Laptop
A location aware peer-to-peer overlay network

Chi-Jen Wu, De-Kai Liu and Ren-Hung Hwang

Eric Lacher

Seminar peer-to-peer Netzwerke
Prof. Dr. Christian Schindelhauer

Albert-Ludwigs Universität Freiburg - Technische Fakultät

29. Juli 2009

Überblick

- Was ist *Laptop*?
- Aufbau des Overlay Netzwerks
- Verwaltung des Netzwerks
- Routing
- Performanz
- Probleme
- Fazit

Overlay Netzwerk

Overlay Netzwerk

Ein Netzwerk das auf einem darunter liegenden Netzwerk (Underlay) aufbaut und dessen Infrastruktur nutzt.

Laptop

Nutzt das Internet als Underlay und versucht dessen Topologie nachzubilden (ortsbezogenheit) um optimale Wege zwischen Knoten herzustellen.

Ziele

Ziele von Laptop

- Ortsbezogenheit (location awareness)
- Selbstorganisation
- Fehlertoleranz
- Minimierung des Organisations-Overheads
- gute Skalierbarkeit

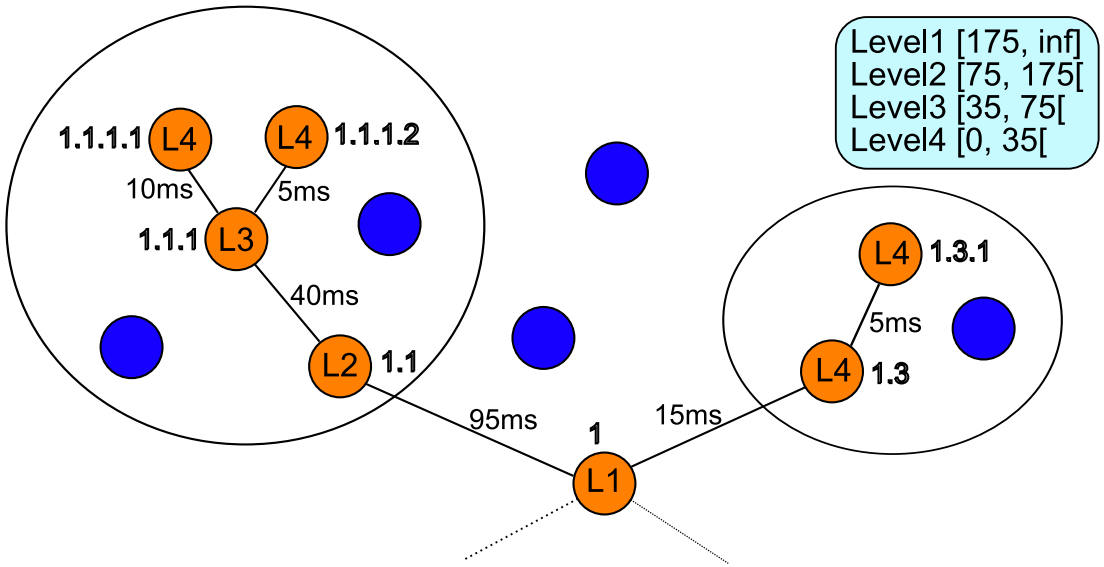
Überblick

- Was ist *Laptop*?
- Aufbau des Overlay Netzwerks
- Verwaltung des Netzwerks
- Routing
- Performanz
- Probleme
- Fazit

Struktur

Struktur des Netzwerks

- hierarchische Baumstruktur mit beschränktem Grad d
- Knotenadresse: Parent (1.2) + sub-ID (1) = 1.2.1
- Beschriftung mit Level: Abhängig von der gemessenen RTT zum Parent (Segmente)



Bildung des Netzwerks: Root-Node

Root-Node

Der Root-Node ist der erste Knoten im Netz.
Er erhält die Adresse “1“ und wird dem Level 1 zugewiesen.
Er wird daher die mit L1 bezeichnet.

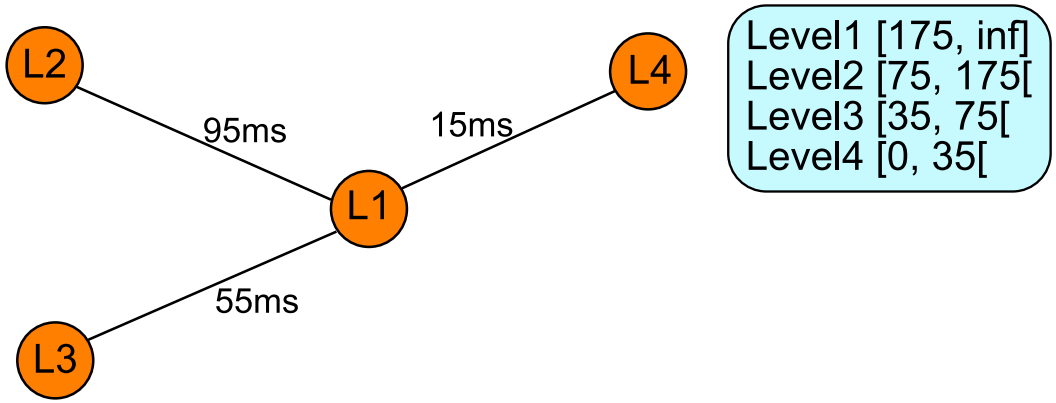
Bildung des Netzwerks: Child-Nodes

Children-Nodes

Jeder Knoten kann bis zu d Kind-Knoten aufnehmen. d ist somit der Maximal-Grad.

Level der Kinder

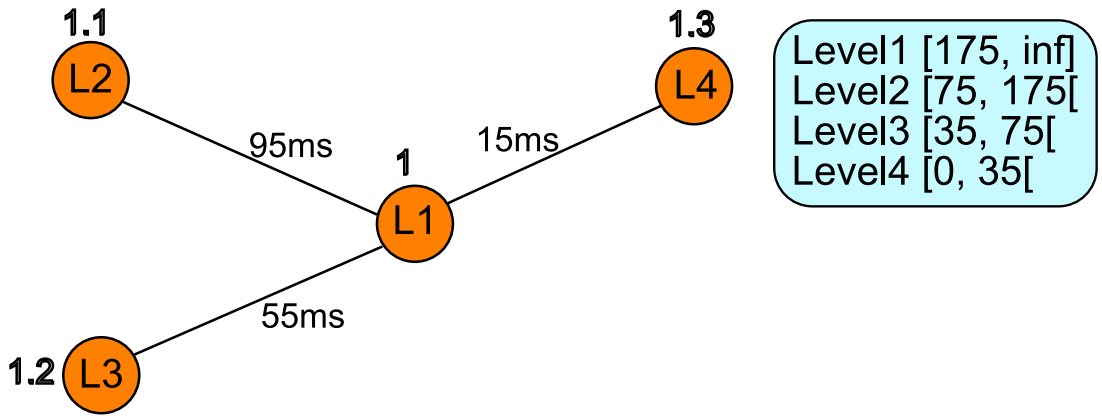
Der Level wird bestimmt durch die RTT zum Elternknoten. Die Einteilung nach Levels erfolg durch sog. Segmente.



Bildung des Netzwerks: Child-Nodes

Adresse der Kinder

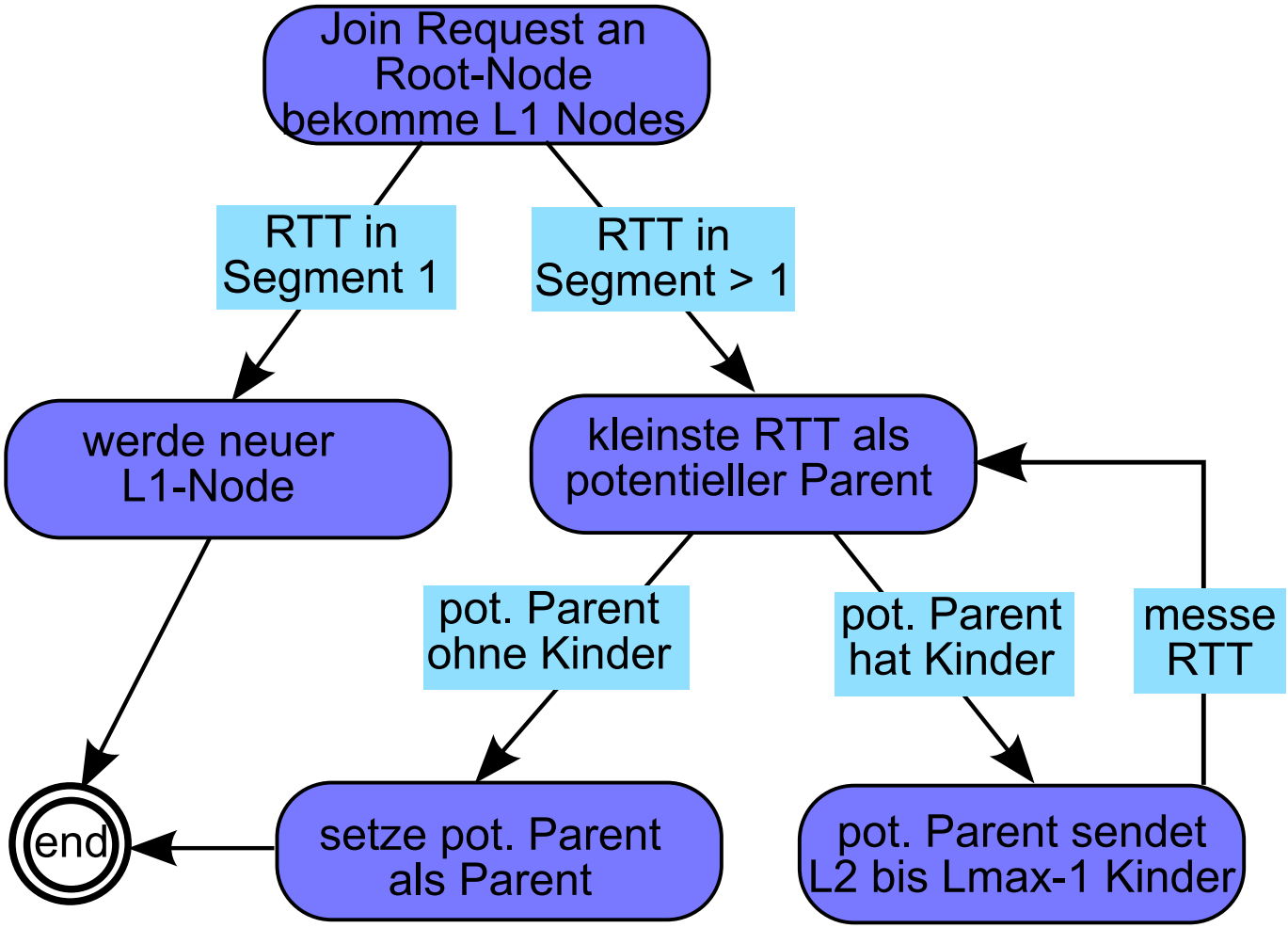
Die Adresse des Kind-Knotens leitet sich vom Elternknoten ab, indem dessen Adresse als Prefix und eine fortlaufende Nummerierung als Suffix dient.



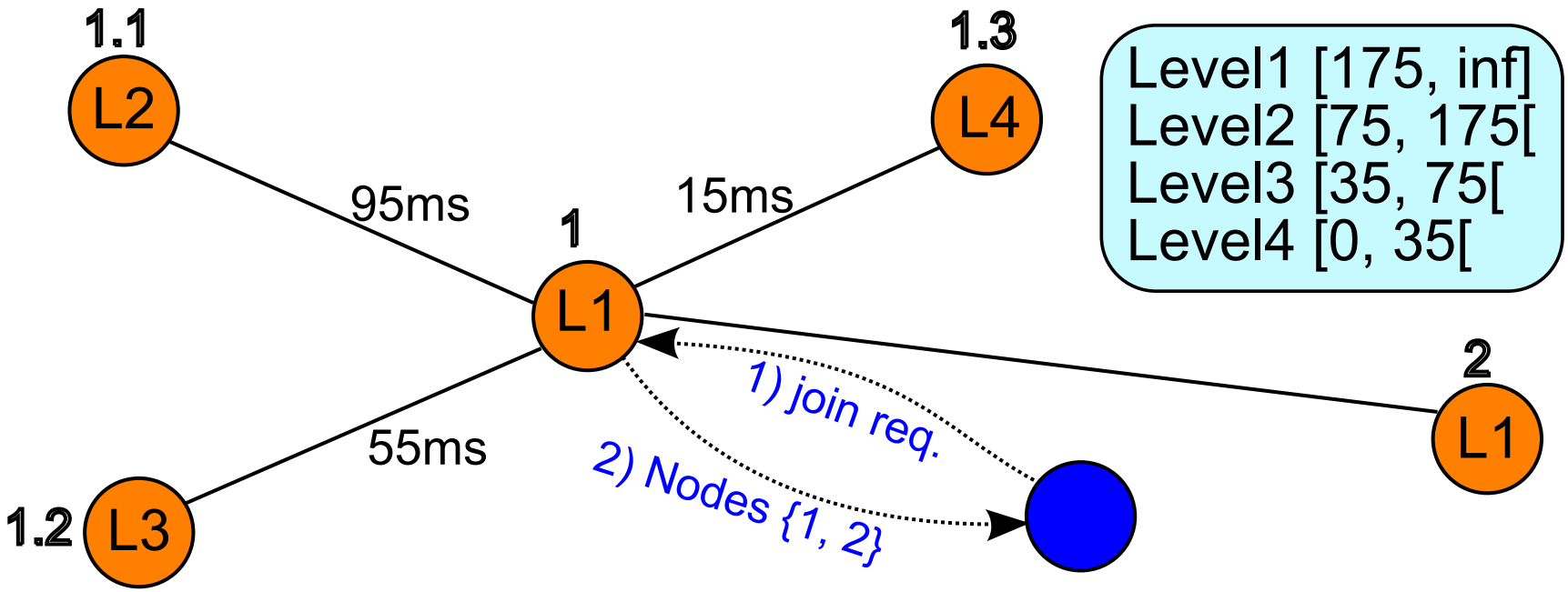
Überblick

- Was ist *Laptop*?
- Aufbau des Overlay Netzwerks
- Verwaltung des Netzwerks
- Routing
- Performanz
- Probleme
- Fazit

Verwaltung des Netzwerks: *Join* eines Knotens (Übersicht)

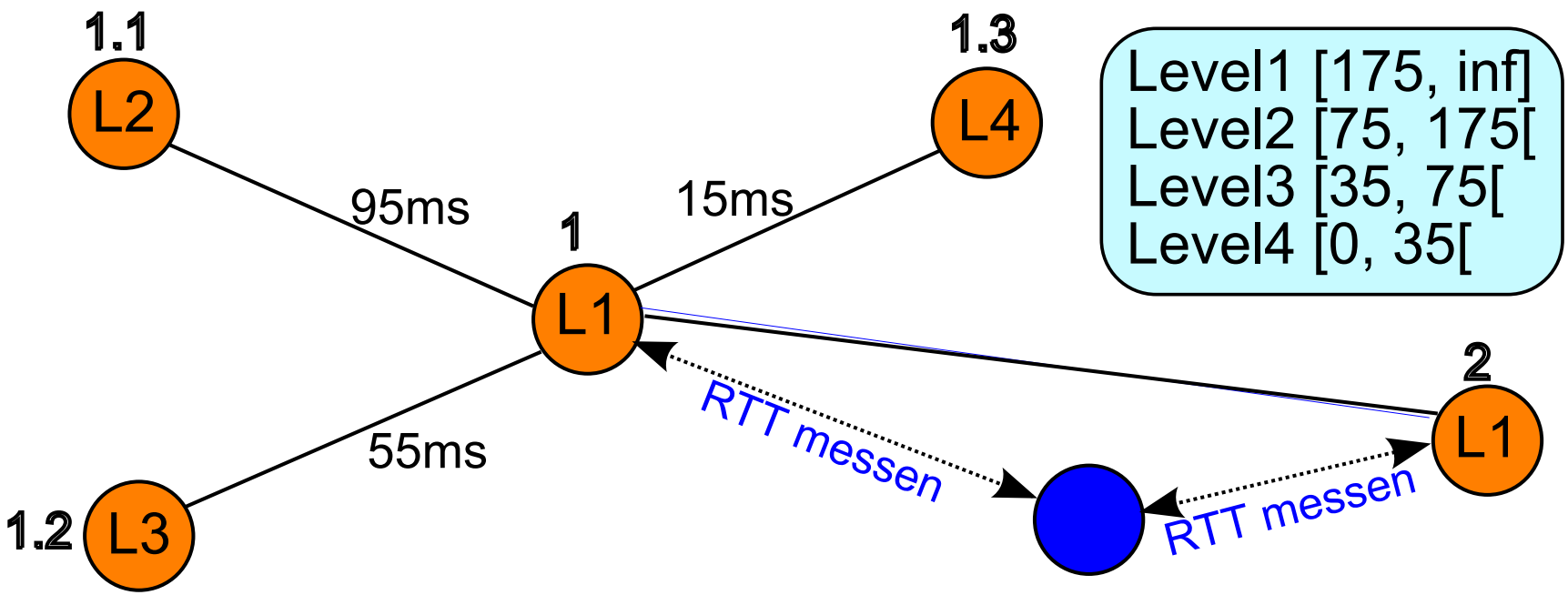


Verwaltung des Netzwerks: *Join* eines Knotens(1)



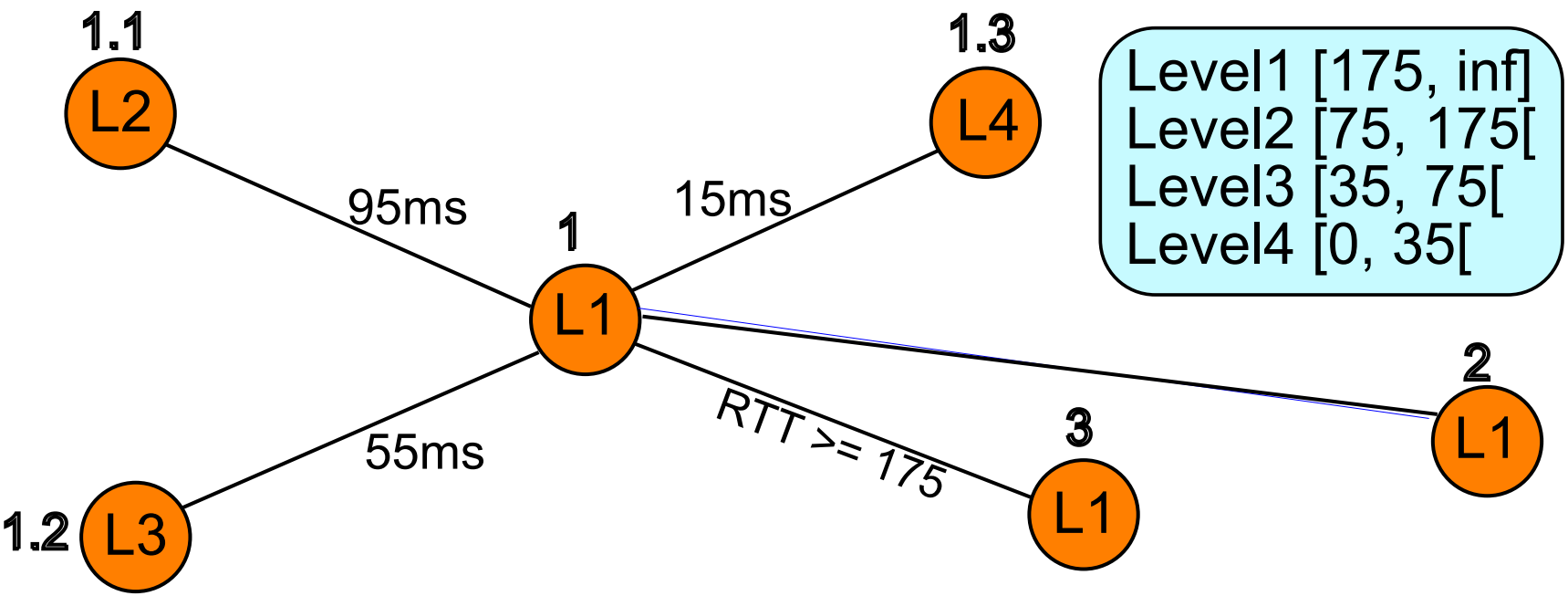
Der Kandidat sendet ein Join-Request an den Root-Node und bekommt eine Liste aller L1 Knoten zurück

Verwaltung des Netzwerks: *Join* eines Knotens(2)



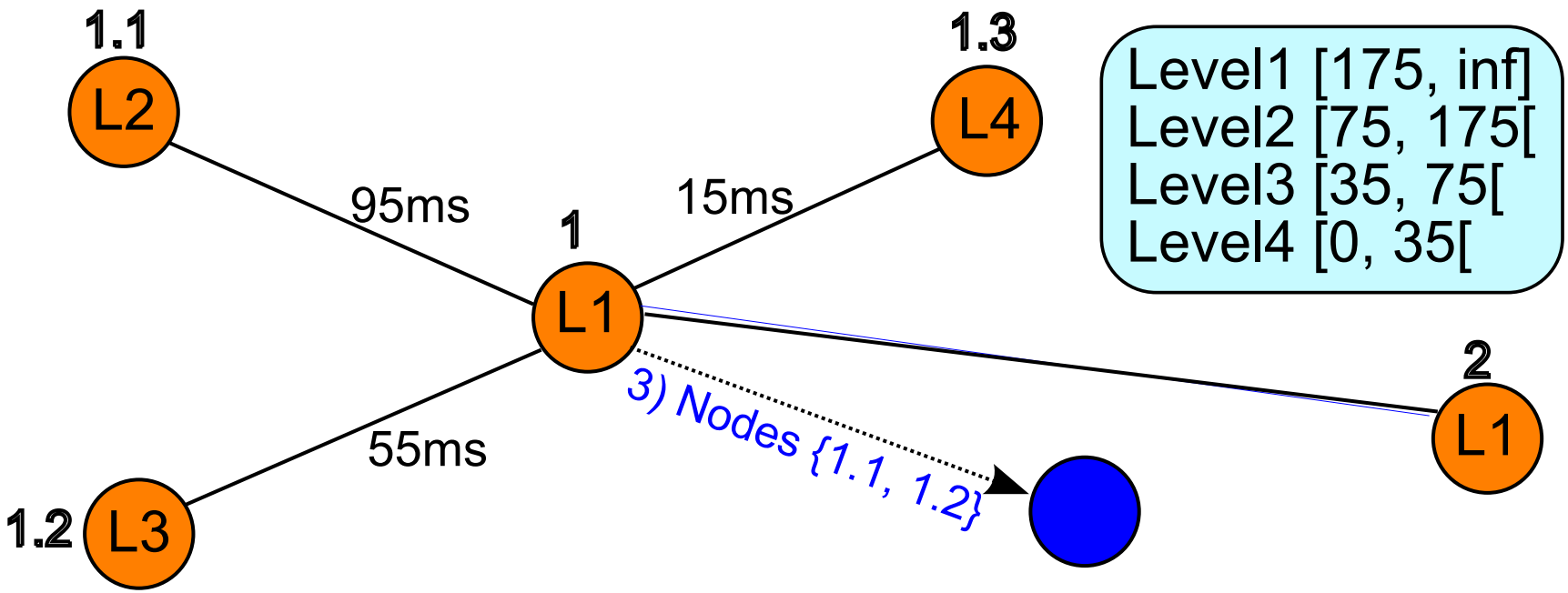
Der Kandidat misst die RTT zu den zurückgegebenen Knoten und wählt den Nähesten als potentiellen Parent

Verwaltung des Netzwerks: *Join* eines Knotens(3)



Falls die kürzeste RTT $\geq L_{max}$, so wird der Kandidat zu neuem L1.

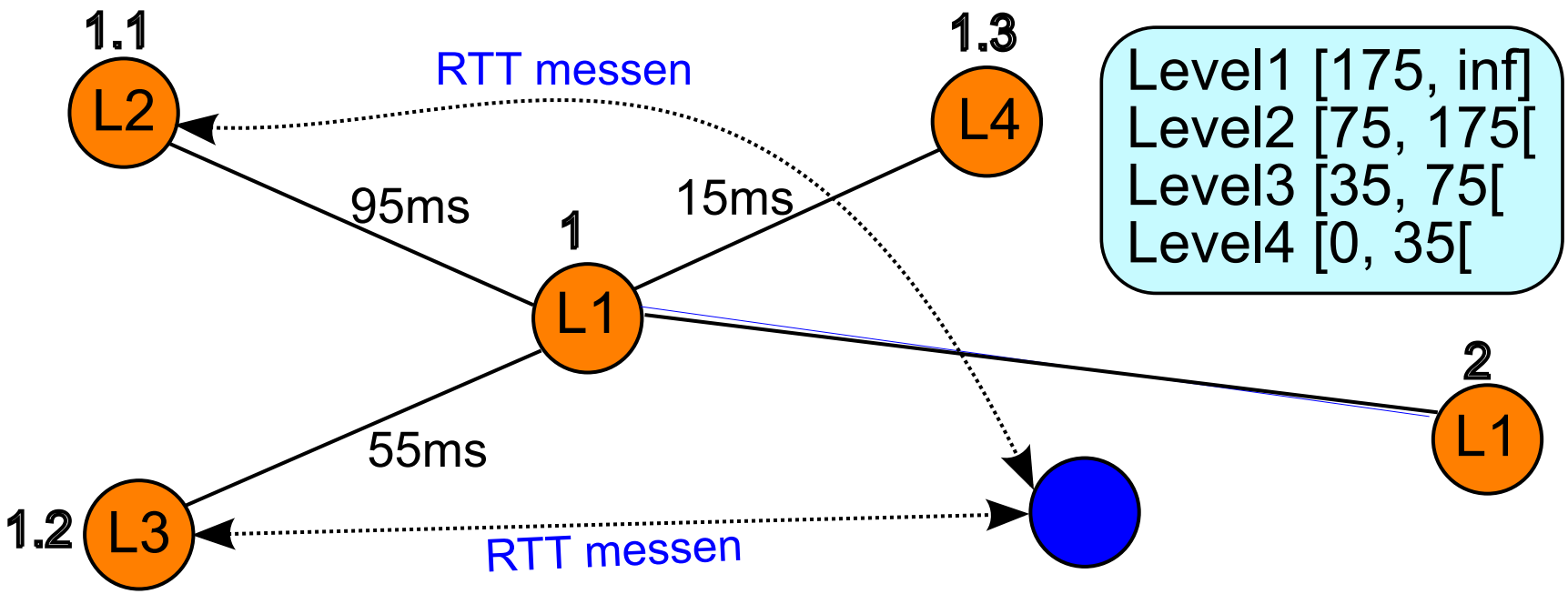
Verwaltung des Netzwerks: *Join* eines Knotens(4)



Ansonsten gibt Knoten 1 alle Knoten aus für deren Level i gilt:

$$2 \leq i < L_{max}$$

Verwaltung des Netzwerks: *Join* eines Knotens(5)



Messen der RTT zu Knoten 1.1 (L2) und 1.2 (L3). Setze Knoten mit kleinster RTT als neuen pot. Parent und Verfahre erneut ab Join(2)

Verwaltung des Netzwerks: Control Overhead beim *Join*

Overhead ist $O(d \log_d(N))$ mit Anzahl Nodes N und maxDegree d

Der Grad des Baumes ist begrenzt durch den Maximalgrad d
$\rightarrow$ max. d RTT Messungen pro Knoten
Die Höhe des Baumes ist begrenzt durch $\log_d(N)$
$\rightarrow$ max. $\log_d(N)$ Knotentests

$\rightarrow$ $d \log_d(N)$ **Messungen sind maximal nötig.**

Verwaltung des Netzwerks: Wegfall eines Knotens

Abmelden vs. Timeout

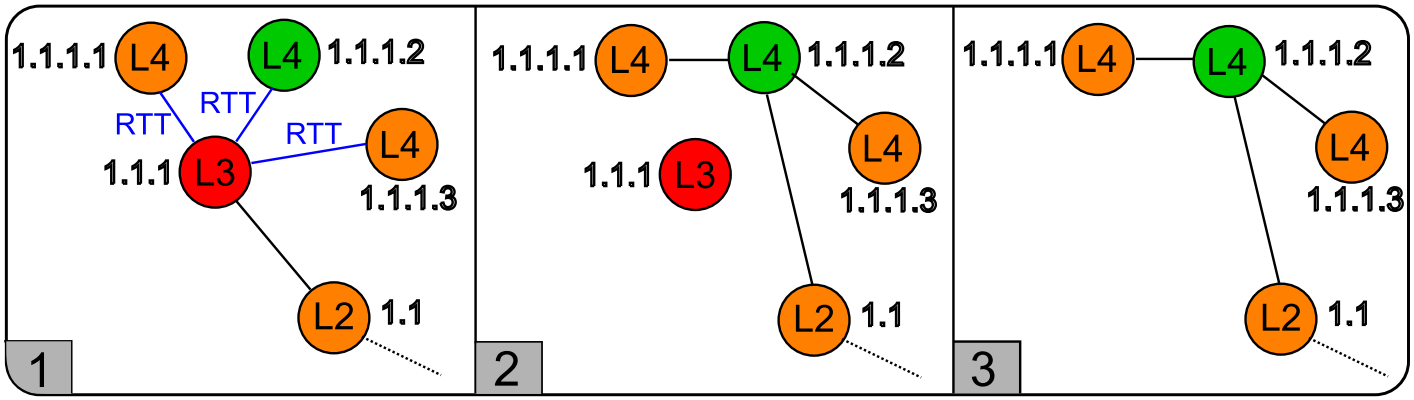
Zwei Möglichkeiten für einen Wegfall:

1. **Abmelden**: Der Knoten meldet sich vom Netzwerk ab.
2. **Timeout**: Der Knoten sendet keinen *HEARTBEAT* mehr zum Parent und wird nach Timeout entfernt.

Abmelden

Abmeldeprozedur: Knoten K meldet sich bei Parent P ab

- K ohne Kinder $\rightarrow$ Abmeldung an P. P löscht K aus Routing Tabelle.
- K mit Kindern $(c_1, ..., c_d)$: c_{min} mit kleinster RTT übernimmt.
- Andere Kinder setzen Parent auf c_{min}, c_{min} setzt Parent auf P.



K (1.1.1) meldet sich ab, c_{min} (1.1.1.2) hat kleinste RTT und übernimmt

Verwaltung des Netzwerks: Control Overhead beim Abmelden eines Knotens

Overhead (bezügl. Knotenkommunikation) ist $O(d)$ mit maxDegree d

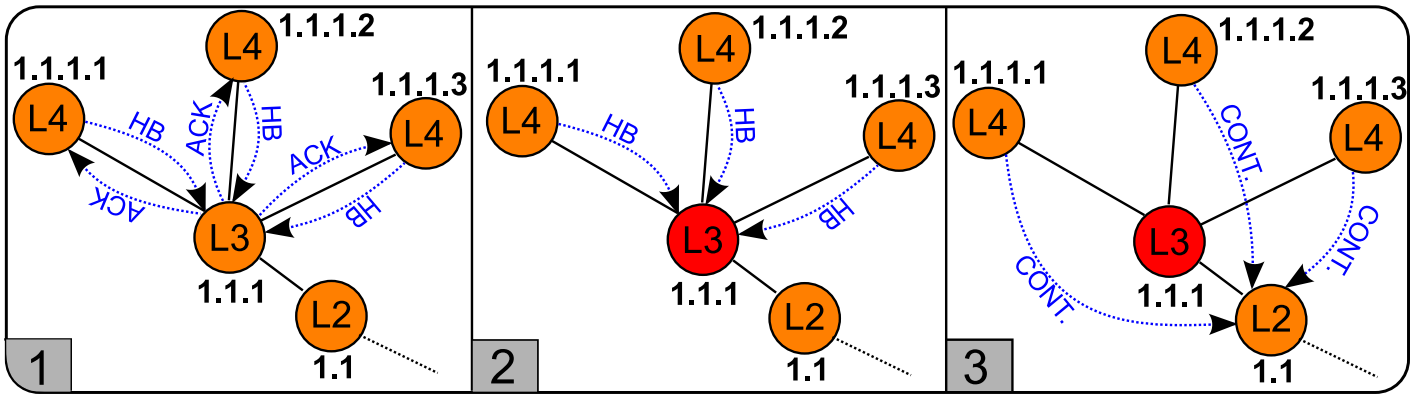
Wenn Knoten ein Blatt war muss nur Parent informiert werden.
$\rightarrow$ Aufwand $O(1)$
Wenn Knoten kein Blatt war müssen maximal d Kinder und 1 Parent informiert werden
$\rightarrow$ Aufwand $O(d)$

Es sind maximal $d+1$ Knoten in Kommunikation involviert
$\rightarrow O(d)$

Timeout(1)

Abmeldeprozedur: Knoten K fällt ohne Abmeldung aus

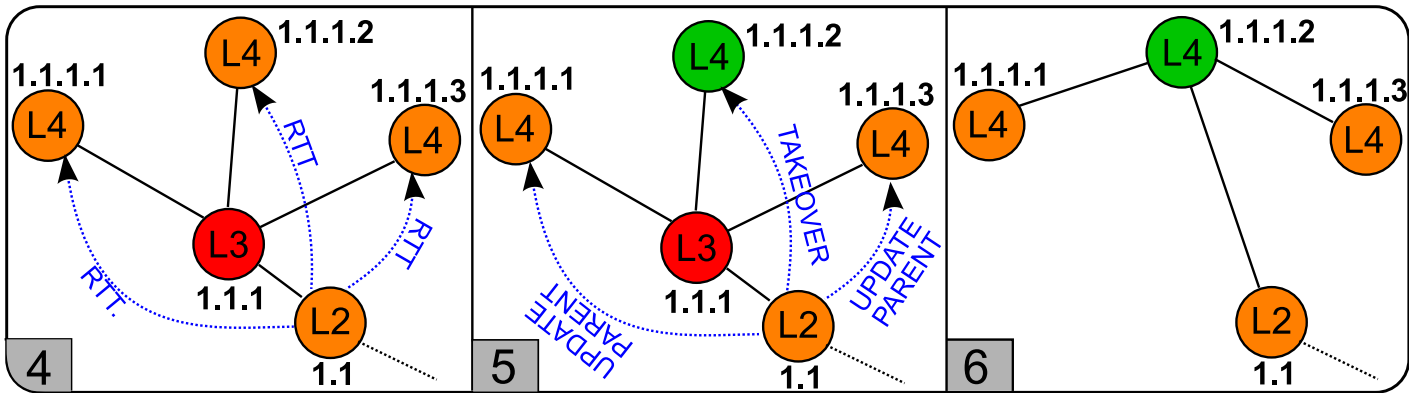
1. Ausfall wird ermittelt durch fehlende ACKs auf HEARTBEAT anfragen.
2. Kinder senden CONTENTION Nachrichten an Parent PP (1.1) von P (1.1.1).
3. Wenn PP auch nicht erreichbar ist $\rightarrow$ JOIN-Prozedur



Timeout(2)

Abmeldeprozedur: Knoten K fällt ohne Abmeldung aus

4. PP misst RTT zu verwaisten Kindern und ermittelt so c_{min}, den Waisen mit der kleinsten RTT.
5. PP sendet TAKEOVER an c_{min} und UPDATE_PARENT an die anderen Waisen.
6. c_{min} setzt Parent auf PP, Waisen setzen Parent auf c_{min}.



Verwaltung des Netzwerks: Control Overhead beim Wegfall durch Timeout

Overhead ist $O(d \log_d(N))$ mit Anzahl Nodes N und maxDegree d

Wenn PP erreichbar: max. d Kinder und PP sind in Kommunikation involviert
$\rightarrow$ Aufwand in $O(d)$
Wenn PP nicht erreichbar: JOIN-Prozedur (siehe vorher)
$\rightarrow$ Aufwand in $O(d \log_d(N))$

$\rightarrow$ **Aufwand in** $O(d \log_d(N))$

Überblick

- Was ist *Laptop*?
- Aufbau des Overlay Netzwerks
- Verwaltung des Netzwerks
- Routing
- Performanz
- Probleme
- Fazit

Routing Tabelle

Routing

- Routing Tabelle
- Caching
- Weitergabe an Knoten mit längstem übereinstimmenden Prefix

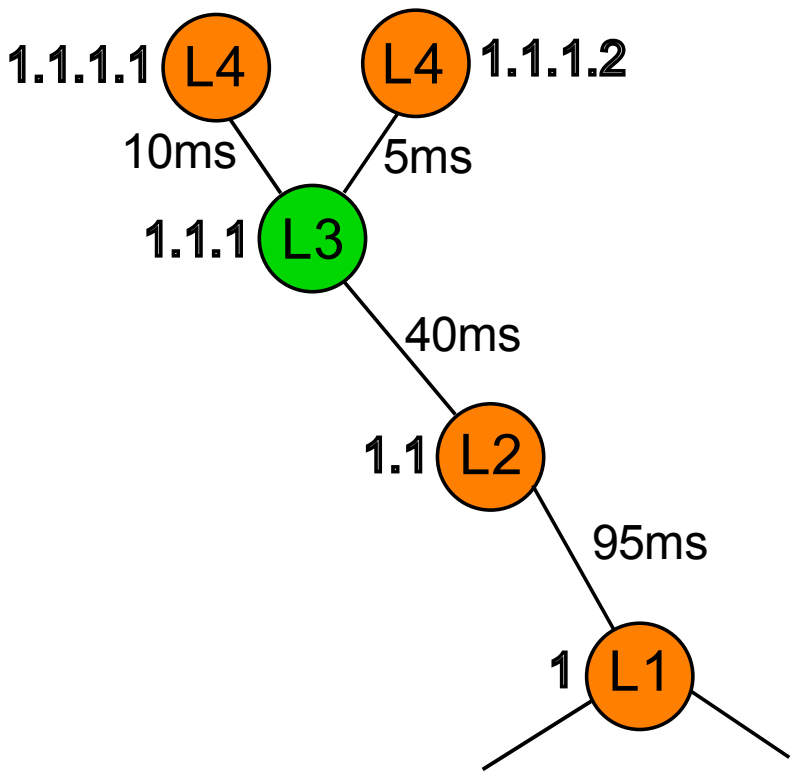
Routing Tabelle

1.1.1 (Level 3)

Root & Parent		
Root	1	140.123.105.14
Level 2	1.1	110.1.105.13

children		
Level 4	1.1.1.1	110.12.10.13
Level 4	1.1.1.2	110.12.14.13

Cache		
Level 3	1.2	18.1.105.13
Level 4	1.3	120.140.190.1
...	...	...



Aufwand beim Routing

Routing-Pfad begrenzt durch $O(\log_d N)$

worst-case: zuerst zum Root-Node, dann aufwärts bis zum Blatt
$\longrightarrow 1 + \log_d N \longrightarrow O(\log_d N)$

Überblick

- Was ist *Laptop*?
- Aufbau des Overlay Netzwerks
- Verwaltung des Netzwerks
- Routing
- Performanz
- Probleme
- Fazit

Testumgebung

BRITE topology generator

- ingesamt 1.000.000 Nodes
- ...verteilt auf 50 AS
- jedes AS hat 200 LANs
- Cache-Grösse: 64

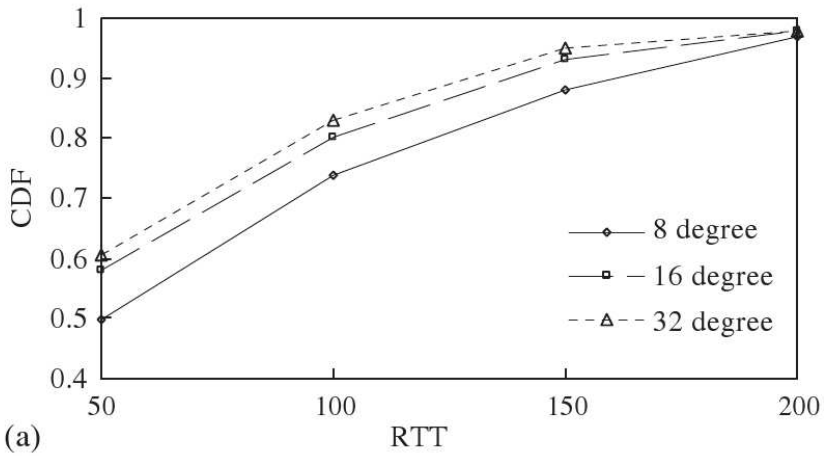
4 durchgeführte Tests

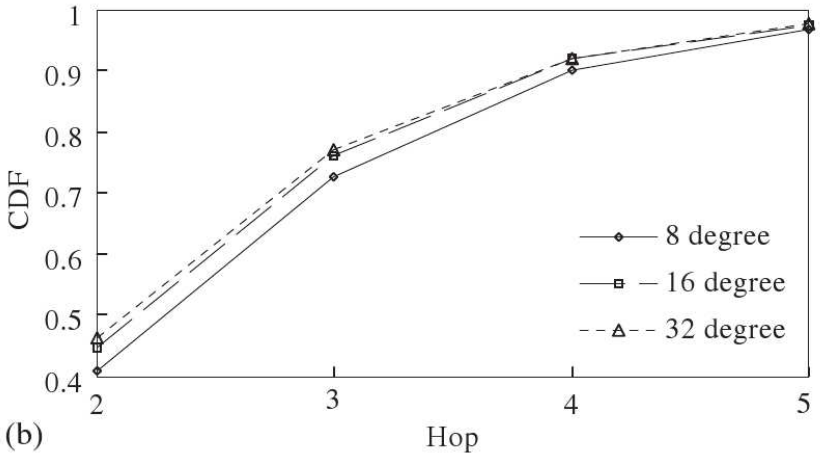
1. „clustering effect“ (RTT + hop count)
2. Routing
3. Ortsbezogenheit („routing stretch“)
4. Fehlerkorrektur

1. „clustering effect“

Ergebnis

80% der Nodes haben eine RTT $\leq 100ms$ zu ihrem Parent
80% der Nodes haben einen Hop-Count ≤ 3 zu ihrem Parent (Underlay-Hops)



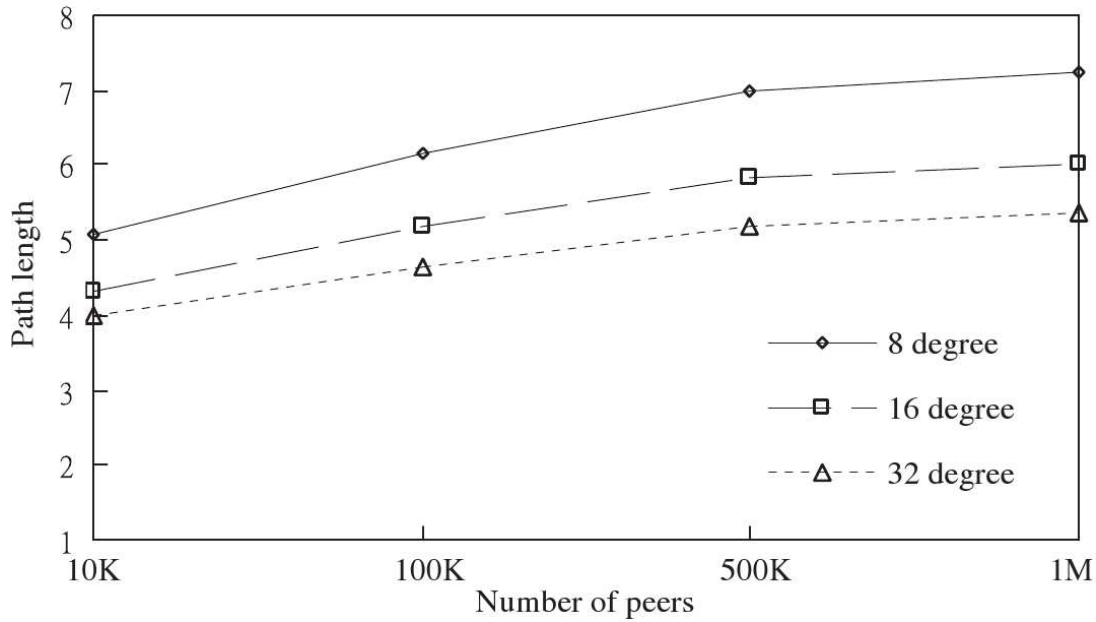


Quelle: Paper[WLH07]

2. Routing

Ergebnis

durchschn. Pfadlänge ist ca. 5 bei $d = 32$ und 1M Nodes.
Theorie: $log_{32}1M = 4$

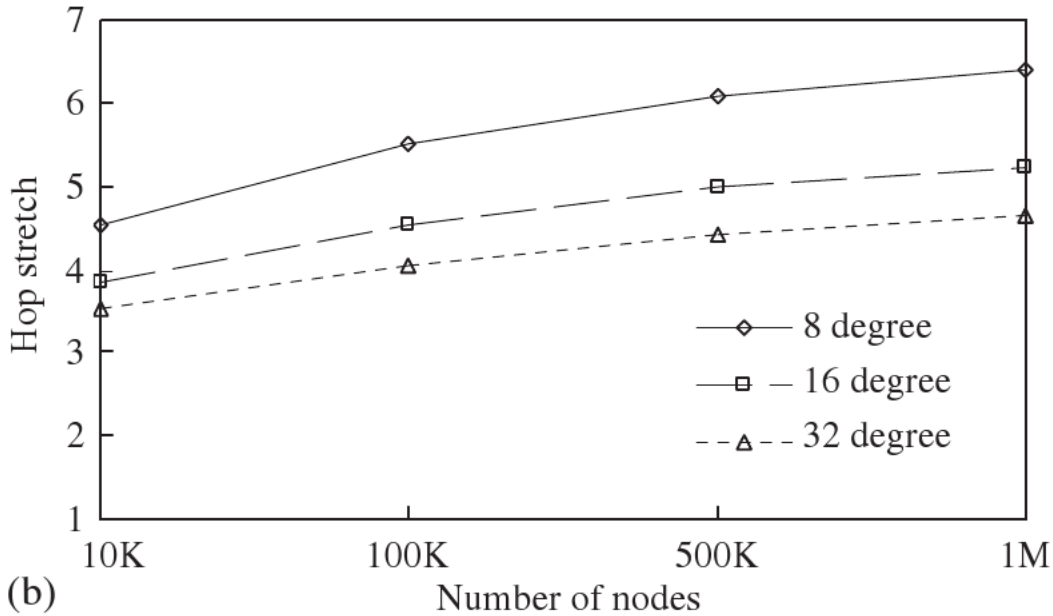


Quelle: Paper[WLH07]

3. Ortsbezogenheit („routing stretch“)

Ergebnis

Hop-Count-Stretch: (Hops in IP) / (Hops in Laptop)
Für 1M Nodes und $d = 32$ ist der Hop-Count-Stretch bei 4.6

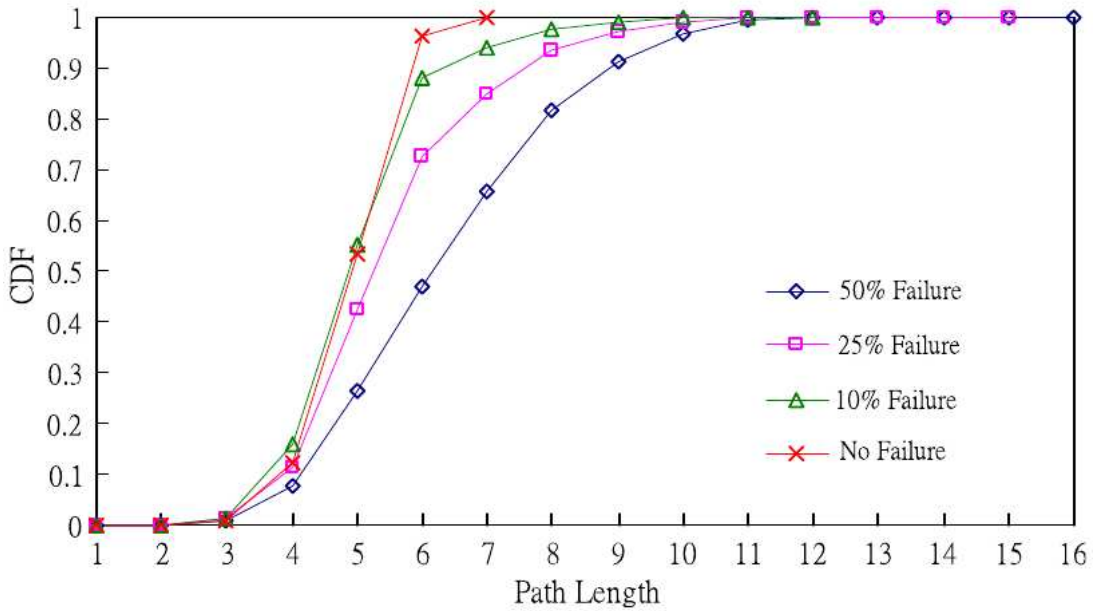


Quelle: Paper[WLH07]

4. Fehlerkorrektur

Ergebnis

bei 50% ausgefallenen Knoten sind 96% der Pfade kürzer als 10



Quelle: Paper[WLH07]

Überblick

- Was ist *Laptop*?
- Aufbau des Overlay Netzwerks
- Verwaltung des Netzwerks
- Routing
- Performanz
- Probleme
- Fazit

Probleme beim Aufbau und der Verwaltung

1. Adressen bei wegfallenden Nodes
2. Ausfall des Root-Node?

Probleme des Designs

1. RTT konstant? Starre Topologie.
2. Root-Node extrem wichtig

Überblick

- Was ist *Laptop*?
- Aufbau des Overlay Netzwerks
- Verwaltung des Netzwerks
- Routing
- Performanz
- Probleme
- Fazit

Fazit

Ziele erreicht?

- **Ortsbezogenheit:** ja, aber: unflexibel bei Underlay-Änderungen
- **Selbstorganisation:** ja
- **Fehlertoleranz:** gut, aber: zentraler Root-Node
- **Minimierung des Organisations-Overheads:** ja, logarithmisch zum Grad, (haupts. Parent-Child Kommunikation)
- **gute Skalierbarkeit:** ja, siehe Kapitel Performanz.

Vielen Dank für die Aufmerksamkeit

Literatur


Chi-Jen Wu, De-Kai Liu, and Ren-Hung Hwang.
A location-aware peer-to-peer overlay network: Research articles.
Int. J. Commun. Syst., 20(1):83–102, 2007.