Freie Universität Berlin

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

Pawel Cofta

04.11.2008

Ausgangspunkt

-Ziel: möglichst viele Routing-Algorithmen aus verschiedenen Bereichen (proaktiv, reaktiv etc.) für das Testbed lauffähig machen

-Geeignete Implementierungen von Routing-Algorithmen suchen:
  -Meist keine Implementierung vorhanden (nur PDF-Dokument)
  -Gelegentlich: ns2 Implementierung verfügbar
  -Selten: Linux Implementierung (meist aber nur für ältere Kernel)

-Erforderliches Grundlagenwissen: Routing im Linux-Kernel

Routing in Linux

-Routingtabellen

-Netlink

-Netfilter

-Socket Buffer, Sockets, ...

Forwarding Information Database

-3 Fälle bei ankommenden Paketen können auftreten

- -Paket ist für den Router bestimmt
- -Paket ist für Endsystem im eigenen Netz bestimmt
- -Paket ist für Endsystem im entfernten Netz bestimmt

-Routinginformationen in Tabellen gespeichert (max. 256)

-3 Standardtabellen (weitere können angelegt werden):

local (höchste Priorität: 0), main, default (niedrigste Priorität)

-Anhand der Routing Policy Database wird entschieden, welche Routingtabelle verwendet wird

-Routingtabellen und die Routing Policy Database bilden die FIB (Forwarding Information Base)

FIB (2)

-Für jedes ankommende Paket: Regeln aus Policy Database anwenden

-Bei Passender Regel in Tabelle nachschauen: Längster Präfix entscheidet

-Kein Eintrag → nächste Tabelle gemäß Priorität (letzte Tabelle default)

-Bei lokalen Paketen wird immer die local Tabelle benutzt

-Das Tool ip: Manipulation von Routen und Regeln

```
atze@ubuntuvm:~$ sudo ip route add prio 1000 from 10.0/16 iif eth3 table 12

atze@ubuntuvm:~$ ip rule show
0:      from all lookup local
1000:   from 10.0.0.0/16 iif eth3 lookup 12
32766:  from all lookup main
32767:  from all lookup default
```

ip verwendet im Hintergrund Netlink

Netlink

-Netlink wird zur Interprozesskommunikation eingesetzt

-Verwendung von Sockets (Berkeley sockets API)

-Adressierung über PIDs (Process ID), daher keine Kommunikation zwischen mehreren Systemen über Netlink möglich

-Warum wichtig für Routing?

- -Ermöglicht Datenaustausch zwischen User- und Kernelspace
- -Routingtabellen können mit Netlink bearbeitet werden
- -Empfohlene Schnittstelle für Netzwerkdaten

Netfilter

-Netfilter ermöglicht es Pakete zu inspizieren und zu manipulieren

-Grundlage für iptables

-5 sogenannte Ketten:

  -PREROUTING

  -LOCAL_IN

  -FORWARD

  -LOCAL_OUT

  -POSTROUTING

-Mögliche Aktionen (u.a.):

  -ACCEPT

  -DROP

  -STOLEN

  -...

Jede Implementierung ist anders...

-Problem: Wie weit ist eine Vergleichbarkeit der Routing Algorithmen möglich?

-Qualität der Implementierung

-Eingesetzte Techniken:

- -Kernel- vs. Userspace
- -Linux Routingtabellen vs. Eigene Implementierung
- -Layer 2, 3 oder 2.5
- -...

Unterscheidungskriterien

Userspace vs. Kernelspace:

-Beliebige Aufteilung des Routing-Algorithmus möglich

  -Komplett im Kernelspace (hohe Performance, kein Kontextswitch)

  -Routenberechnung im Userspace, Routing mit Kernel-Routingtabellen

  -Andere Varianten denkbar

-Verwaltung der Routingtabelle (Kernel / eigene)

-Virtuelles Netzwerkinterface (ja / nein)

-Eigene ARP-Implementierung (ja / nein)

-Layer (2 / 3 / 2.5)

-Netlink / Netfilter / ...

OLSR

-OLSR (Optimized Link State Routing Protocol)
  -proaktiv
  -RFC 3626
  -Implementierung existiert wird gepflegt (http://www.olsr.org/)
  -Nutzt Kernel-Routingtabellen
  -Layer 3
  -Userspace Deamon

AODV

-AODV (Ad hoc On-Demand Distance Vector Routing)

-reaktiv

-RFC 3561

-Keine Implementierung für aktuellen Kernel verfügbar

-Am besten geeignet: Impl. Uppsala University:
http://core.it.uu.se/core/index.php/AODV-UU

-Kernelmodul und Deamon

-Problem bei reaktiven Protokollen: Pakete können nicht direkt geroutet werden → zuerst Path Discovery

  -Was tun mit dem Paket in der Zeit?

Probleme bei AODV-Anpassung

-Keine aktuelle Netlink Dokumentation vorhanden

-Quellcode nahezu ohne Kommentare

```
extern struct sock *netlink_kernel_create(struct net *net,
                                          int unit,unsigned int groups,
                                          void (*input)(struct sk_buff *skb),
                                          struct mutex *cb_mutex,
                                          struct module *module);
```

- Wie errät man die Bedeutung der Parameter?

LXR - Linux Cross Reference


```
1359 netlink_kernel_create(struct net *net, int unit, unsigned int groups,
1360                        void (*input)(struct sk_buff *skb),
1361                        struct mutex *cb_mutex, struct module *module)
1362 {
1363         struct socket *sock;
1364         struct sock *sk;
1365         struct netlink_sock *nlk;
1366         unsigned long *listeners = NULL;
1367
1368         BUG_ON(!nl_table);
1369
1370         if (unit < 0 || unit >= MAX_LINKS)
1371                 return NULL;
1372
1373         if (sock_create_lite(PF_NETLINK, SOCK_DGRAM, unit, &sock))
1374                 return NULL;
1375
1376         /*
1377          * We have to just have a reference on the net from sk, but don't
1378          * get_net it. Besides, we cannot get and then put the net here.
1379          * So we create one inside init_net and the move it to net.
1380          */
1381
1382         if (__netlink_create(&init_net, sock, cb_mutex, unit) < 0)
1383                 goto out_sock_release_nosk;
1384
1385         sk = sock->sk;
1386         sk_change_net(sk, net);
1387
1388         if (groups < 32)
1389                 groups = 32;
1390
1391         listeners = kzalloc(NLGRPSZ(groups), GFP_KERNEL);
1392         if (!listeners)
1393                 goto out_sock_release;
```

Aktuelle Arbeiten

-AODV-UU lauffähig machen

  -Kompiliert bereits

  -Mit Hilfe von LXR Unterschiede zu Älteren Kernelversionen aufspüren und Dokumentieren

  -Größte Änderungen in 2.6.24: u.a. Einführung von Namespaces

-Socket Buffer

  -Kleines Testmodul um Änderungen an Socket Buffern nachvollziehen zu können

Ausblick: Weitere Algorithmen

-DSR (Dynamic Source Routing)

- -reaktiv
- -Ebenfalls Uppsala University Impementierung
- -Technisch stark unterschiedlich zu AODV-UU
- -Stand: Dezember 2007 (daher keine großen Schwierigkeiten zu erwarten)

-ZRP (Zone Routing Protokoll)

- -Einziges hybrides Verfahren mit Linux-Implementierung
- -Version 0.1 vom Mai 2004 (auf dem Originalserver nicht mehr verfügbar – 404)

-DYMO (AODV Nachfolger)

- -Universität Murcia, Stand: Mai 2006

Danke! Fragen?