# 09 TCP States

## 1 Thema des Praktikums

Im folgenden Praktikum werden die in der Theorie besprochenen Mechanismen von TCP untersucht. Die besprochenen Server- und Client-Programme werden in diesem Praktikum verwendet.

Die Schwerpunkte des Praktikums sind:

- TCP-State-Machine
- Verbindungsauf und Abbau (three-way hand-shake)
- Verwendung von Sequenznummern

## 2 Vorbereitung

### 2.1 Beobachtung von TCP-Protokollzuständen

Das Ziel dieser Aufgabe ist es, alle Zustände in der TCP-State-Machine beim Verbindungsauf- und -Abbau sowohl auf Client- wie auf Server-Seite zu beobachten.

Um die Zustände im TCP-State-Diagramm beobachten zu können, verwenden wir den Befehl *iptables*, indem wir gezielt Pakete im IP-Layer verwerfen und damit die TCP-State-Machine in einen bestimmten Zustand (sowohl auf Client wie Server-Seite) bringen können.

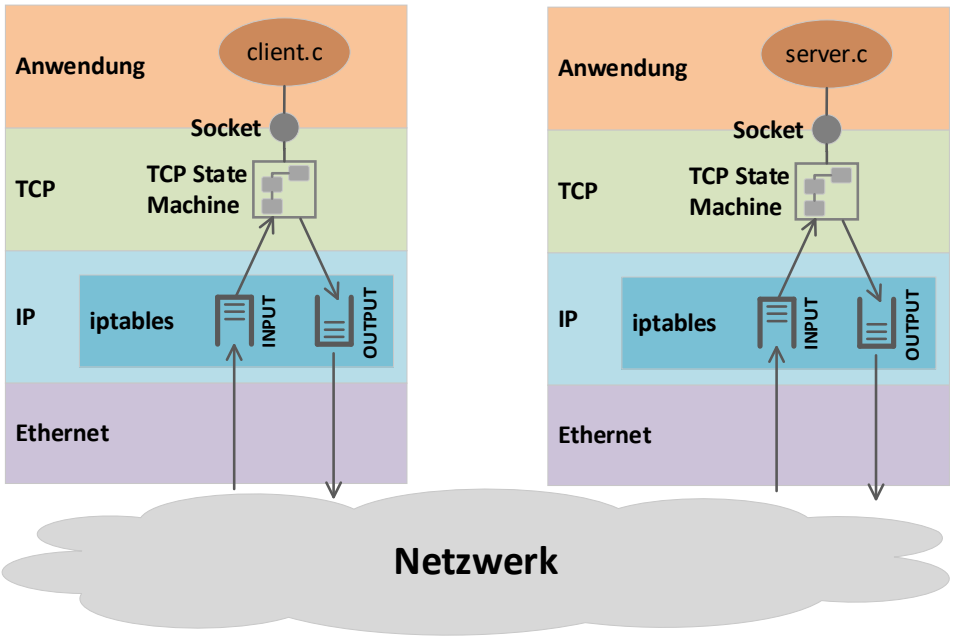



Abbildung 1

Der Befehl *iptables* der gehört eigentlich zur Linux-Firewall und erlaubt es, beim Empfangen (INPUT) oder Senden (OUTPUT) anhand von verschiedenen Kriterien (Protokoll, Flags etc.) gezielt Pakete zu verwerfen (Abbildung 1). Die Änderung der Filter-Regel benötigt root Rechte, also mit **`sudo`** ausführen.

Beispiel:

```
sudo iptables -A INPUT -p tcp --tcp-flags SYN,ACK SYN,ACK -j DROP
```

Der obige Befehl fügt im IP-Layer eine Filter-Regel hinzu, so dass alle empfangenen (-A INPUT) TCP-Pakete (-p tcp) mit gesetzten SYN und ACK Flags (--tcp-flags SYN,ACK SYN,ACK) ausgefiltert werden (-j DROP).

Der Parameter –tcp-flags ist erklärungsbedürftig:

Die erste kommagetrennte Liste bezeichnet die Flags, die für eine Regel betrachtet werden. Die zweite Liste gibt an, wie diese gesetzt sein müssen, damit die Regel aktiv wird. Das heisst: falls ein Flag in der 2 Liste aufgeführt ist, muss es *True* sein und falls es nicht aufgeführt ist, muss es *False* sein.

Beispiel:

Der folgende Befehl filtert zu sendende TCP-Pakete aus, bei denen das ACK-Flag gesetzt und das FIN-Flag *nicht* gesetzt ist:

```
sudo iptables –A OUTPUT –p tcp --tcp-flags FIN,ACK ACK –j DROP
```

Mit folgendem Befehl können Sie alle aktiven Filterregeln anzeigen:

```
sudo iptables -L
```

Mit –D statt –A können Sie eine eingegebene Filter-Regel wieder löschen. Wenn Sie eine Regel 2-mal eingegeben haben, müssen Sie diese auch 2-mal löschen:

```
sudo iptables –D INPUT –p tcp --tcp-flags SYN,ACK SYN,ACK –j DROP
```

**TCP-State-Machine für den Verbindungsaufbau:**

- Studieren Sie Abbildung 2: TCP-State-Machine für den Verbindungsaufbau und leiten Sie die *iptables* Befehle her, um auf Client und Server die jeweiligen Protokollzustände beim Verbindungsaufbau beobachten zu können.

  Tragen Sie die fehlenden iptables-Parameter in Tabelle 1 ein, damit der Client oder der Server im angegebenen Zielzustand stehen bleibt.

  **Hinweis:** Es gibt mehrere Lösungen: Hier soll jeweils beim Empfänger derjenige Event abgefangen werden, der zum Verlassen des gewünschten Zielzustands führen würde.

Damit Server und Client im Zustand ESTABLISHED stehen bleiben, muss im Code eine Zeile

```
wait("Text");
```

eingefügt werden. Wobei `wait(…)` blockiert, bis eine Tastatureingabe gemacht wird.

- Geben Sie in Tabelle 2 an, wo der Code ergänzt werden muss.

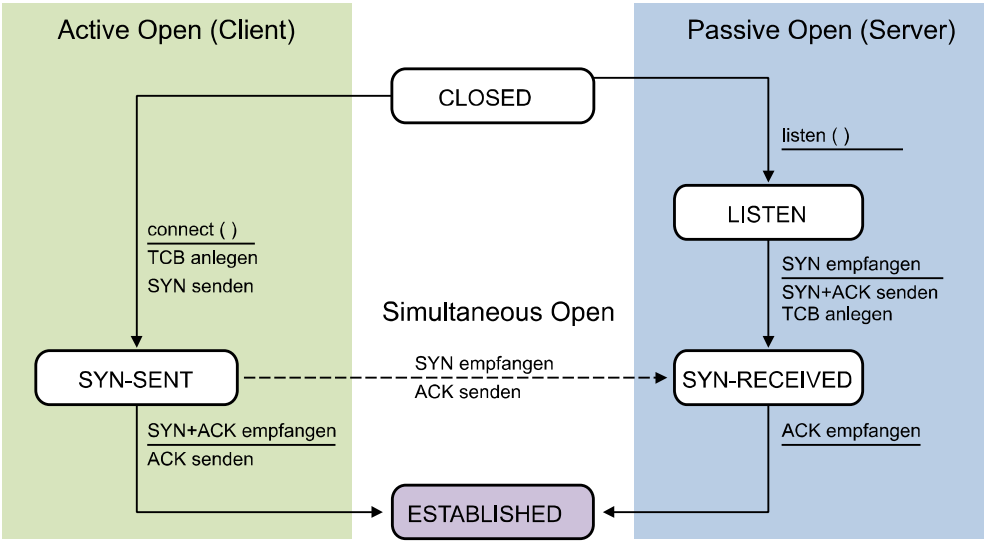



Abbildung 2: TCP-State-Machine für den Verbindungsaufbau

| Zustand | Client / Server | iptables-Befehl |
|---|---|---|
| SYN-SENT | Client | `iptables -A INPUT -p tcp -j DROP` --tcp-flags SYN,ACK SYN,ACK |
| LISTEN | Server | `iptables -A INPUT -p tcp -j DROP` --tcp-flags SYN SYN |
| SYN-RECEIVED | Server | `iptables -A INPUT -p tcp -j DROP` --tcp-flags ACK ACK |

Tabelle 1

syn-ack ack

| Zustand | Client / Server | Befehle vor/nach wait(…) |
|---|---|---|
| ESTABLISHED | Client | status = connect(....)<br>ExitOnError()<br>wait(); |
| ESTABLISHED | Server | visits++;<br>printf("....");<br>wait(); |

Tabelle 2

*Braucht es einen iptables-Befehl, um auf dem Server den Zustand LISTEN zu beobachten, oder lässt sich der LISTEN Zustand auch anders beobachten?*

**TCP-State-Machine für den Verbindungsabbau:**

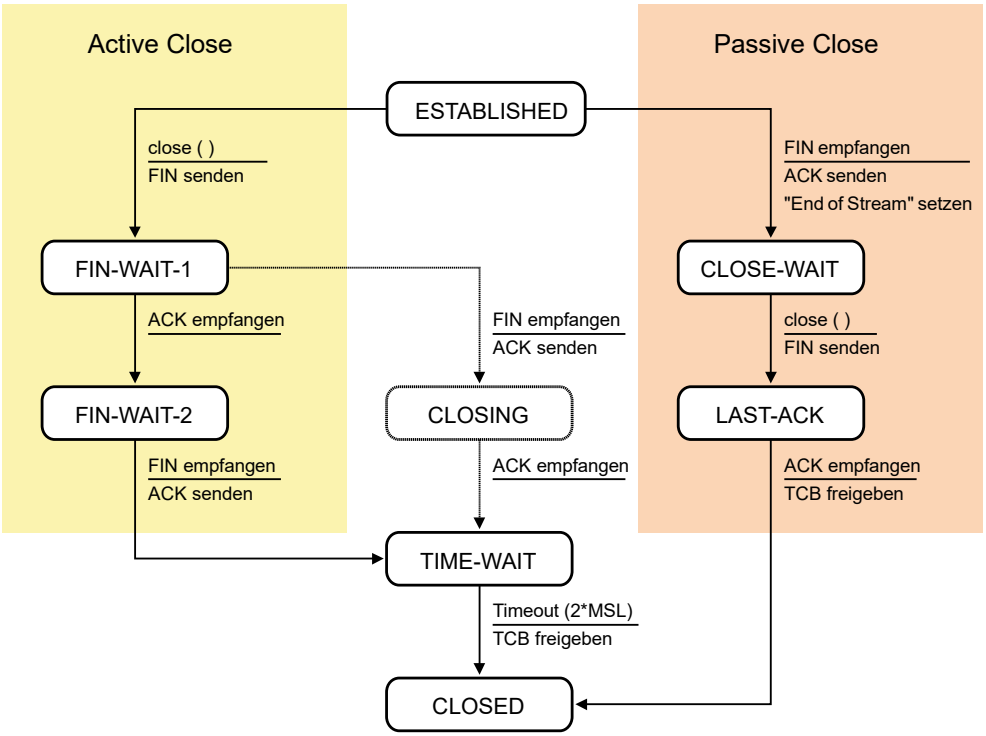



Abbildung 3: TCP-State-Machine für den Verbindungsabbau

Leiten Sie die *iptables*-Befehle her, um auf der aktiven und passiven Seite die Protokollzustände gemäss Abbildung 3: TCP-State-Machine für den Verbindungsabbau beobachten zu können.

Es wird für jeden Zustand von einer ESTABLISHED-Verbindung ausgegangen. Das heisst Filterregeln werden erst eingegeben, wenn die TCP-State-Machines auf beiden Seiten im Zustand ESTABLISHED sind. In diesem Zustand können Sie die Programme anhalten, indem Sie an geeigneter Stelle einen wait()-Befehl einfügen (siehe Vorbereitung Tabelle 2).

| Zustand | Active / Passive Seite | iptables-Befehl bzw. wait(…) |
|---|---|---|
| FIN-WAIT-1 | Active | … input … ACK ACK |
| FIN-WAIT-2 | Active | … input … FIN FIN |
| TIME-WAIT | Active | … output … TCB TCB |
| CLOSE-WAIT | Passive | … output … FIN FIN |
| LAST-ACK | Passive | … input … ACK ACK |

Tabelle 3

Zeigen Sie die Vorbereitungen dem Laborbetreuer.

## 3 Versuchsdurchführung

- Bauen Sie die Versuchsanordnung gemäss Abbildung 4 auf. Die Rechner werden über eth1 und einen Hub verbunden sowie mit Linux gestartet Das Ethernet-Interface eth0 bleibt mit dem ZHAW-Netz verbunden.

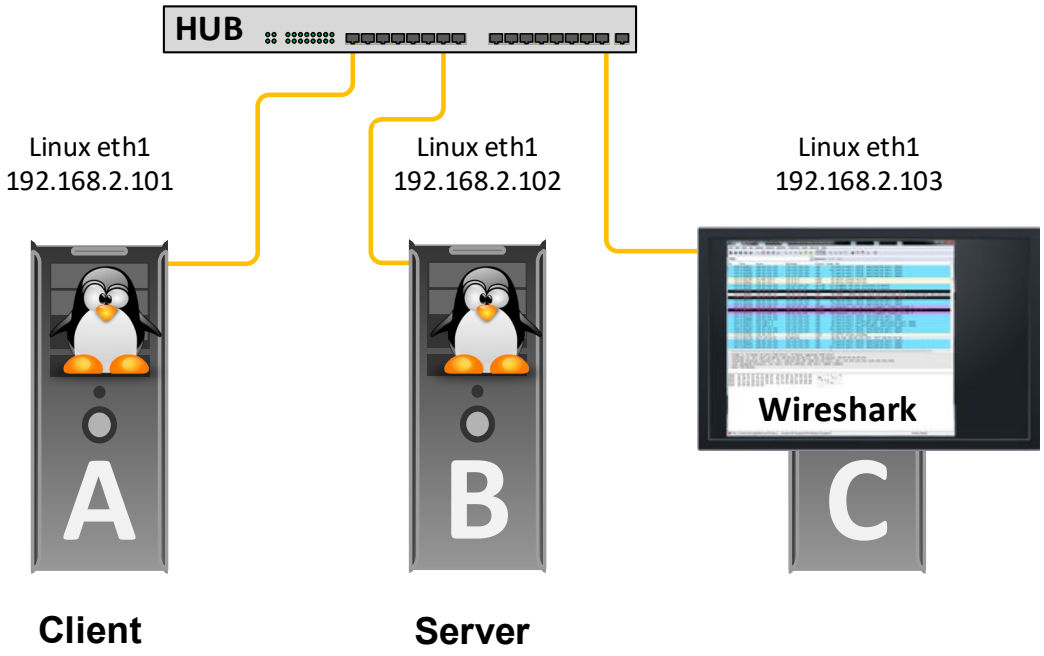



Abbildung 4

- Kopieren Sie auf den Rechnern A und B alle Client- und Server-Programme in ein lokales Verzeichnis . Öffnen Sie dazu ein Terminal und geben Sie **`download-kt`** ein. Das Script kopiert die benötigten Files nach /home/ktlabor/praktika/09_tcp_states.
- **Am Ende des Praktikums** führen Sie das Scrip **`reset-kt-home`** aus. Das Script löscht das Home-Verzeichnis des Users ktlabor und erstellt es neu.
- Im Folgenden benötigen wir die Programme server.c und client.c. Wechseln Sie auf den Rechnern A und B ins Praktikumsverzeichnis:
  **`cd /home/ktlabor/praktika/09_tcp_states`**
- Übersetzen Sie die Programme mit dem GNU Compiler:
  Auf den Rechner A:
  **`gcc -Wall –o client client.c`**
  Auf den Rechner B:
  **`gcc -Wall –o server server.c`**
  Wiederholen Sie dies nach jeder Änderung an den Programmen.

### 3.1 Durchführung: Zustände beim Verbindungsaufbau

Um die Zustände der TCP -State-Machine anzuzeigen, wird auf dem Server- und dem Client-Rechner je eine Konsole mit dem ss-Befehl (Socket State) zusammen mit dem watch-Befehl (wiederholtes Aufrufen) gestartet.

- Auf dem Client (Rechner A) betrachten Sie die Sockets mit Destination-Port 4711 mit folgendem Befehl:
  `watch –n 0.1 ss –t –a –o state all dport == :4711`
- Analog starten Sie auf dem Server eine Konsole für das Source-Port 4711 mit folgendem Befehl:
  `watch –n 0.1 ss –t –a –o state all sport == :4711`
- Auf dem Rechner C nutzen Sie Wireshark, um die Kommunikation beobachten zu können.
- Starten Sie nun auf dem Rechner B in einer weiteren Konsole den Server:
  `./server 4711`

*In welchem Zustand ist der Server?*

LISTEN

---

- Starten Sie auf dem Rechner A in einer weiteren Konsole den Client und versuchen Sie in den «watch»-Windows die verschiedenen Client- und Server- Zustände von Abbildung 2 zu verfolgen:
  `./client 192.168.2.102 4711`
- Studieren Sie die erfolgte Kommunikation im Wireshark.

*Warum konnten Sie die verschiedenen Zustände auf den Konsolen nicht sehen, obwohl Wireshark die Übertragung der TCP-Pakete anzeigt?*

Es wurden noch keine Daten versendet, sondern erst durch den three Handshake eine Verbindung aufgebaut

---

- Geben Sie auf dem Client PC den folgenden Befehl ein.
  `sudo iptables –A INPUT –p tcp --tcp-flags SYN,ACK SYN,ACK –j DROP`

*Was für Meldungen (Quelle/Ziel und Inhalt) werden damit ausgefiltert?*

Wenn im Paket SYN / ACK SYN / ACK vorkommen, dann werden diese Pakete gedropped. Diese Einstellung wurde direkt auf der Firewall betätigt

- Starten Sie den Client erneut.

*In welchen Zuständen befinden sich Server und Client nun?*

PC A: SYN-SENT B: SYN-RECV

---

*Was geschieht Protokoll-mässig (siehe Wireshark)?*

Durch den iptables-Rule werden die Pakete gedropped, daher probiert TCP die Pakete weiterzuversenden

Um Client und Server in den Zustand ESTABLISHED zu bringen, sollen Sie in nun den Programmen Stopps einbauen. Dazu kann die vordefinierte Routine **`wait(…)`** verwendet werden. Sie gibt einen String aus und blockiert bis eine Eingabe erfolgt.

```
/* Beispiel für die Anwendung von wait()*/
wait("close");
close(SocketXY);
```

- Fügen Sie im Client- und Server-Programm ein **`wait`**-Routinen gemäss Ihrer Vorbereitung (Tabelle 1) ein, so dass die Verbindung im Zustand ESTABLISHED verweilt. (Compilieren nicht vergessen).
- Entfernen Sie auf dem Client PC die iptables Regel wieder:
  `sudo iptables –D INPUT –p tcp --tcp-flags SYN,ACK SYN,ACK –j DROP`

- Starten Sie Client und Server erneut und beobachten Sie die Protokollzustände.

*Wie lange verweilt die Verbindung auf Client- und Server-Seite im Zustand ESTABLISHED?*

Der Client bleibt solange hängen, bis die Close-Methode aufgerufen wird

### 3.2 Durchführung: Zustände beim Verbindungsabbau

*Studieren Sie den Code. Wer (Client oder Server) durchläuft den Active-Close- resp. den Passive-Close-Pfad gemäss Abbildung 3: TCP-State-Machine für den Verbindungsabbau?*

- Bringen Sie Client und Server in den Zustand ESTABLISHED. Geben Sie in Tabelle 4 die Zustände der passiven und der aktiven Seite an, indem Sie diese Zustände vor Ausführung von close() bestimmen.

| | Server | Client |
|---|---|---|
| Vor dem 1. Close: | | |
| Nach dem 1. Close: | FIN Wait 2 | CLosed Wait |
| Nach dem 2. Close: | Time wait | Closed |

Closed

Tabelle 4

*Wie nennt man den Zustand, in dem sich die TCP-Verbindung nach dem 1. Close befindet?*

Half-Close

*Wie lange dauert es, bis der Socket der Active-Close-Seite endgültig geschlossen ist?*

1 Minute

- Erzwingen Sie die transitiven Zustände FIN-WAIT-1 und LAST-ACK, indem Sie die vorbereiteten iptable-Befehle vom Tabelle 3 anwenden.
  **Hinweis:** Es wird von einer ESTABLISHED-Verbindung ausgegangen. Das heisst, die Filterregeln werden erst eingegeben, wenn die TCP-State-Machines auf beiden Seiten im Zustand ESTABLISHED sind, also vor dem close() warten.

*Wie lautet der Befehl auf der Server-Seite:*

sudo iptables -a INPUT -p tcp --tcp-flags ACK ACK -j DROP

*Wie lautet der Befehl auf der Client-Seite:*

sudo iptables -a INPUT -p tcp -j DROP --tcp-flags FIN, ACK ACK

Zeigen Sie diesen Zustand und die obigen Resultate dem Laborbetreuer.

# 4 Durchführung: Verwendung der Sequence- und Acknowledgement-Nummern

## 4.1 Sequence- und Acknowledgement-Nummern beim Verbindungs-Aufbau/-Abbau

- Starten Sie auf Rechner C Wireshark, um den zeitlichen Ablauf einer TCP-Verbindung aufzuzeichnen. Starten Sie den Server (Rechner B) und greifen Sie mit dem Client (Rechner A) auf den Server zu.
- Ändern Sie im Wireshark die TCP-Einstellungen zwischen absoluten und relativen Sequenznummern.

*Welche Vor- und Nachteile haben die beiden Darstellungen?*

Vorteil: sehr genaue Sequenznummer

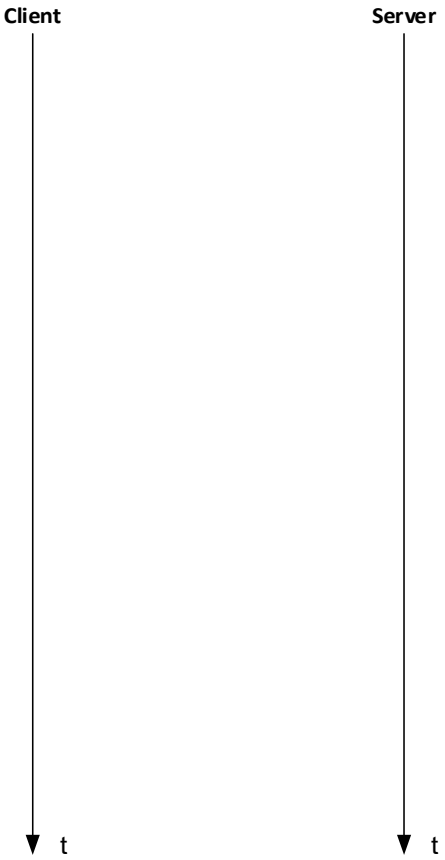
Nachteil: zu viele Informationen

- Betrachten Sie den Verbindungsaufbau und -abbau sowie die Datenübertragung der TCP-Verbindung.

*Wie wird die Initial Sequence Number (ISN) bestimmt? Gilt diese für beide Richtungen?*

*Wie verändert sich die Sequence-Nummer beim Verbindungsaufbau und –abbau?*

- Zeichnen Sie das Sequenzdiagramm. Notieren Sie jeweils die Sequence- und Acknowledgement-Nummern (relativ), die Datenlänge sowie die TCP-Flags jeder Meldung



### 4.2 Sequence- und Acknowledgement-Nummern bei der Datenübertragung

- Erweitern Sie den Server, so dass er mehrmals Daten sendet (z.B. mit einer for-Schleife).
- Betrachten Sie das Flow-Diagramm in Wireshark.

*Was für eine Funktion hat die Sequence-Nummer?*

*Welche Funktion hat die Acknowledgement-Nummer resp. wie wird sie im Regelfall berechnet?*

- Zeigen Sie die Resultate dem Laborbetreuer.

## 5 Zusatzaufgabe: Bidirektionale Kommunikation

- Ändern Sie die Server- und Client-Programme für bidirektionale Kommunikation, so dass beide alternierend Daten senden und empfangen. Oder verwenden Sie die beiden fertigen Programme **`server_duplex.c`** und **`client_duplex.c`**.
- Betrachten Sie das Flow-Diagramm in Wireshark und vergleichen Sie es mit den oben erstellten.

*Was hat sich am Sequenzdiagramm geändert?*

- Geben Sie im Spreadsheet TCP-Beispiel.xls von der obigen Verbindung die Initialwerte und einige der Datentransfers ein.

*Gibt es Abweichungen?*