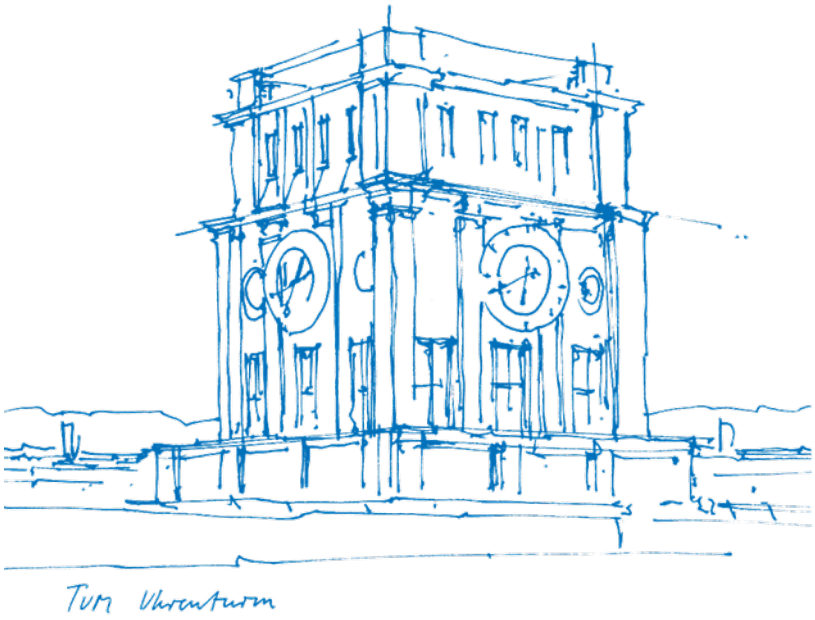
GRNVS Tutorium 10 - SS21

Fabian Sauter
Technische Universität München
Grundlagen: Rechnernetze und Verteilte Systeme (IN0010)
Lehrstuhl für Netzarchitekturen und Netzdienste
Garching, 28.06.2021

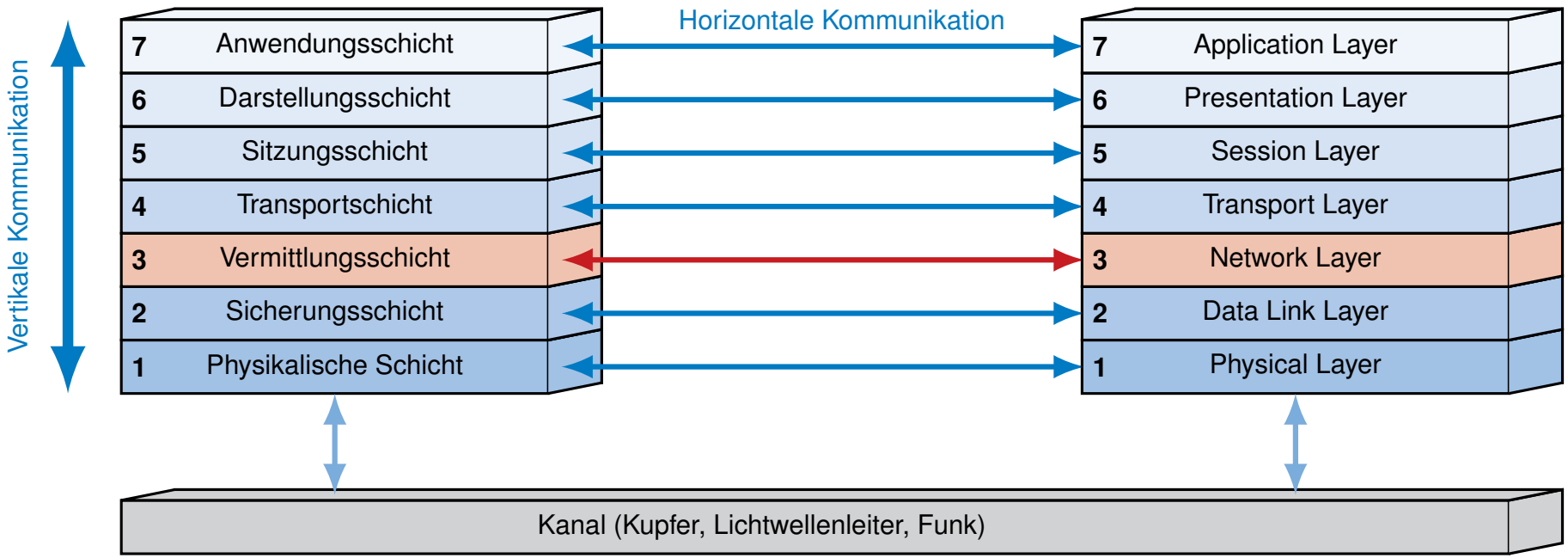
Slides & Notes: http://grnvs.uwpx.org



Wiederholung

Kapitel 3: Vermittlungsschicht

Einordnung im ISO/OSI-Modell



Kapitel 3

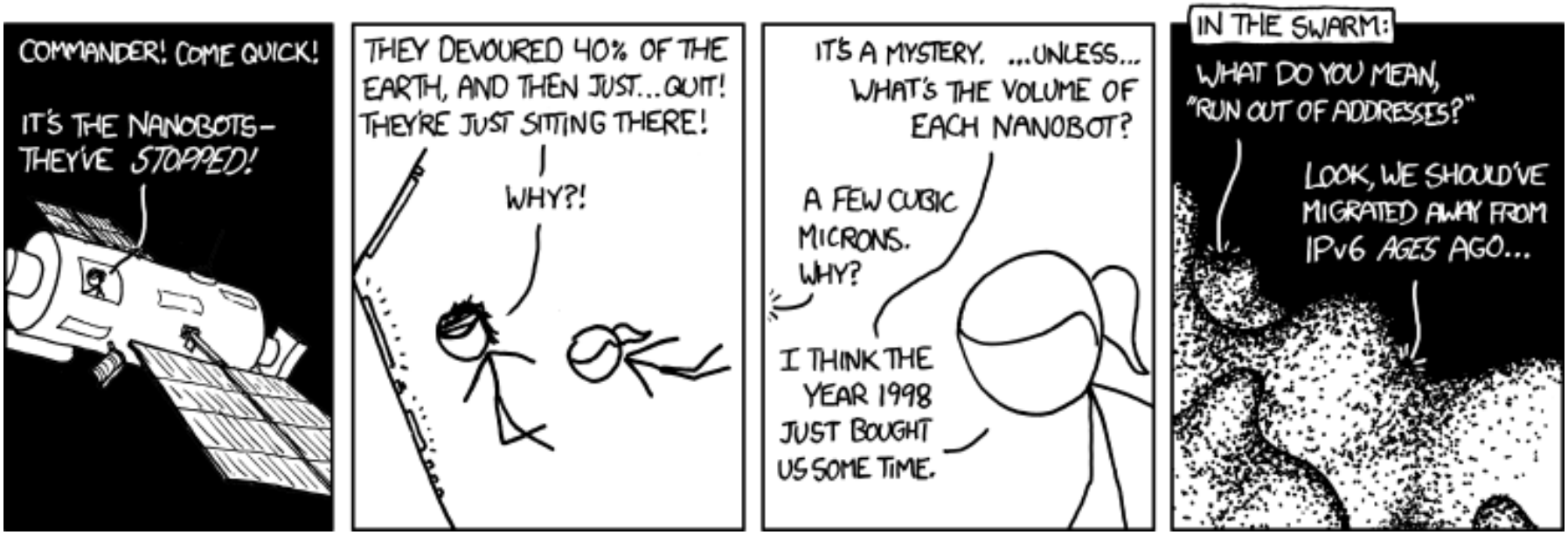


Abbildung: Nanobots (xkcd)[1]

[1] https://xkcd.com/865/

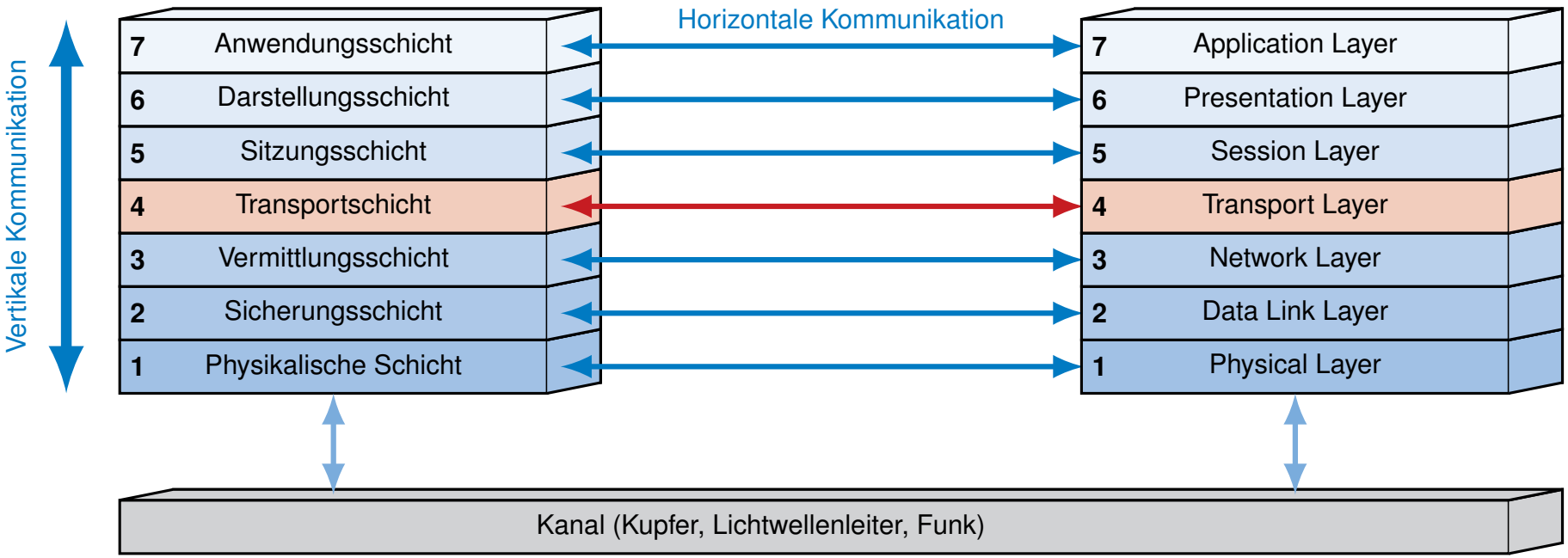
Kapitel 3

Abbildung: 100 years later, this story remains terrifying–not because it's the local network block, but because the killer is still on IPv4. (Campfire (xkcd))[2]

[1] `https://xkcd.com/742/`

Motivation

Einordnung im ISO/OSI-Modell



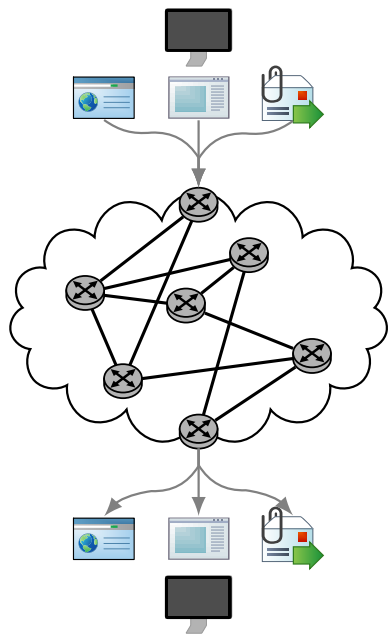
Aufgaben der Transportschicht

Die wesentlichen Aufgaben der Transportschicht sind

- Multiplexing von Datenströmen unterschiedlicher Anwendungen bzw. Anwendungsinstanzen,

Multiplexing:

- Segmentierung der Datenströme unterschiedlicher Anwendungen (Browser, Chat, Email, . . .)
- Segmente werden in jeweils unabhängigen IP-Paketen zum Empfänger geroutet
- Empfänger muss die Segmente den einzelnen Datenströmen zuordnen und an die jeweilige Anwendung weiterreichen



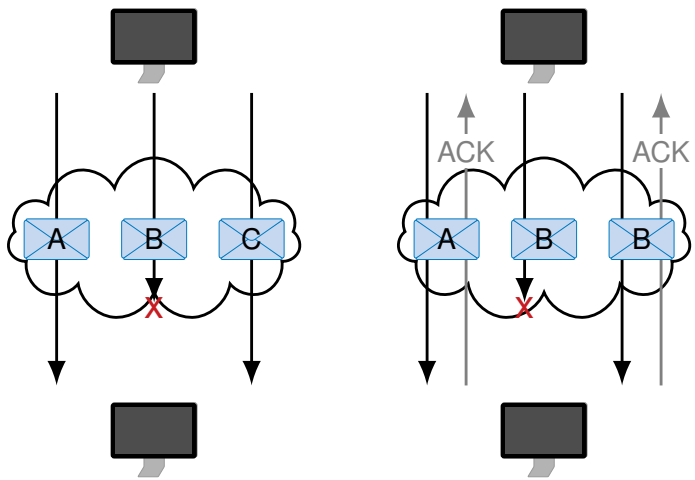
Aufgaben der Transportschicht

Die wesentlichen Aufgaben der Transportschicht sind

- Multiplexing von Datenströmen unterschiedlicher Anwendungen bzw. Anwendungsinstanzen,
- Bereitstellung verbindungsloser und verbindungsorientierter Transportmechanismen und

Transportdienste:

- Verbindungslos (Best Effort)
  - Segmente sind aus Sicht der Transportschicht voneinander unabhängig
  - Keine Sequenznummern, keine Übertragungswiederholung, keine Garantie der richtigen Reihenfolge
- Verbindungsorientiert
  - Übertragungswiederholung bei Fehlern
  - Garantie der richtigen Reihenfolge einzelner Segmente



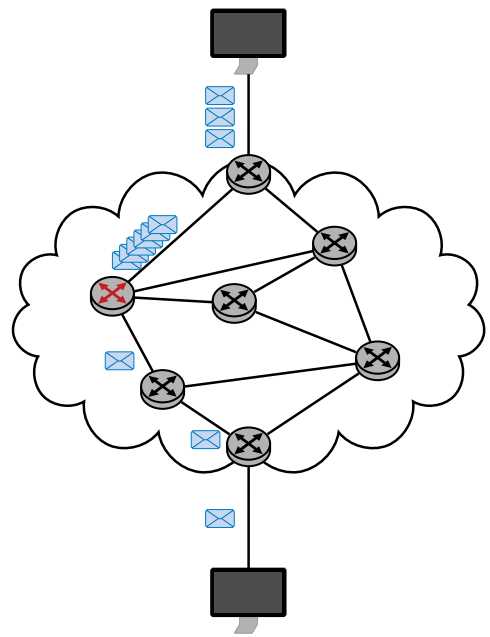
Aufgaben der Transportschicht

Die wesentlichen Aufgaben der Transportschicht sind

- Multiplexing von Datenströmen unterschiedlicher Anwendungen bzw. Anwendungsinstanzen,
- Bereitstellung verbindungsloser und verbindungsorientierter Transportmechanismen und
- Mechanismen zur Stau- und Flusskontrolle.

Stau- und Flusskontrolle:

- Staukontrolle (Congestion Control)
  - Reaktion auf drohende Überlast im Netz
- Flusskontrolle (Flow Control)
  - Laststeuerung durch den Empfänger

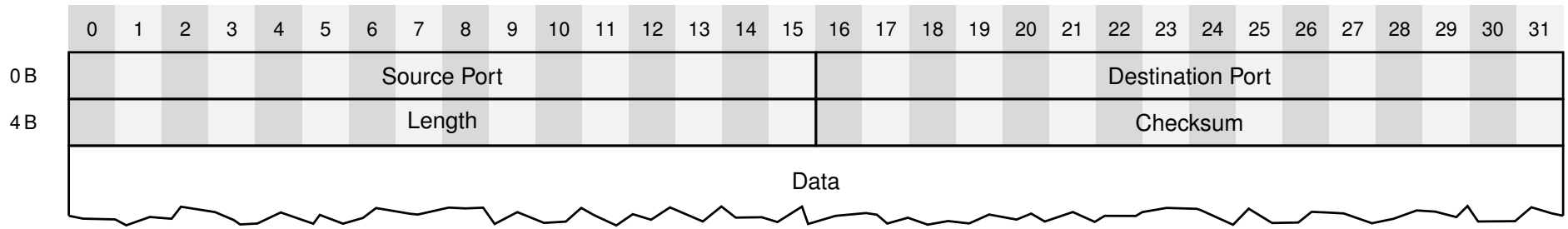


User Datagram Protocol (UDP)

Das User Datagram Protcol (UDP) ist eines der beiden am häufigsten verwendeten Transportprotokolle im Internet. Es bietet

- ungesicherte und nachrichtenorientierte Übertragung
- bei geringem Overhead.

UDP-Header:



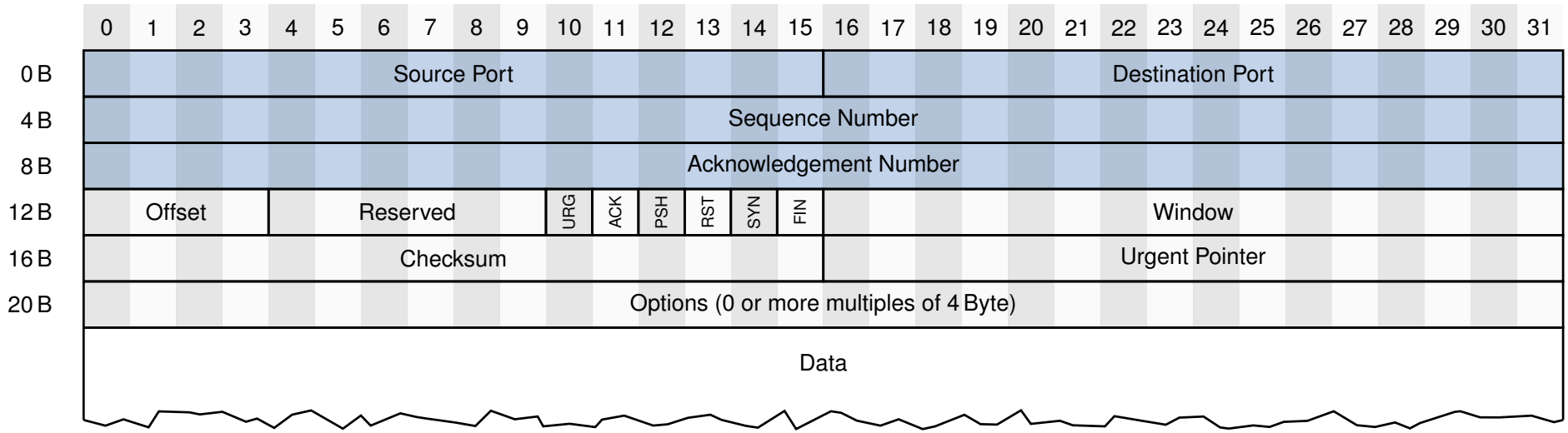
- „Length“ gibt die Länge von Header und Daten in Vielfachen von Byte an.
- Die Prüfsumme erstreckt sich über Header und Daten.
  - Die Verwendung der UDP-Prüfsumme ist bei IPv4 optional, wird für IPv6 jedoch vorausgesetzt.
  - Wird sie nicht verwendet, wird das Feld auf 0 gesetzt.
  - Wird sie verwendet, wird zur Berechnung ein Pseudo-Header genutzt (eine Art „Default-IP-Header“ der nur zur Berechnung der Prüfsumme dient). Er beinhaltet folgende Felder des IP-Headers: Quell- und Ziel-IP-Adresse, ein 8-Bit-Feld mit Nullen, Protocol-ID und Länge des UDP-Datagramms.

Transmission Control Protocol (TCP)

Das Transmission Control Protocol (TCP) ist das dominierende Transportprotokoll im Internet (rund 90 % des Datenverkehrs im Internet [1]). Es bietet

- gesicherte / stromorientierte Übertragung mittels Sliding-Window und Selective Repeat sowie
- Mechanismen zur Fluss- und Staukontrolle.

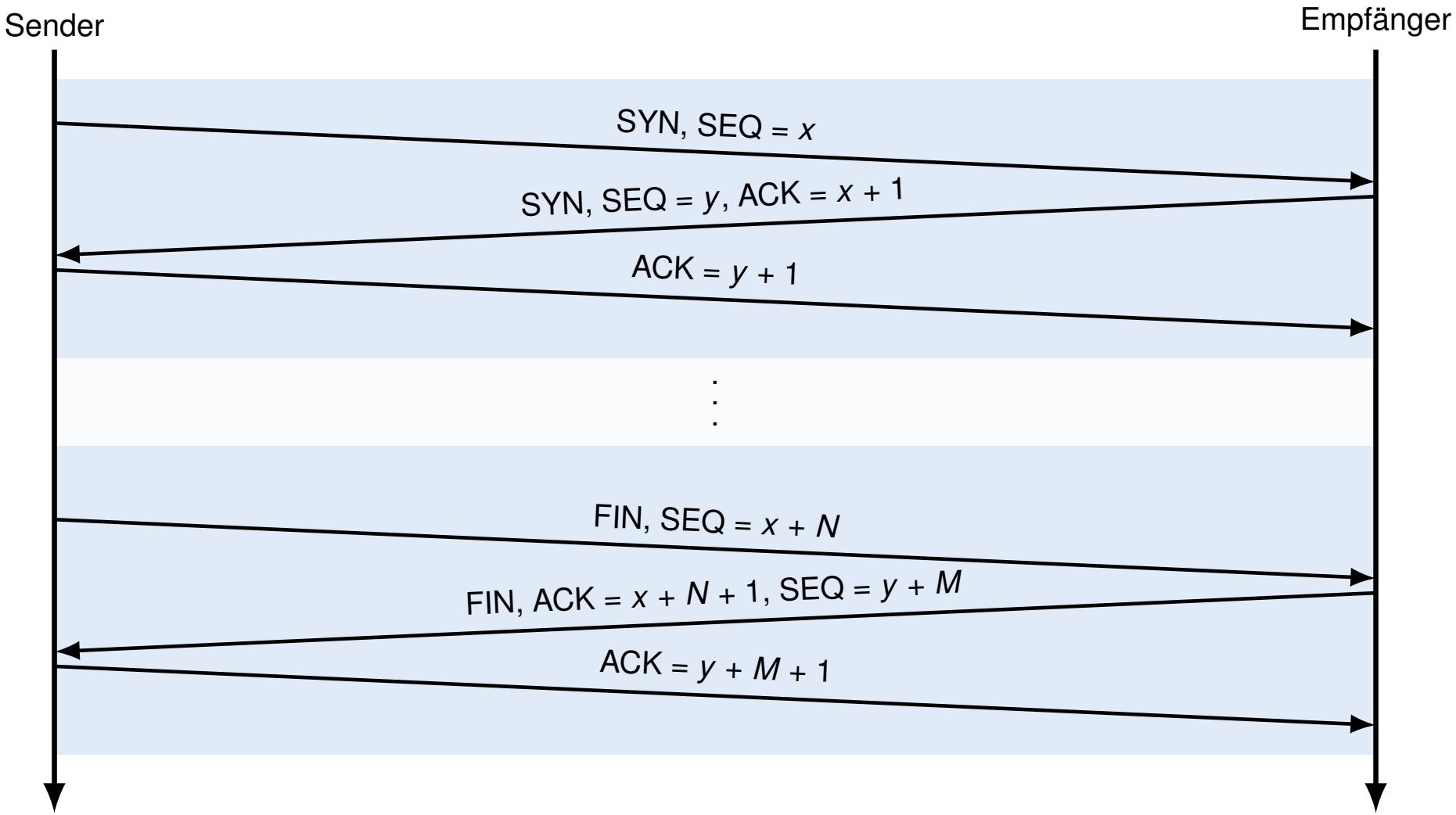
TCP-Header:



- Quell- und Zielport werden analog zu UDP verwendet.
- Sequenz- und Bestätigungsnummer dienen der gesicherten Übertragung. Es werden bei TCP nicht ganze Segmente sondern einzelne Bytes bestätigt (stromorientierte Übertragung).

Verbindungsorientierte Übertragung

Beispiel: Aufbau und Abbau einer Verbindung

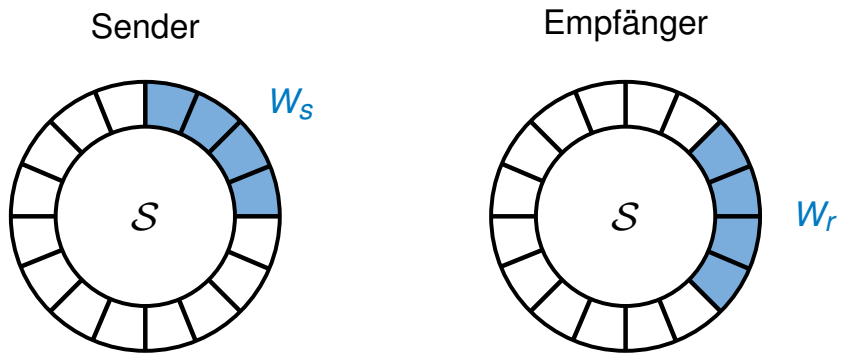


Diese Art des Verbidnungsaufbaues bezeichnet man als 3-Way-Handshake.

Zur Notation:

- Sender und Empfänger haben denselben Sequenznummernraum $\mathcal{S} = \{0,1,2,\ldots,N-1\}$.

  Beispiel: $N = 16$:



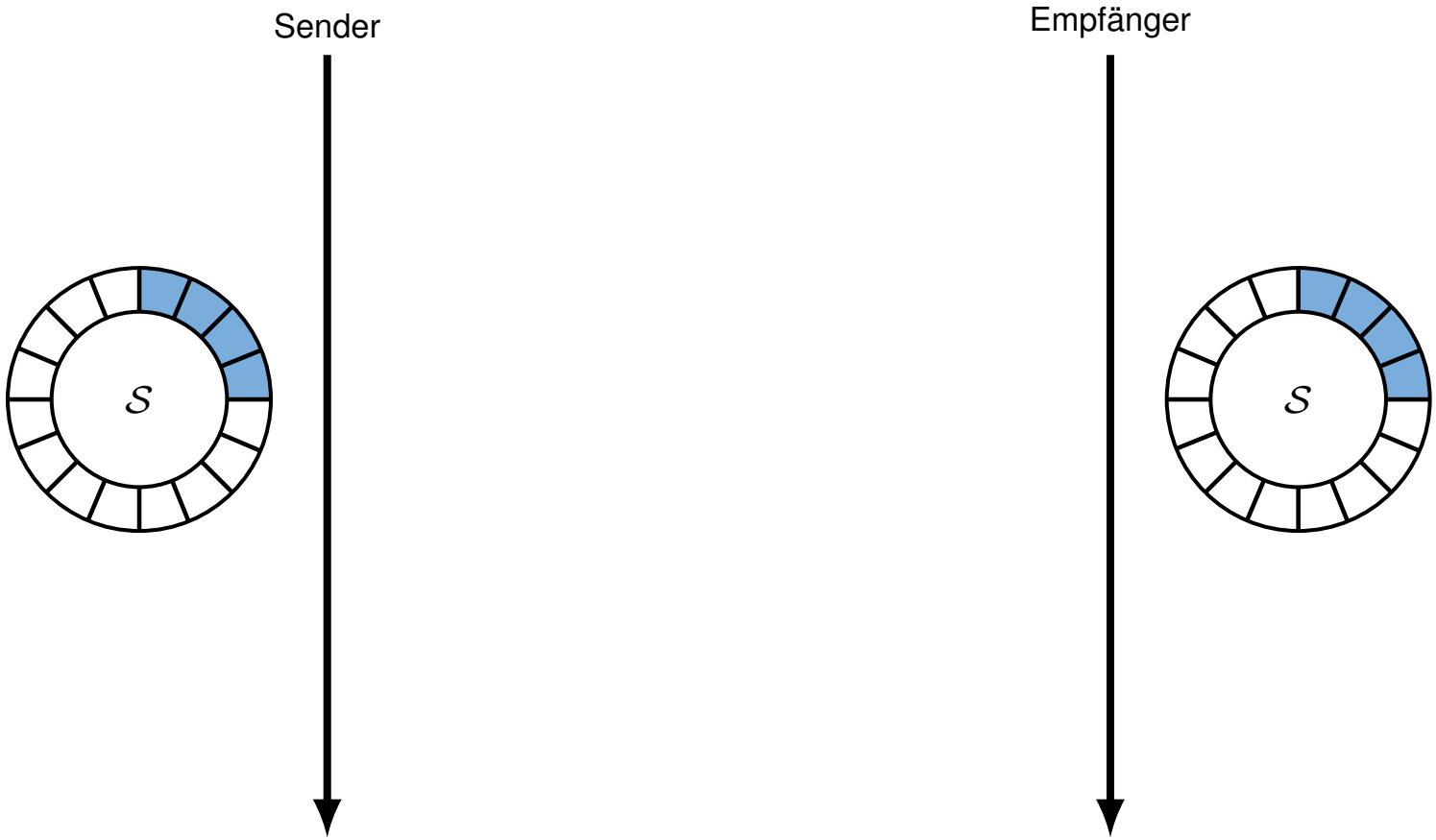
- Sendefenster (Send Window) $W_s \subset \mathcal{S}$, $|W_s| = w_s$:
  Es dürfen w_s Segmente nach dem letzten bestätigten Segment auf einmal gesendet werden.
- Empfangsfenster (Receive Window) $W_r \subset \mathcal{S}$, $|W_r| = w_r$:
  Sequenznummern der Segmente, die als nächstes akzeptiert werden.
- Sende- und Empfangsfenster „verschieben“ und überlappen sich während des Datenaustauschs.

Sliding-Window-Verfahren

Sendefenster W_s bzw. Empfangsfenster W_r

gesendet aber noch nicht bestätigt

gesendet und bestätigt/empfangen

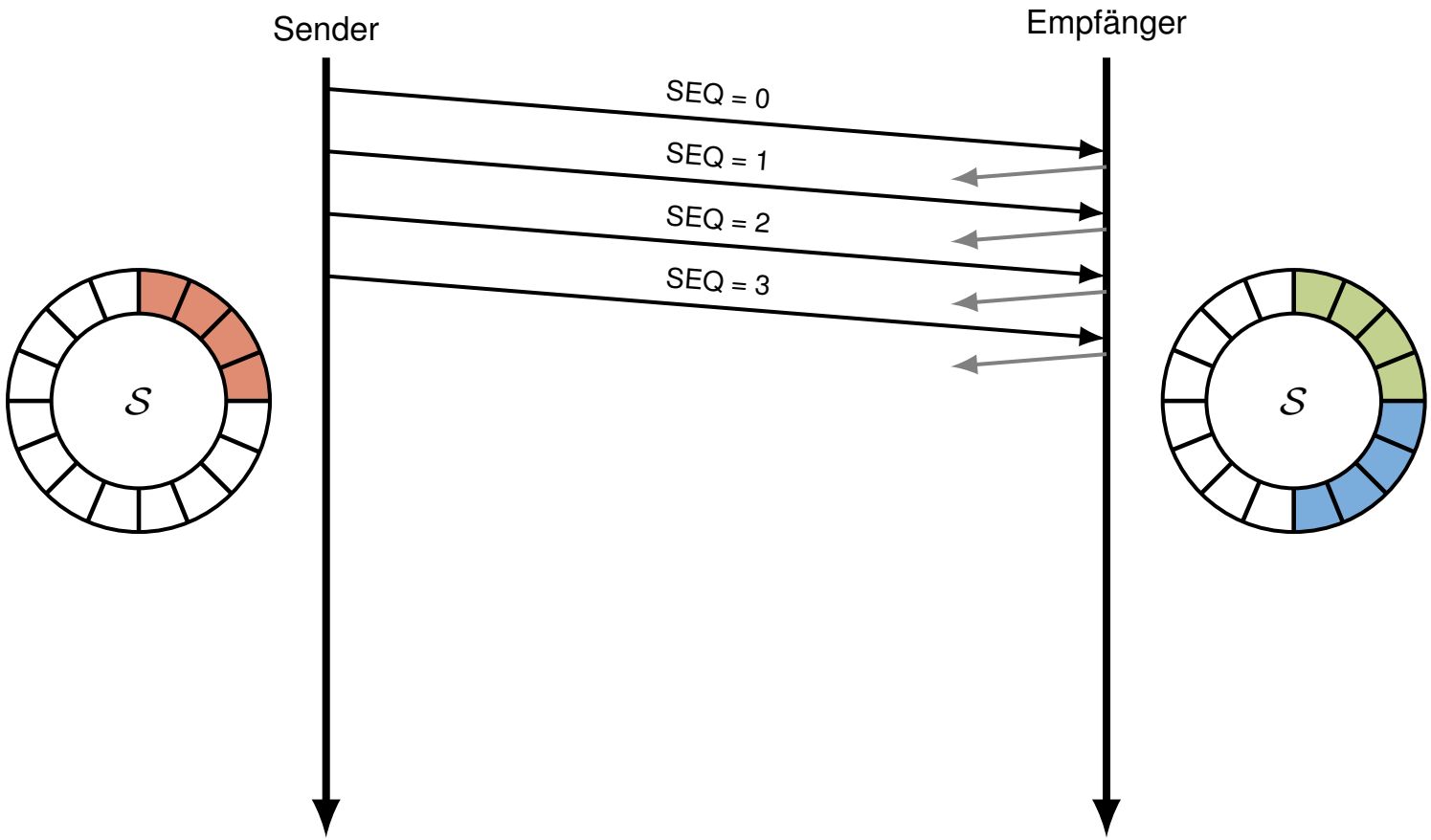


Sliding-Window-Verfahren

Sendefenster W_s bzw. Empfangsfenster W_r

gesendet aber noch nicht bestätigt

gesendet und bestätigt/empfangen

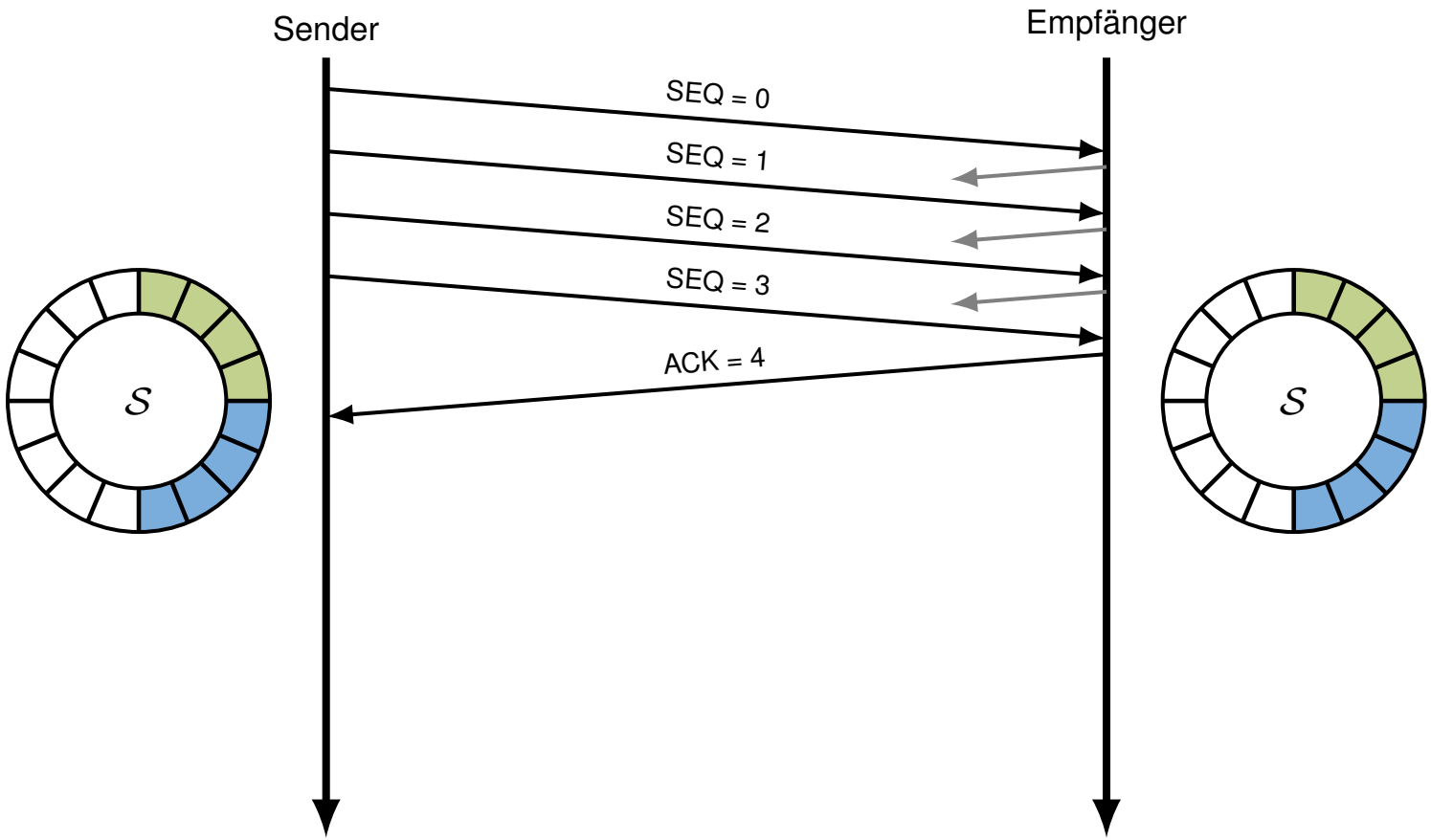


Sliding-Window-Verfahren

Sendefenster W_s bzw. Empfangsfenster W_r

gesendet aber noch nicht bestätigt

gesendet und bestätigt/empfangen

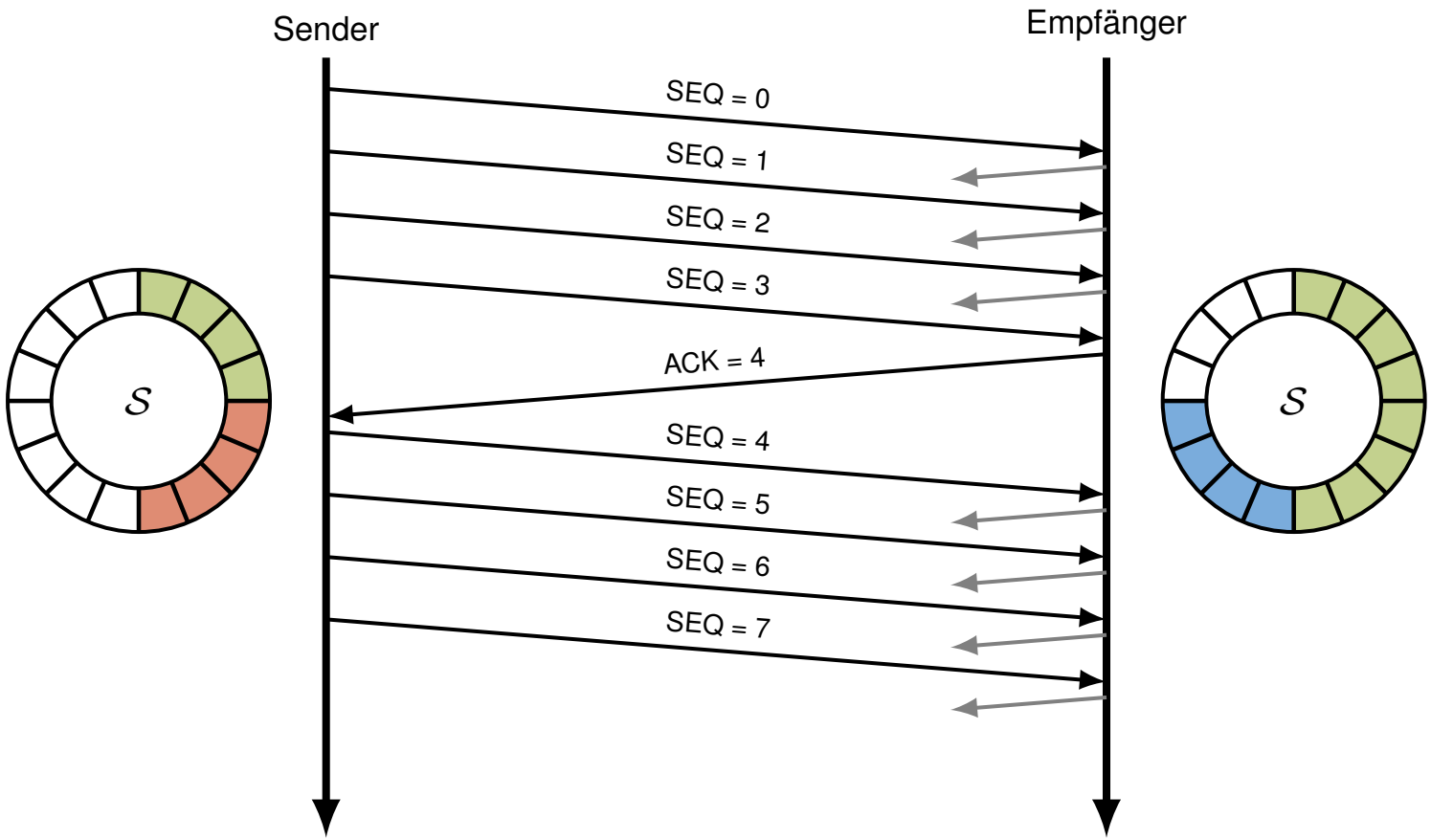


Sliding-Window-Verfahren

Sendefenster W_s bzw. Empfangsfenster W_r

gesendet aber noch nicht bestätigt

gesendet und bestätigt/empfangen

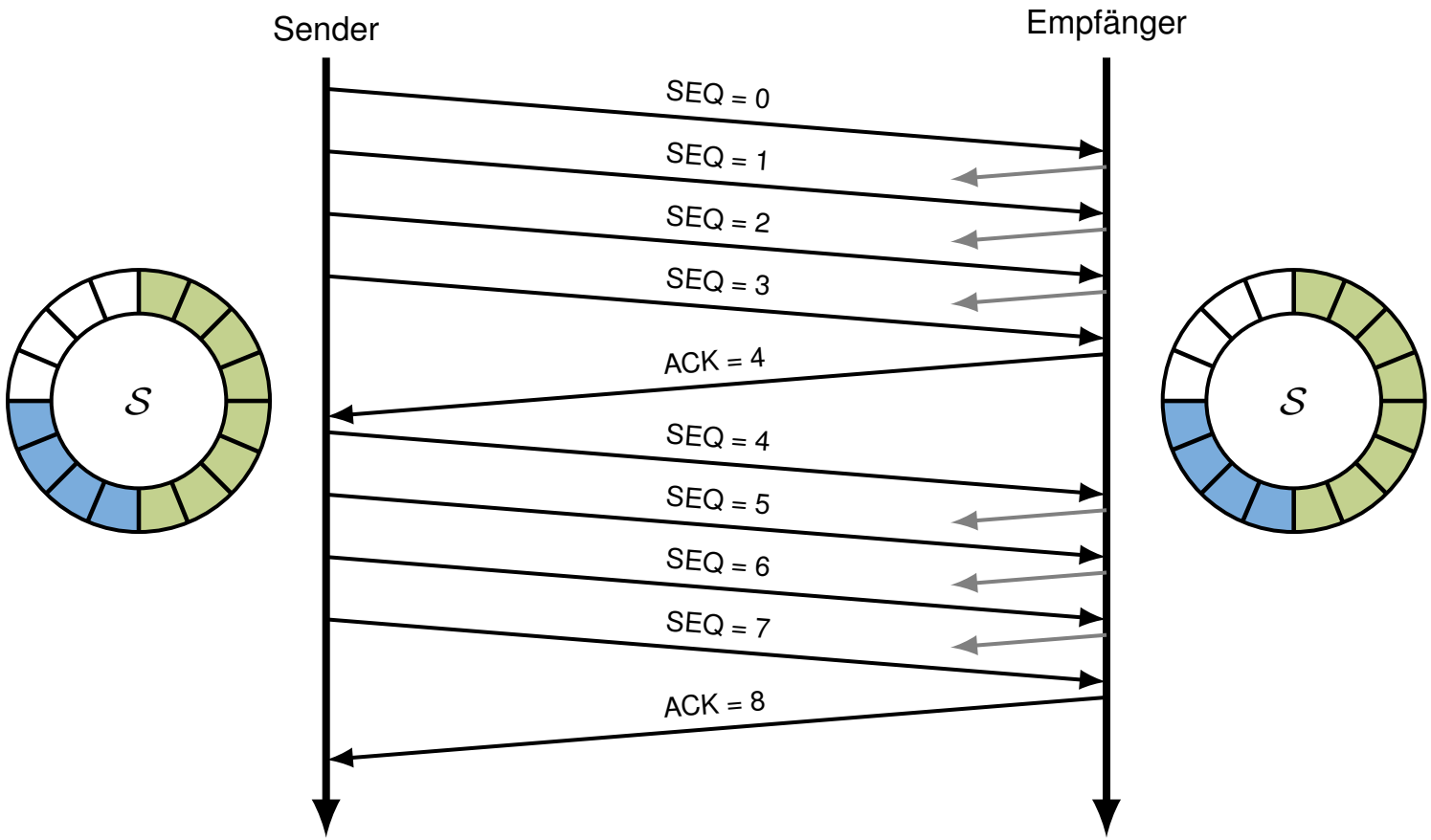


Sliding-Window-Verfahren

Sendefenster W_s bzw. Empfangsfenster W_r

gesendet aber noch nicht bestätigt

gesendet und bestätigt/empfangen



Sliding-Window-Verfahren

Neues Problem: Wie wird mit Segmentverlusten umgegangen?

Zwei Möglichkeiten:

1. Go-Back-N
   - Akzeptiere stets nur die nächste erwartete Sequenznummer
   - Alle anderen Segmente werden verworfen
2. Selective-Repeat
   - Akzeptiere alle Sequenznummern, die in das aktuelle Empfangsfenster fallen
   - Diese müssen gepuffert werden, bis fehlende Segmente erneut übertragen wurden

Wichtig:

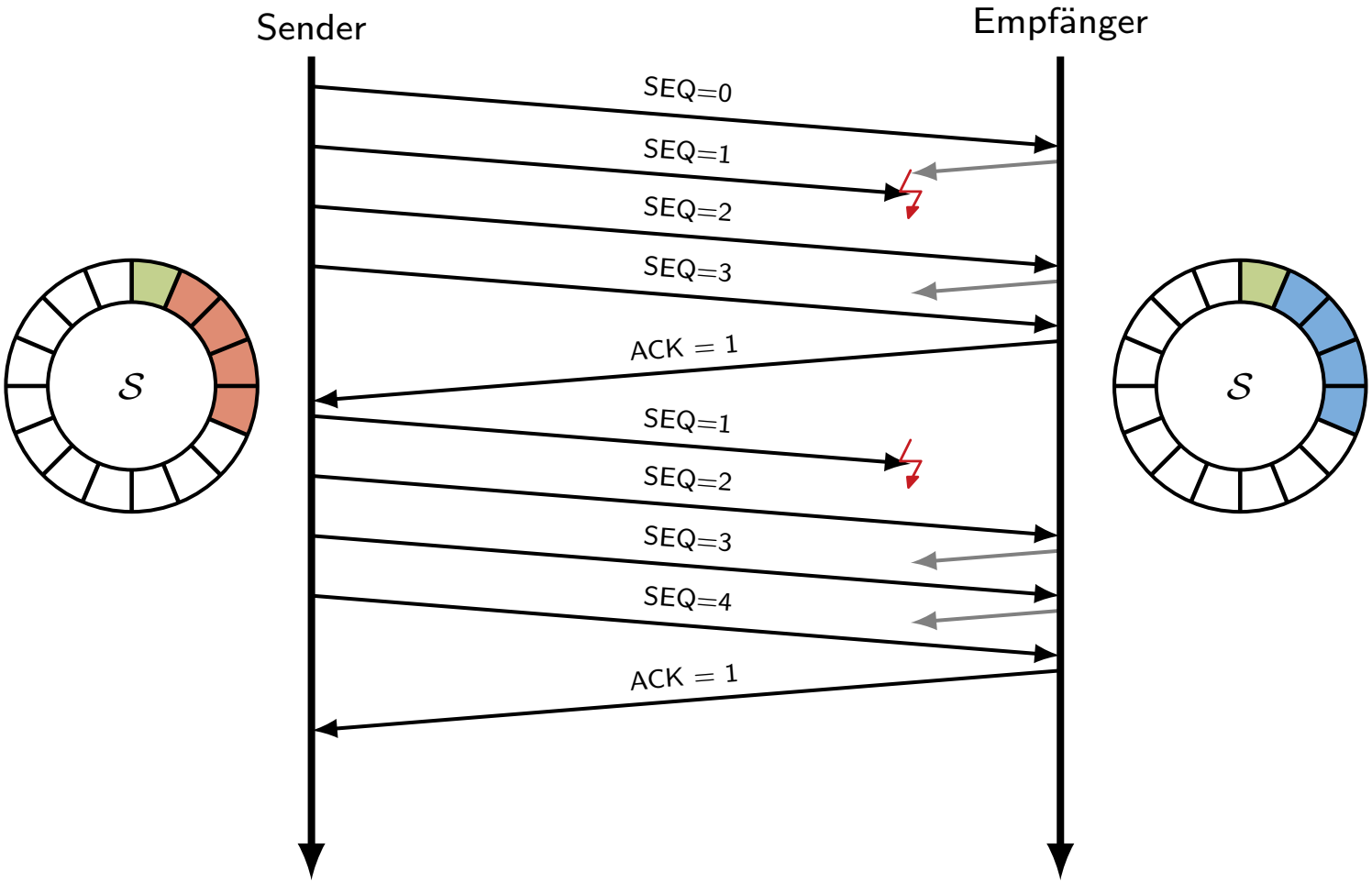
- In beiden Fällen muss der Sequenznummernraum so gewählt werden, dass wiederholte Segmente eindeutig von neuen Segmenten unterschieden werden können.
- Andernfalls würde es zu Verwechslungen kommen
  $\rightarrow$ Auslieferung von Duplikaten an höhere Schichten, keine korrekte Reihenfolge.

Frage: (siehe Übung)

Wie groß darf das Sendefenster W_s in Abhängigkeit des Sequenznummernraums $\mathcal{S}$ höchstens gewählt werden, so dass die Verfahren funktionieren?

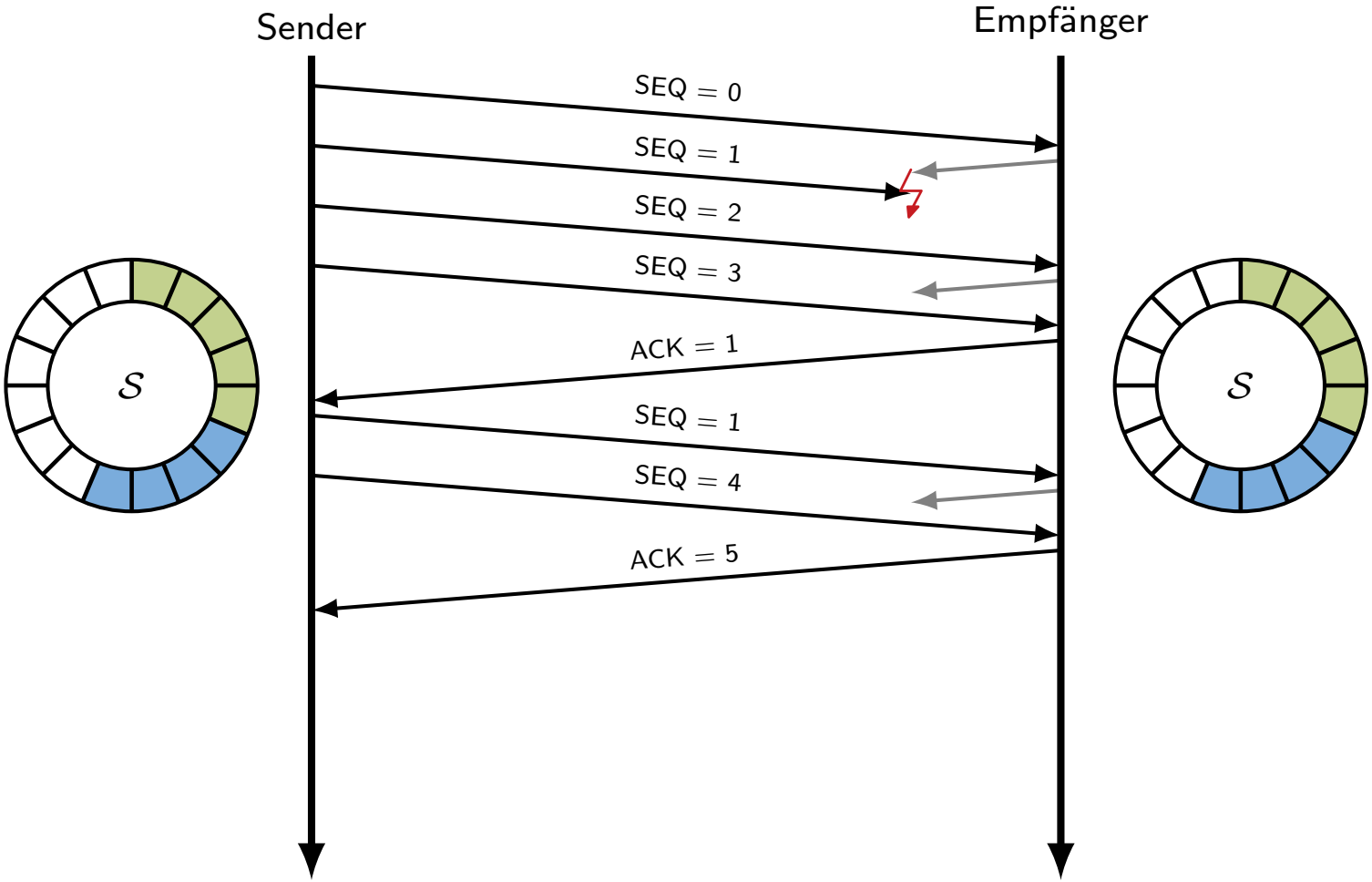
Sliding-Window-Verfahren

Go-Back-N: $N = 16$, $w_s = 4$, $w_r = 4$



Sliding-Window-Verfahren

Selective Repeat: $N = 16$, $w_s = 4$, $w_r = 4$

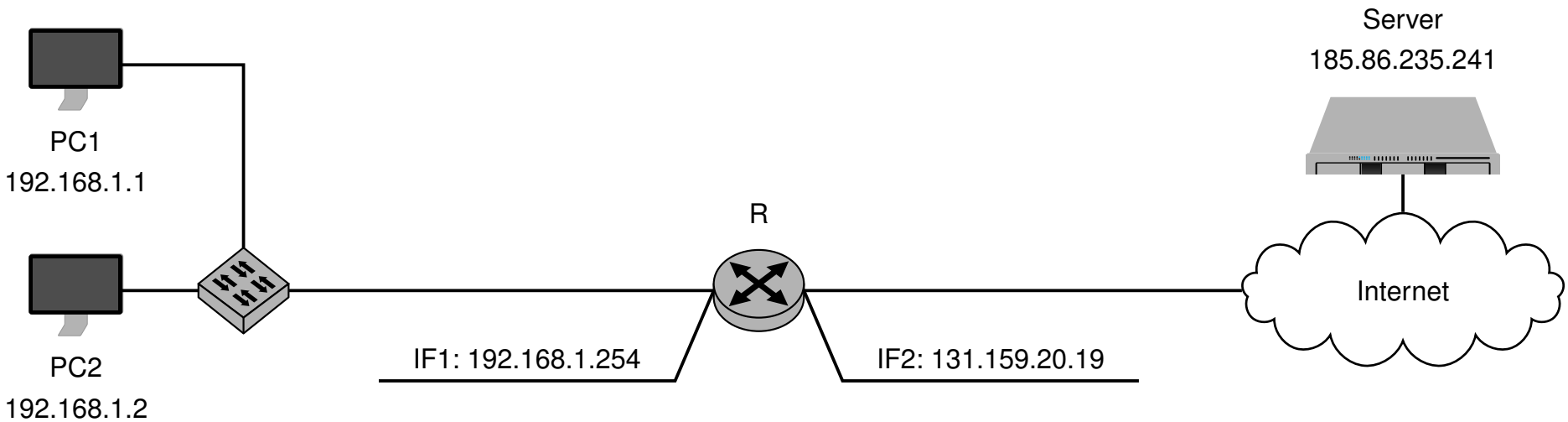


Kapitel 4: Transportschicht

Network Address Translation (NAT)

Wie funktioniert NAT im Detail?

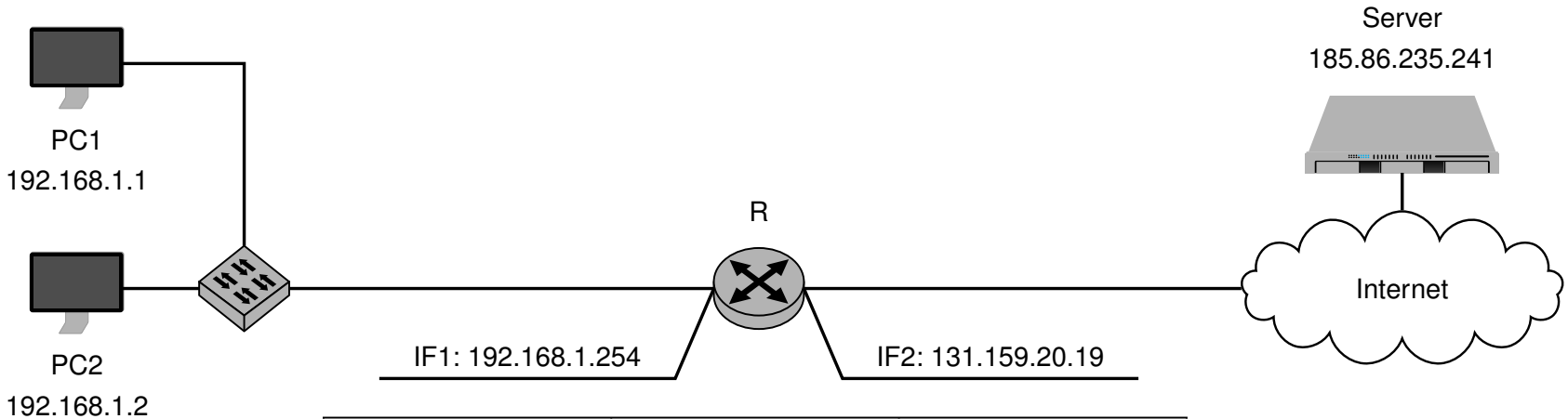
Üblicherweise übernehmen Router die Netzwerkadressübersetzung:



- PC1, PC2 und R können mittels privater IP-Adressen im Subnetz 192.168.1.0/24 miteinander kommunizieren.
- R ist über seine öffentliche Adresse 131.159.20.19 global erreichbar.
- PC1 und PC2 können wegen ihrer privaten Adressen nicht direkt mit anderen Hosts im Internet kommunizieren.
- Hosts im Internet können ebensowenig PC1 oder PC2 erreichen – selbst dann, wenn sie wissen, dass sich PC1 und PC2 hinter R befinden und die globale Adresse von R bekannt ist.

Network Address Translation (NAT)

PC1 greift auf eine Webseite zu, welche auf dem Server mit der IP-Adresse 185.86.235.241 liegt:

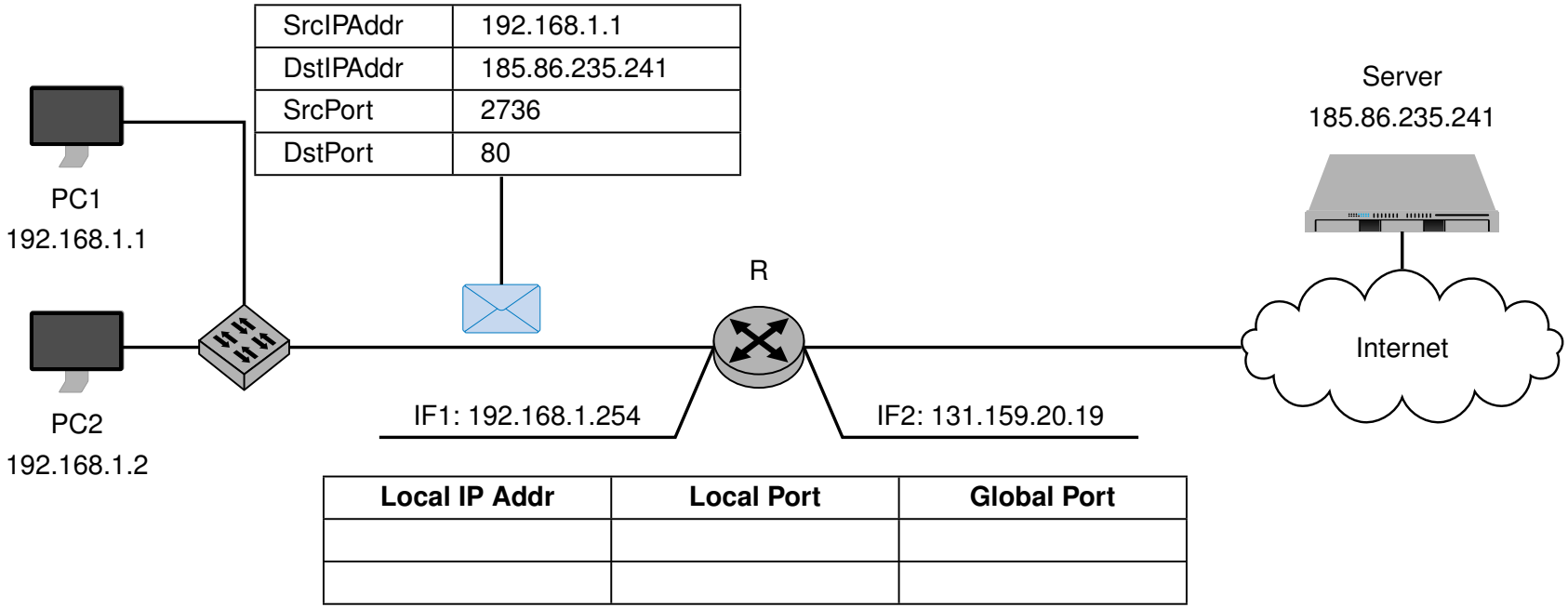


Local IP Addr	Local Port	Global Port

- Die NAT-Tabelle von R sei zu Beginn leer.

Network Address Translation (NAT)

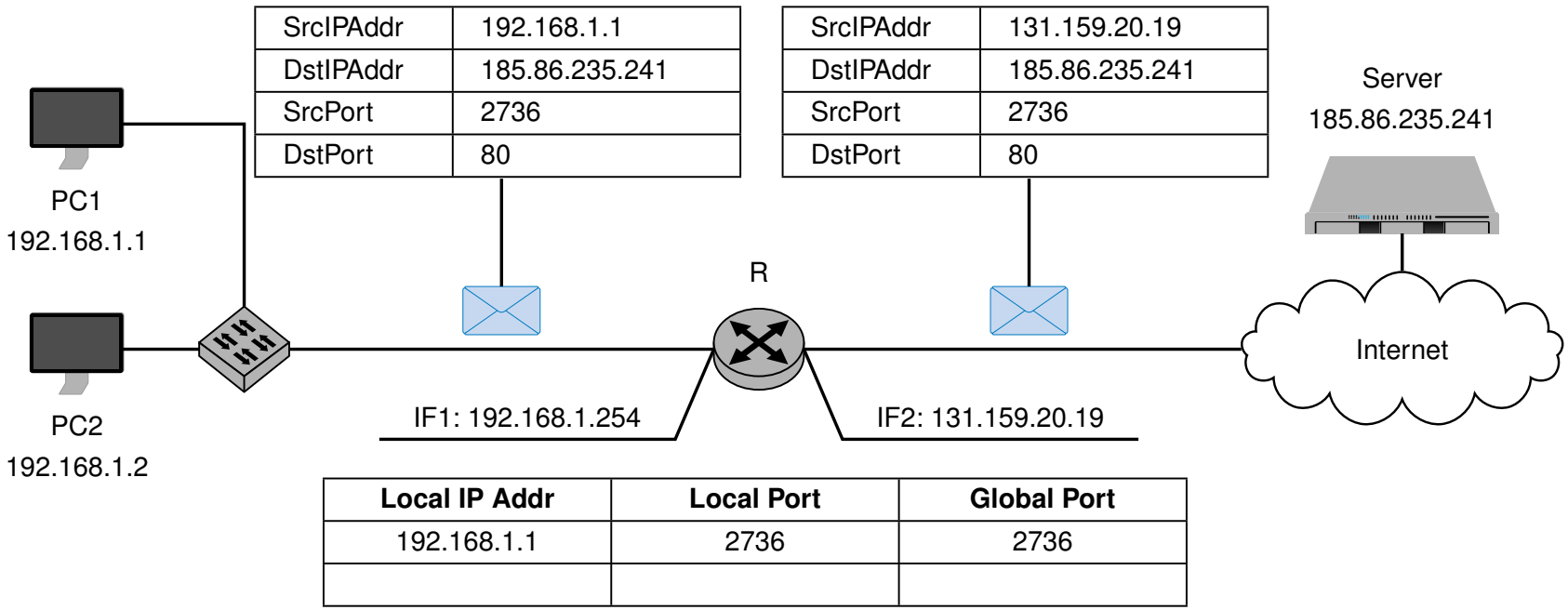
PC1 greift auf eine Webseite zu, welche auf dem Server mit der IP-Adresse 185.86.235.241 liegt:



- Die NAT-Tabelle von R sei zu Beginn leer.
- PC1 sendet ein Paket (TCP SYN) an den Server:
  - PC1 verwendet seine private IP-Adresse als Absenderadresse
  - Der Quellport wird von PC1 zufällig im Bereich [1024,65535] gewählt (sog. Ephemeral Ports)
  - Der Zielport ist durch das Application Layer Protocol vorgegeben (80 = HTTP)

Network Address Translation (NAT)

PC1 greift auf eine Webseite zu, welche auf dem Server mit der IP-Adresse 185.86.235.241 liegt:

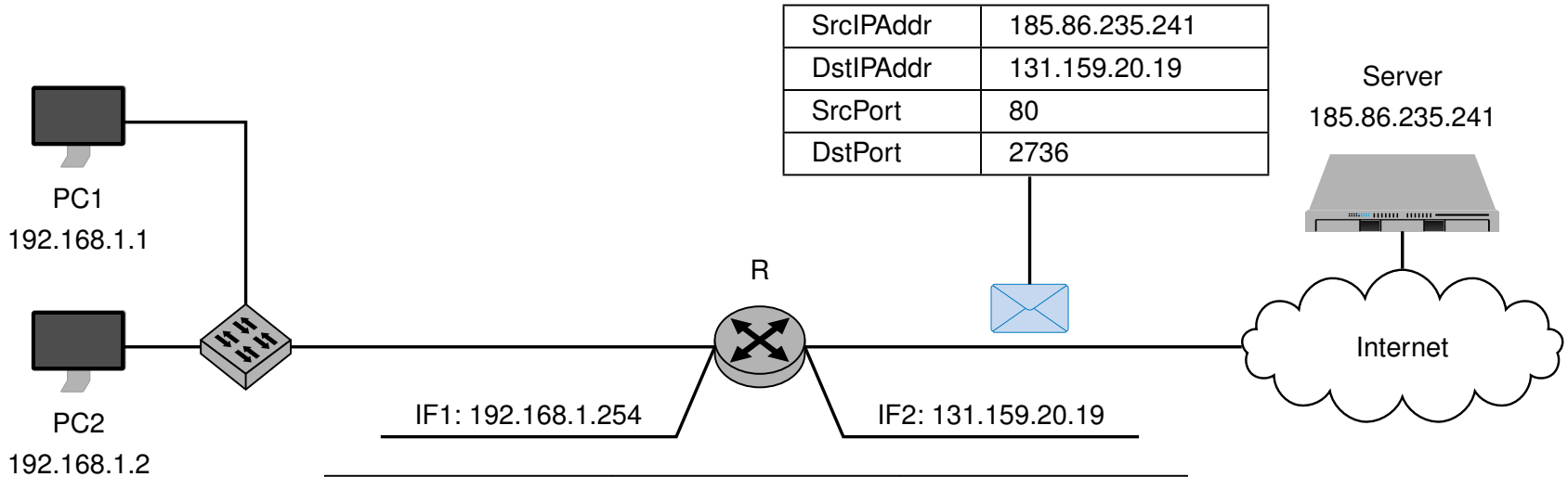


Local IP Addr	Local Port	Global Port
192.168.1.1	2736	2736

- Die NAT-Tabelle von R sei zu Beginn leer.
- PC1 sendet ein Paket (TCP SYN) an den Server:
  - PC1 verwendet seine private IP-Adresse als Absenderadresse
  - Der Quellport wird von PC1 zufällig im Bereich [1024,65535] gewählt (sog. Ephemeral Ports)
  - Der Zielport ist durch das Application Layer Protocol vorgegeben (80 = HTTP)
- Adressübersetzung an R:
  - R tauscht die Absenderadresse durch seine eigene globale Adresse aus
  - Sofern der Quellport nicht zu einer Kollision in der NAT-Tabelle führt, wird dieser beibehalten
  - R erzeugt einen neuen Eintrag in seiner NAT-Tabelle, welche die Änderungen an dem Paket dokumentieren

Network Address Translation (NAT)

Antwort vom Server an PC1

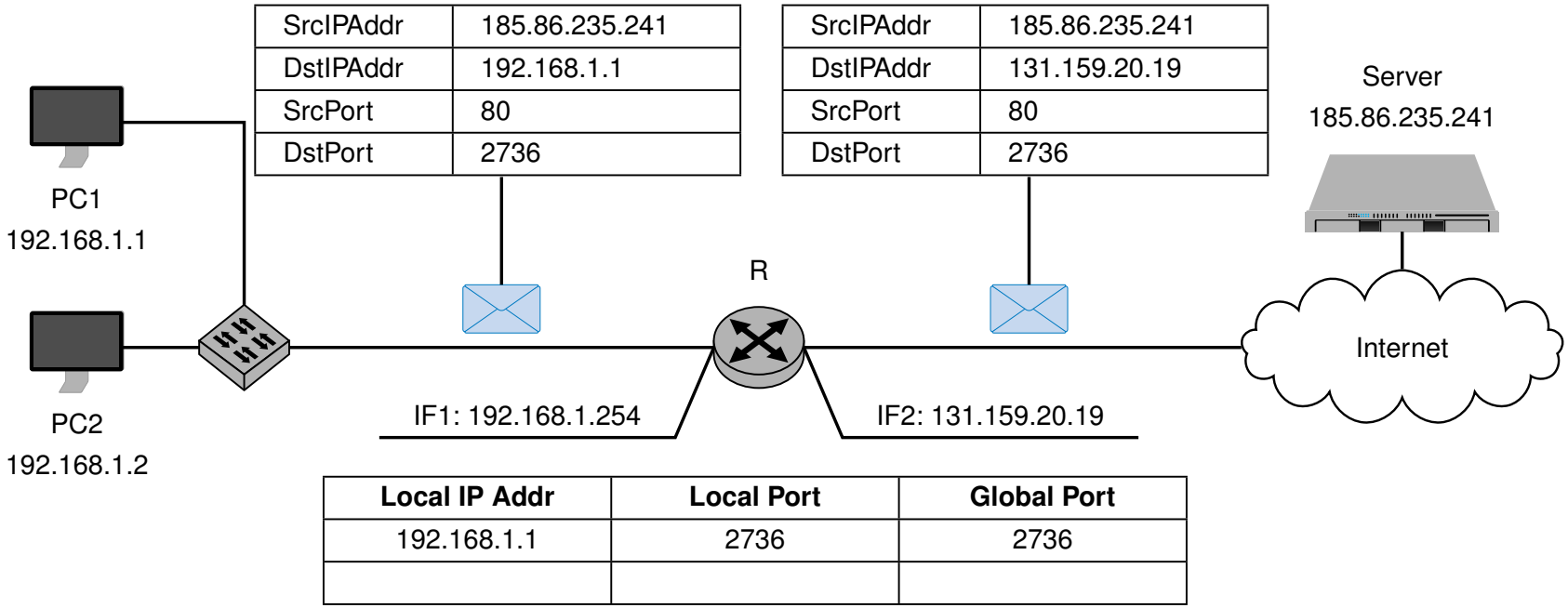


Local IP Addr	Local Port	Global Port
192.168.1.1	2736	2736

- Der Server generiert eine Antwort:
  - Der Server weiß nichts von der Adressübersetzung und hält R für PC1.
  - Die Empfängeradresse ist daher die öffentliche IP-Adresse von R, der Zielport der von R übersetzte Quellport aus der vorherigen Nachricht.

Network Address Translation (NAT)

Antwort vom Server an PC1

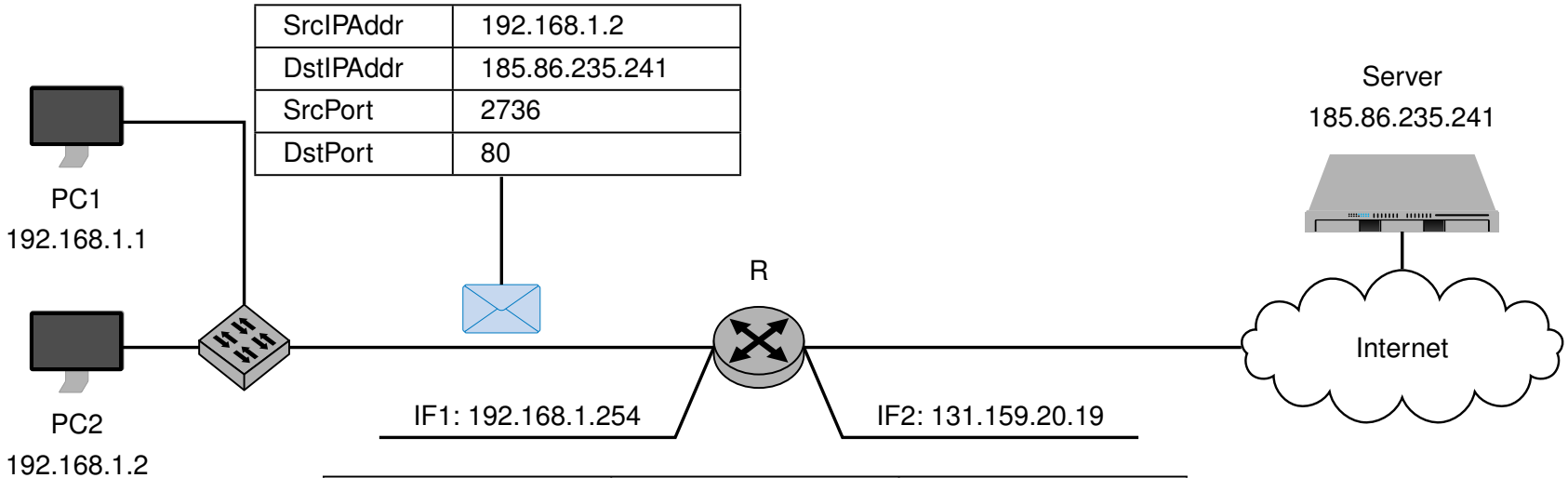


Local IP Addr	Local Port	Global Port
192.168.1.1	2736	2736

- Der Server generiert eine Antwort:
  - Der Server weiß nichts von der Adressübersetzung und hält R für PC1.
  - Die Empfängeradresse ist daher die öffentliche IP-Adresse von R, der Zielport der von R übersetzte Quellport aus der vorherigen Nachricht.
- R macht die Adressübersetzung rückgängig:
  - In der NAT-Tabelle wird nach der Zielportnummer in der Spalte Global Port gesucht, dieser in Local Port zurückübersetzt und die Ziel-IP-Adresse des Pakets gegen die private IP-Adresse von PC1 ausgetauscht.
  - Das so modifizierte Paket wird an PC1 weitergeleitet.
  - Wie der Server weiß auch PC1 nichts von der Adressübersetzung.

Network Address Translation (NAT)

PC2 greift nun ebenfalls auf den Server zu:

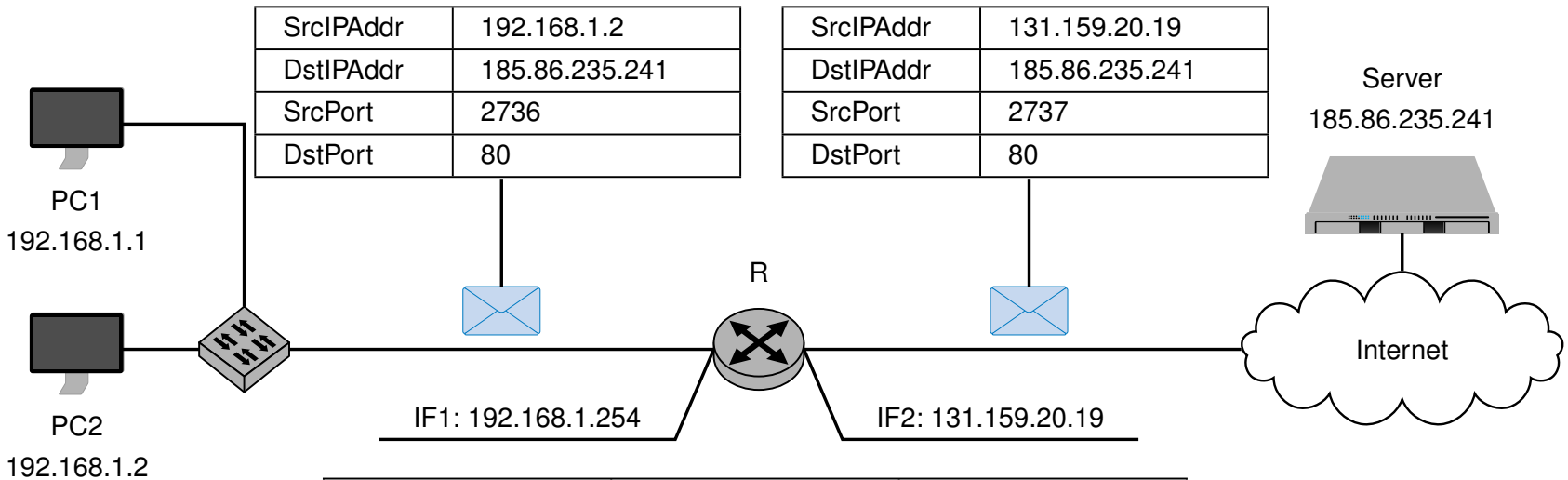


Local IP Addr	Local Port	Global Port
192.168.1.1	2736	2736

- PC2 sendet ebenfalls ein Paket (TCP SYN) an den Server:
  - Rein zufällig wählt PC2 denselben Quell-Port wie PC1 (Portnummer 2736)

Network Address Translation (NAT)

PC2 greift nun ebenfalls auf den Server zu:



Local IP Addr	Local Port	Global Port
192.168.1.1	2736	2736
192.168.1.2	2736	2737

- PC2 sendet ebenfalls ein Paket (TCP SYN) an den Server:
  - Rein zufällig wählt PC2 denselben Quell-Port wie PC1 (Portnummer 2736)
- Adressübersetzung an R:
  - R bemerkt, dass es bereits einen zu PC1 gehörenden Eintrag für den lokalen Port 2736 gibt
  - R erzeugt einen neuen Eintrag in der NAT-Tabelle, wobei für den globalen Port ein zufälliger Wert gewählt wird (z. B. der ursprüngliche Port von PC2 + 1)
  - Das Paket von PC2 wird entsprechend modifiziert und an den Server weitergeleitet
- Aus Sicht des Servers hat der „Computer“ R einfach zwei TCP-Verbindungen aufgebaut.

Ablauf:

- Aufgabe 1
- Aufgabe 2
- Aufgabe 3

Studenten zählen

(Nur als Erinnerung für mich.)