Freie Universität Berlin

2. Zwischenbericht Diplomarbeit
Implementierung von Routing-Algorithmen für das DES-Testbed

Pawel Cofta

03.02.2009

Struktur der Arbeit

- Einleitung (Motivation, Ausgangspunkt, …)
- Allgemeines zu Routing (Konzepte, Klassifizierungen, Anforderungen WMN)
- Einsatzort (Testbed, Linux, Knoten)
- Routing in Linux (Techniken, Netfilter, Netlink, …)
- Eingesetzte Implementierungen
    - Funktionsweise
    - Welche Techniken?
    - Ggf. welche Anpassungen
- Evaluation
    - Bewertung der Implementierungen
- Ausblick

Implementierungen

- OLSR: olsrd

http://www.olsr.org/

Proaktives Protokoll, wird gepflegt und ist ohne Änderungen einsetzbar

- AODV: AODV-UU

http://core.it.uu.se/core/index.php/AODV-UU

Reaktives Protokoll, Für Kernel 2.6.27 Änderungen durchgeführt. Läuft.

- DSR: DSR-UU

http://core.it.uu.se/core/index.php/DSR-UU

Reaktives Protokoll, Für Kernel 2.6.27 Änderungen durchgeführt. Läuft.

Implementierungen (2)

- Babel:

http://www.pps.jussieu.fr/~jch/software/babel/

Proaktives Protokoll, basiert auf Ideen von DSDV, AODV und EIGRP

Läuft ohne Anpassungen unter Linux 2.6.27

- DYMO: NIST-DYMO

http://sourceforge.net/projects/nist-dymo/

Reaktives Protokoll, Stand August 2005, viele Änderungen für 2.6.27

Kurz vor Abschluss

- TODO:
    - AWDS: http://awds.berlios.de/
    - ZRP: http://www.zrp.be/

AODV

- 1999: Charles E. Perkins, Elizabeth M. Royer
http://www.cs.ucsb.edu/~ravenben/classes/papers/aodv-wmcsa99.pdf
- Reaktives Protokoll
- Knoten nicht auf aktivem Pfad „ruhen sich aus“
- Schleifenfrei (durch Sequenznummern)
- Route Discovery per Broadcast (wie bei DSR)
- Routingstabellen auf den Knoten
- Spezifiziert als RFC 3561:
http://tools.ietf.org/html/rfc3561

DYMO

Dynamic MANET On-demand Routing

http://ianchak.com/dymo/draft-ietf-manet-dymo-12.html

Reaktives Routing-Protokoll

Nachfolger von AODV

Keine großen Änderungen

Keine neuen Features, sondern Vereinfachungen

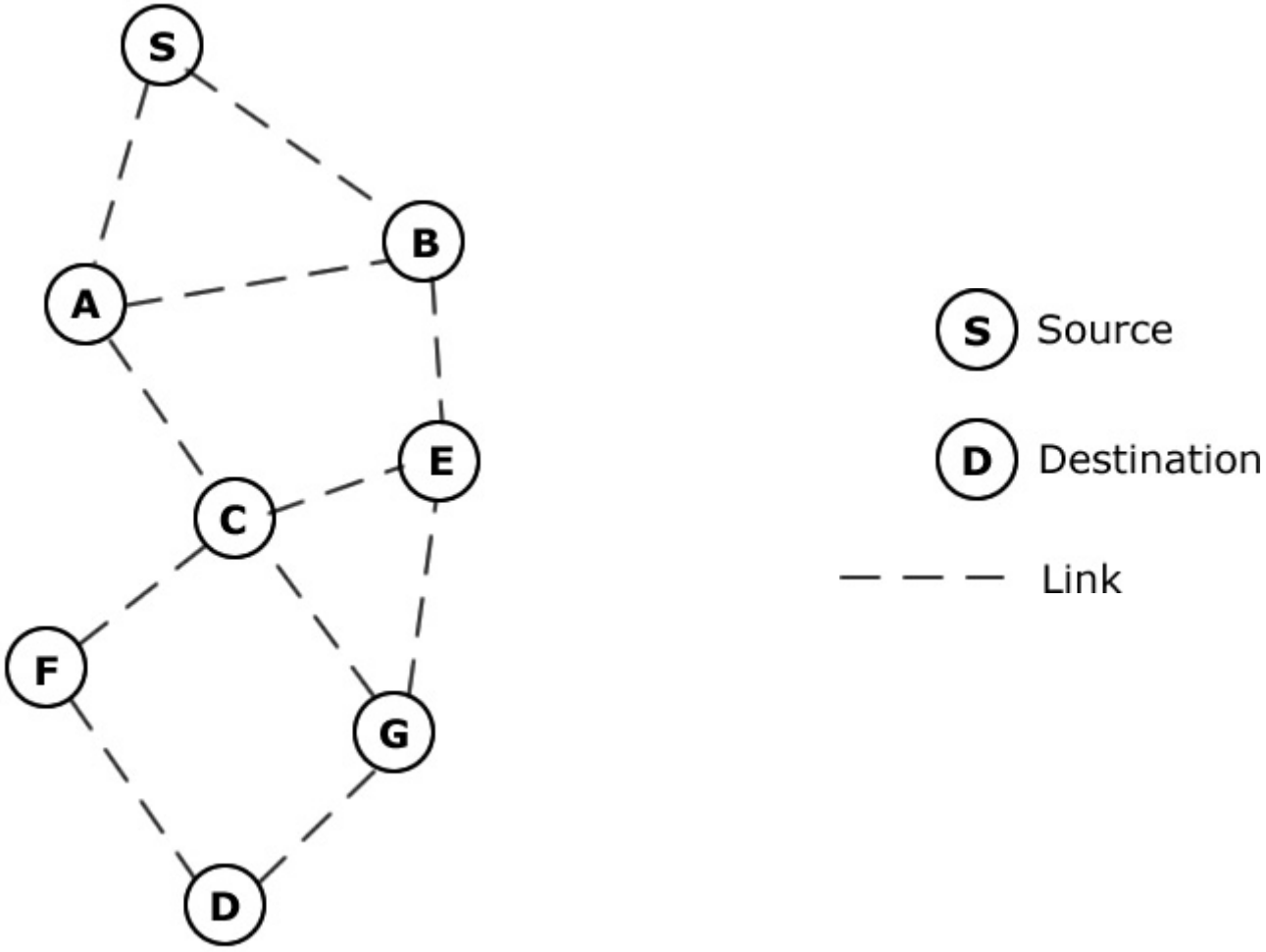
Ziel: einfachere Implementierung – weniger Overhead

Weitere Ansätze:

AODVjr (ohne Sequenznummern, Hello Pakete, Hop Count, ...)

AODV-PA (Path accumulation, wie bei DSR)

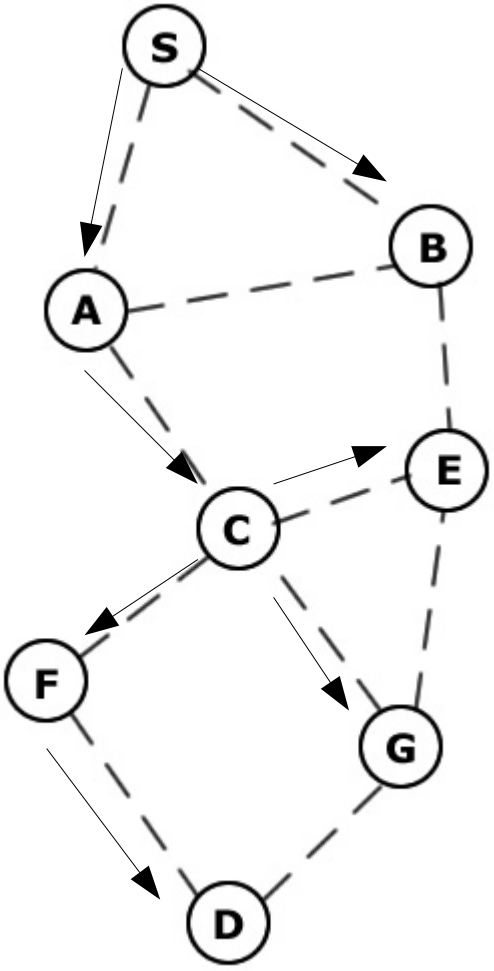
DYMO: Beispiel



DYMO: Route Discovery

- Knoten haben Routingtabelle, wenn aber kein Eintrag für das Ziel: Route Discovery
- Jeder Knoten hat eigene Sequenznummer
- Versenden von RREQ Broadcast und erhöhen eigener Sequenznummer
- Empfänger eines RREQ, der keine Route zum Ziel kennt leitet Paket per Broadcast weiter (nur einmal, Duplikaterkennung Seq.nummer)
- Spätestens beim Ziel wird der Rückweg aufgebaut (RREP)
- Zwischenknoten hängen Ihre Adresse mit an das Paket (Path accumulation, wie bei DSR)
- Quelle, kann eine min. Sequenznummer des Zielrouters angeben, um sicherzugehen, dass keine alte Information propagiert wird

DYMO: Route Discovery RREQ



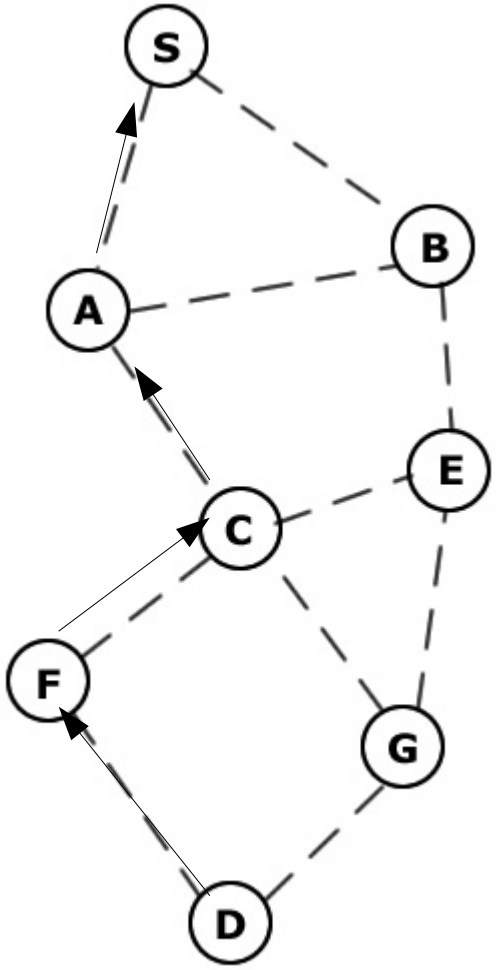
S: RREQ Source S, Dest D

A: RREQ Source S, A, Dest D

C: RREQ Source S, A, C, Dest D

RREQs, die verworfen werden sind nicht eingezeichnet: z.B. B → A

DYMO: Route Reply RREP



Route Reply als Unicast

Knoten die ein RREP weiterleiten aktualisieren ihre Routingtabelle

Wegen RREQ / RREP Verfahren unterstützt DYMO nur bidirektionale Links

DYMO: Route Maintenance

- DYMO muss auf Topologieänderungen reagieren (Mobilität, Ausfälle von Knoten)
- Timer zum Löschen von inaktiven Routen
- Knoten, die Datenpakete für ein Ziel mit unbekannter Route erhalten erzeugen eine Route Error Message (RERR)
- Nicht erreichbarer Knoten und alle davon betroffenen Knoten werden aufgeführt
- RERR wird per Broadcast versendet
- Bekommt ein Knoten eine RERR-Nachricht: Vergleich mit seiner Routing Tabelle → Alle nicht betroffenen Knoten werden aus der RERR Nachricht entfernt, falls welche übrig bleiben → Broadcast

Debianisieren und Verteilen der Software

- Wie verteilt man die Software auf die Knoten?
    - Debianpaket
- Starten und Stoppen der Routingimplementierungen mit Hilfe von Startskripten wie z.B.: startDSR-UU.sh und stopDSR-UU.sh
- Ein Dateisystem für alle Knoten. Problem?
    - Bisherige Implementierungen: Konfiguration aller Knoten gleich
    - Einzige Ausnahme: IP bei virtuellen Schnittstellen
    - Beispiel DSR-UU: ifconfig dsr0 192.168.42.23 up
    - Kann man per bash-Skript individuell aus eth0-Adresse ableiten

Erfahrungsbericht: Portierung

Wie portiert man (als Kernel-Neuling) effizient Routing Implemtierungen für neue Kernel-Versionen?

- Kompilieren
- Welche Datenstrukturen und Funktionen betroffen?
- Was tun, wenn diese schlecht dokumentiert sind?
  - Viele Möglichkeiten, aber beste Methode: Gleiche Änderung im Linux-Kernel finden
  - Mit LXR geht das erstaunlich schnell: z.B.: http://lxr.linux.no/

Erfahrungsbericht: Portierung (2)

Oft ist es schwer zu ergründen, WARUM etwas im Kernel geändert worden ist

Patches enthalten selten Begründungen, meist nur Beschreibung was geändert worden ist:

```
[NETLINK]: Add "groups" argument to netlink_kernel_create

Signed-off-by: Patrick McHardy <[EMAIL PROTECTED]>

---
commit 5719d60b114683e7c1bf1aa9a553efb641184e1b
tree c6a56c893ae404e6767f3cefbebd2a88a2981775
parent c366740a65d35924ee4efce970db8a738dd4b384
author Patrick McHardy <[EMAIL PROTECTED]> Sat, 13 Aug 2005 01:50:00 +0200
committer Patrick McHardy <[EMAIL PROTECTED]> Sat, 13 Aug 2005 01:50:00 +0200

 drivers/w1/w1_int.c             |    2 +-
 include/linux/netlink.h         |    2 +-
...
```

Erfahrungsbericht: Portierung (3)

Einige für das Routing relevante Änderungen treten immer wieder auf

z.B.: Netlink, proc_net_create, Netfilter, sk_buff, ...

Beschleunigt die Arbeit

Diese Änderungen sind leider eine eintönige Beschäftigung...

ABER: Verschiedene Implementierungen → viele Techinken

Das ist der interessante Aspekt der Arbeit

Danke! Fragen?