Humboldt Universität Informatik

Humboldt University

Computer Science Department
Systems Architecture Group
http://sar.informatik.hu-berlin.de

DTN-Routing-Verfahren

Florian Holzhauer, Ralf Cremerius
(fh@fholzhauer.de, cremeriu@informatik.hu-berlin.de)

Interplanetary Internet Seminar, 2006

08.06.2006

Gliederung

- Context-aware Adaptive Routing (CAR)

⇒ Metrik der Zustellungswahrscheinlichkeit

- Minimum Estimated Expected Delay (MEED)

⇒ Metrik der Verzögerung

- Shared Wireless Infostation Model (SWIM)

⇒ Queue-Behandlung

- Probabilistic Routing Protocol using History of Encounters and Transitivity (PROPHET)

⇒ Metrik über Begegnungswahrscheinlichkeit

CAR -> Einführung

Ziel

- Ermöglichung asynchroner Kommunikation
- Nachrichtenzustellung mit Kontextinformationen optimieren

Routing

- Abschätzungen über Mobilität und Knotenparameter
- hopweises Routing an besten Nachbarknoten

Annahmen

- Keine absoluten Positionsinformationen
- Annahme mobiler Knoten zwischen unverbundenen Netzen
- Netzwerk kooperierender Knoten

CAR –> Routing

- „Kontext“: Für Nachrichtenzustellung relevante Attribute

Routingprinzipien der Knoten

- synchrones Routing für netzinterne Zielknoten

  asynchrones Routing für netzexterne Zielknoten
- berechnen Zustellungswahrscheinlichkeiten durch Kontext
- abschätzen von Kontextinformationen zwischen Updates
- berechnen Routingtabelle:

  (Ziel, Nachbarknoten, Zustellungswahrsch.)
- Nachrichtenweiterleitung an maximal mobilen Netzknoten

CAR –> Kontextbewertung

- **Statisch** gewichtete Kontextbewertung

$$Maximise\left\{ f\left(U\left(x_i \right) \right) = \sum_{i=1}^{n} \underline{\underline{w_i}} U_i\left(x_i \right) \right\}$$

- **Dynamisch** gewichtete Kontextbewertung

$$Maximise\left\{ f\left(U\left(x_i \right) \right) = \sum_{i=1}^{n} \underline{\underline{a_i\left(x_i \right)}} U_i\left(x_i \right) \right\}$$

$a_i\left(x_i \right)$ dabei zusammengesetzter Parameter, z.B. aus:

– Dringlichkeit

– Vorhersagbarkeit

– Verfügbarkeit

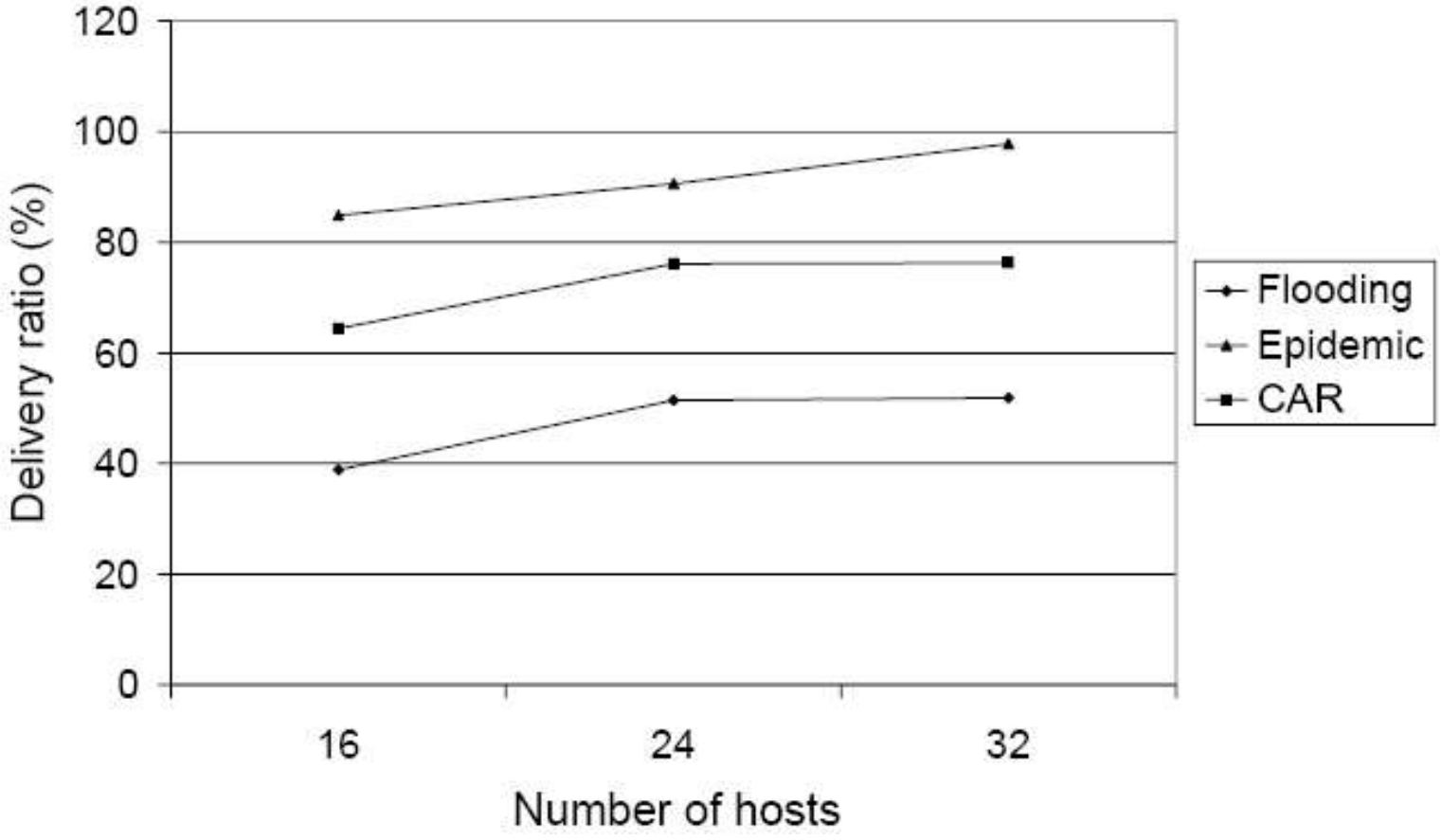
CAR –> Mobilitätsmodell

- Standardannahme völliger Zufälligkeit unrealistisch

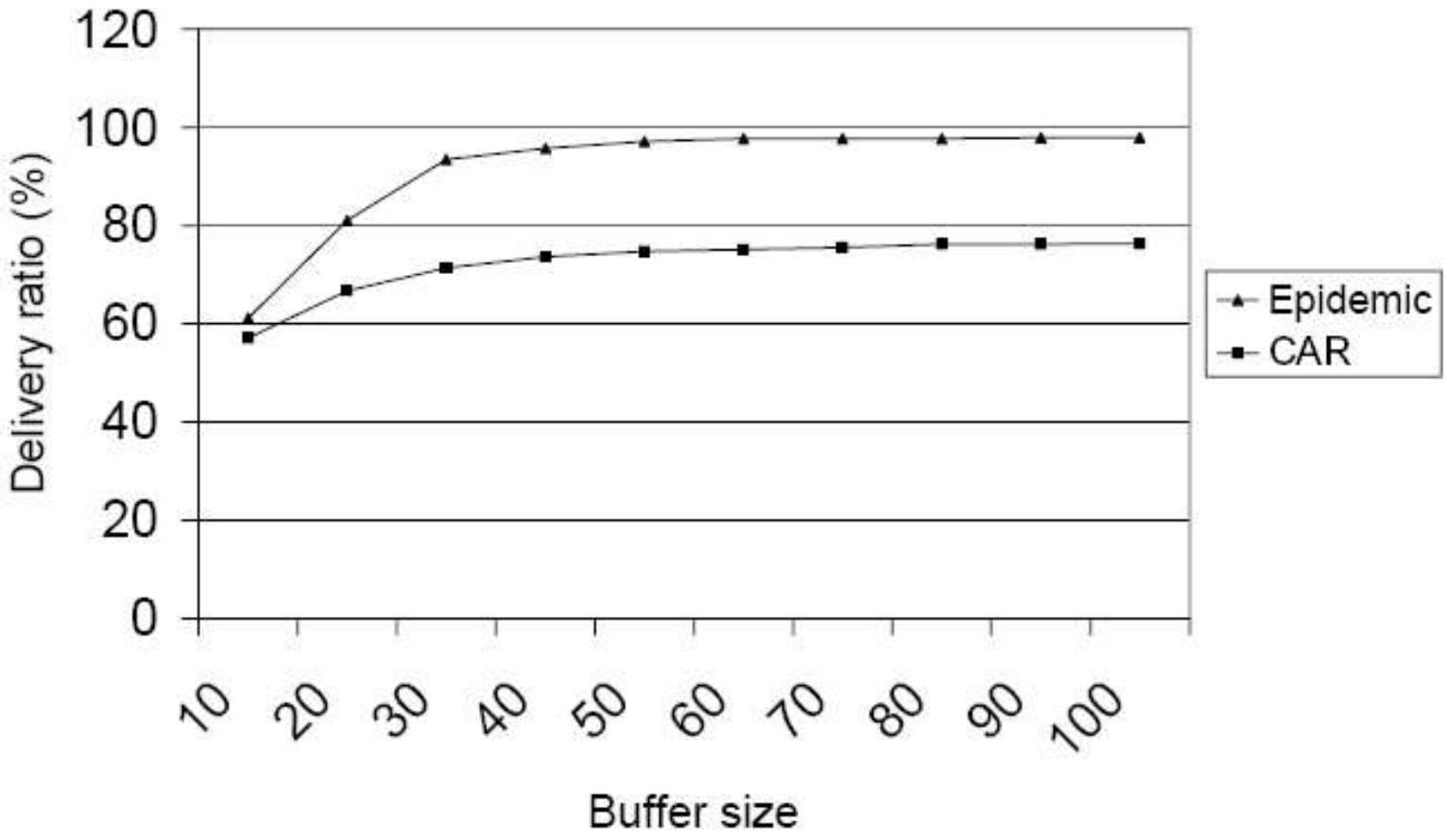
- Knoten streben zufällige Zielpunkte an
- Neuorientierung bei Erreichen der Zielpunkte
- Entscheidungen durch Zufallszahlenvergleich mit Treshold

- Einzelne Netze bewegen sich im Ganzen

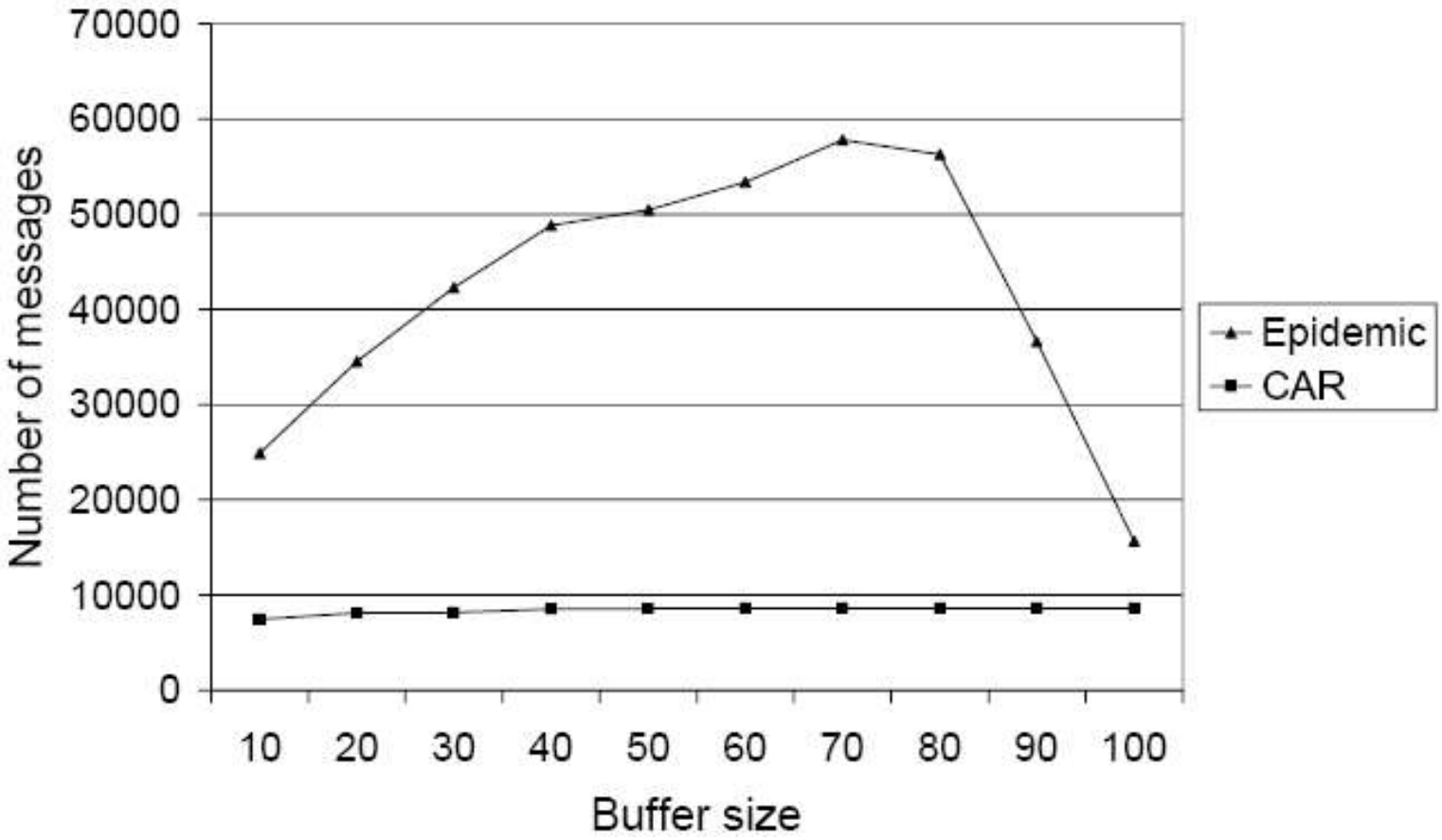
CAR –> Ergebnisse Simulation (1)



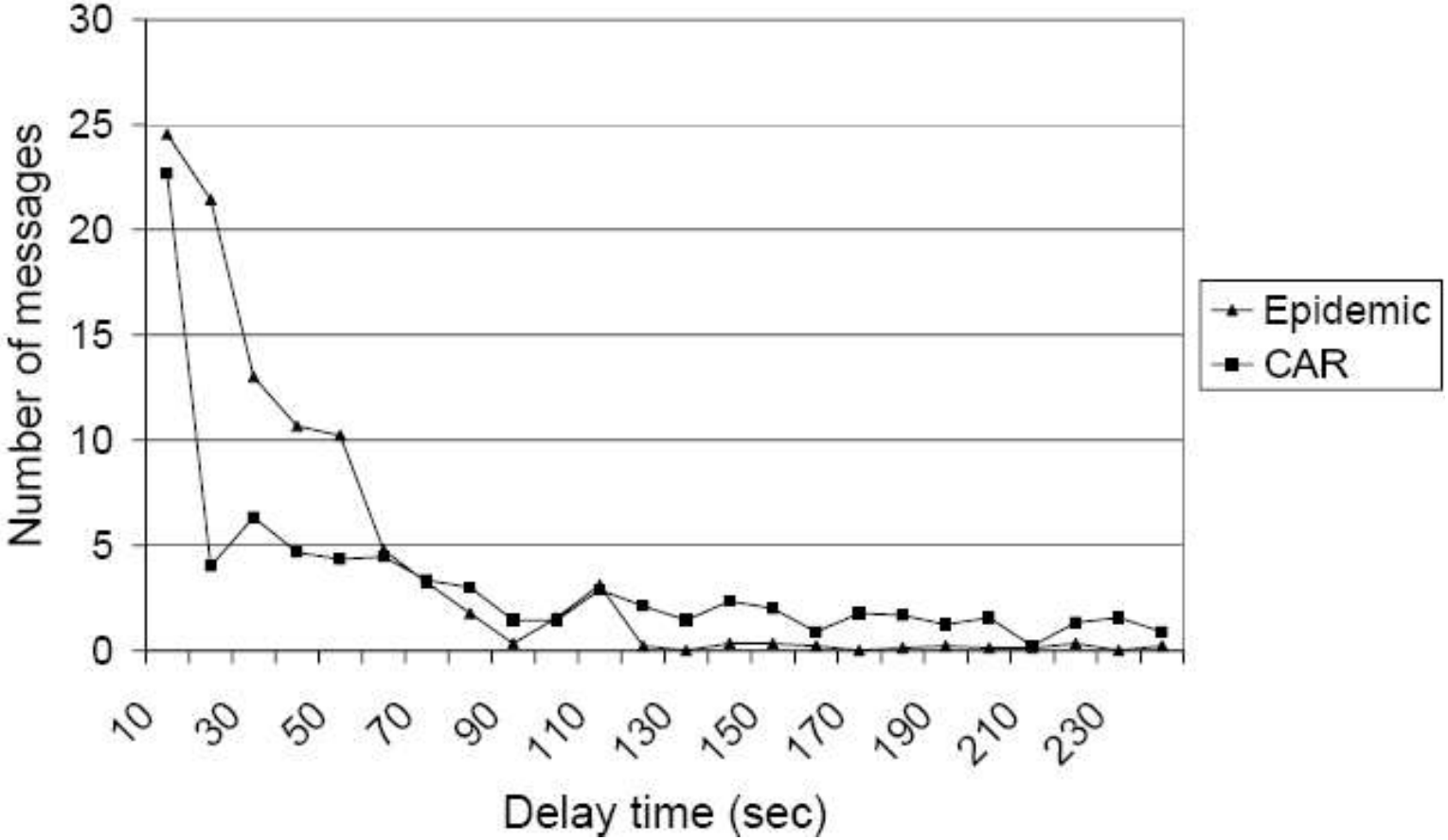
CAR –> Ergebnisse Simulation (2)



CAR –> Ergebnisse Simulation (3)



CAR –> Ergebnisse Simulation (4)



CAR -> Zusammenfassung

- Effizientes Routing durch Bewertung und Vorhersage von Kontextinformationen
- Wenig Overhead
- Mäßiger Berechnungsaufwand
- Gute Zustellungsleistung

MEED -> Einführung

Designziele

- Routing selbstkonfigurierend
- Performanz
- Ressourceneffizienz
- Skalierbarkeit

Annahmen

- Kein Wissen über Netzwerk
- Netzwerk als ungerichteter Graph
- Konstante Bandbreite der Verbindungen
- Verzögerung wird vernachlässigt

MEED -> Ansatz

Optimierung der Zustellungszeit

- Komponenten der Zustellungszeit:
  - Wartezeit
  - Abarbeitungsverzögerung
  - Übertragungszeit
  - Latenz

⇒ berechen- und quantifizierbar

Mobilitätsabschätzung gemäß Vergangenheit

- Knoten beobachten An- und Abmeldungen anderer Knoten
- Begrenzung auf Beobachtungsfenster

Optimierung Link-State-Protokoll

⇒ Knoten können irrelevante Updates unterdrücken

MEED -> Routing

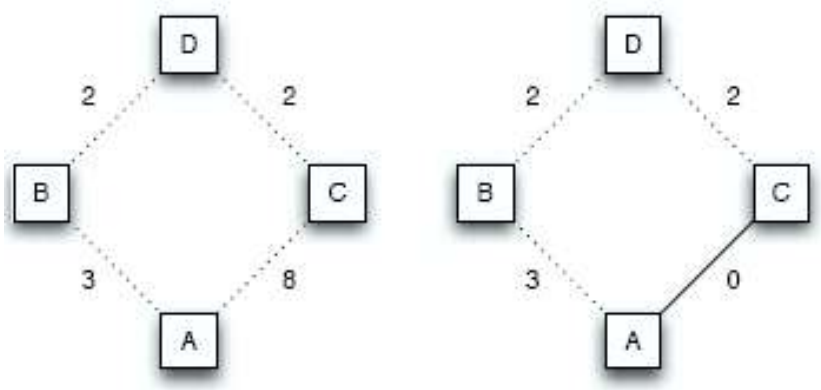
Technik: Per-contact routing

- Routingtabelle bei Nachrichtenankunft berechnen

⇒ Entscheidung auf Basis aktuellster Information

- Verfügbare Links mit 0 bewerten

⇒ Spontane Nutzung von „Abkürzungen“



MEED -> Routing (2)

Vorteil

- Alternativenfindung für tote Links

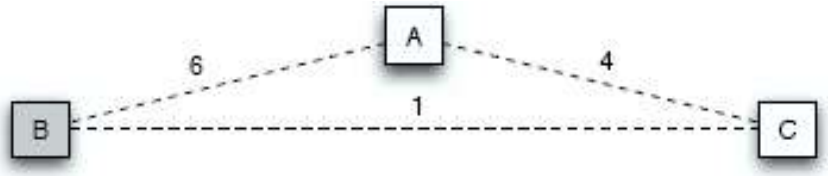
Nachteil

- Verwendung Link-State-Protokoll + veränderliche Topologie

⇒ Anfälligkeit für Zyklen

  - Wenn sich Kantenbewertungen schnell genug ändern

  ⇒ Besonders bei Abkürzungen problematisch:



MEED -> Simulation (1)

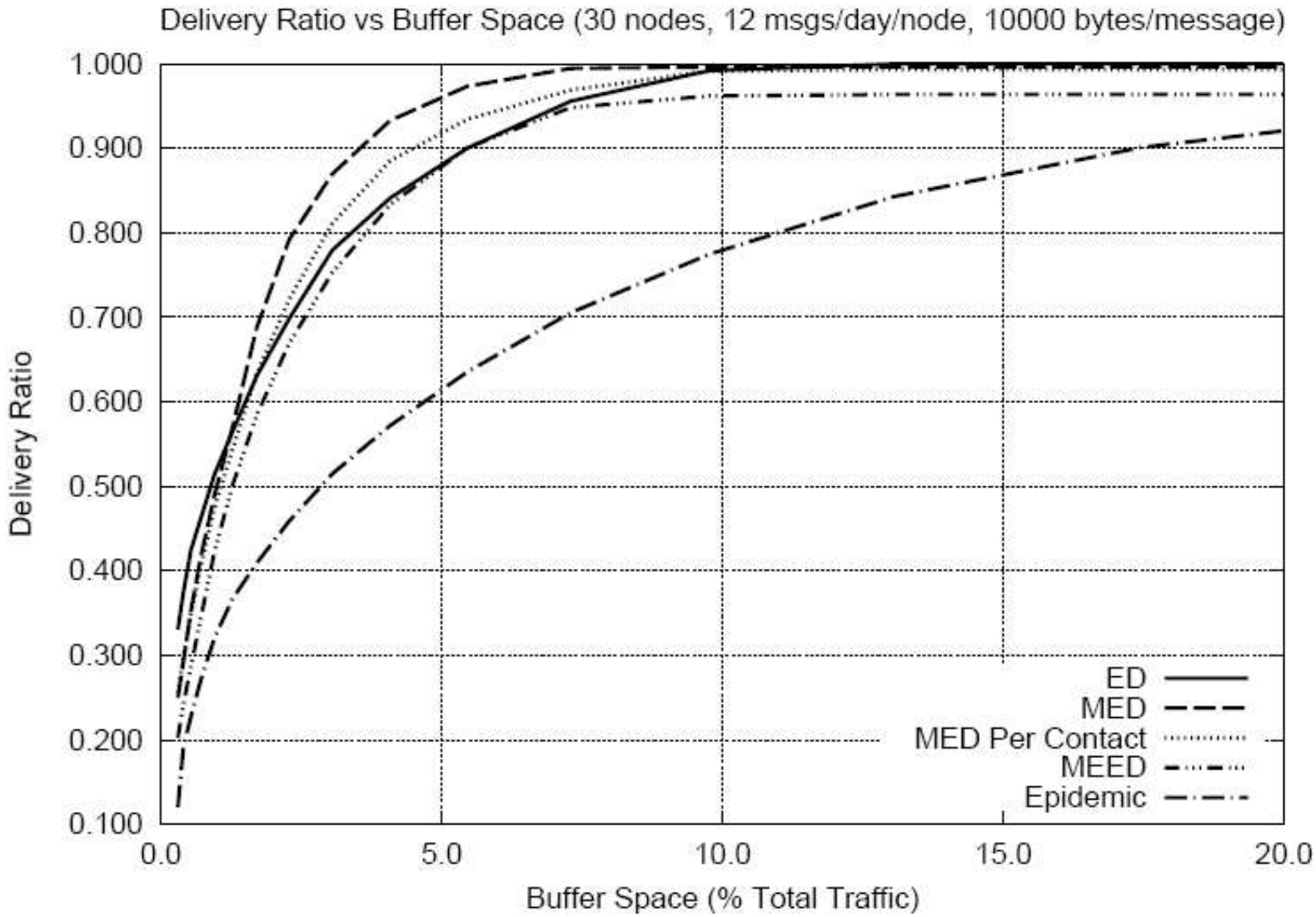
Vertreter

- ED – Earliest delivery
- MED - Minimum expected delay
- MED mit per-contact routing
- Epidemic
- MEED

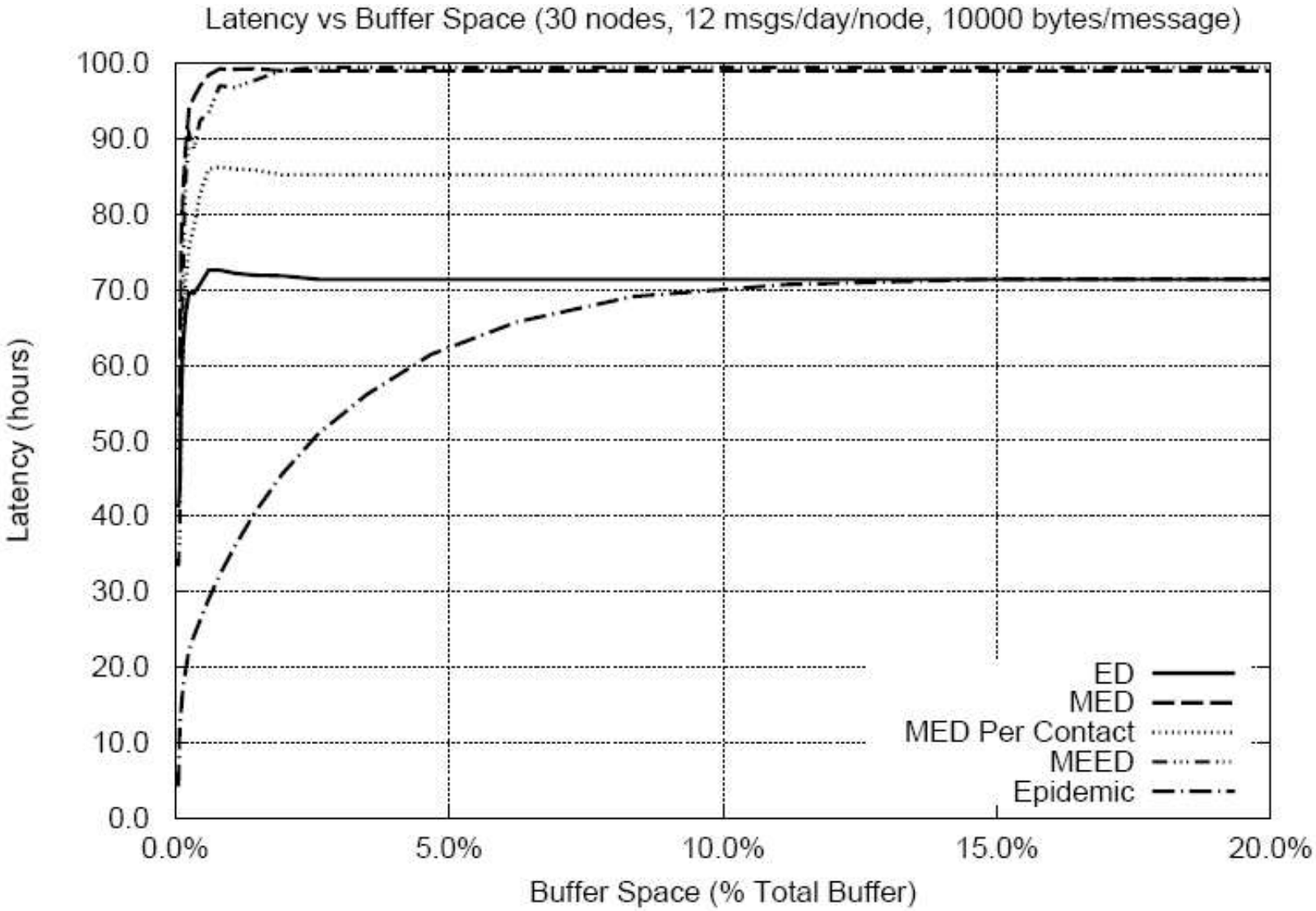
Bedingungen

- Abgeleitete Daten aus WLAN-Traces
- Betrachtung von 30 Knoten

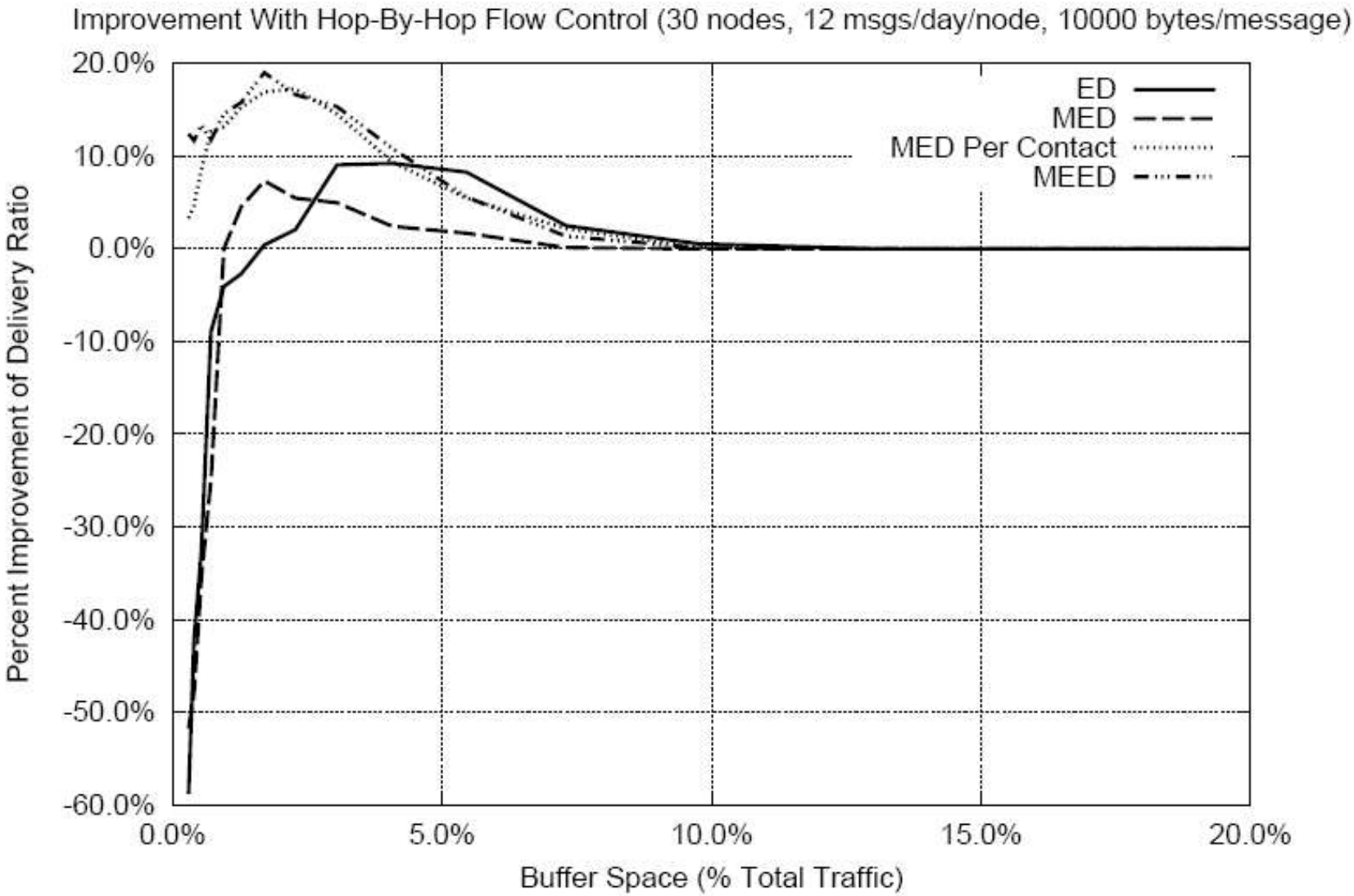
MEED -> Simulation (2)



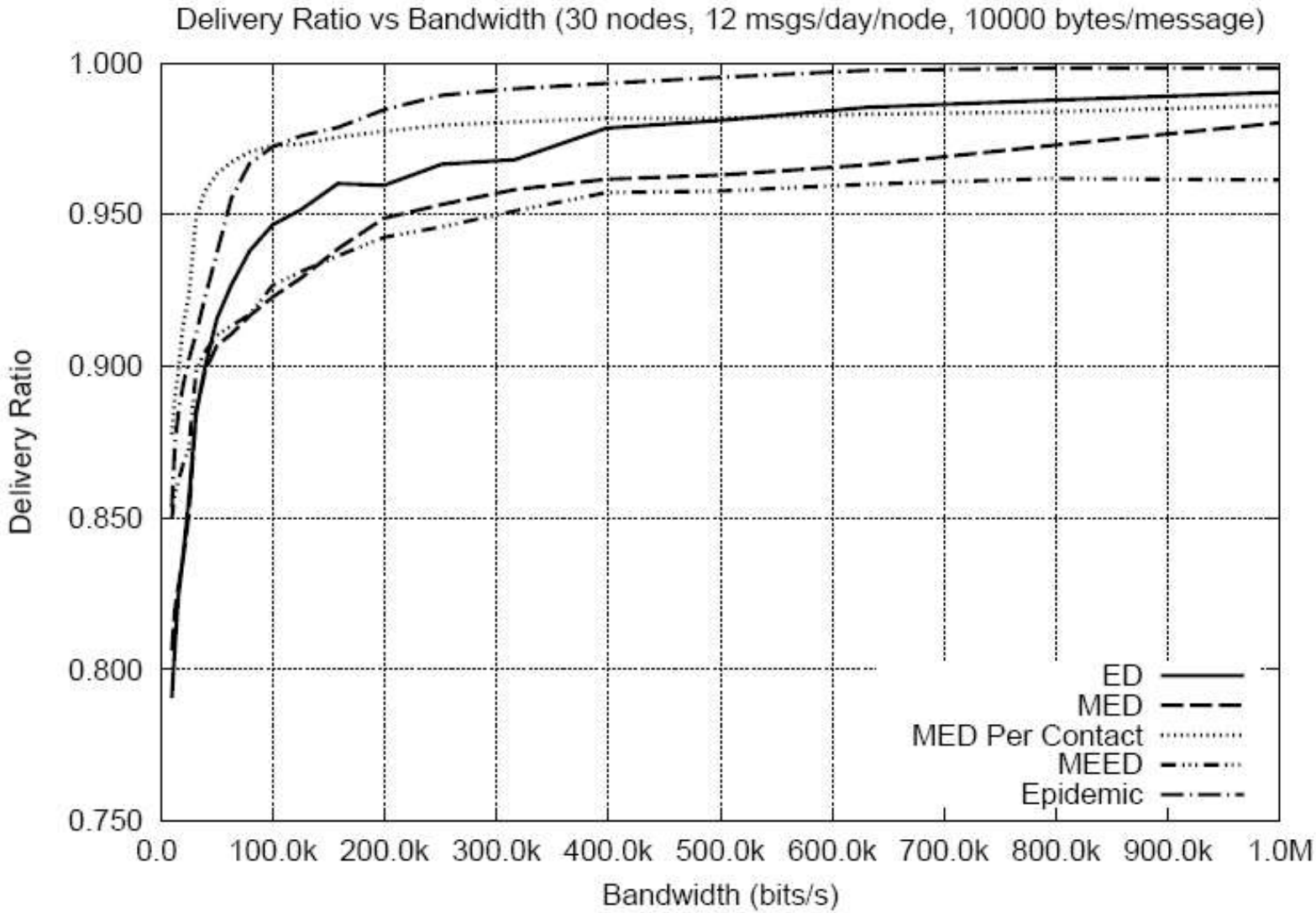
MEED -> Simulation (3)



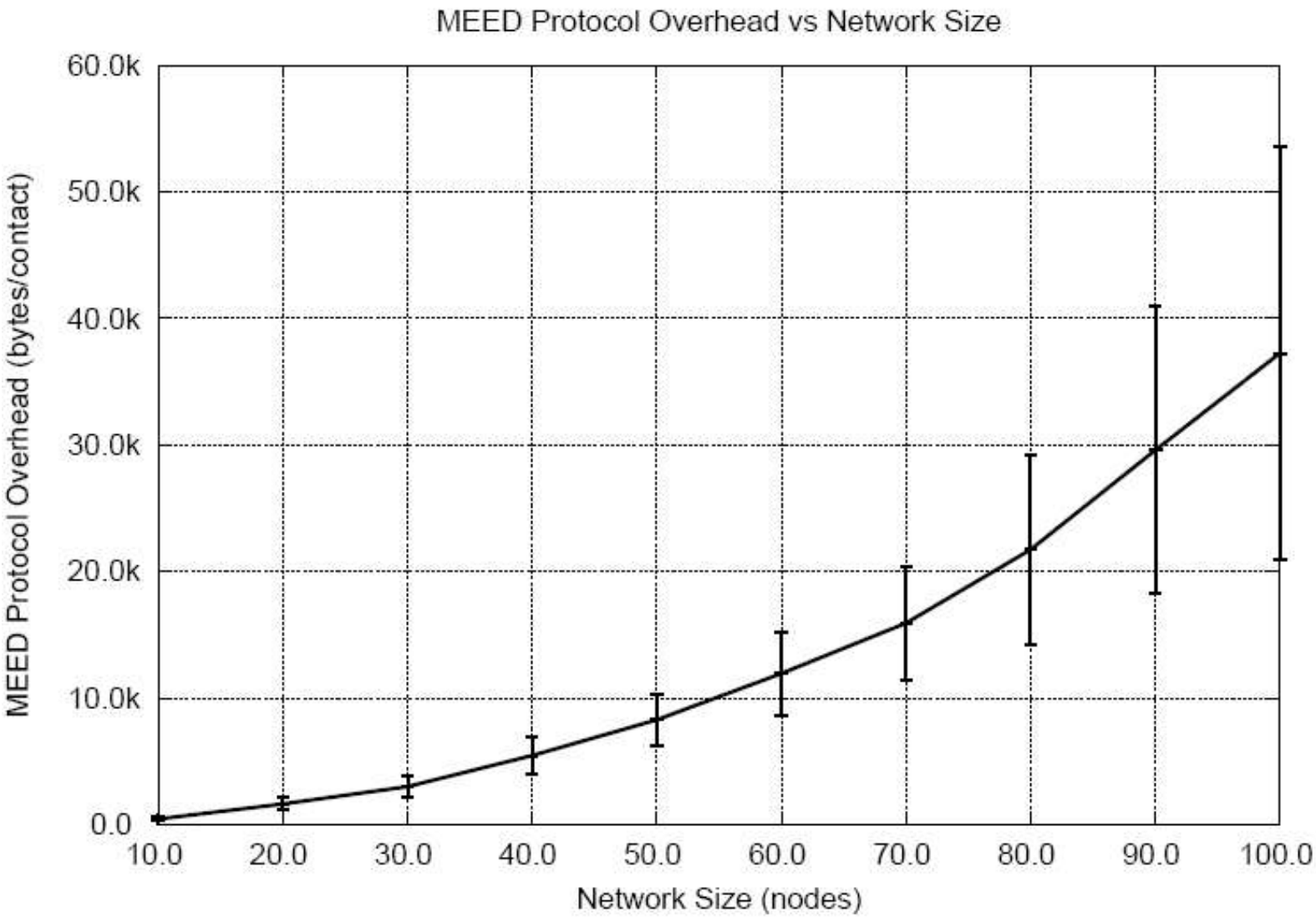
MEED -> Simulation (4)



MEED -> Simulation (5)



MEED -> Simulation (6)



MEED -> Zusammenfassung

- Effizientes Routing durch Beobachtung von Wartezeiten
- Hohe Zustellungswahrscheinlichkeit mit nur einer Nachricht
- Flexibel gegenüber Topologieänderungen

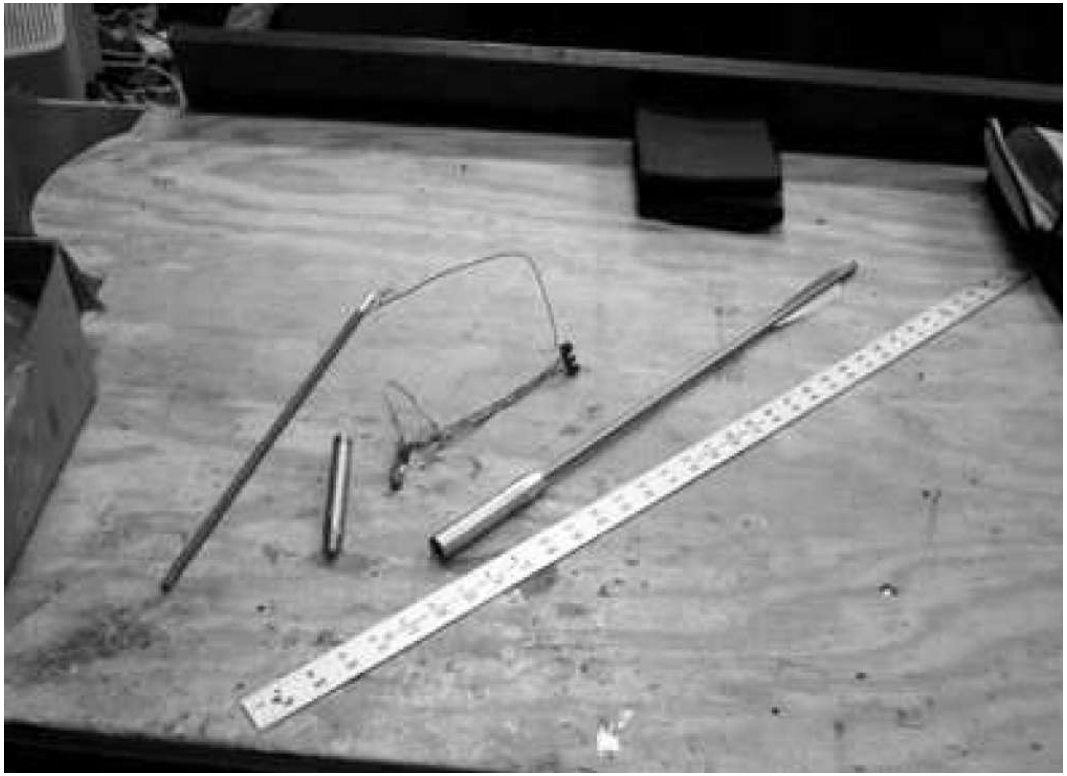
- große Routingtabellen durch Link-State-Routing

⇒ Großer Speicher- und Verarbeitungsaufwand bei Updates

⇒ Viel Overhead in großen Netzwerken

- Zyklen können auftreten

Swim



Swim-Sensor mit Pfeil

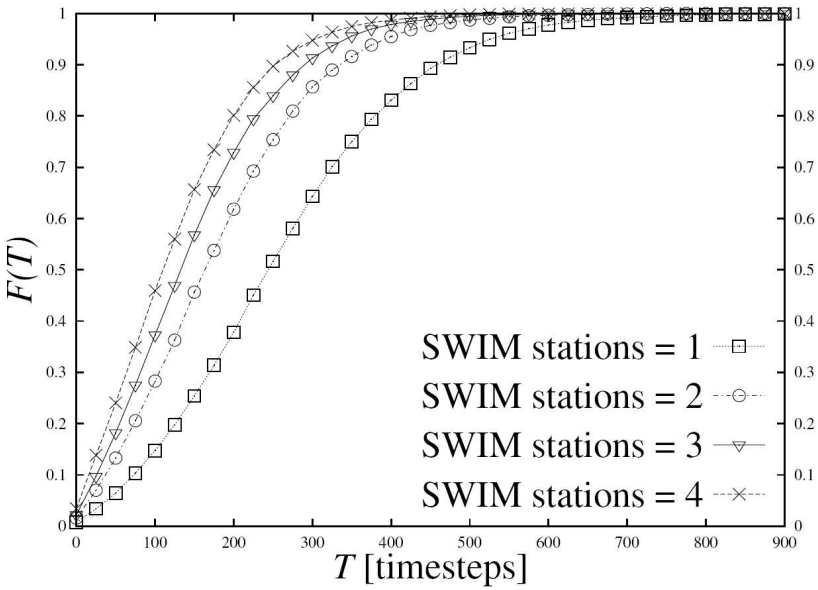
Swim - Bedingungen

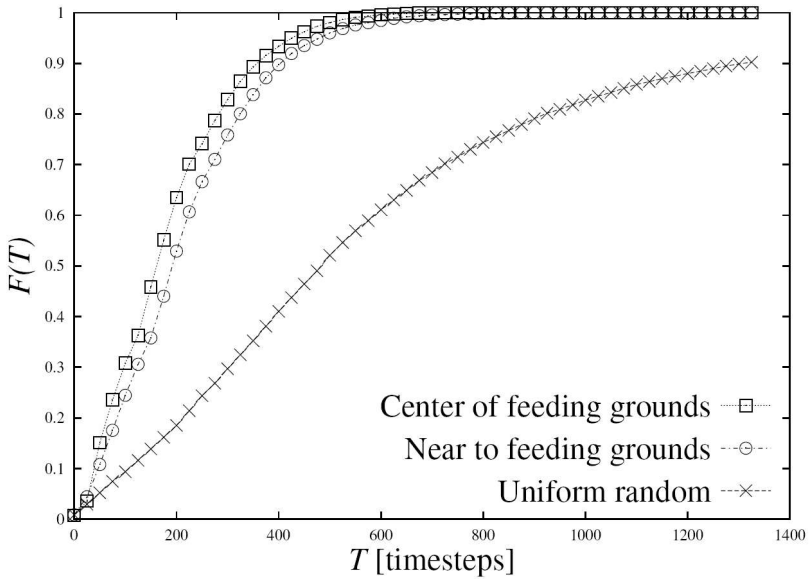
- Wenig Speicherplatz (60 KB)
- Wenig Energie (8-35 mA bei Übertragung)
- „Schnelle“ Datenübertragung (25Kbit/s)
- Maximal 3 Monate Zeit zur Übertragung

Swim - Netztopologie

- Datenaustausch zwischen den Walen
  - Soziale Faktoren bei Begegnungswahrscheinlichkeit
  - Keine echter Zufall
  - Löschung einzelner Pakete bei vollem Speicher
- Endstation „Swim-Station“
  - Schwimmende Tonnen
  - Mehrere Tonnen verteilt im Wasser
  - Komplettleerung der Queue

SWIM Stations



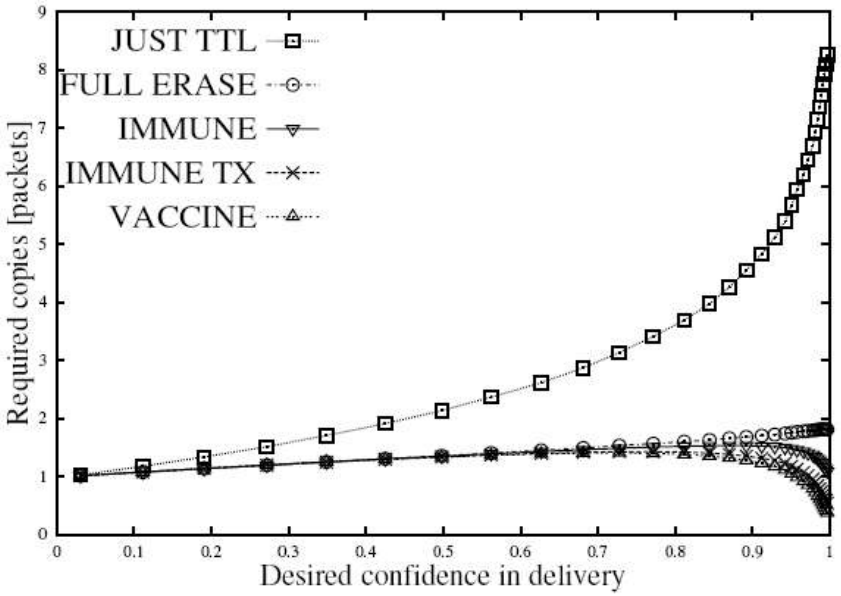


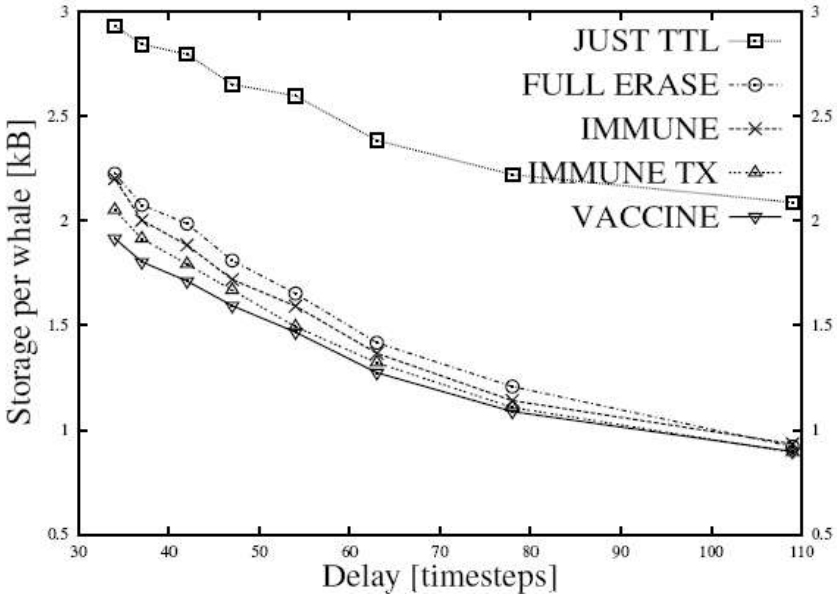
- Gute Wahl der Swim Stations sinnvoller als Anzahl
- F(T): Wahrscheinlichkeit der Auslieferung nach T Timeticks

Queue Management

- **JUST_TTL**: Paket wird nach TTL gelöscht
- **FULL_ERASE**: Beim Übergeben an die Swim Station wird Speicher komplett gelöscht
- **IMMUNE**: Nach Übergabe wird Paket nicht mehr angenommen
- **IMMUNE_TX**: Wie IMMUNE, zusätzlich Weitergabe der Information über erfolgte Auslieferung an andere Wale mit der Nachricht
- **VACCINE**: Wie IMMUNE_TX, aber Auslieferungsinfo an alle Wale

Queue Management II





- Delivery Confidence hängt von F(T) ab -> Zeitdauer unterschiedlich
- Mehr Puffer -> weniger Delay

Prophet - Routing

- Bisher erfolgte Begegnungen als Grundlage der Routingtabelle
- Nicht zufälliges („Soziales“) Netzwerk
- Umweltfaktoren
- „Transitiv“

Prophet - Vorhersagbarkeit?

- „delivery predictability“
- Tabelle mit P(a,b) – Wahrscheinlichkeit des Wiedersehens mit bereits getroffenen Knoten
- Treffen zweier Knoten -> Austausch und Aktualisierung der Routingtabelle
- Keine Berücksichtigung von Umweltfaktoren
- P_{Init}, Beta, Gamma? Unklar.

Prophet – Update I

- Update beim Treffen zweier Nodes
- $P_{(a,b)} = P_{(a,b)old} + (1- P_{(a,b)old}) * P_{Init}$
- $P_{init} = (O,1]$
- „Häufige Begegnung = Hohe Wahrscheinlichkeit“

Prophet – Update II

- $P_{(a,b)} = P_{(a,b)old} * \gamma^k$
- γ = „Alterungskonstante“ (0,1)
- K = Zeiteinheiten seit letzter Begegnung
- Alterung wird regelmässig ausgeführt

Prophet - Transivitität

- $P_{(a,c)} = P_{(a,c)old} + (1 - P_{(a,c)old}) * P_{(a,b)} * P_{(b,c)} * \beta$
- B „scaling constant“ - entscheidet über Einfluss der Transitivität [0,1]
- A sieht häufig B, B häufig C => Routing von A nach C am besten über B

Quellen

- **Adaptive Routing for Intermittently Connected Mobile Ad Hoc Networks**
  Mirco Musolesi, Stephen Hailes, Cecilia Mascolo
- **Practical Routing in Delay-Tolerant Networks**
  Evan P. C. Jones, Lily Li, Paul A. S. Ward
- **Probabilistic Routing in Intermittently Connected Networks**
  Anders Lindgren, Avri Doria, Olov Schelén
- **A Sensor Network for Biological Data Acquisition**
  Tara Small, Zygmunt J. Haas, Alejandro Purgue, Kurt Fristrup
- http://de.wikipedia.org/wiki/Link-State