netcrashd

Björn Meier, Johannes Tenschert, Sergey Datsevich

16. März 2012

Problem & Ziele

Problem:

- Half-Opened-Socket

Ziele:

- Wichtigste Fehlerszenarien erkennen und
- im Hintergrund als Daemon
- nach gewählter Strategie
- idealerweise nicht als root
- an verwaiste IP-Adressen gebundene Sockets schließen.
- Strategien und Szenarien sollten konfigurierbar sein.

High-Level-Events (1)

1: Kurzfristiger Verbindungsabbruch

- Verbindung nur kurz weg, funktioniert danach wieder
- keine geänderte IP
- gleiches Netzwerk
- *kann* ignoriert werden, sollte nicht mit anderen Szenarien verwechselt werden

High-Level-Szenarien (2)

2.1: Längerer Verbindungsabbruch mit gleicher IP

- Verbindung länger weg als nur „kurz“
- gleiches Netzwerk
- in Strategie kann entschieden werden, wie zu reagieren ist

High-Level-Szenarien (2)

2.1: Längerer Verbindungsabbruch mit gleicher IP

- Verbindung länger weg als nur „kurz“
- gleiches Netzwerk
- in Strategie kann entschieden werden, wie zu reagieren ist

2.2: Längerer Verbindungsabbruch mit neuer IP

- Verbindung länger weg als nur „kurz“
- gleiches Netzwerk
- Es muss reagiert werden.

High-Level-Szenarien (3)

3: Kompletter Verbindungsabbruch

- Im beobachteten Zeitraum keine Verbindung mehr zum Netzwerk aufgebaut.
- Es muss reagiert werden.

4: Wechsel des Netzwerks

- Eine Verbindung abgebrochen,
- eine andere geöffnet.
  - z. B. Wechsel LAN $\rightarrow$ WLAN
  - oder Wechsel des verwendeten WLAN:
    FAU-STUD $\rightarrow$ eduroam, gleiches Interface wlanX
- Es muss reagiert werden.

Low-Level-Szenarien

Nr	Beschreibung	Ablauf	HL-Nummer
1	kurz abstecken *ignoriert*	NL(down) NL(up)	1
2	lang abstecken ($\leq 30s$) **ShortDisconnect**	[NL(down)] DELADDR(IP1) [NL(up)] NEWADDR(IP1)	1, 2.1
3	$30s \leq$ lang abstecken $\leq 60s$ **LongDisconnect**	siehe 2 mit Wartezeit $\geq 30s$ bzw. $\leq 60s$	2.1
4	lag abstecken, neue IP **LongDisconnectNewIP**	siehe 2 Wartezeiten egal, nur gleiches Netzwerk	2.2
5	Verbindungsabbruch **LostConnection**	[NL(down)] DELADDR(IP1)	3
6	neue Verbindung **NewConnection**	[NL(up)] NEWADDR(IP1)	-
7	Wechsel des Netzwerks **ChangedNetwork**	[NL(down)] DELADDR(IP1) [NL(up)] NEWADDR(IP2)	4

Aufbau

Event-Erkennung

Über Netlink auf NEWLINK, DELADDR, NEWADDR warten. Eigener Thread.

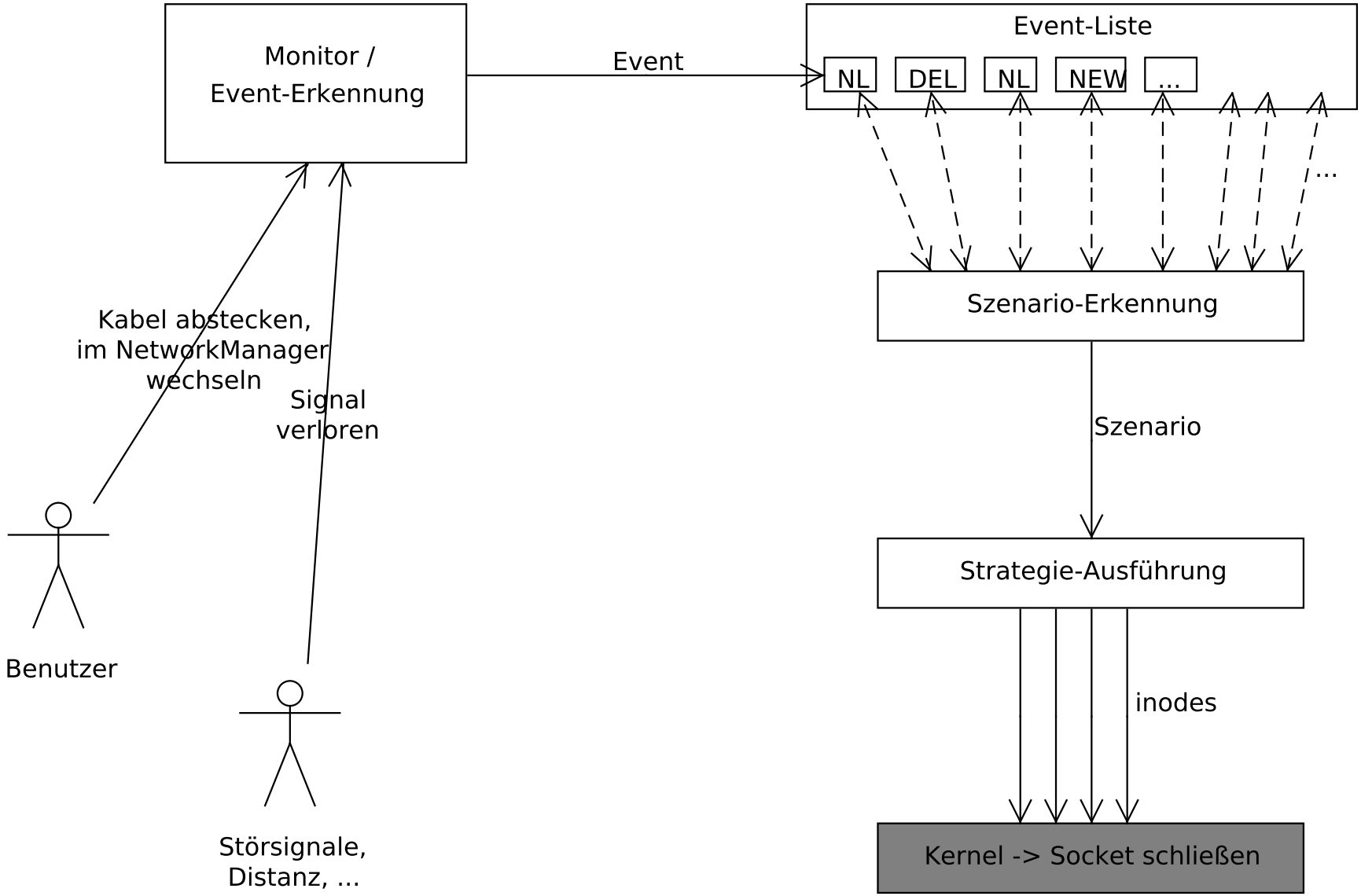
Szenario-Erkennung

Events interpretieren (siehe Low-Level-Szenarien) und an Strategie-Auswahl übergeben.

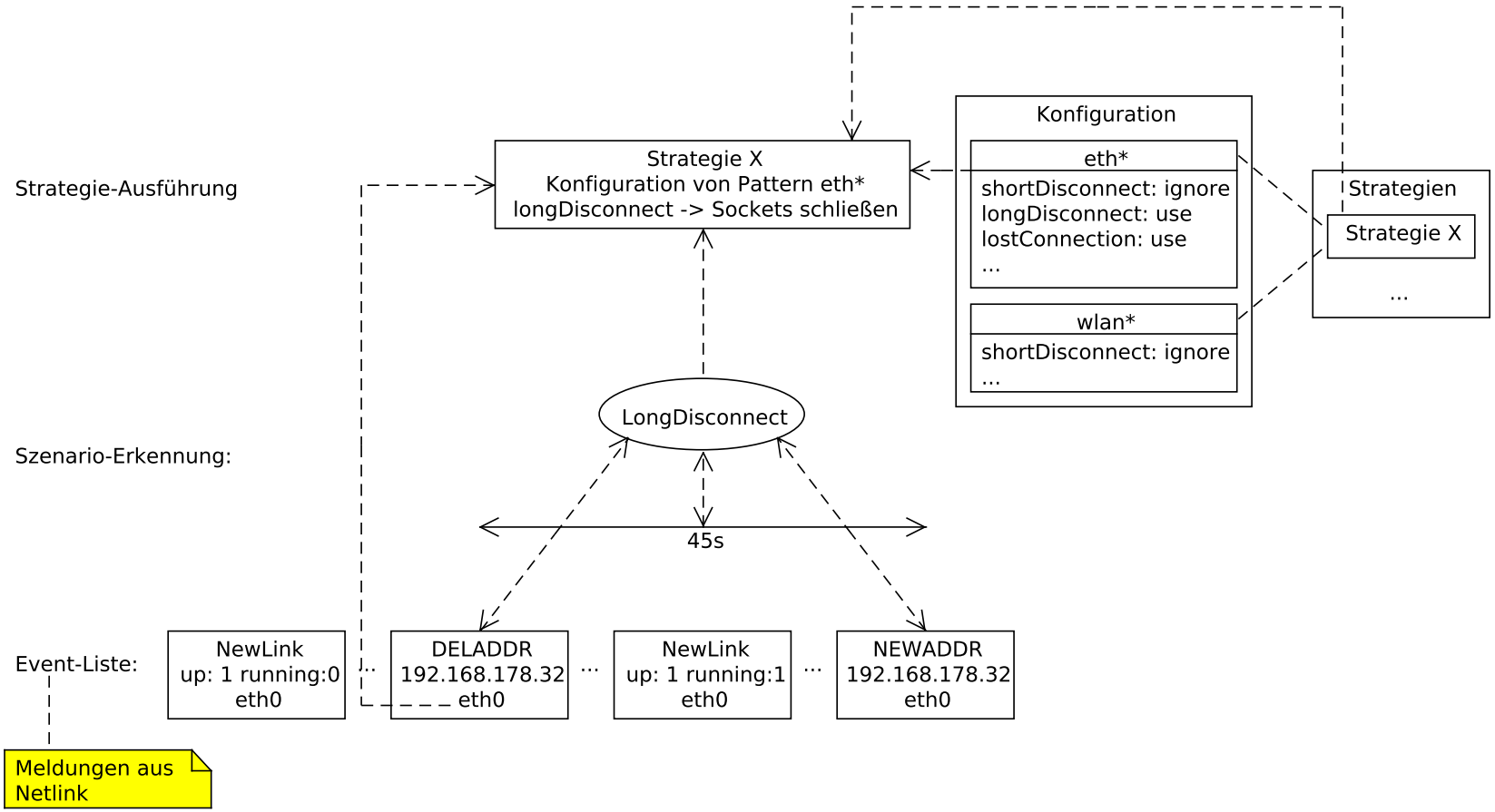
Strategie-Ausführung

Auswahl der passenden Konfiguration. Strategie entscheidet, ob und wie auf Szenario reagiert werden soll.

Aufbau


Monitor /
Event-Erkennung
Event
Event-Liste
NL
DEL
NL
NEW
...
...
Szenario-Erkennung
Szenario
Strategie-Ausführung
inodes
Kernel -> Socket schließen
Kabel abstecken,
im NetworkManager
wechseln
Signal
verloren
Benutzer
Störsignale,
Distanz, ...

Längerer Verbindungsabbruch



Konfiguration (Beispiel)

```
scenario : {
        newnetwork_wait = 10;
        changednetwork_timespan = 20;
        shortdisconnect_timespan = 45;
        longdisconnect_timespan = 90;
        lostConnection_after = 90;
        longdisconnectNewIP_timespan = 60; };
strategy : {
  eth0 : {
    name = "default";
    shortDisconnect = "ignore";
    longDisconnect = "ignore";
    longDisconnectNewIP = "use";
    lostConnection = "default";
    changedNetwork = "use"; };
  eth* : { ... };
  wlan* : { ... };
  default : { ... }; };
```

Features

- Wichtigste Fehlerszenarien erkannt
- Erkennung, Szenarien, Strategien im Userspace
  $\rightarrow$ Kernel nicht weiter aufblähen; Fehler, Änderungen
- keine Root-Rechte notwendig
- Daemon
- konfigurierbar
- Dokumentation des Quellcodes
  $\rightarrow$ Erweitern leichter

Umfang

```
sa67dusy@faui49man1:~/passt/netcrashd$ make me.proud
   197    461   3685 init.c
    10     16    109 log.c
   387    993   8194 monitor.c
   169    459   3646 netlink.c
   615   1529  13926 scenario.c
   289    676   6135 strategy.c
   158    407   3041 timeout.c
   360    885   9288 config.c
   428   1105   8808 cache.c
   361    935   6800 proc.c
    71    167   1372 scenarios/LostConnection.c
    91    223   2056 scenarios/ShortDisconnect.c
   106    257   2315 scenarios/LongDisconnect.c
   105    259   2332 scenarios/LongDisconnectNewIP.c
   139    364   3392 scenarios/ChangedNetwork.c
   157    408   3126 netlink/addr.c
   114    282   2168 netlink/link.c
   129    317   2518 netlink/neigh.c
   125    308   2428 netlink/route.c
   127    299   2981 strategies/default.c
    43    112    883 Makefile
    24     55    438 DoxyConfig
    83    214   1552 cache.h
     8      9     87 config.h
    26     97    607 ip.h
    27     52    375 log.h
    61    184   1396 monitor.h
    92    220   1471 netlink.h
    52    143    947 proc.h
   113    344   3250 scenario.h
    47    131   1492 strategy.h
     7     17    145 timeout.h
  4721  11928 100963 total
```

Kernel - SocKet Remote ShutDown (skrsd)

Schnittstelle

- /proc/net/shutdown_socket (WUGO)
  - für alle schreibbar
- Rechteüberprüfung zur Laufzeit

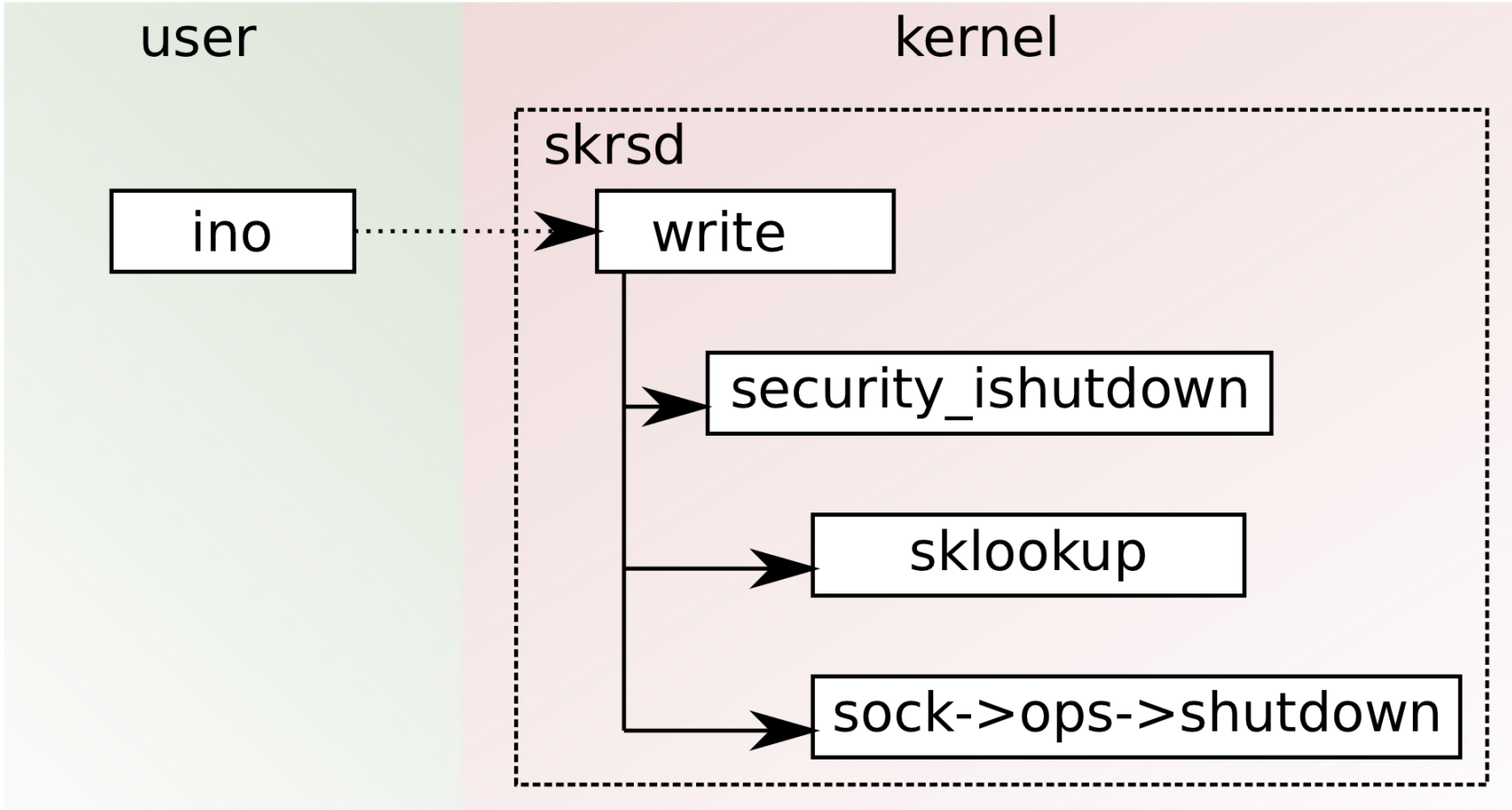
Verwenden:

Statisch, Loadable Kernel Module

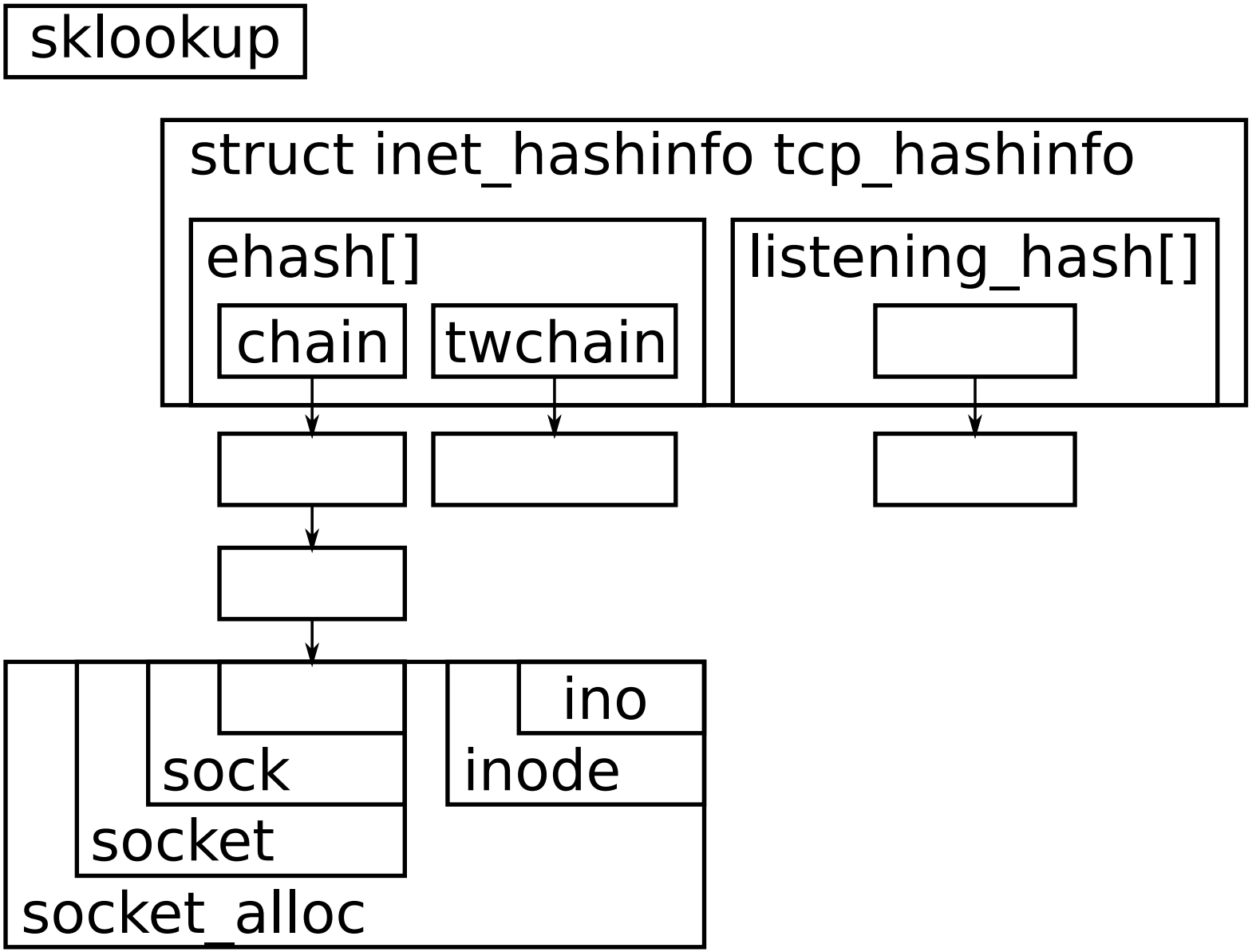
Umfang

```
$ wc skrsd.c
 190  440 4009 skrsd.c
```

Kernelmodul - Übersicht



Kernelmodul - Lookup



Testen / Vorführen

- Python-Skript
- ssh
- wget