

Ralf Spenneberg

# Linux-Firewalls mit iptables & Co.

Sicherheit mit Kernel 2.4 und 2.6  
für Linux-Server und -Netzwerke



 ADDISON-WESLEY

---

An imprint of Pearson Education

München • Boston • San Francisco • Harlow, England  
Don Mills, Ontario • Sydney • Mexico City  
Madrid • Amsterdam



# 19 Connection Tracking

Das Connection Tracking ist eine wesentliche Funktion der Iptables/Netfilter-Firewall. Deshalb gebührt dieser Funktion auch ein eigenes Kapitel. Da diese Funktionalität aber für viele andere Funktionen (zustandsorientiertes Filtern, Network Address Translation) benötigt wird, wurden in den entsprechenden Kapiteln bereits ausführlich verschiedene Aspekte betrachtet. Hier soll es daher nur um die fortgeschrittene Konfiguration des Connection Tracking gehen.

## 19.1 Connection Tracking – Überblick

Das Connection Tracking ist verantwortlich für die Überwachung der Netzwerkverbindungen durch die Iptables-Firewall. Das Connection Tracking kann TCP-, UDP-, ICMP- und alle anderen Protokolle überwachen. Hierzu erkennt das Connection Tracking, aus welcher Richtung das erste Paket kommt. Dieses Paket erhält den virtuellen Zustand `NEW`. Wenn das Paket von den Firewall-Regeln akzeptiert wird, wird die Verbindung auch in der Zustandstabelle eingetragen. Das zweite Paket muss aus der anderen Richtung kommen. Nur dann erhalten das zweite Paket und die Verbindung in der Zustandstabelle den virtuellen Zustand `ESTABLISHED`. Alle weiteren Pakete erhalten unabhängig von ihrer Richtung den Zustand `ESTABLISHED`. Diese Funktion kann genutzt werden, um Verbindungen nur in bestimmten Richtungen zu erlauben.

Das Connection Tracking unterscheidet insgesamt 5 verschiedene Zustände:

- `NEW`: Ein Paket ist neu, wenn die dazugehörige Verbindung noch nicht in der Verbindungstabelle existiert und das Paket bestimmte andere Eigenschaften hat, die für ein neues Paket verlangt werden. Diese Eigenschaften sind von dem Protokoll abhängig und werden weiter unten besprochen.
- `ESTABLISHED`: Ein Paket hat den Zustand `ESTABLISHED`, wenn die Verbindung bereits in der Verbindungstabelle existiert und mindestens ein Antwortpaket bereits von dem Connection Tracking gesehen wurde oder dieses Paket das erste Antwortpaket ist.
- `RELATED`: Ein Paket ist mit einer Verbindung verwandt, wenn es, obwohl es ein anderes Protokoll, eine andere IP-Adresse und andere Ports verwendet, sich auf eine Verbindung bezieht, die den Zustand `ESTABLISHED` hat. Dies können ICMP-Fehlermeldungen sein oder Verbindungen, die von einem Connection Tracking-Helfermodul hinzugefügt wurden. Auf modernen Kernen werden die letzten

Verbindungen in einer eigenen Verbindungstabelle `/proc/net/ip_conntrack_expect` verwaltet.

- **INVALID:** Falls ein Paket zu keiner der in der Verbindungstabelle aufgeführten Verbindungen passt, aber dennoch nicht die Anforderungen für eine neue Verbindung beim verwendeten Protokoll erfüllt, ist es ungültig. Dies können zum Beispiel ICMP-Fehlermeldungen sein, die sich auf eine in der Verbindungstabelle nicht vorhandene Verbindung beziehen.
- **UNTRACKED:** Wenn Ihr Kernel über die `raw`-Tabelle verfügt, kennt das Connection Tracking auch den Zustand `UNTRACKED`. Das bedeutet, dass diese Verbindung nicht von dem Connection Tracking überwacht wird und daher auch nicht in der Verbindungstabelle auftaucht.

Da die verschiedenen Protokolle unterschiedliche Arten der Verbindung unterstützen und diese unterschiedlich auf- und abbauen, unterscheidet das Connection Tracking im Linux-Kernel auch diese Protokolle in der Behandlung.

### 19.1.1 Das TCP-Connection Tracking

Bei dem TCP-Connection Tracking gibt es zwei Varianten. Dieser Abschnitt bespricht die alte Version, die bis Kernel 2.6.8 einschließlich genutzt wurde. Die neue Variante wird im Abschnitt TCP-Window-Tracking (siehe Abschnitt 19.3) besprochen.

Das alte TCP-Window-Tracking erkennt eine neue Verbindung daran, dass die verwendeten IP-Adressen und Ports des Pakets mit keiner der in der Verbindungstabelle eingetragenen Verbindungen übereinstimmen. Zusätzlich muss das Paket mit dem Zustand `NEW` entweder ein `SYN`-, ein `ACK`- oder ein Null-Paket sein. Ein Null-Paket hat kein TCP-Flag gesetzt.

Damit die Verbindung anschließend in den Zustand `ESTABLISHED` versetzt wird, muss auf ein `SYN`-Paket immer ein `SYN/ACK`-Paket aus der entgegengesetzten Richtung folgen. Wurde die Verbindung mit einem `ACK`-Paket im Zustand `NEW` begonnen, so befindet sich die Verbindung anschließend direkt in dem Zustand `ESTABLISHED`, ohne dass ein Antwort-Paket erforderlich ist. Dies erlaubt die Wiederaufnahme von Verbindungen nach einem Reboot der Firewall oder nach einem Failover in einem Firewall-Cluster. Weitere Informationen hierzu finden Sie im Abschnitt 26.3, »Hochverfügbarkeit bei zustandsorientierten Firewalls«.

Die aufgebaute Verbindung verbleibt für bis zu 5 Tage in der Verbindungstabelle, wenn keine weiteren Pakete von dem Connection Tracking erkannt werden. Jedes weitere Paket setzt die Lebensdauer des Eintrags in der Tabelle wieder auf 5 Tage zurück. Sobald das Connection Tracking ein `TCP-RST`-Paket oder ein `TCP-FIN`-Paket erkennt, wird die Verbindung auch aus der Zustandstabelle entfernt. Um diese verschiedenen Zustände sauber trennen zu können, verwaltet das Connection Tracking intern die Zustände viel detaillierter und versieht die einzelnen Zwischenzustände auch mit unterschiedlichen Timeouts. Diese verschiedenen Zwischenzustände werden im Abschnitt 19.4.14, »TCP-Zustände«, besprochen.

### 19.1.2 Das UDP-Connection Tracking

Das Protokoll UDP unterscheidet sich von dem Protokoll TCP dadurch, dass dieses Protokoll keine Verbindungen kennt. Daher gibt es auch keine Signalisierung eines Verbindungsaufbaus oder -abbaus in dem UDP-Protokoll. Die Überwachung der Verbindung kann nur durch reine Beobachtung der Pakete und Verwendung von Timeouts erfolgen.

Das Connection Tracking erkennt eine neue Verbindung daran, dass das Paket IP-Adressen und Portnummern verwendet, die zu keiner in der Tabelle gespeicherten UDP-Verbindungen passen. Ist das der Fall, erhält das Paket den Zustand `NEW` und die Verbindung wird, wenn die Regeln es erlauben, als neue Verbindung in die Tabelle eingetragen.

Da es durchaus Applikationen gibt, die über UDP lediglich Informationen versenden und nie eine Antwort erhalten, muss das Connection Tracking die Verbindung selbstständig nach einiger Zeit wieder entfernen. Eine Verbindung, für die das Connection Tracking keine Antwort-Pakete aus der entgegengesetzten Richtung gesehen hat, wird automatisch nach 30 Sekunden aus der Tabelle entfernt. Kommt das Antwort-Paket erst nach 31 Sekunden, so kommt es zu spät und wird nicht mehr als `ESTABLISHED` erkannt.

Erhält das Connection Tracking innerhalb der 30 Sekunden ein Antwort-Paket, so wird die Lebensdauer für die Verbindung in der Tabelle auf 180 Sekunden heraufgesetzt. Solange nun alle 180 Sekunden ein weiteres Paket mit identischen IP-Adressen und Portnummern ausgetauscht wird, verbleibt die Verbindung in der Tabelle. Ist das nicht der Fall, wird die Verbindung nach 180 Sekunden aus der Zustandstabelle entfernt.

### 19.1.3 Das ICMP-Connection Tracking

Auch beim ICMP-Protokoll gibt es ähnlich wie beim UDP-Protokoll keine echten Verbindungen. Jedoch ist zum Beispiel der Ping mit den Nachrichten Echo-Request und Echo-Reply einer Verbindung ähnlich. Das Connection Tracking behandelt daher alle ICMP-Request- und -Reply-Pakete auch entsprechend. Sobald das Connection Tracking ein Request-Paket erkennt, trägt es dieses mit seinen IP-Adressen, seiner Identifikations- und Sequenznummer in der Verbindungstabelle ein. Das Request-Paket erhält den Zustand `NEW` zugewiesen. Die Identifikationsnummer erlaubt es, die Request-Pakete verschiedener Ping-Befehle auseinander zu halten. Die Sequenznummer ermöglicht es, die Reply-Pakete den entsprechenden Request-Paketen zuzuordnen. Das Connection Tracking richtet in der Verbindungstabelle für jedes eindeutige Request-Paket eine Verbindung ein. Sobald nun das Connection Tracking ein Reply-Paket erkennt, prüft es die IP-Adressen, die Identifikationsnummer und die Sequenznummer und vergleicht diese mit den Verbindungen in der Tabelle. Stimmen diese mit einer Verbindung überein, so erhält das Paket den Zustand `ESTABLISHED` und die Verbindung wird aus der Tabelle entfernt, denn pro Echo-Request-Paket ist nur ein Reply-Paket erlaubt. Stimmen die Daten nicht mit einer bekannten Verbindung überein, so erhält das Reply-Paket den Zustand `INVALID`.

Damit die Verbindungen nicht unendlich lange in der Tabelle verbleiben, wenn kein Reply-Paket gesendet wird, erhalten diese Verbindungen eine Lebensdauer von 30 Sekunden. Das Reply-Paket muss innerhalb dieser 30 Sekunden erkannt werden, sonst entfernt das Connection Tracking die Verbindung aus der Tabelle.

Das ICMP-Protokoll wird aber auch dafür verwendet, um Fehler in TCP- und UDP-Verbindungen anzuzeigen. Diese Fehlermeldungen können von IP-Adressen versendet werden, die mit der eigentlichen Verbindung nichts zu tun haben (z.B. Router). Das Connection Tracking erkennt die Verbindung, auf die sich die ICMP-Fehlermeldung bezieht, an dem Inhalt des ICMP-Pakets. Eine ICMP-Fehlermeldung muss immer den kompletten IP-Header und die ersten 64 Bit der IP-Daten enthalten. Bei TCP- oder UDP-Paketen befinden sich hier die Ports. Diese Header werden von dem Connection Tracking ausgewertet. Existiert eine Verbindung in der Tabelle, auf die diese Informationen passen, so erhält das Paket den Zustand `RELATED` und die Verbindung, auf die sich die Fehlermeldung bezieht, wird aus der Tabelle entfernt.

### 19.1.4 Das Connection Tracking für alle weiteren Protokolle

Für alle weiteren Protokolle kennt das Connection Tracking mit einigen wenigen Ausnahmen (z.B. GRE bei Anwendung der PPTP-Patches) keine besonderen Regelungen. Sie werden alle gleich behandelt. Ein neues Paket erkennt das Connection Tracking an der Tatsache, dass sich in der Tabelle noch keine Verbindung befindet, die die gleichen IP-Adressen und das gleiche Protokoll verwendet. Sobald sich eine derartige Verbindung in der Tabelle befindet, werden alle weiteren Pakete, die identische IP-Adressen und dasselbe Protokoll verwenden, mit dem Zustand `ESTABLISHED` versehen. Die Verbindung bekommt ab dem ersten Paket die Lebensdauer von 600 Sekunden zugewiesen. Das bedeutet, dass entweder innerhalb von 600 Sekunden ein weiteres Paket dieser Verbindung erkannt werden muss oder die Verbindung aus der Tabelle entfernt wird. Jedes weitere Paket setzt die Lebensdauer wieder auf 600 Sekunden zurück.

## 19.2 Das `ip_conntrack`-Kernelmodul

Das `ip_conntrack`-Kernelmodul unterstützt beim Laden die Definition des Parameters `hashsize`. Um diesen Parameter zu verstehen, müssen wir uns zunächst mit der Verbindungstabelle beschäftigen. Die Verbindungstabelle wird als Hash-Tabelle angelegt, damit der Kernel schnell in der Tabelle suchen kann. Während Sie die Größe der Verbindungstabelle während der Verwendung der Tabelle ändern können, ist die Größe des Hashes fix und wird bei der Erzeugung der Verbindungstabelle festgelegt. Anschließend kann die Größe des Hashes nicht geändert werden.

Auf einem normalen Linux-System berechnet der Kernel die Größe der Verbindungstabelle und des Hashes aus der Arbeitsspeichergröße. Auf der Intel 32-Bit-Architektur wird die Größe der Verbindungstabelle wie folgt berechnet:

```
CONNTRACK_MAX = RAM / 16384
```

Auf einem 512-MByte-Rechner hat die Verbindungstabelle also eine Größe von maximal 32.768 Verbindungen. Diese Größe sinkt unabhängig von der Speichergröße nie unter 128, und auch auf Systemen mit mehr als 1 GByte Speicher beträgt dieser Wert maximal immer 65.536. Ist dieser Wert erreicht, werden weitere Verbindungen nicht mehr von der Firewall akzeptiert. Sie können diesen Wert aber manuell in der Variablen `ip_conntrack_max` setzen:

```
echo 128000 > /proc/sys/net/ipv4/ip_conntrack_max
```

Bei einigen Kernel müssen Sie alternativ auf diese Datei zugreifen:

```
echo 128000 > /proc/sys/net/ipv4/netfilter/ip_conntrack_max
```

Den aktuellen Wert können Sie hier auch auslesen.

Die Größe des Hashs (`hashsize`) wird ebenfalls von dem Kernel berechnet und beträgt auf der Intel 32-Bit-Architektur `HASHSIZE=CONNTRACK_MAX / 8`. Diese Berechnung wird nur einmal beim Laden des Moduls durchgeführt, da anschließend die Hashsize nicht mehr geändert werden kann. Wenn Sie anschließend aber den Wert `CONNTRACK_MAX` erhöhen, ist die Größe des Hashs nicht mehr optimal, und die Suche nach einer Verbindung in der Tabelle durch den Kernel dauert unnötig lange.

Um dies zu verhindern, sollten Sie vor dem Laden des Kernelmoduls bereits entscheiden, wie viele Verbindungen Ihre Verbindungstabelle später unterstützen soll. Berechnen Sie nun den passenden Wert für die Hashsize, indem Sie die maximale Anzahl der Verbindungen durch 8 teilen. Bei einem Kernel älter als 2.4.21 sollten Sie auf Grund des verwendeten Hash-Algorithmus möglichst eine Primzahl benutzen. Bei jüngeren Kerneln sollte es sich bei der Zahl um eine Potenz von 2 handeln. Wählen Sie die geeignete Zahl aus, und laden Sie das Modul `ip_conntrack` manuell, bevor Ihre Regeln es benötigen:

```
modprobe ip_conntrack hashsize=16384
```

Auslesen können Sie den Hashsize-Wert entweder aus den Kernelmeldungen beim Laden des Moduls oder, wenn Ihr Kernel es unterstützt, über die Variable `/proc/sys/net/ipv4/netfilter/ip_conntrack_buckets`.

### Tipp



Falls Sie das Modul `ip_conntrack` fest in den Kernel einkompiliert haben, können Sie auch die Option `ip_conntrack.hashsize=16384` als Boot-Option dem Kernel übergeben.

## 19.3 TCP-Window-Tracking

Lange Zeit warfen Anwender und Hersteller kommerzieller Paketfilter dem Linux-Paketfilter vor, dass die Zustandsüberwachung speziell des TCP-Protokolls nicht ausreichend sei, da sie die TCP-Sequenznummern nicht überwachen würde. Um dies zu ermöglichen, schrieb Jozsef Kadlecsek bereits im August 2000 den TCP-Window-Tracking-Patch. Ursprünglich war dieser Patch recht kritisch in der Anwendung, da er das exakte Verhalten des TCP-Protokolls entsprechend des TCP-Standards nachbildete und viele Clients sich nicht zu 100 Prozent an diesen Standard halten. Der Patch wurde in den nächsten Jahren stark verbessert und vor allem um die Funktion erweitert, die Timeout-Werte für die verschiedenen Protokolle über das `/proc`-Verzeichnis einstellen zu können. Mit dem Kernel 2.6.9 wurde der Patch als fester Bestandteil in den Kernel übernommen. Für ältere Versionen steht der Patch noch auf der Netfilter-Homepage zur Verfügung.



### Achtung

Mit dem TCP-Window-Tracking werden TCP-Null-Pakete nicht mehr als neue Pakete akzeptiert!

Das TCP-Window-Tracking analysiert die von den Kommunikationspartnern angegebenen TCP-Windows und die Sequenznummern der Pakete. Alle Pakete, die eine Sequenznummer verwenden, die nicht in die aktuellen TCP-Windows passt, werden von dem Connection Tracking als `INVALID` gekennzeichnet.



### Tipp

Die Funktionsweise des Patches basiert auf dem Artikel von Guido van Rooij »Real Stateful TCP Packet Filtering in IP-Filter« ([http://www.iaa.nl/users/guido/papers/tcp\\_filtering.ps.gz](http://www.iaa.nl/users/guido/papers/tcp_filtering.ps.gz)). Der Artikel beschreibt diese Funktionalität in dem IP-Filter für BSD und Solaris. Wenn Sie sich für den Hintergrund interessieren, ist dies eine sehr ausführliche Quelle der angestellten Überlegungen.

## 19.4 /proc-Variablen

Der TCP-Window-Tracking-Patch fügt in seiner aktuellen Version ein neues Unterverzeichnis `netfilter` dem Verzeichnis `/proc/sys/net/ipv4/` hinzu. Da das Connection Tracking das Protokoll IPv6 noch nicht unterstützt, gibt es in dem Verzeichnis `/proc/sys/net/ipv6` kein entsprechendes Verzeichnis.

**Achtung**

Denken Sie daran, dass nach einem Neustart des Systems diese Werte wieder auf Ihre Default-Werte zurückgesetzt werden. Es ist sinnvoll, zu Beginn Ihres Firewall-Skripts die Werte manuell zu setzen, die Sie verwenden möchten.

Dieses Verzeichnis enthält die folgenden neuen Variablen:

**19.4.1 ip\_conntrack\_buckets**

In dieser Variablen können Sie die Hashsize der Verbindungstabelle auslesen (siehe Abschnitt 19.2).

**19.4.2 ip\_conntrack\_count**

Diese Variable zeigt die Anzahl der aktuell in der Verbindungstabelle vorgehaltenen Verbindungen an.

**19.4.3 ip\_conntrack\_generic\_timeout**

Diese Variable definiert das Timeout der Verbindungen, die nicht das TCP-, UDP- oder ICMP-Protokoll verwenden.

Default: 600 Sekunden.

**19.4.4 ip\_conntrack\_icmp\_timeout**

Diese Variable definiert das Timeout von ICMP-Verbindungen wie Echo-Request/Echo-Reply.

Default: 30 Sekunden.

**19.4.5 ip\_conntrack\_log\_invalid**

Mit diesem Parameter können Sie die Protokollierung ungültiger Pakete für ein bestimmtes Protokoll anschalten. Hierzu müssen Sie die Nummer des Protokolls in diese Variable schreiben (TCP=6, siehe `/etc/protocols`). Um diese Funktion abzustellen, nutzen Sie das Protokoll 255.

Default: 0.

### 19.4.6 ip\_contrack\_max

Diese Variable definiert die Größe Ihrer Verbindungstabelle. Mehr Verbindungen kann das Connection Tracking nicht überwachen. Sie können diesen Wert online ändern. Jedoch sollten Sie dann auch die Hashsize anpassen und den Abschnitt über das `ip_contrack`-Kernelmodul (siehe Abschnitt 19.2) lesen.

### 19.4.7 ip\_contrack\_tcp\_be\_liberal

Hiermit schalten Sie die Überwachung der Sequenznummern und der TCP-Windows für alle Pakete außer RST-Paketen ab. Wenn Sie Netzwerkgeräte verwenden, die sich nicht ganz an den TCP-Standard halten, kann dies notwendig sein.

Default: 0.

### 19.4.8 ip\_contrack\_tcp\_loose

Wie bereits mehrfach erwähnt wurde, akzeptiert der Linux-Kernel auch ein ACK-Paket als ein neues Paket. Das können Sie mit dieser Variable abschalten. Setzen Sie dazu diese Variable einfach auf 0. Wenn Sie das Verhalten wünschen, können Sie hier angeben, wie viele Pakete in beide Richtungen geflossen sein müssen, bevor das TCP-Window-Tracking aktiv werden darf.

Default: 3.

### 19.4.9 ip\_contrack\_tcp\_max\_retrans

Dieser Wert definiert die maximale Anzahl von wiederholten TCP-Paketen ohne Bestätigung des Empfängers. Sobald dieser Wert erreicht wurde, fängt der `ip_contrack_timeout_max_retrans` an zu zählen.

Default: 3.

### 19.4.10 ip\_contrack\_tcp\_timeout\_close

Diese Variable definiert, wie lange eine Verbindung in dem Zustand CLOSE verbleibt (siehe Abschnitt 19.4.14).

Default: 10 Sekunden.

### 19.4.11 ip\_contrack\_tcp\_timeout\_close\_wait

Diese Variable definiert, wie lange eine Verbindung in dem Zustand CLOSE\_WAIT verbleibt (siehe Abschnitt 19.4.14).

Default: 60 Sekunden.

### 19.4.12 ip\_conntrack\_tcp\_timeout\_established

Diese Variable definiert, wie lange eine Verbindung in dem Zustand ESTABLISHED verbleibt (siehe Abschnitt [19.4.14](#)).

Default: 432000 Sekunden; das entspricht 5 Tagen.

### 19.4.13 ip\_conntrack\_tcp\_timeout\_fin\_wait

Diese Variable definiert, wie lange eine Verbindung in dem Zustand FIN\_WAIT verbleibt (siehe Abschnitt [19.4.14](#)).

Default: 120 Sekunden.

### ip\_conntrack\_tcp\_timeout\_last\_ack

Diese Variable definiert, wie lange eine Verbindung in dem Zustand LAST\_ACK verbleibt (siehe Abschnitt [19.4.14](#)).

Default: 30 Sekunden.

### ip\_conntrack\_tcp\_timeout\_max\_retrans

Der Timeout für die Verbindung in der Verbindungstabelle, wenn nur Paketwiederholungen ohne Bestätigung des Empfängers beobachtet wurden.

Default: 300 Sekunden.

### ip\_conntrack\_tcp\_timeout\_syn\_recv

Diese Variable definiert, wie lange eine Verbindung in dem Zustand SYN\_RECV verbleibt (siehe Abschnitt [19.4.14](#)).

Default: 60 Sekunden.

### ip\_conntrack\_tcp\_timeout\_syn\_sent

Diese Variable definiert, wie lange eine Verbindung in dem Zustand SYN\_SENT verbleibt (siehe Abschnitt [19.4.14](#)).

Default: 120 Sekunden.

### ip\_conntrack\_tcp\_timeout\_time\_wait

Diese Variable definiert, wie lange eine Verbindung in dem Zustand TIME\_WAIT verbleibt (siehe Abschnitt [19.4.14](#)).

Default: 120 Sekunden.

### **ip\_conntrack\_udp\_timeout**

Hiermit definieren Sie den Timeout einer UDP-Verbindung, für die noch nicht Pakete in beiden Richtungen erkannt wurden.

Default: 30 Sekunden.

### **ip\_conntrack\_udp\_timeout\_stream**

Hiermit definieren Sie den Timeout einer UDP-Verbindung, für die bereits Pakete in beiden Richtungen erkannt wurden.

Default: 180 Sekunden.

## **19.4.14 TCP-Zustände**

Das Connection Tracking unterscheidet für TCP-Verbindungen die folgenden Zustände:

1. **SYN-SENT**: Ein erstes **SYN**-Paket wurde erkannt.
2. **SYN-RCV**: Das **SYN/ACK**-Paket wurde erkannt. Dies ist das zweite Paket des TCP-Handshakes.
3. **ESTABLISHED**: Das dritte Paket des TCP-Handshakes, das erste **ACK**-Paket, wurde erkannt.
4. **FIN\_WAIT**: Das erste **FIN**-PAKET wurde erkannt. Die Verbindung wurde abgebaut.
5. **CLOSE\_WAIT**: Das zweite **FIN**-Paket wurde erkannt.
6. **LAST\_ACK**: Das letzte **ACK**-Paket als Antwort auf das zweite **FIN**-Paket wurde erkannt.
7. **TIME\_WAIT**: Nach Ablauf des Zustands **LAST\_ACK** wechselt die Verbindung in den Zustand **TIME\_WAIT**, um Pakete, die während der Übertragung in eine falsche Reihenfolge gebracht wurden, noch zu akzeptieren.
8. **CLOSE**: Nach dem Ablauf des Zustands **TIME\_WAIT** wechselt die Verbindung in den Zustand **CLOSE**, um sicherzustellen, dass es nicht zum Konflikt mit neuen Verbindungen kommen kann.

Der Übergang der verschiedenen Zustände wird durch das entsprechende Paket ausgelöst. Lediglich die letzten Übergänge von **LAST\_ACK** nach **CLOSE** erfolgen durch den Ablauf des Timeouts. Kommt es vorher zum Ablauf eines anderen Timeouts, wird die Verbindung aus der Tabelle entfernt!