

Energieeffizientes Routing in Ad-Hoc Netzen

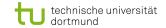
Eine simulationsbasierte Analyse von AODV und OLSR

Marcel Ebbrecht

TU Dortmund, Bachelorabschlussvortrag

20. Mai 2018

Übersicht



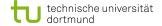
- Vorarbeit
- 2 Routing
- Wo ist das Problem
- Omnet
- Simulation
- 6 Abschluss

Whoami



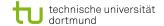
- Marcel Ebbrecht
- Viele Jahre LAN, WAN, WLAN, TCP/IP
- n Jahre Informatik
- marcel.ebbrecht@googlemail.com
- https://github.com/marcelebbrecht/powerrouting

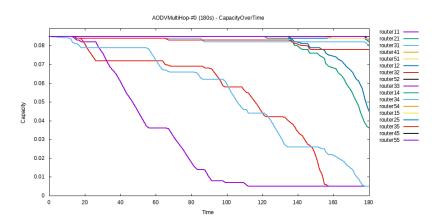
Warum sind wir hier?



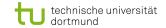
- Was ist das Problem und warum ist das relevant?
- OLSR und AODV im Wesentlichen
- Optimierung nach Ladezustand
- Nach dem Vortrag gerne mehr Videos und Auswertungen

Warum sind wir hier?



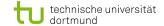


Übersicht



- Vorarbeit
- 2 Routing
- Wo ist das Problem
- Omnet
- Simulation
- 6 Abschluss

Allgemein



- TCP/IP Layer 3
- IPv4 (IPv6)
- Statisches Routing
- Dynamisches Routing (reaktiv oder proaktiv)

Ad-hoc On-demand Distance Vector technische universität dortmund

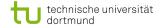
- Reaktiv
- Geringer Overhead / Hohe Latenz
- Loop avoidance
- Precursors-Listen
- HopCount maßgebend

AODV Ablauf



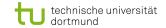
- Route benötigt -> RREQ
- Route bekannt -> RREP
- Route nicht mehr verfügbar -> RERR
- Sammelt Informationen aus empfangenen Nachrichten
- Entfernt Routen durch Timeout oder RERR

Optimized Link State Routing



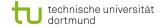
- Proaktiv
- OLSR Paket mit Messages
- MID, HELLO, TC
- 1HNB, 2HNB, MPRSet ("optimized")
- Willingness

Übersicht



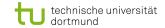
- Wo ist das Problem

Warum wichtig



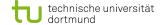
- Steigender Bedarf nach Bandbreite und Verfügbarkeit
- Viele mobile Geräte
- ABER: Routing braucht Strom
- Akkuladung begrenzt
- Routingverfahren berücksichtigen es nicht

Übersicht



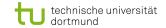
- Omnet

Ganz kurz: Omnet



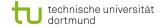
- Freie Software, C++, nette Community
- Ereignisbasiert, Modular
- Framework INET kann alles, wirklich alles, also viel
- Inetmanet gibt MANET (AODV okay/buggy, OLSR korrekt)
- Unbrauchbar: Auswertung -> PERL/Gnuplot (R Versionskonflikte)

Übersicht



- Simulation

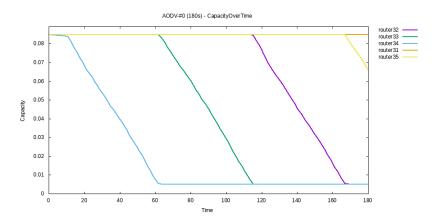
AODV Vanilla



Video AODV INIT und FAST

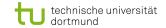
AODV Vanilla



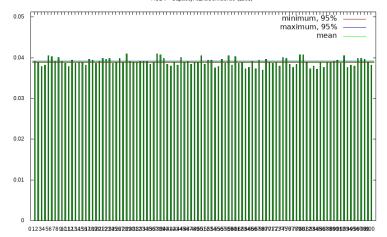


AODV Vanilla

Stddev

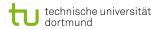






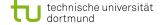
Run

Da geht doch was ...



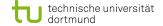


Ziele



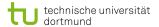
- Lastverteilung nach relativem Akkustand
- Kompatibilität zu anderen Systemen
- Steuerung der Empfindlichkeit und Grad der Anpassung

Ideen



- Es muss ein vorhandener Parameter eingesetzt werden
- AODV: HopCount künstlich anheben (begrenzt möglich)
- OLSR: Willingness ändern

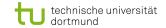
Mit Mathe muss man rechnen...







- Drei Parameter
 - Trigger $t \in [0.1, 0.9]$
 - ullet Sensitivity $s\in\mathbb{R}^+$
 - Bias $B \in \mathbb{N}_0$
- Ein Ziel
 - HopCount $H_{neu} = H_{alt} + 1 \rightarrow H'_{neu} = H_{alt} + 1 + P$
 - wobei Penalty $P = \lceil \frac{s}{C(i)} + B \rceil$



- Trigger t = 0.2 bedeutet, dass bei $C(i) \in \{0.8, 0.6, ..., 0.2\}$ eine Anpassung stattfinden soll
- Allgemeiner: Es wird eine Anpassung ausgelöst, wenn gilt: $(C(i)*100) \mod (100t) = 0$
- $C(i) \in]0,1]$ entspricht dem relativen Ladezustand
- Anpassung meint das Versenden von RERR, dadurch Ausfall und Neuberechnung



- Penalty $P(s, C(i), B) = \lceil \frac{s}{C(i)} + B \rceil$ ist der Wert, der auf den neuen Hopcount H'_{neu} aufgeschlagen wird
- $P(2,0.6,0) = \lceil \frac{2}{0.6} + 0 \rceil = 4$
- $H'_{neu} = H_{alt} + 1 + 4 > H_{neu} = H_{alt} + 1$
- Nulldivision unmöglich (Akku leer!)

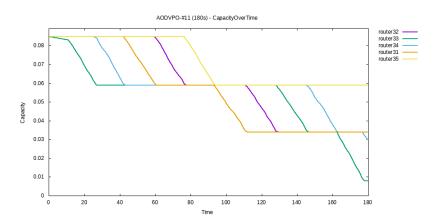


- Implementierung relativ einfach
- Funktioniert nicht bzw. zufällig
- Diagnose: AODV fehlerhaft
- ullet Therapie: Wartezeit W(P) bevor RREP erzeugt wird

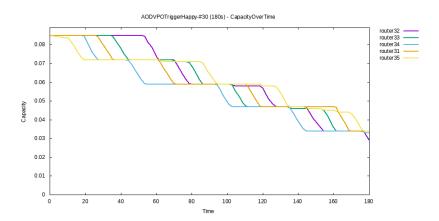


Video AODVPO FAST, RENEG

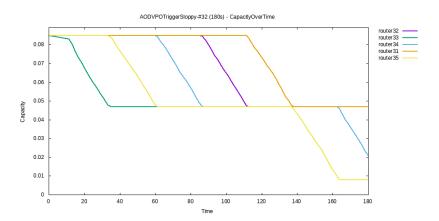




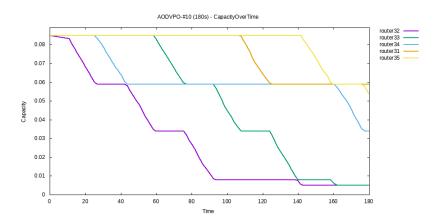




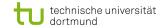




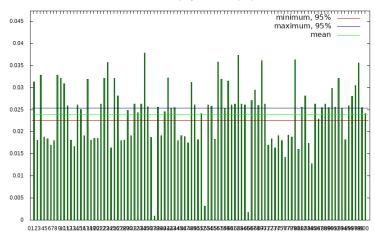




Stddev



AODVPO - CapacityAtEndConfidence (180s)

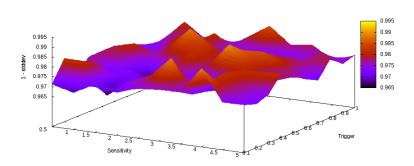


Run

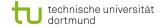
Studies



AODVPOParameterStudy (180s, Interpolation: 10, 5 repetitions) - CapacityOverTime

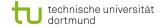


Studies

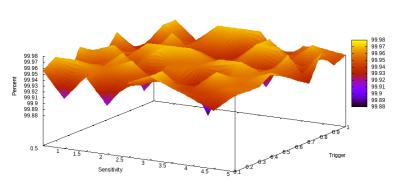


0 ... ist auch ein Prozentsatz

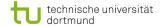
Studies







Performance

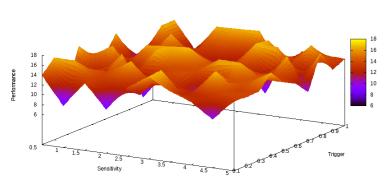


- Performance F = (1 (L + c))/(M + c)
- Standardabweichung der Ladestände $M = \sqrt{\sum_{i=1}^n \frac{(C(i) \bar{\mathbf{x}})^2}{n}}$
 - Sei n die Anzahl der Router,
 - C(i) die Restladung des jeweiligen Routers i und
 - $\bar{x} = \sum_{i=1}^{n} \frac{C(i)}{n}$ das arithmetische Mittel der Restladungen
- L ist der durchschnittliche PaketLoss aller Übertragungen
- $c = 10^{-10}$ (kontant)

Performance

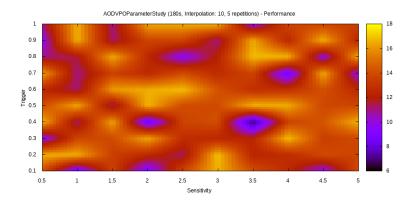






Performance



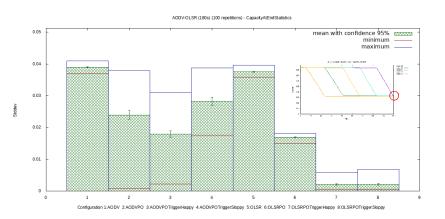


OLSR

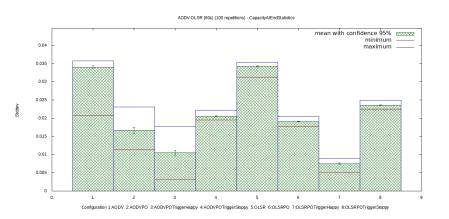


- Prinzipiell stabiler
- Statt HopCount-Penalty neue Willingness $W_{neu} = \lfloor \max(1, (7 \cdot C(i) \cdot (1-s) B)) \rfloor$ berechnet
- Fließender Übergang
- Keine Hacks

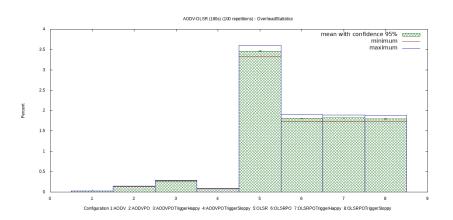




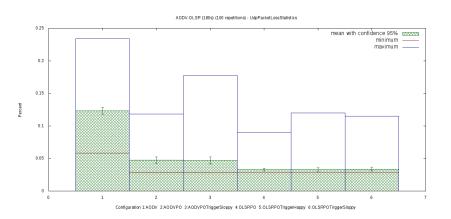










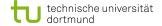


Übersicht



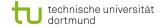
- Vorarbeit
- 2 Routing
- Wo ist das Problem
- Omnet
- Simulation
- 6 Abschluss

Fazit



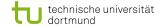
- Funktioniert in den gezeigten Szenarien
- Funktioniert in gemischten Netzen (Kompatibel)
- Funktioniert in größeren Netzen (Skaliert)

Das wars! Fragen?





Möchten Sie mehr wissen?



- RFC 3626 OLSR
- RFC 3561 AODV, Draft 01 IPv6
- Nitin Kumar, Parvez Ahmed Alvi, Ajay Singh Yadav, and Anupam Swami. A study of routing protocols for ad-hoc network. International Journal of Application or Innovation in Engineering and Management, 2(6):154–159, Juni 2013.
- Stefano Basagni, Marco Conti, Silvia Giordano, and Ivan Stojmenovic. IEEE 802.11 AD HOC Networks: Protocols, Performance, and Open Issues, page 416ff. WileyIEEE Press, Piscataway, NJ, USA, 2004.
- DIY: https://github.com/marcelebbrecht/powerrouting