  - Scribe: ∼250.000 pkt/s
  - BIDIR-SAM: ∼290.000 pkt/s
- geringer Paketverlust

  (∼ 1 Paket in 19.2 sek)

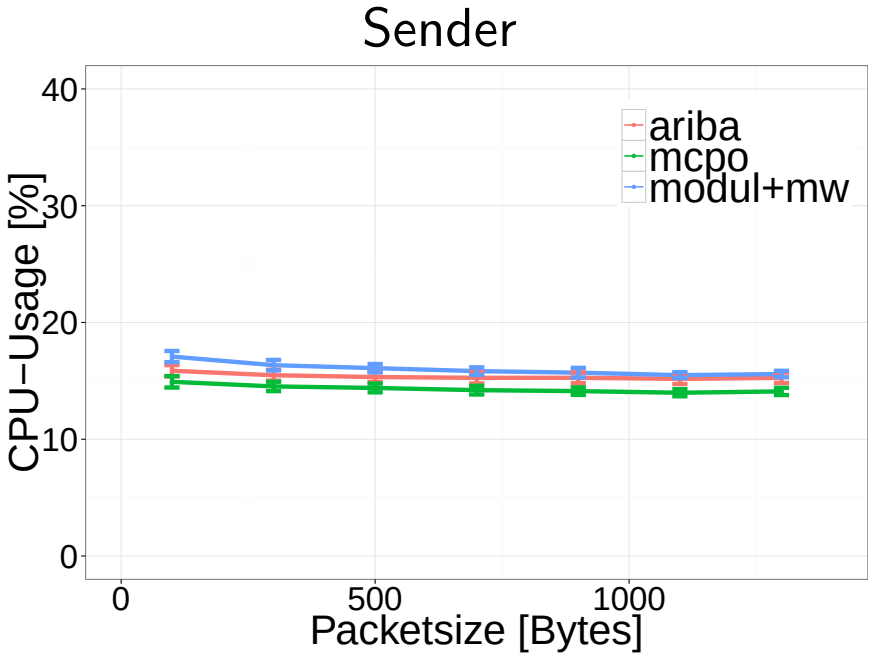


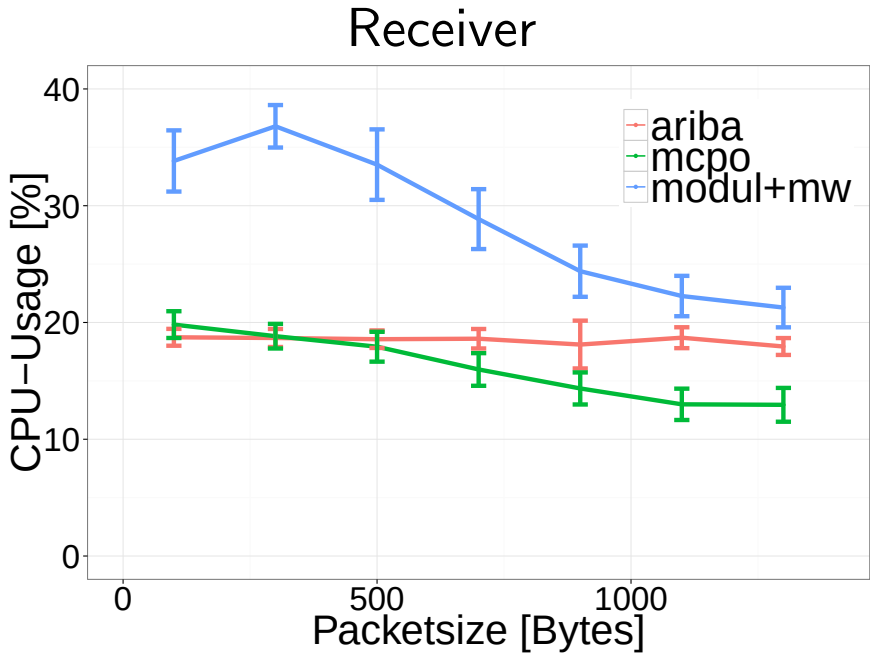
Folge der Komplexität aus:

- Anzahl der Netzwerkschichten
- Garantierten Eigenschaften der unteren Ebenen

Test: Durchsatz

CPU-Belastung





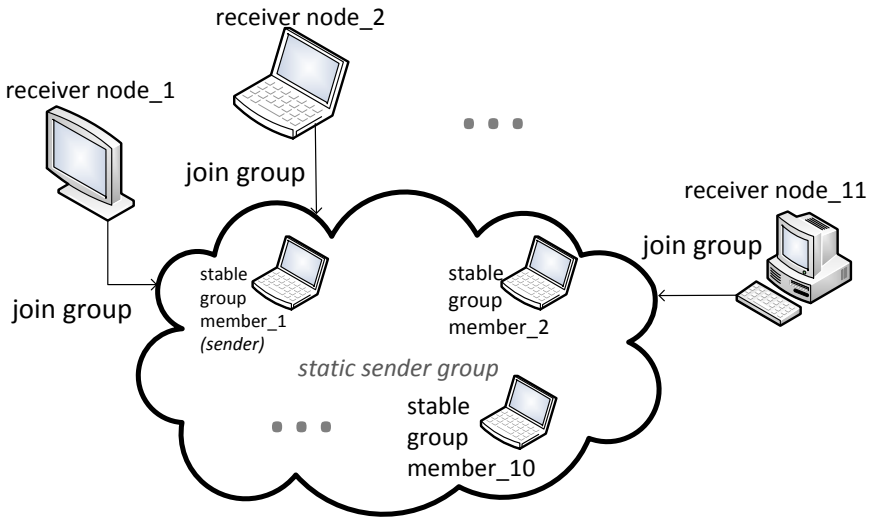
- Sender:
  - ausgeglichene CPU-Belastung
- Empfänger:
  - CPU-Belastung abhängig von der Paketanzahl (bei MCPO und H∀Mcast)
  - Zusätzliche Belastung durch Kopieroperationen der H∀Mcast-Middleware

Test: Join-Latency

- Konstanter Aufwand bei Join durch MCPO-Objekt
  - Objektinitialisierung (MCPO)
  - Binden an Ariba-Node
  - Beitreten des MCPO-Verteilbaums
  - Auf Daten warten

Komplexität folgt aus:

- Abbildung der Adressierung
- Abbildung der Join-Funktion



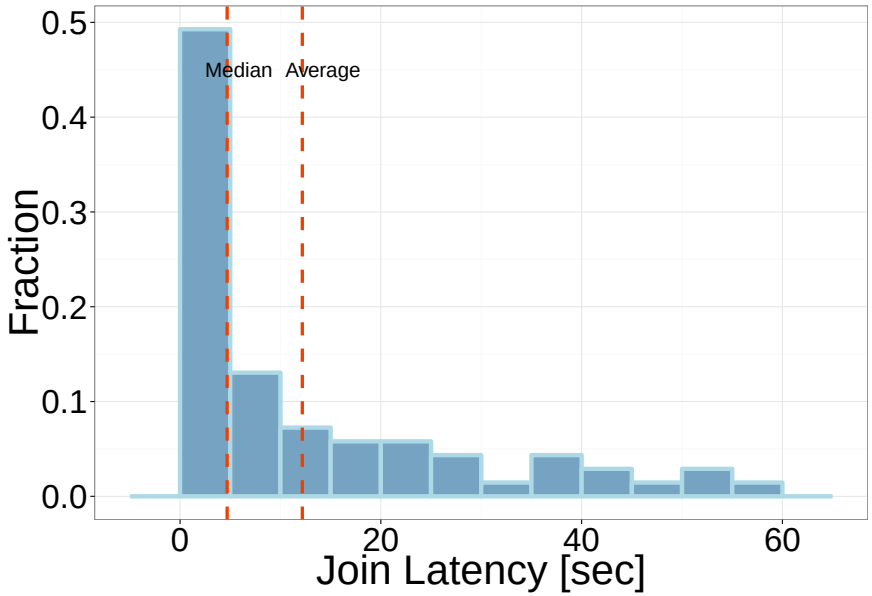
Aufbau:

- Statische Gruppe mit 10 MCPO-Knoten
  - → enthält 1 Sender
- 11 Knoten treten der Gruppe bei
  - → Messen der Zeit bis zur Ankunft der ersten Daten

Test: Join-Latency

Testergebnisse

- nicht alle Knoten sind der Gruppe erfolgreich beigetreten
  - Knoten liefern keine Messdaten
- Daten aus vergleichbaren Verteilbäumen
  - mind. 8 erfolgreiche *joins*



- 44% aller Knoten < 1 *sek*
- 45% aller Knoten > 10 *sek*

Test: Join-Latency

Einflussfaktoren

Node A:

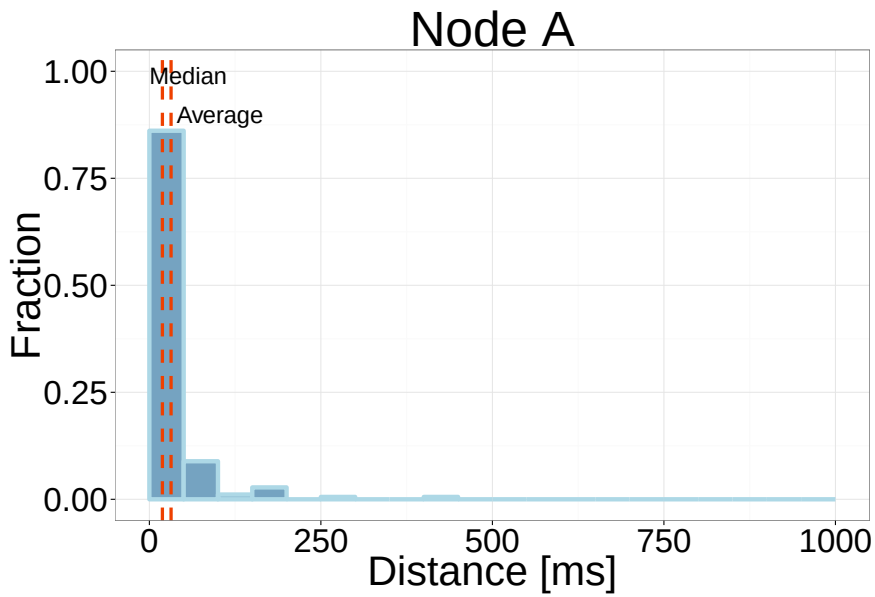
- Join Latency: 0.3 sek
- mittl. Distanz: 19 ms

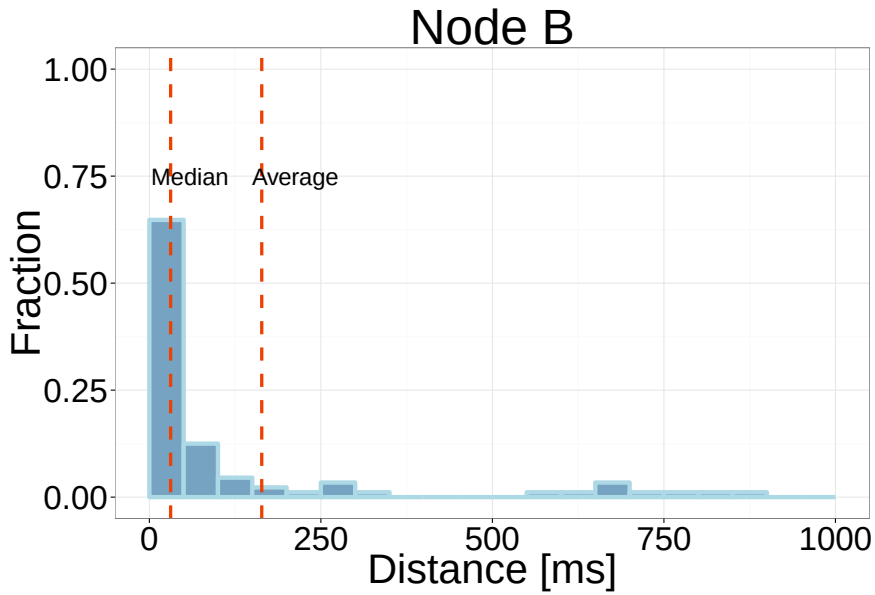
Node B:

- Join Latency: 55.3 sek
- mittl. Distanz: 163 ms

Timeout

- Ariba-Timeout zum Senden: 10 Sekunden
- MCPO-Timeout bei Eintritt in Verteilbaum: 15 Sek
- beide Timeouts statisch
- MCPO nutzt die Info nicht, ob Nachricht gesendet wurde





Zusammenfassung

- Einflussfaktoren auf Komplexität
  - Verschiedenartigkeit der Funktionen und deren Anforderungen
  - durchbrochene Netzwerkebenen
  - Abbildung der Parameter und Funktionen
  - widersprüchliche Konzepte der Ebenen
- Folgen der Komposition
  - nicht vollständig auflösbare Konflikte
  - sehr hoher Aufwand in einzelnen Funktionen, um Konzepte zu erhalten
  - Eigenschaften werden an suboptimalen Plätzen im Stack implementiert
  - komplizierte Anwendungsschnittstellen
  - schwierige Fehlersuche

Ausblick

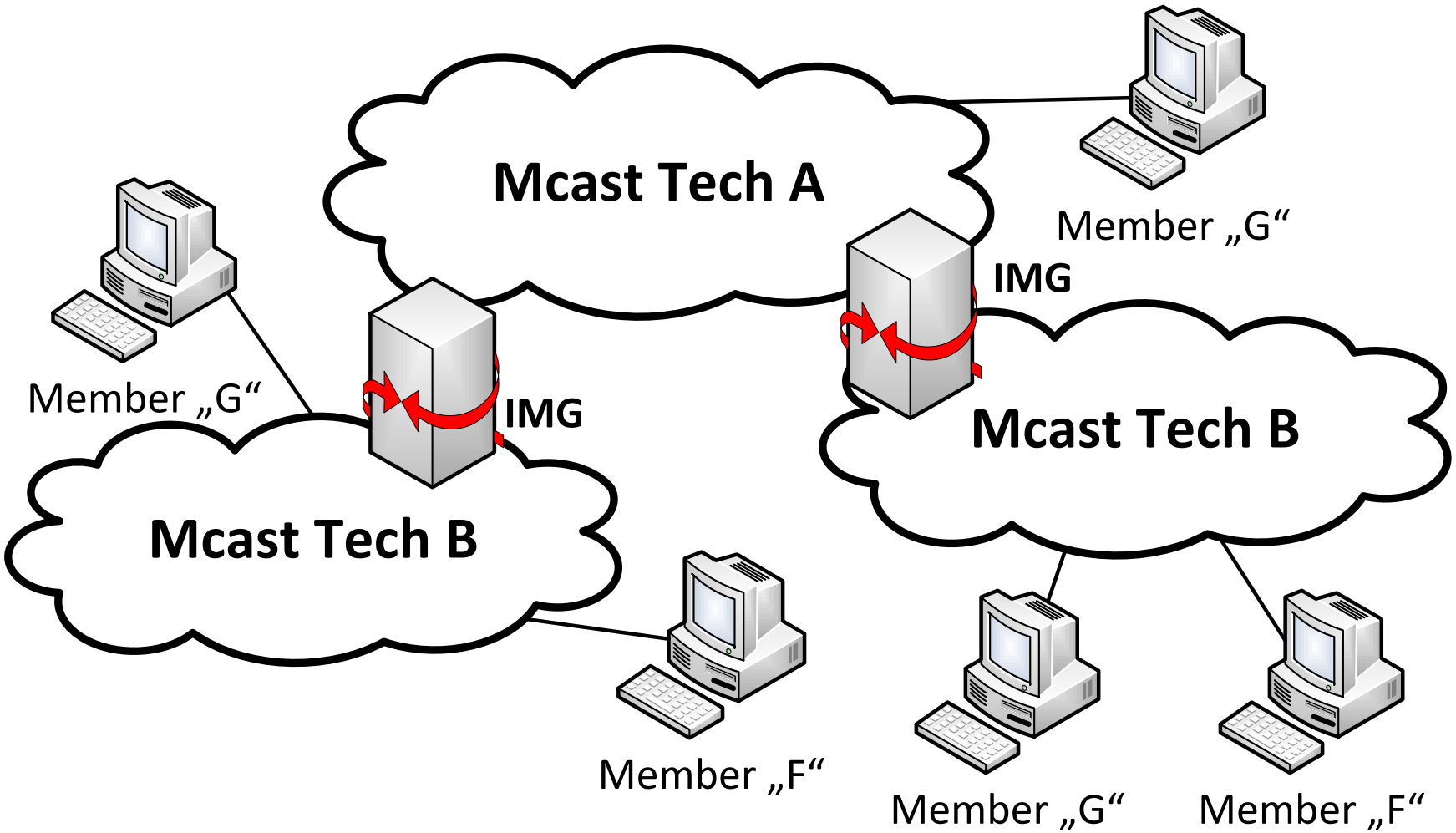
- Maß für Komplexität?
- Tests: Topologie der Overlays mit Einbeziehen
- Analyse, wie man Netzwerk-APIs so gestalten kann, dass sie möglichst wenig Komplexität erzeugen

Thank you for your attention. Questions?

iNET: `http://inet.cpt.haw-hamburg.de`

HAMcast

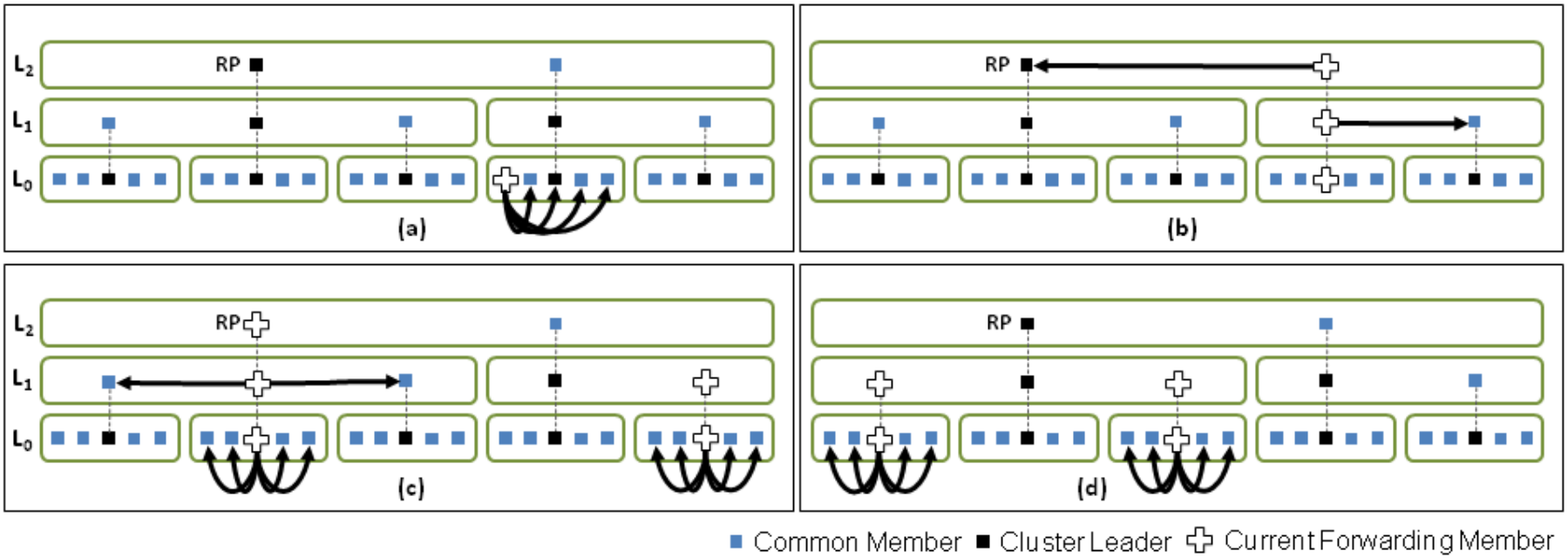
Architektur



MCPO

Verteilbaum

- Multicast-Overlay über Ariba
- Datenverteilung ähnlich des NICE-Protokolls
- Ersetzung der Ariba-Unicast-Schnittstelle durch eine Multicast-Schnittstelle



Netzwerk-Stack-Ebenen

Strukturierung und Durchlässigkeit

Ariba/MCPO-Anwendungsschnittstelle

- Die Ariba/MCPO-Schnittstelle initialisiert 3 Hauptkomponenten
- Ariba/SpoVNet ist kann für mehrere Anwendungen genutzt werden
- Widerspruch zwischen Transparenzprinzip und E2E-Prinzip

Ariba/MCPO-Adaptermodul

- SpoVNet als Netzwerkinterface
- Initialisierung der SpoVNet Komponenten bei Middleware-Start

H∀Mcast-Schnittstelle

- H∀Mcast bietet einfache Schnittstelle für MC-Anwendungen an
- Die Komplexität wird in die Module verlagert
- Technologiemodule müssen zusätzliche Informationen Anwendungsunabhängig herausfinden

Konflikt:

Wie kann man transparent Module verbergen, wenn externe Informationen benötigt werden?