

Das Linux Netfilter Paket

Seminar zu ausgewählten Themen der
Netzwerkwelt

Klaus Rechert

Firewalls

- Eigenschaften
 - Einzige Schnittstelle zwischen zwei Netzen
 - Implementiert eine Policy
- Verschiedene Typen
 - Paketfilter
 - Proxy / Application Firewalls

Paketfilter

- Eigenschaften
 - Passiv
 - Überprüfen Netzwerkpakete anhand von Regeln
- Vorteile
 - Sehr schnell
 - Einfach zu implementieren
 - Einfache Regeln
- Nachteile
 - Können Paketinhalt nicht bewerten
- Bsp: iptables

Application Firewalls

- Eigenschaften
 - Meistens Proxy Server
 - Überwachung nur auf Anwendungsebene
- Vorteile
 - Inhalte der übertragenen Daten werden überprüft
 - Zugriffskontrolle auf Inhalte (Webseiten)
- Nachteile
 - Hohe Spezialisierung
 - Aufwändige Implementierung und Wartung
 - Kein Ersatz für Paketfilter
 -
- Beispiel: FWTK

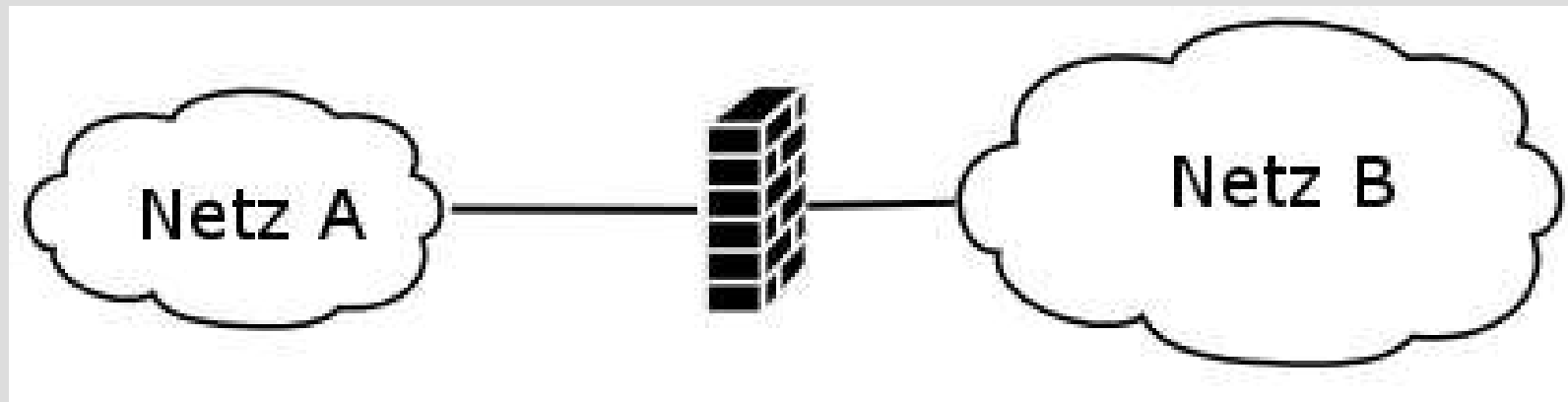
Policy

- Eigenschaften:
 - Implementierung der Firewall
 - Geordnete Menge von Regeln
 - Erste erfüllende Regel wird angewendet
 - Falls Regeln nicht terminieren, wird die nächste passende gesucht

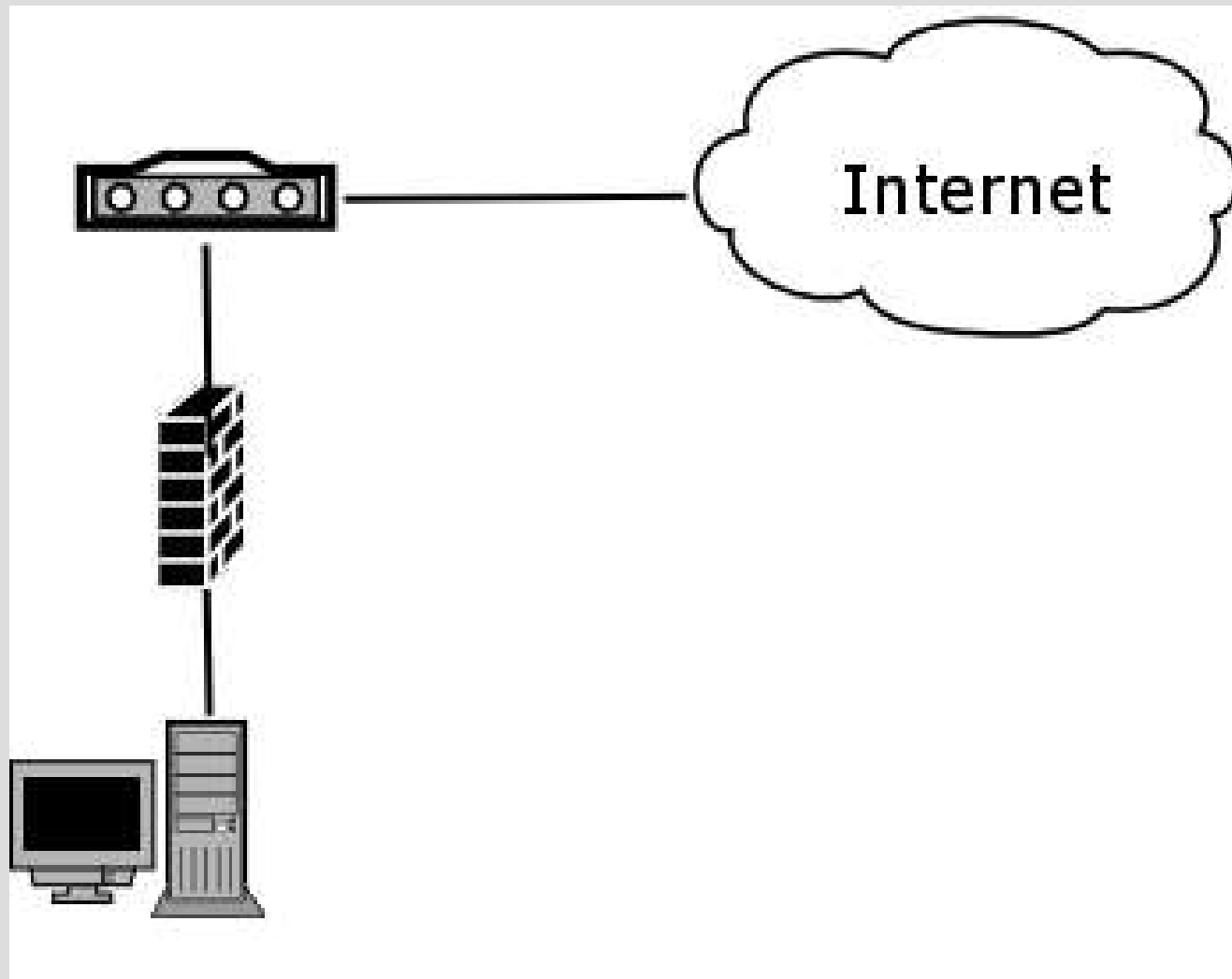
Nachteile / Einschränkungen

- Nur eine von vielen Sicherheitsvorkehrungen
- Kann nicht vor Angriffen von “innen” schützen
- Möglicher Engpass
- Wählverbindungen von einzelnen Rechnern können nicht zentral überwacht werden
- Ausfallsicherheit

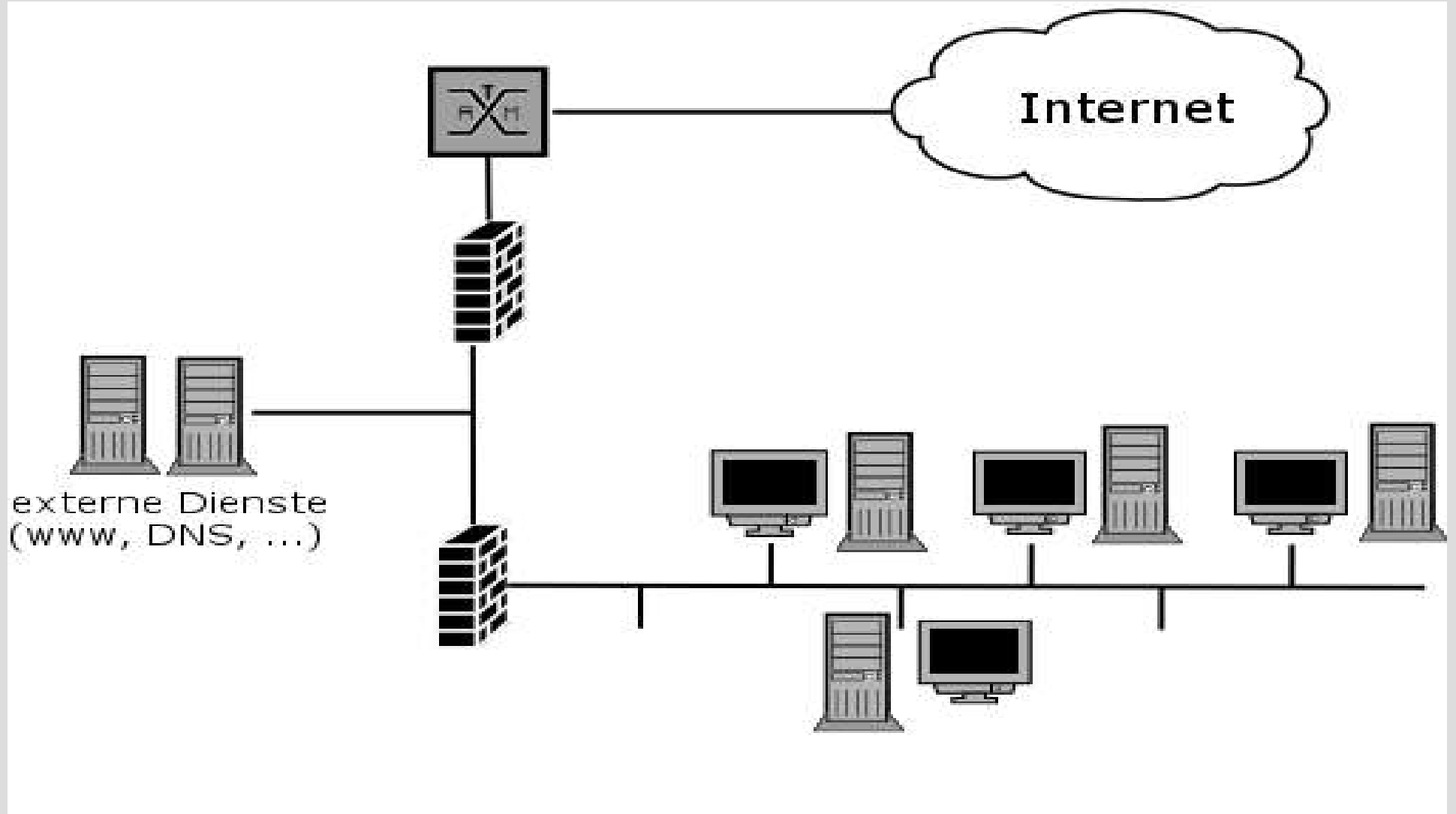
Firewall



Beispiele



Beispiele



Linux Netfilter

- Neu seit Kernel 2.4
- Bietet Infrastruktur zur Paketbehandlung
- Unterstützt IPv4, IPv6, DECnet
- Entwickelt von Paul “Rusty” Russel
- Betreut vom Netfilter Core Team
- www.netfilter.org

Linux Socket Buffer `sk_buff`

- Repräsentation von Netzwerkpaketen im Kernel
- Enthält Paketdaten und Protokolldaten
- Erlaubt flexible Handhabung der Daten

Die sk_buff Struktur

- next / prev - Listenzeiger
- list - Zeiger auf die aktuelle Liste
- sk - Zeiger auf den erzeugenden Socket
- dev - Netzwerkschnittstelle
- real_dev- Erweiterung für dev (VLAN)
- h - Transport Layer Header; h.th für TCP; h.uh für UDP
- nh - Networklayer; nh.ipv4 für Ipv4
- mac - Link Layer Header
- dst - Informationen über den weiteren Weg durch den Stack
- cb - Control Buffer (privat)
- len, data_len - Paketgrößen
- csum - Checksummen Informationen
- protokol - Ethernet ProtokollID (IP 0x0800)
- data, tail - Zeiger auf Beginn und Ende des Datenbereichs
- head, tail - Zeiger auf Beginn und Ende des Pufferbereichs
- nfct - Netfilter Conntrack Informationen
- nfmark, nfcache - Felder werden von Netfilter benutzt

Netfilter Hooks

- Kontrollpunkte im Netzwerk-Kode
- Jedes Paket muss mind. einen passieren
- Dazu gibt es noch Funktionen zur Verwaltung

Netfilter Hooks 2

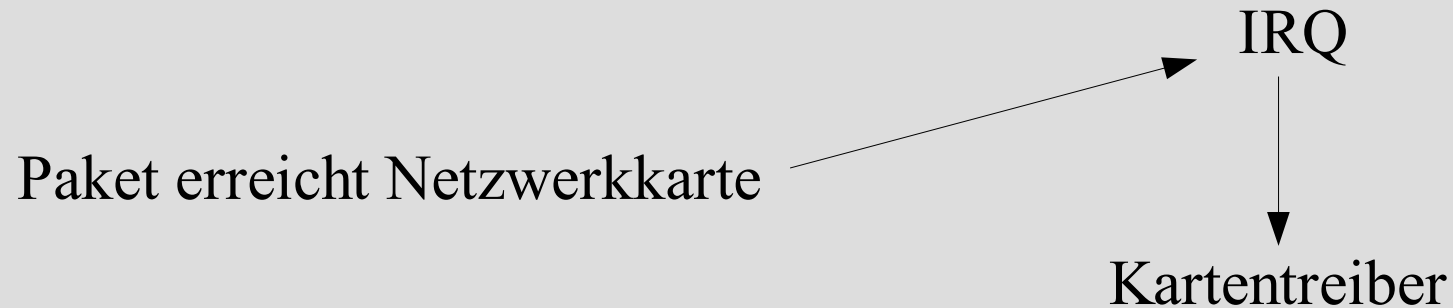


1. NF_IP_PRE_ROUTING
2. NF_IP_LOCAL_IN
3. NF_IP_FORWARD
4. NF_IP_POST_ROUTING
5. NF_IP_LOCAL_OUT

Netfilter Hooks 3

- um Netfilter Hooks zu nutzen, registriert man callback-Funktionen
- diese Funktionen bekommen alle Pakete, die den Hook passieren
- Funktion muss einen dieser Werte zurückgeben:
 - `NF_ACCEPT`
 - `NF_DROP`
 - `NF_STOLEN`
 - `NF_QUEUE`
 - `NF_REPEAT`
- siehe Beispiel Modul:
<http://www.informatik.uni-freiburg.de/~rechert/traffic.c>

Reise durch den Linux Kernel 1



`netif_rx()`

`sk_buff` wird in die Liste eingehender Pakete eingefügt

- Interrupts abgeschaltet
- erzeugt neuen `sk_buff`
- schreibt Daten in `sk_buff`
- `netif_rx()`
- Interrupts wieder eingeschaltet

Softirq NET_RX

Reise durch den Linux Kernel 2

Softirq NET_RX

Kernel Scheduler
wenn Softirq NET_RX

net_rx_action()

- entnimmt einen sk_buff der Paketliste
- sucht in ptype_all und ptype_base einen Protokollhandler

**Protokollhandler
Ipv4**

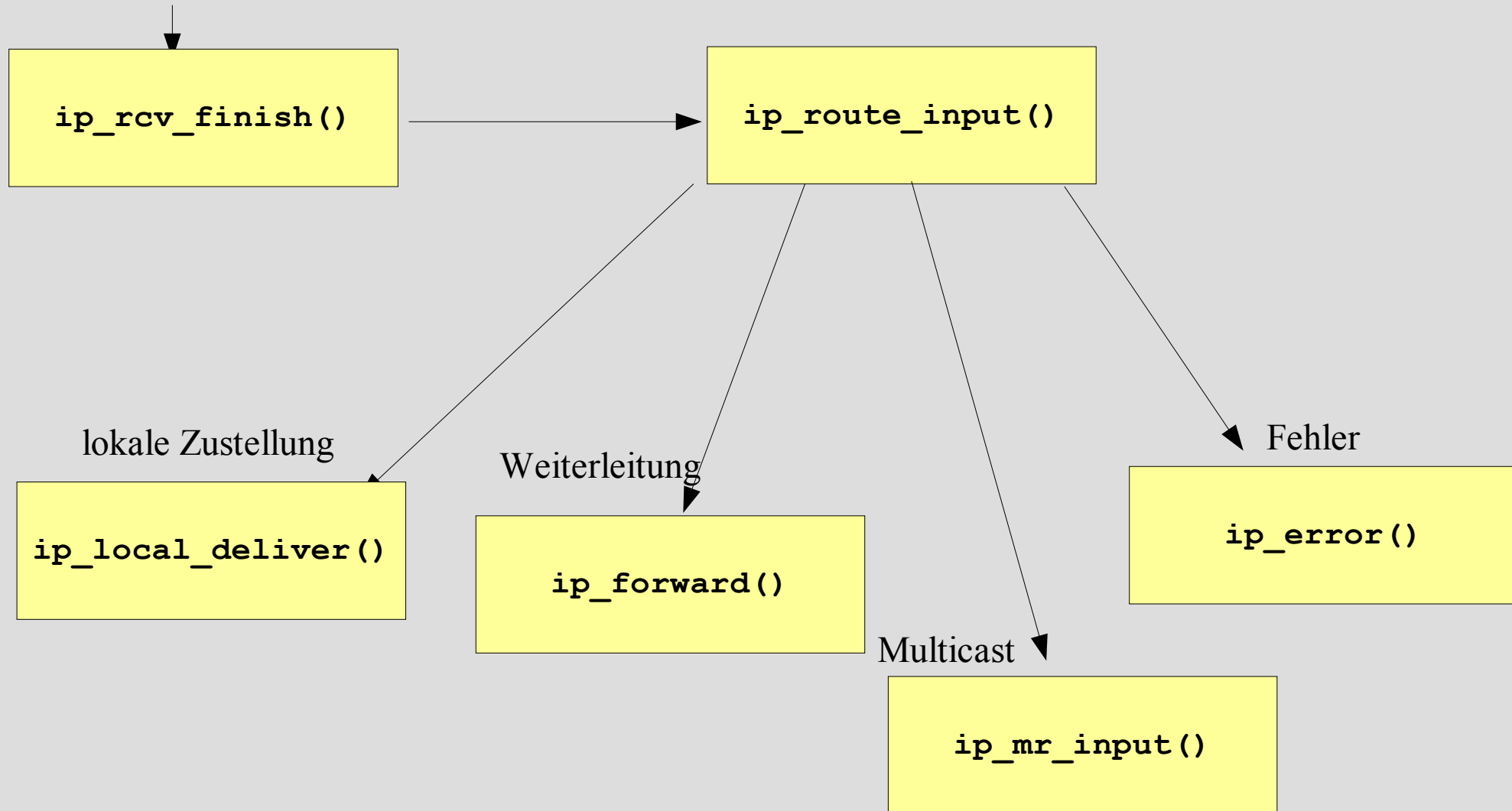
ip_rcv()

ip_rcv()

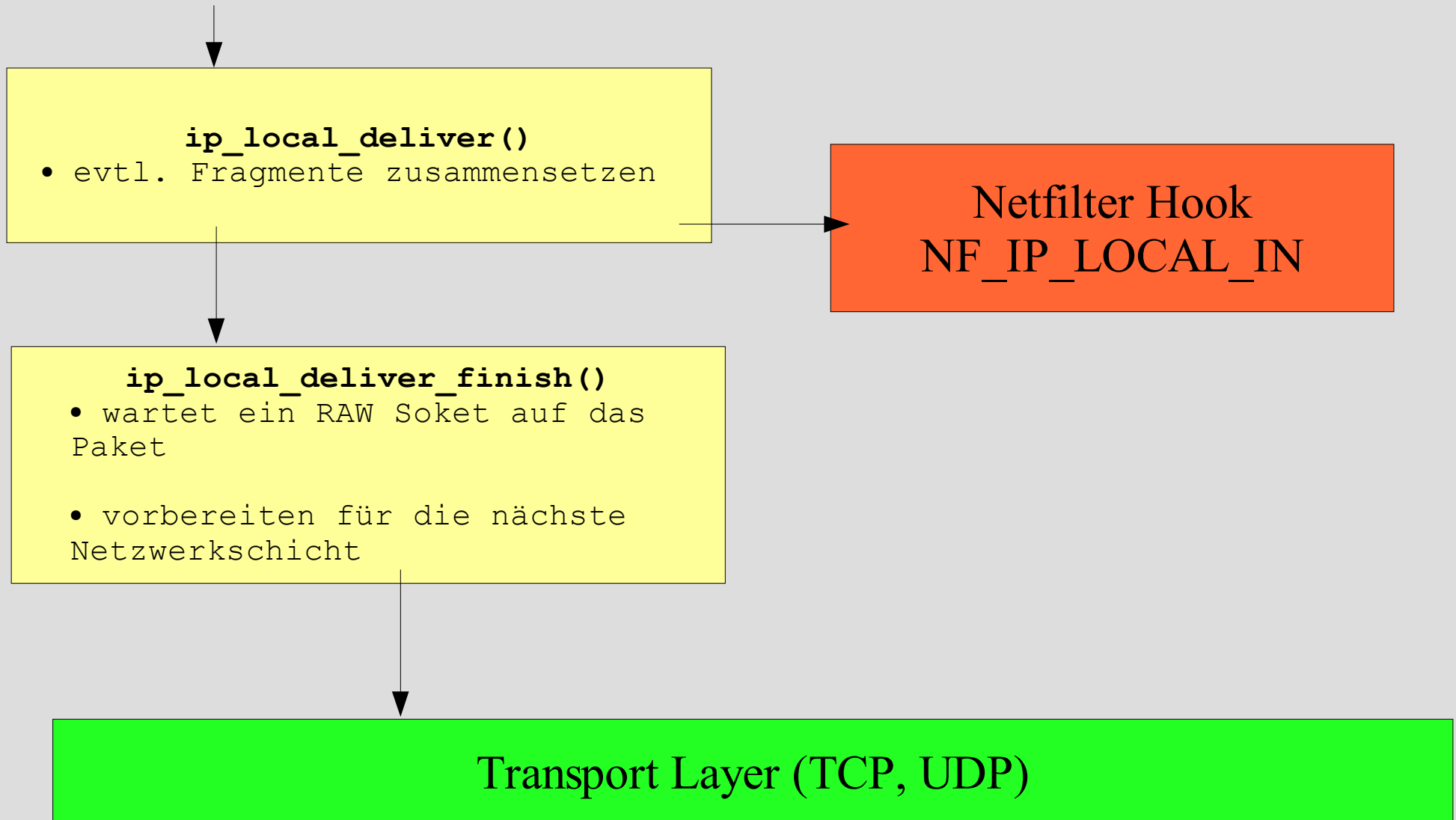
- ist das Paket für diesen Host
- Header und Checksumme werden berechnet

**Netfilter Hook
NF_IP_PREROUTING**

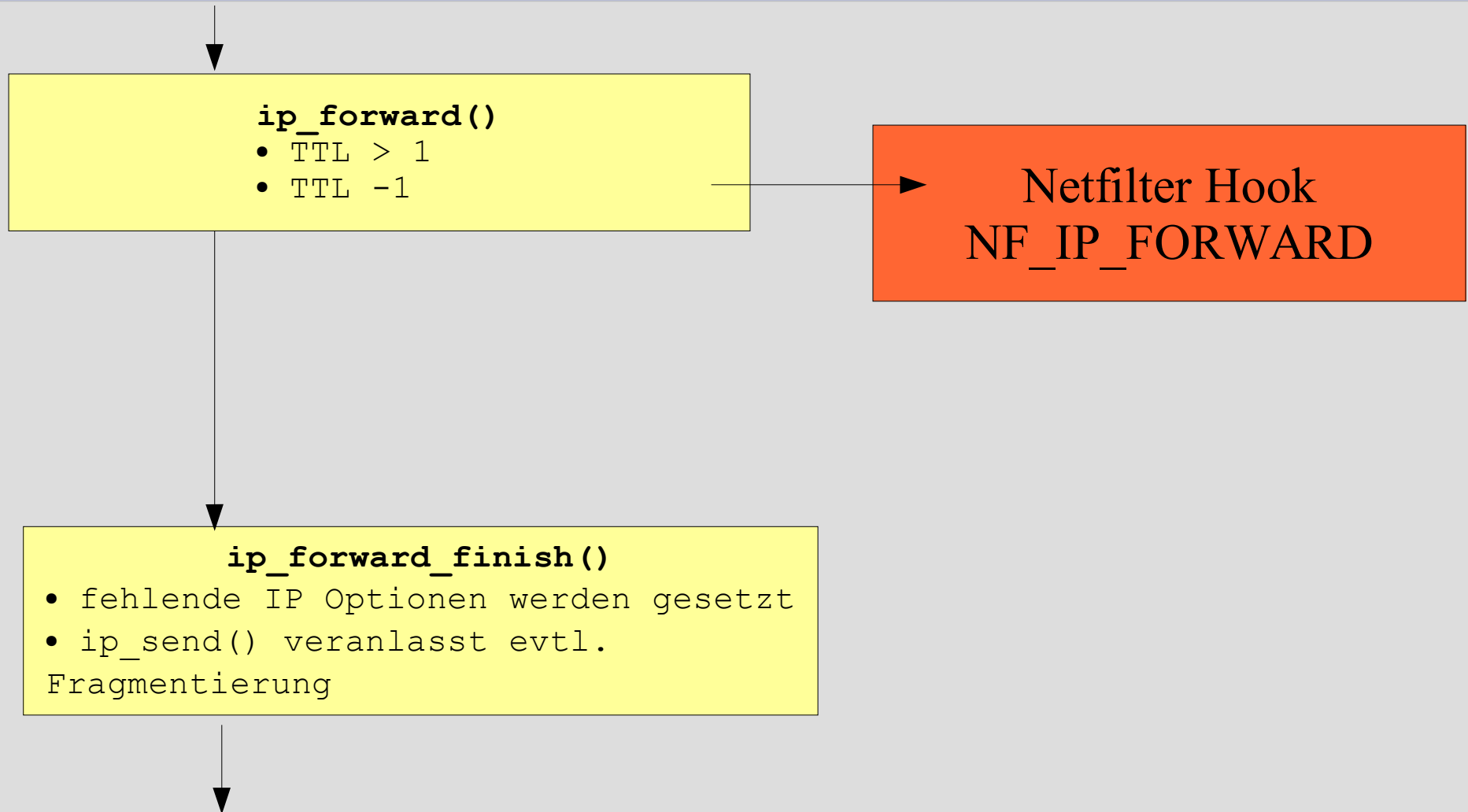
Reise durch den Linux Kernel 3



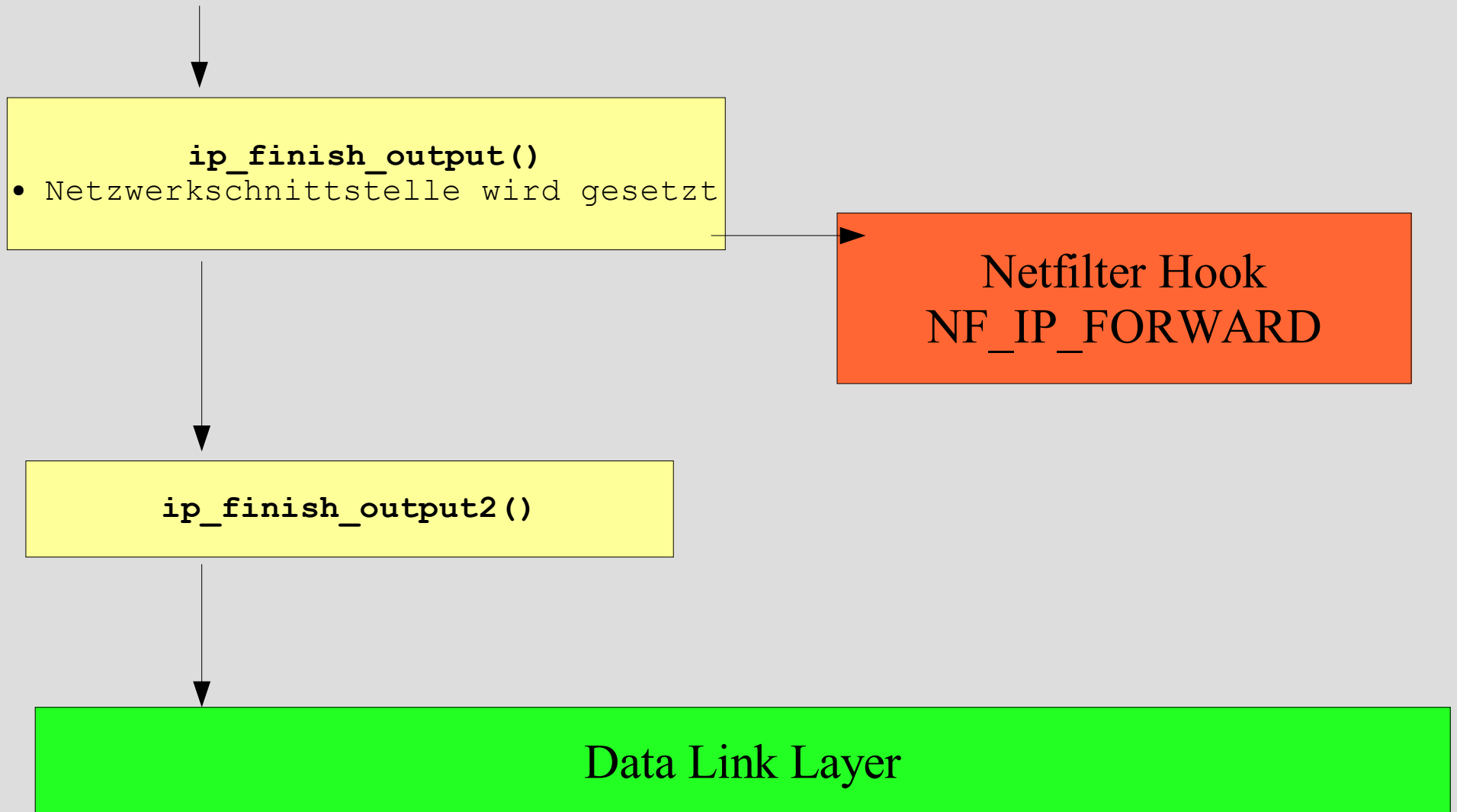
Lokale Zustellung



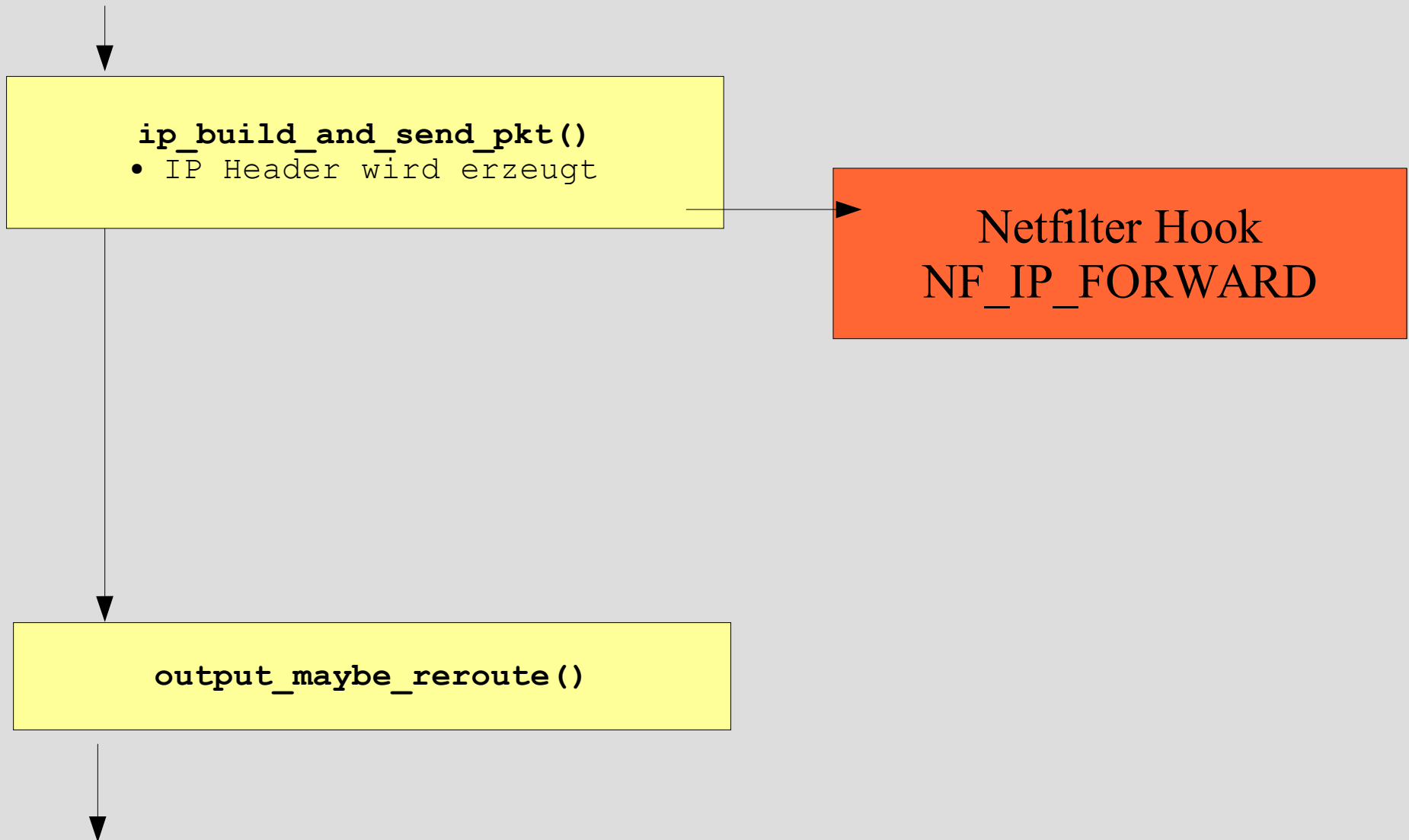
Weiterleitung



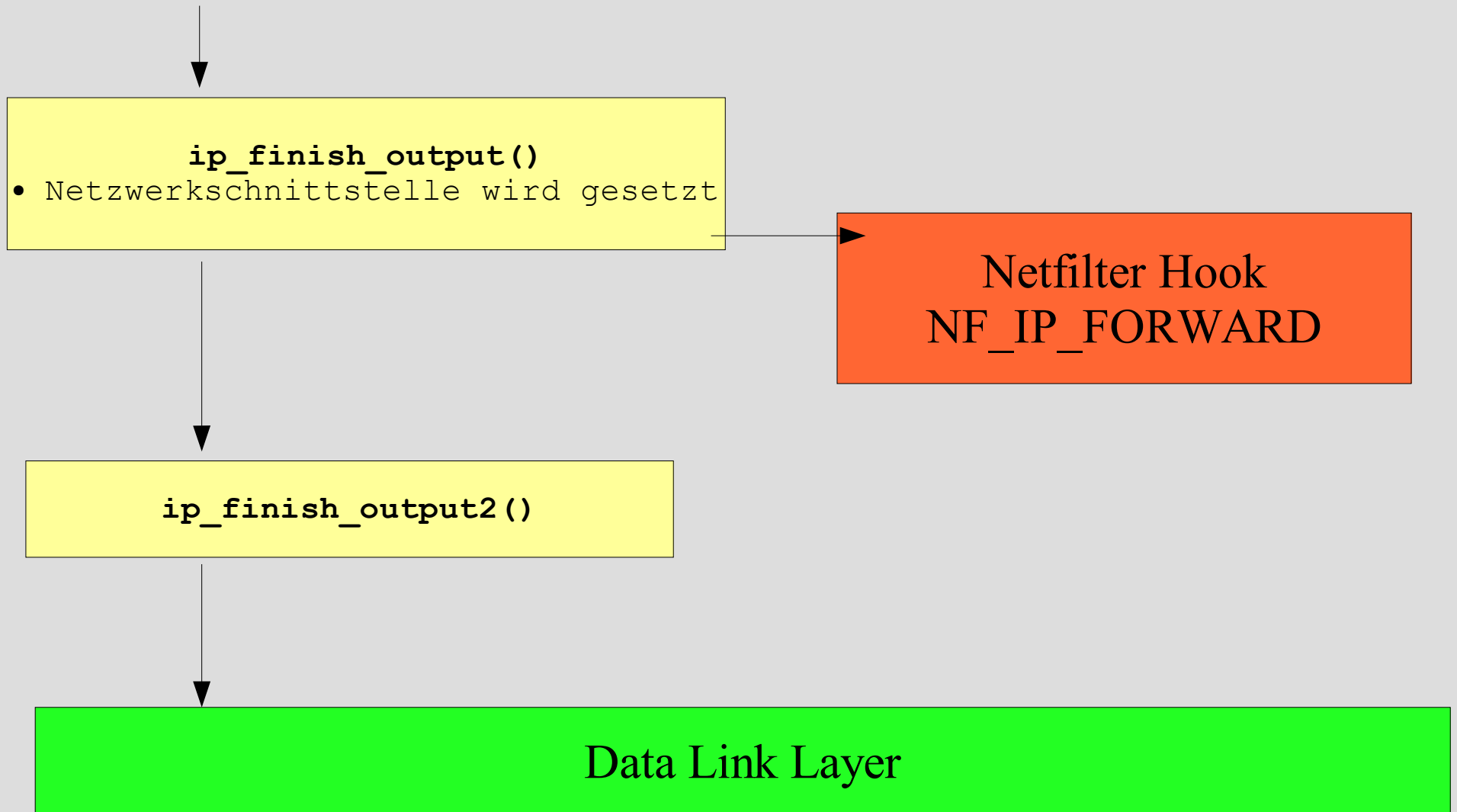
Weiterleitung ff.



Lokal generierte Pakete



Lokal generierte Pakete ff.



iptables

- Infrastruktur zur Paketauswahl
- Tabellen aus Regel-Ketten (chains)
- eine Kette für jeden Netfilter Hook
- drei Tabellen werden automatisch erzeugt
 - i. filter* -Tabelle – Paketauswahl
 - ii. nat* -Tabelle – Network Translation Table
 - iii. mangle* -Tabelle – allgemeine Behandlung, Paketmanipulation

iptables

- iptables ist auch ein Programm zur Verwaltung von Regeln
- Eine Regel besteht aus:
 - match(es)
 - (höchstens ein) targetRegeln können nur ein *match* oder nur ein *target* enthalten. (siehe Beispiele)

iptables Regel Syntax:

```
iptables -t table -operation chain -j target match(es)
```

Beispiele

```
iptables -t filter -A INPUT -j ACCEPT -p tcp -dport ssh
```

```
iptables -t filter -A INPUT -j DROP
```

```
iptables -t filter -A OUTPUT -m owner --pid 741
```

```
iptables -L -v
```

filter-Tabelle

- Tabelle zur allg. Paketauswahl
- Paket kann nur an einer Stelle gefiltert werden
je nach Ursprung

NF_IP_LOCAL_IN = INPUT chain

NF_IP_LOCAL_OUT = OUTPUT chain

NF_IP_FORWARD = FORWARD chain

nat-Tabelle

- Zwei Arten von NAT
 1. SNAT (Source Address NAT)
MASQUERADE als spezielles Beispiel
 2. DNAT (Destination Address NAT)
REDIRECT als spezielles Beispiel
- Nur das erste Paket einer Verbindung muss die Tabelle durchlaufen

SNAT

Adressänderung der Absenderadresse

Beispiel:

```
iptables -t nat -A POSTROUTING -j SNAT --to-source \  
10.0.0.1 -o eth0
```

```
iptables -t nat -A POSTROUTING -j MASQUERADE -o ppp0
```

DNAT

Änderung der Zieladresse

Beispiel:

```
iptables -t nat -A PREROUTING -j DNAT --to-destination \
  10.0.0.1:8080 -p tcp --dport 80 -i eth0
```

```
iptables -t nat -A PREROUTING -j REDIRECT --to-port \
  8080 -i eth0 -p tcp --dport 80
```

mangle-Tabelle

Tabelle zur Paketmanipulation

Beispiel:

```
iptables -t mangle -A OUTPUT -j DSCP 1 -p tcp \  
--dport 22
```

Connection Tracking

- dient der Überwachung von Verbindungen
- nutzt LOKAL_OUT und PRE_ROUTING Hook

Mögliche Zustände einer Verbindung:

IP_CT_NEW

IP_CT_ESTABLISHED

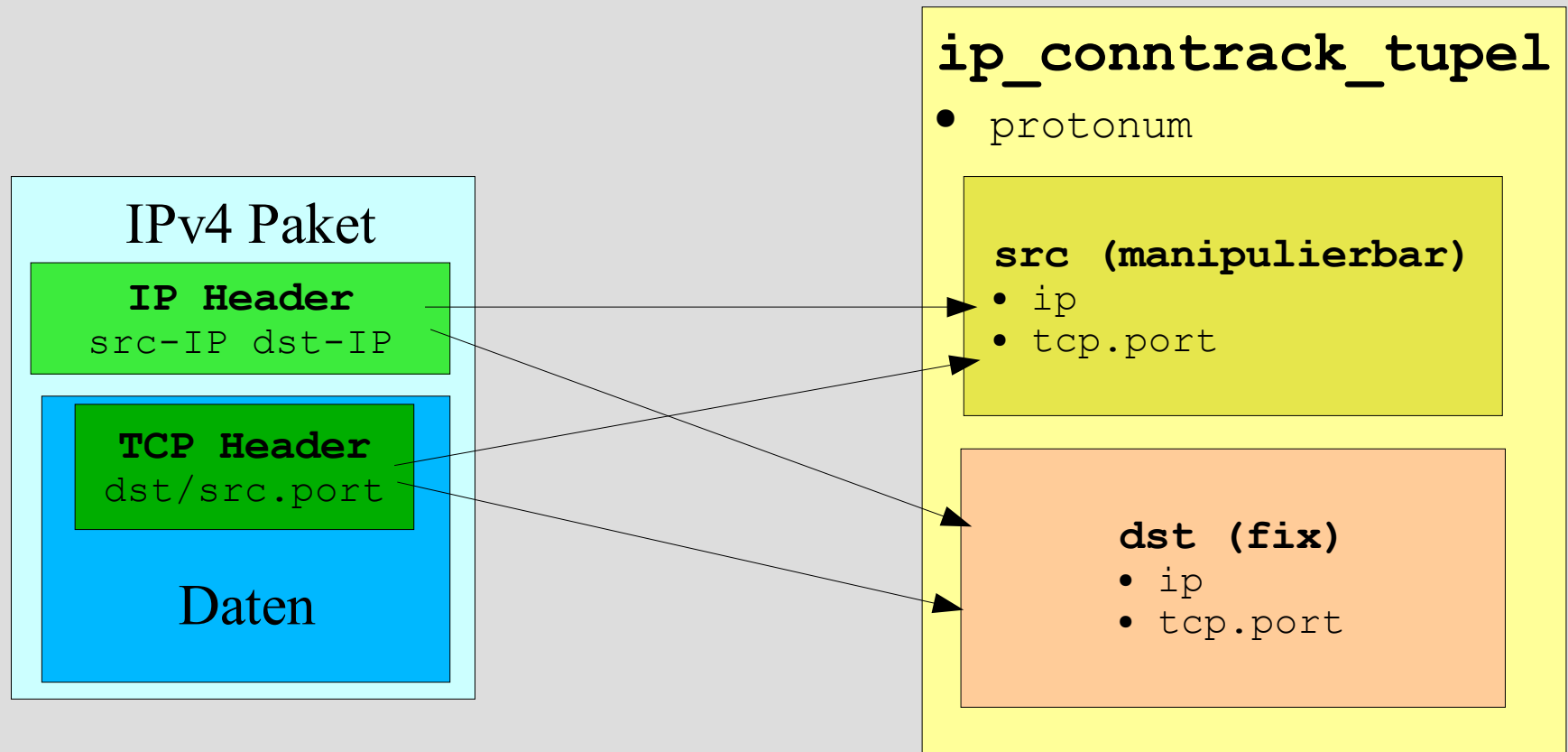
IP_CT_ESTABLISHED + IP_CT_IS_REPLY

IP_CT_RELATED

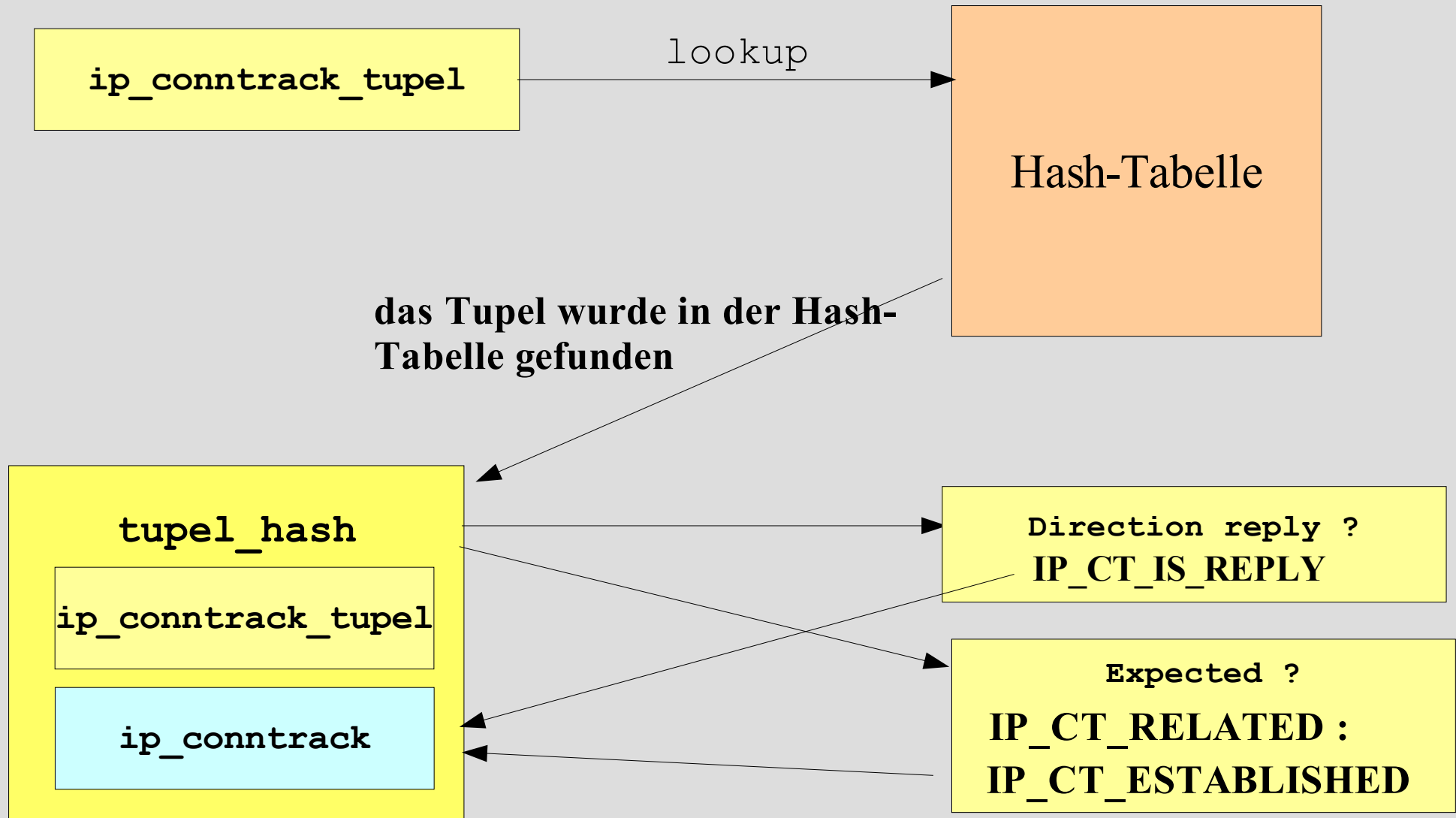
IP_CT_RELATED + IP_CT_IS_REPLY

IP_CT_INVALID

Connection Tracking ff.

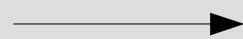


Connection Tracking ff.



Connection Tracking ff.

das Tupel wurde nicht
gefunden

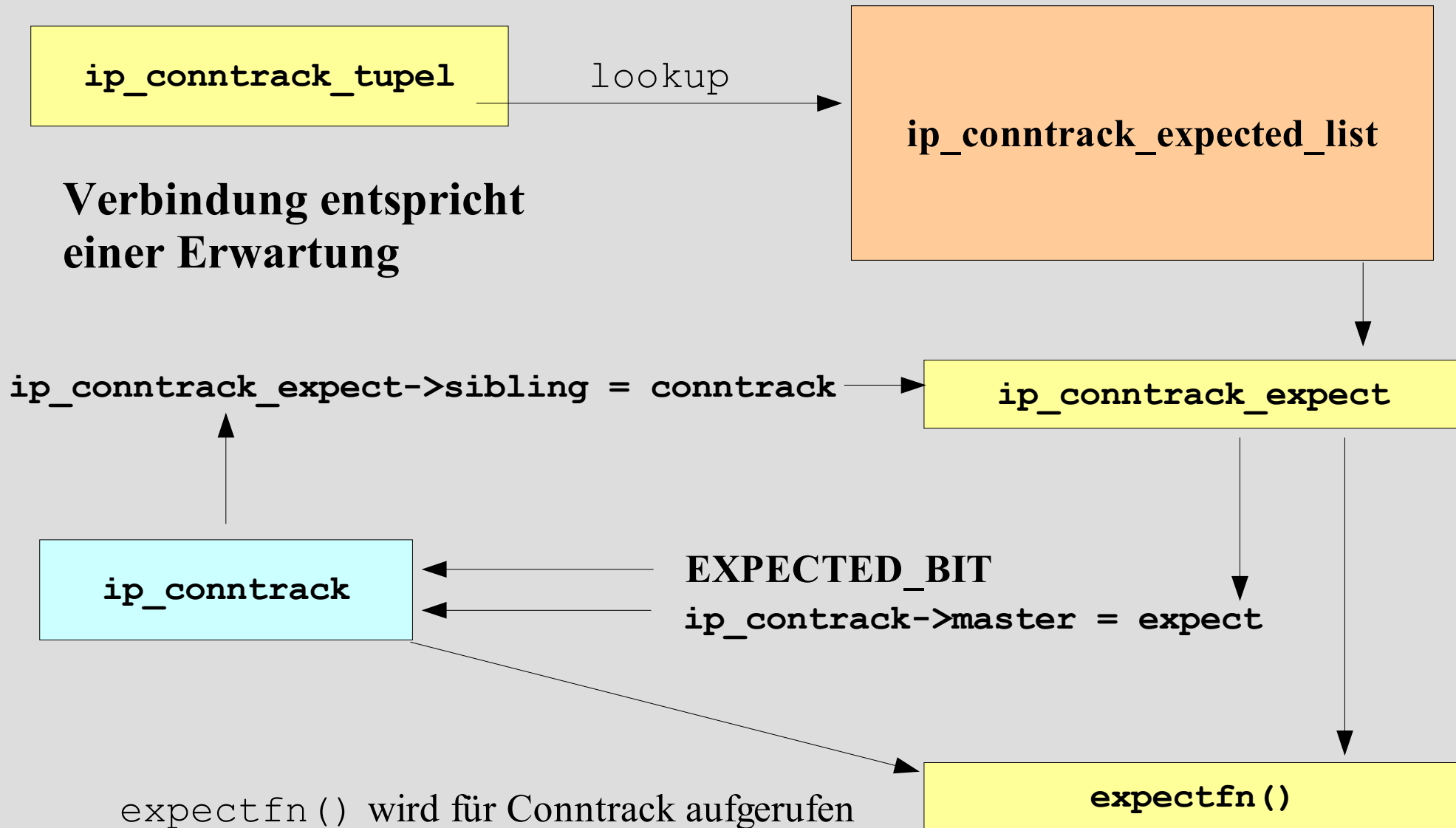


neue Conntrack Struktur muss angelegt
werden

ip_conntrack

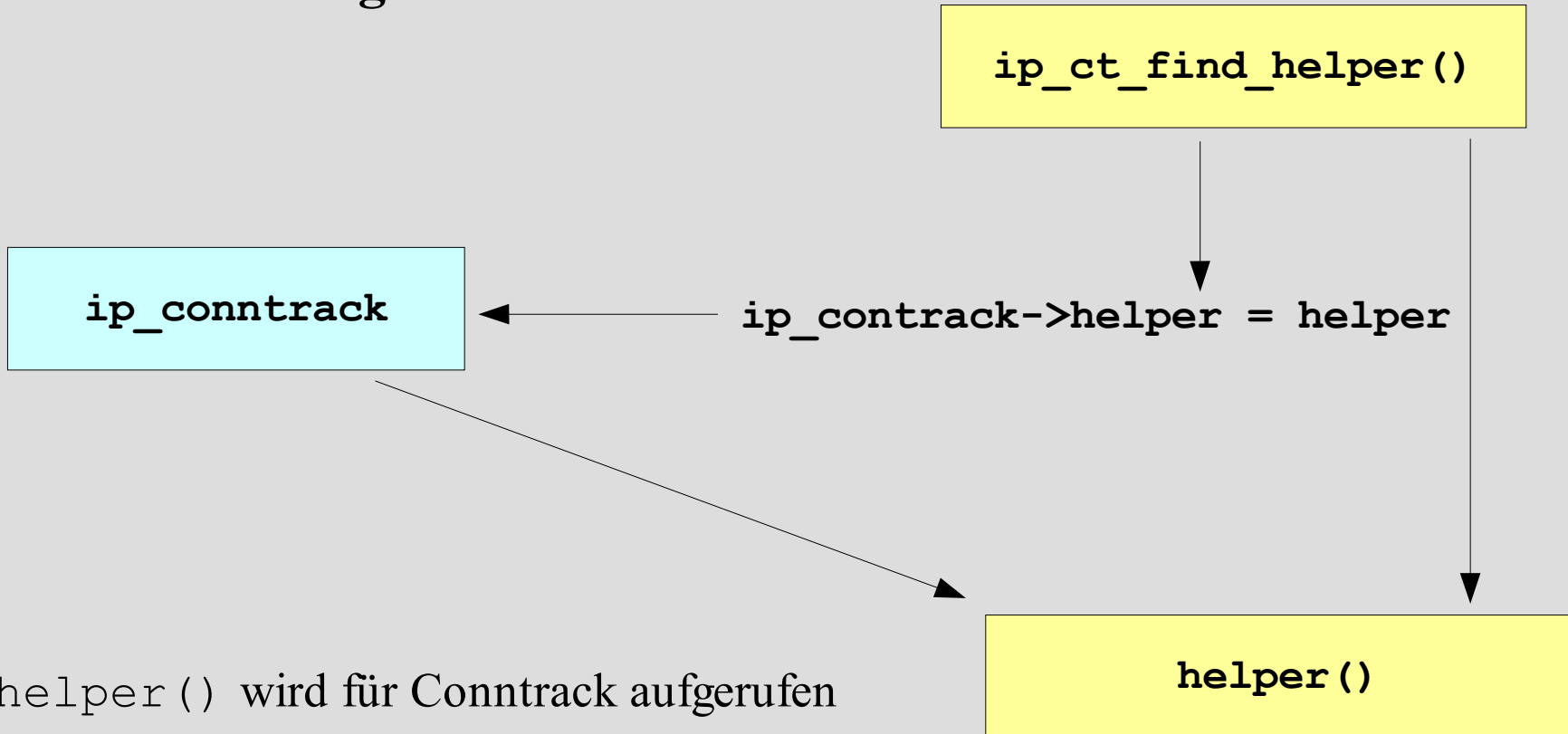
- **ct_general** (usage Counter)
- **tupelhash[]** (eines für jede Richtung)
- **status** (Status Flags)
- **expected_list** (Liste aller *expectations*)
- **timeout** (Timeout der Verbindung)
- **sibling_list** (Listenkopf für weitere *exp.*)
- **expecting** Anzahl *expectations*
- **master** *expectation*
- **helper** *Helper* Struktur
- **infos[]** ein Feld für jede Bewertung
- **proto** Protokoll
- **help** private Daten für *Helper*

Connection Tracking ff.



Connection Tracking ff.

keine Erwartung



`helper()` wird für Contrack aufgerufen

Connection Tracking ff.

- Nach eingegangenem ACK für das Paket bekommt der Conntrack-Eintrag das Bit “ASSURED”
- Das `tuple_hash` wird erst in die Tabelle eingetragen, wenn das Paket LOCAL_IN oder POST_ROUTING passiert hat.
- Die Hash-Tabelle enthält immer zwei `tuple_hash` pro Verbindung

Stateful Firewall

- Stateful Firewalling wird möglich durch Connection Tracking
- Erlaubt:
 - Verbindungsaufbau von “innen”
 - Verbindungen von “innen” nach “ausen”
 - Bestehende Verbindungen aller Richtungen
- Beispiel

```
iptables -A FORWARD -j ACCEPT -m state --state \
ESTABLISHED,RELATED
```

Conntrack Helper

- Erweitern die Fähigkeiten des Connection Trackings
- Assoziation zwischen ESTABLISHED und RELATED Verbindungen
- Registrieren Erwartungen (*expectation*)

Conntrack Helper

`ip_conntrack_helper`

- **list** (Listenzeiger)
- **name** (Name des Helpers)
- **flags**
- **me** (*this module*)
- **max_expected** (max Anz. gleichz. erw. Verb.)
- **timeout**
- **tupel** (*ip_conntrack_tupel*)
- **mask** (*ip_conntrack_tupel*)
- **help** (Funktionszeiger)

`ip_conntrack_expect`

- **list** (Listenzeiger)
- **use** (RefCount)
- **expected_list** (Liste aller Erwartungen)
- **expectand** Master
- **sibling** (Zeiger auf erw. Verbindung)
- **ct_tupel**
- **timeout**
- **tupel** (*ip_conntrack_tupel*)
- **mask** (*ip_conntrack_tupel*)
- **expectfn** (Funktionszeiger)
- **seq** (Sequenznummer)
- **proto** (Protokoll)
- **help** (Helper Struktur)

NAT Helper

- Erweitert die Möglichkeiten der Adressmanipulation

ip_nat_helper

- **list** (Listenzeiger)
- **name** (Name der Helpers)
- **flags**
- **me** (*this module*)
- **max_expected** (max Anz. gleichz. erw. Verb.)
- **timeout**
- **tupel** (*ip_conntrack_tupel*)
- **mask** (*ip_conntrack_tupel*)
- **help** (Funktionszeiger)
- **expect** (Funktionszeiger)

Beispiel FTP

Client

PORT h1,h2,h3,h4,p1,p2



Gateway

Server

Beispiel FTP

Client

Conntrack Helper

```
PORT h1,h2,h3,h4,p1,p2
```



Gateway

Server

Beispiel FTP

Client

Conntrack Helper

PORT h1,h2,h3,h4,p1,p2

Expectation



Gateway

Server

Beispiel FTP

Client

Nat - Helper

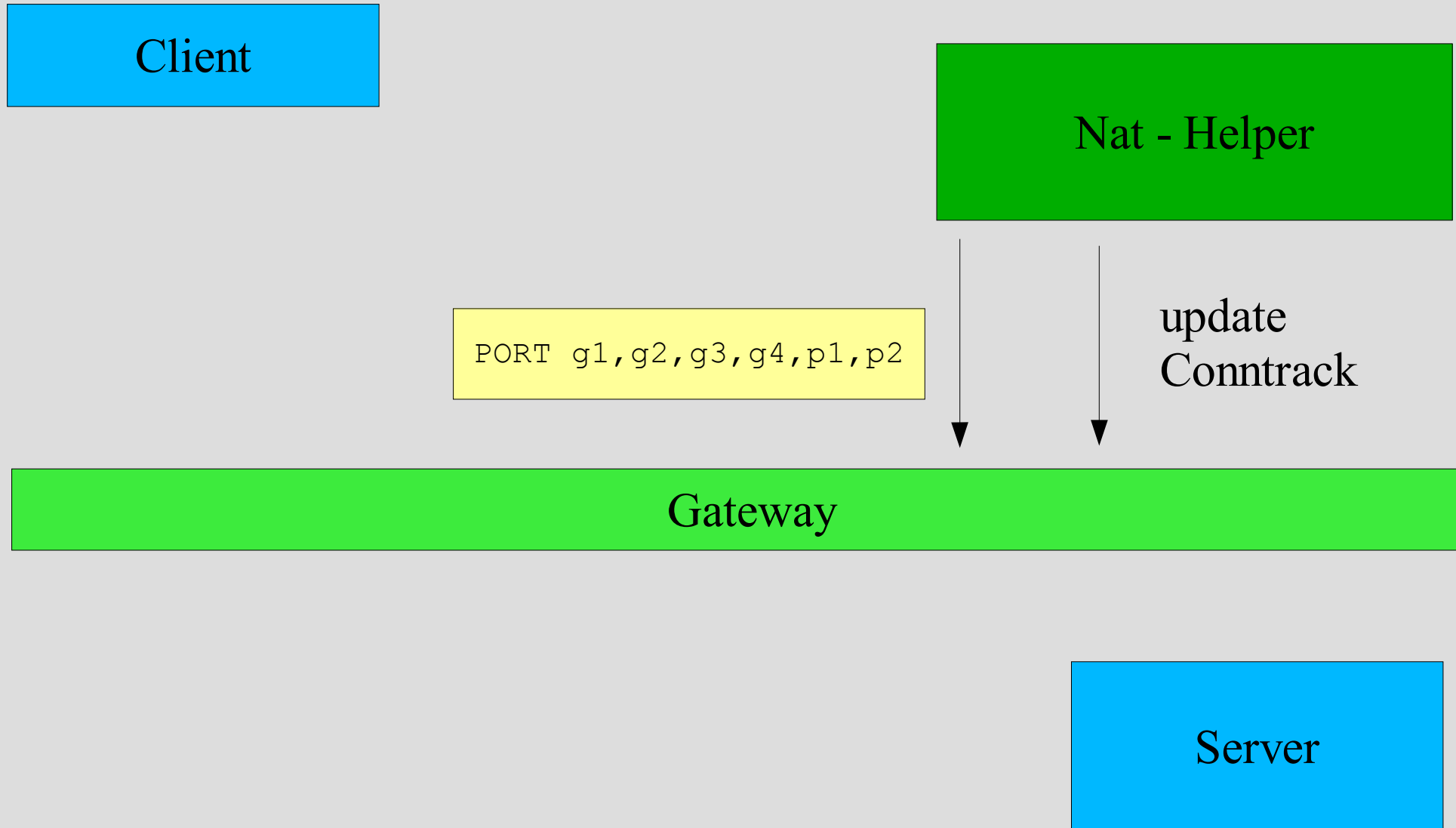
PORT h1,h2,h3,h4,p1,p2

Gateway

Server



Beispiel FTP



Beispiel FTP

Client

Gateway

PORT g1,g2,g3,g4,p1,p2

Server



Beispiel FTP

Client

Gateway

DATA g1,g2,g3,g4,p1,p2

Server



Beispiel FTP

Client

Conntrack Helper

DATA g1,g2,g3,g4,p1,p2

Gateway

Server



Beispiel FTP

Client

Conntrack Helper

RELATED

DATA g1,g2,g3,g4,p1,p2

Gateway

Server



Beispiel FTP

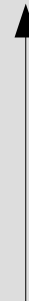
Client

Nat - Helper

DATA g1,g2,g3,g4,p1,p2

Gateway

Server



Beispiel FTP

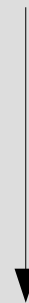
Client

Nat - Helper

DATA h1,h2,h3,h4,p1,p2

Gateway

Server



Beispiel FTP

Client

DATA h1,h2,h3,h4,p1,p2

Gateway

Server



iptables Module

- *owner*
- *length*
- DSCP