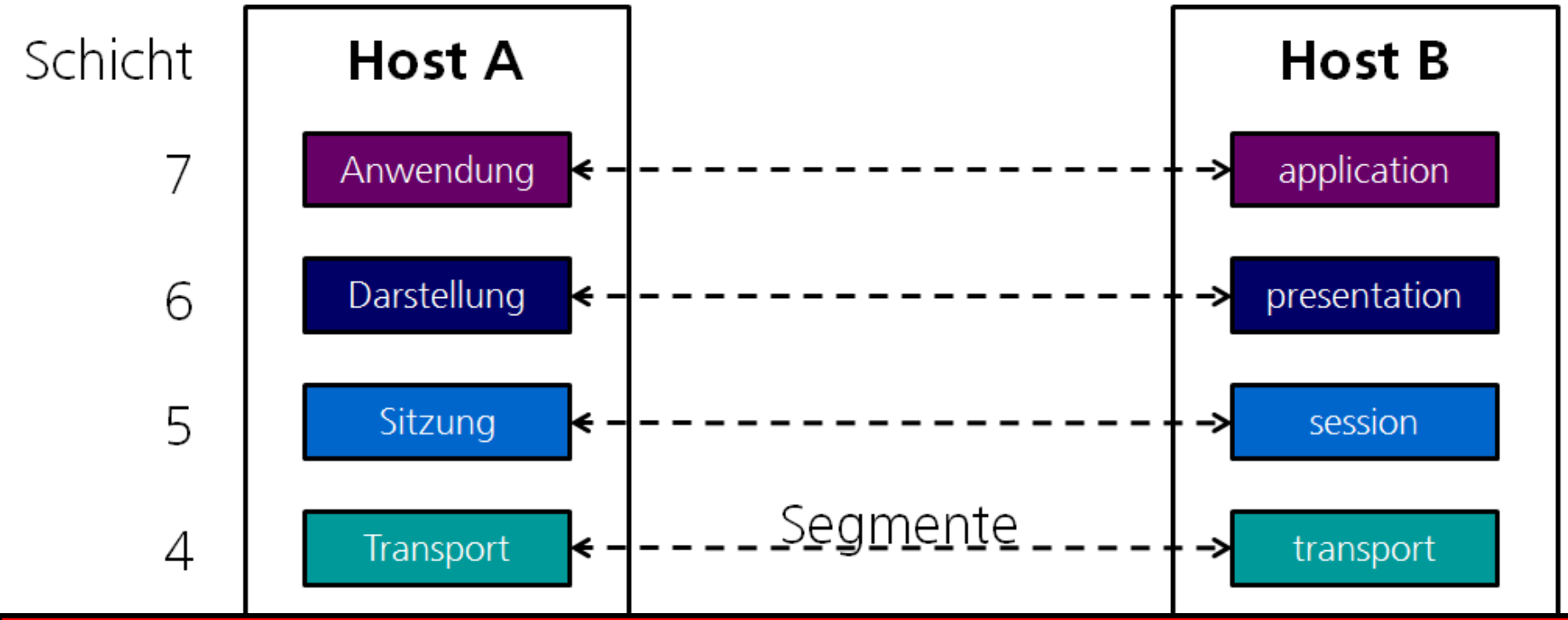
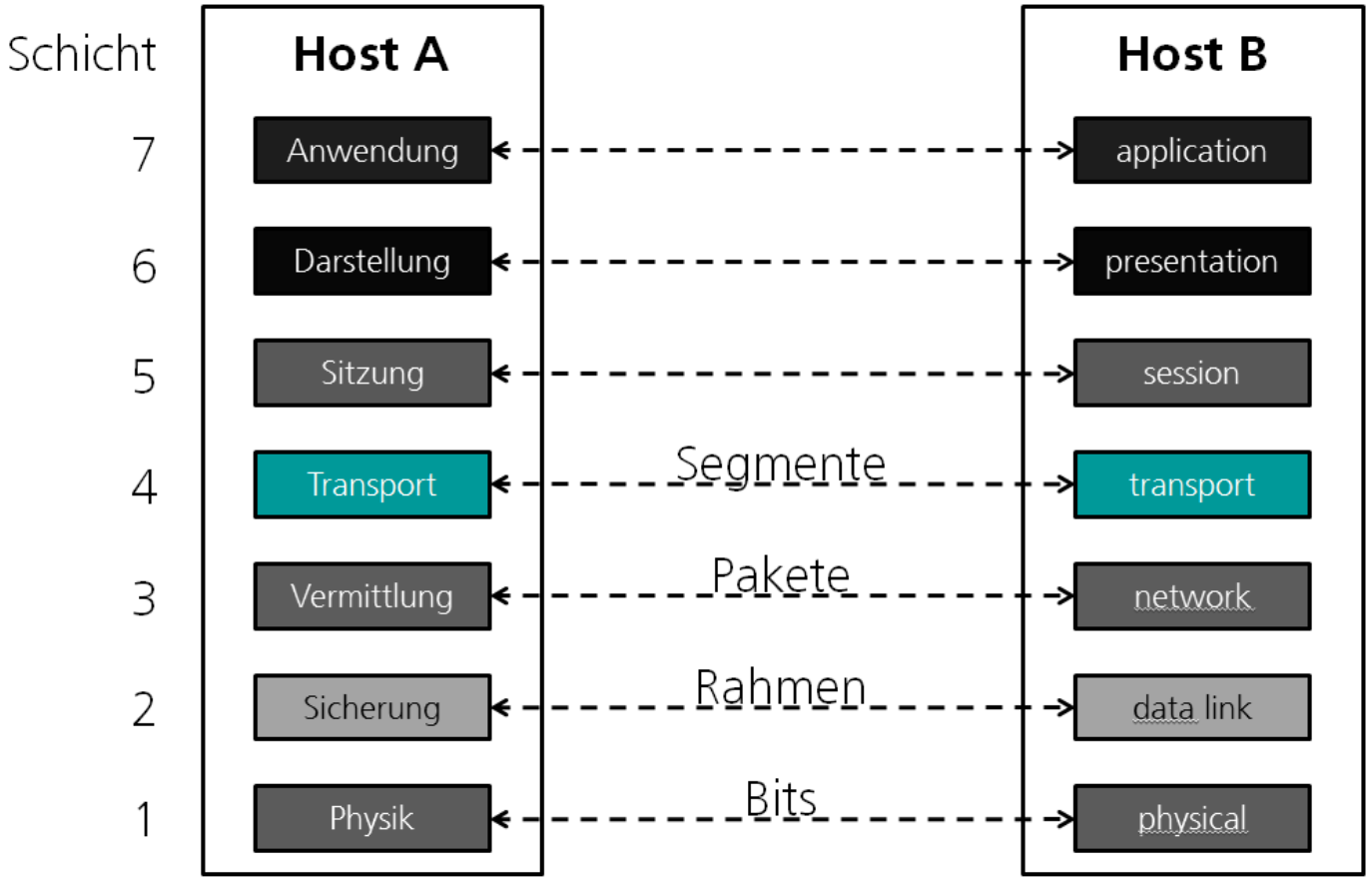


Transportschicht bis Anwendungsschicht



Schicht 4 - Transportschicht

Schicht 4: Transportschicht

Die **Aufgaben** der Transportschicht sind:

- **Datenströme** der übergeordneten Schichten zu **segmentieren**, d.h. in Pakete aufzuteilen, die über die Vermittlungsschicht zum Zielknoten geleitet werden können,
- **Prozessen** auf einem Knoten zu **adressieren**,
- **virtuelle Verbindungen** in paketvermittelnden Netzen bereitzustellen,
- **Fehlererkennung**, **Fehlerkorrektur** und **Flusskontrolle** bereitzustellen.

Schicht 4: Transportschicht

Hinweis: In leitungsvermittelnden Netzen werden die meisten Aufgaben der Transportschicht nicht benötigt, sie ist auf **paketvermittelnde** Netze zugeschnitten.

Adressierung: Ports und Sockets

- Adressen der Vermittlungsschicht bestimmen einen **Knoten** eindeutig, jedoch muss ein eintreffendes Paket letztlich einer der laufenden **Anwendungen** auf dem Knoten zugeordnet werden können.
- Daher wird auf der Transportschicht jeder lokalen Anwendung, die die Transportschicht nutzt, ebenfalls eine Adresse zugeordnet.
- Diese Adresse ist eine 16-Bit Zahl und wird als **Port** (Hafen, Anschluss) bezeichnet. Beispiele für Standardports:
  - HTTP-Server: Port 80
  - SMTP-Server: Port 25
  - DNS-Server: Port 53
- Als **Socket** (Buchse) wird ein Paar aus IP-Adresse für den Knoten und Port-Nummer für die Anwendung bezeichnet.

TCP und UDP

Im Internet haben sich zwei Transportprotokolle durchgesetzt:

- Das **Unified-Datagram-Protocol (UDP)** stellt nur Segmentierung von Datenströmen und Adressierung von Anwendungen bereit. UDP-Pakete können verloren gehen, Reihenfolgeerhalt ist nicht garantiert.

- Darüber hinaus stellt das **Transmission-Control-Protocol (TCP)** die Zustellung und den Reihenfolgeerhalt sicher. Es besitzt außerdem Mechanismen zur Flusskontrolle.

UDP ist schnell aber unzuverlässig, TCP ist durch die Zustellungszusicherung langsamer aber zuverlässiger. UDP wird daher meist bei paketfehlertoleranten Anwendungen eingesetzt (bspw. Videostreaming).

Transmission-Control-Protocol – Grundlagen

- TCP sicher durch Rückbestätigung die Zustellung von Segmenten zu, es wird also eine **virtuelle Verbindung** bereitgestellt.
- Die **Datenmenge** der übertragenen Nutzlast wird **gezählt**. Jedes gesendete Paket enthält eine **Sequenznummer**, die die fortlaufende Nummer des ersten Bytes der Nutzlast enthält (Sequenznummer).
- Der Empfänger bestätigt Pakete, indem er die Sequenznummer des **nächsten** erwarteten Paketes sendet.

Transmission-Control-Protocol – Segmentgröße

- Ein TCP-Segment muss vollständig in ein IP-Paket passen, das wiederum vollständig in ein Layer-2-Rahmen passen muss.
- Ethernet erlaubt im Standard maximal 1500 Bytes Nutzlast. Davon gehen durch den IPv4-Header*) 20 Byte verloren (40 Byte bei IPv6).
- Der TCP-Header selbst ist ebenfalls 20 Byte groß.
- Die effektiv nutzbare maximale **Nutzlastgröße** beträgt daher in der Praxis **1460 Bytes**.
- Da bei DSL weitere 8 Byte auf Schicht 2 benötigt werden, stehen hier nur 1452 Byte für TCP-Nutzlast zur Verfügung.

*) Als Header werden die protokollspezifischen Daten zu Beginn eines Pakets (bzw. Rahmens, Segments, etc) bezeichnet, die keine Nutzlast enthalten..

Verbindungsaufbau und 3-Wege-Handschlag

TCP-Server warten im laufenden Betrieb kontinuierlich darauf, dass Clients diese unter einem Standardport kontaktieren, um eine Verbindung aufzubauen. Der Verbindungsaufbau erfolgt also auf Anfrage von Clients.

Problem: Auf der Vermittlungsschicht können Pakete verloren gehen. Pakete zum Verbindungsaufbau erreichen ggf. den Server nicht.

Lösungsansatz: Der Client lässt sich sein Paket zum Aufbau der Verbindung durch den Server bestätigen.

Folgeproblem: Bestätigungspakete können ebenfalls verloren gehen.

Lösung?

Verbindungsaufbau und 3-Wege-Handschlag

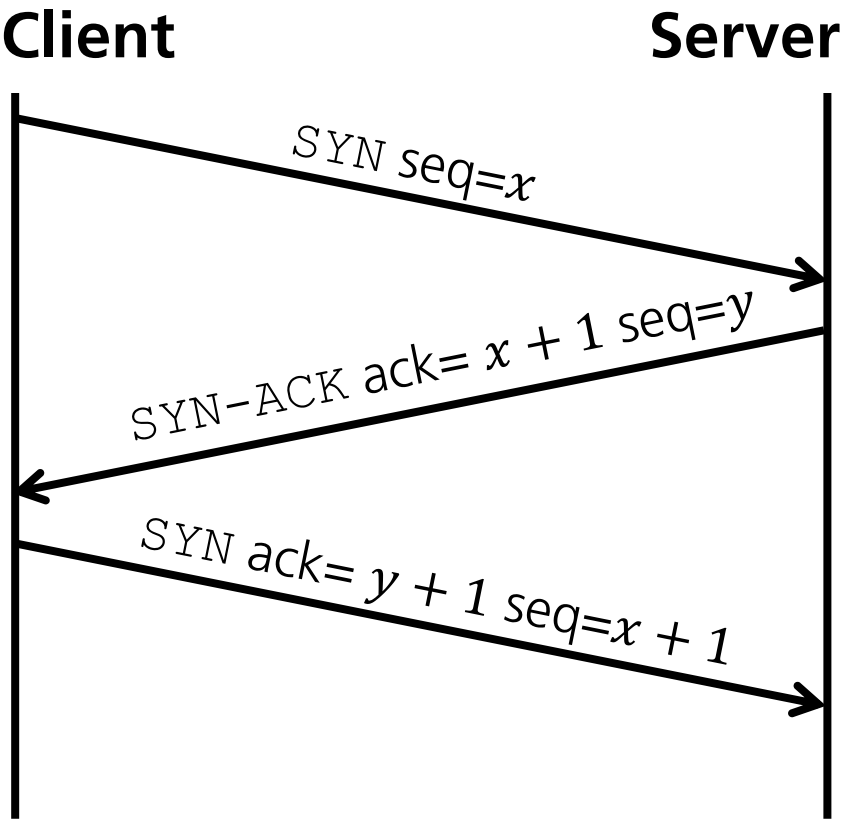
Es gibt keine endgültige Lösung für dieses Problem, mit der ein Verbindungsaufbau garantiert werden kann![*)] TCP nimmt daher einen **Abbruch** in Kauf:

- Die Verbindung gilt erst dann als aufgebaut, wenn sich dies Client und Server **gegenseitig bestätigt** haben.
- Es wird ein 3-Wege-Handschlag aus **Anfrage**, **Bestätigung** und **Gegenbestätigung** durchgeführt.
- Geht eines dieser drei Segmente verloren, bricht der Verbindungsaufbau nach einem **Timeout** ab, die Verbindung kommt nicht zustande.

[*)] Es handelt sich hier um ein 2-Armeen-Problem. Dieses ist nicht lösbar.

3-Wege-Handschlag

- Client sendet SYN- Segment mit zufälliger Sequenznummer x an Server.
- Server bestätigt mit einem SYN-ACK Segment , das neben $x + 1$ auch eine eigene, zufällige Sequenz-nummer y enthält.
- Client sendet ACK- Segment mit Sequenznummer $x + 1$ und bestätigt das SYN-ACK mit $y + 1$.



- Ab diesem Zeitpunkt ist die Verbindung aufgebaut. Client und Server sind jetzt gleichberechtigte Kommunikationspartner, jeder kann senden und empfangen.

Datenübertragung und Sequenznummern

Sender:

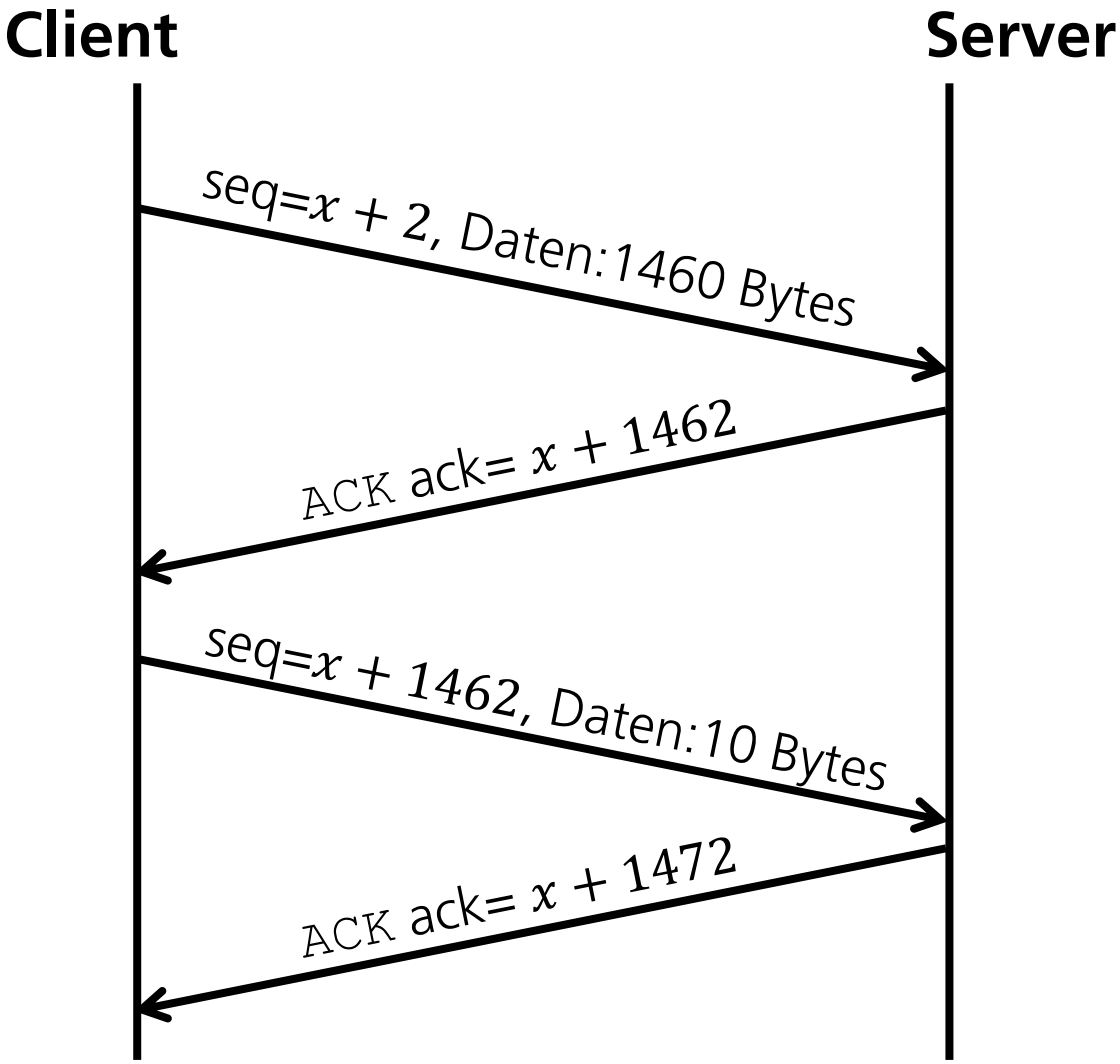
- Beim ersten Segment nach Verbindungsaufbau wird als Sequenznummer $x + 2$ verwendet (bzw., $y + 1$ aus Sicht des „Servers“).

- Alle weiteren Segmente erhalten als Sequenznummer die Sequenznummer des vorangegangenen Segmentes **plus** die Größe der darin übertragenen **Nutzlast** in Bytes (maximal 1460).

Empfänger:

- Der Empfänger **bestätigt** ein Segment, indem er ein `ACK`-Segment sendet, dass die erwartete Sequenznummer des **nachfolgenden** Segmentes enthält.

Datenübertragung und Sequenznummern



Sequenznummern

- Intuitiv würde man vermuten, dass die Sequenznummer mit 1 beginnt und dann entsprechend der Menge der gesendeten Nutzdaten hochgezählt wird.
- Tatsächlich aber wird die initiale Sequenznummer **zufällig gewählt** („gewürfelt“).
- **Grund:** Ein Angreifer, der die laufende Sequenznummer kennt, kann seinerseits scheinbar vom Originalsender stammende gültige Segmente an den Empfänger senden und von diesem verarbeiten lassen (*TCP sequence prediction attack*). Aus diesem Grund darf die initiale Sequenznummer nicht vorhersagbar sein. *)

*) Es gibt weitere Gegenmaßnahmen, die hier aber nicht besprochen werden.

Paketverlust und erneute Übertragung

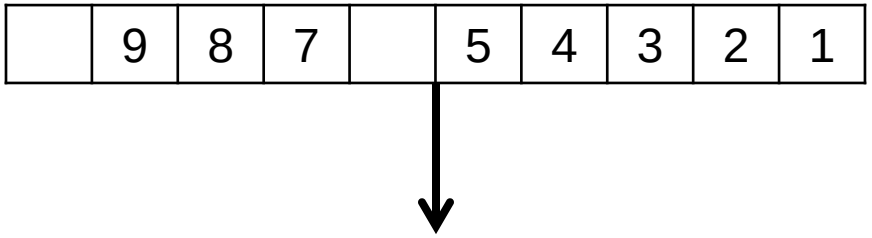
- Der Sender vermerkt den **Sendezeitpunkt** jedes Pakets.
- Nach Ablauf einer als **Retransmission Timeout (RTO)** bezeichneten Wartezeit wird das Segment als verloren angenommen und erneut gesendet.
- Ein geeigneter Wert für den RTO muss **dynamisch** ermittelt werden, denn erhängt von den Eigenschaften des **Übertragungswegs** ab.
  - Ein zu kleiner Wert würde bei langsameren Verbindungen zum erneuten Senden führen, obwohl der Empfänger das Segment erhalten und bestätigt hat.
  - Ein zu hoher Wert würde bei schnelleren Verbindungen zu unnötigen Wartezeiten im Fehlerfall führen.

Kumulative Bestätigung

- Auf die Bestätigung jedes einzelnen Segments zu warten, bevor das nächste Segment gesendet wird (*Stop-and-Wait*), führt zu einer schlechten Ausnutzung des Übertragungskanals.
- Daher darf der Sender **mehrere Segmente** hintereinander senden (*Burst*). Der Empfänger bestätigt kumulativ, d.h. nur das letzte Segment eines Bursts wird bestätigt.
- Der Empfänger kann also das letzte Segment vor einer Lücke bestätigen und bestätigt damit automatisch auch alle vorangegangenen Segmente. Die Segmente hinter der Lücke werden jedoch nicht bestätigt. Dies geschieht erst, wenn die Lücke gefüllt wird.

Kumulative Bestätigung

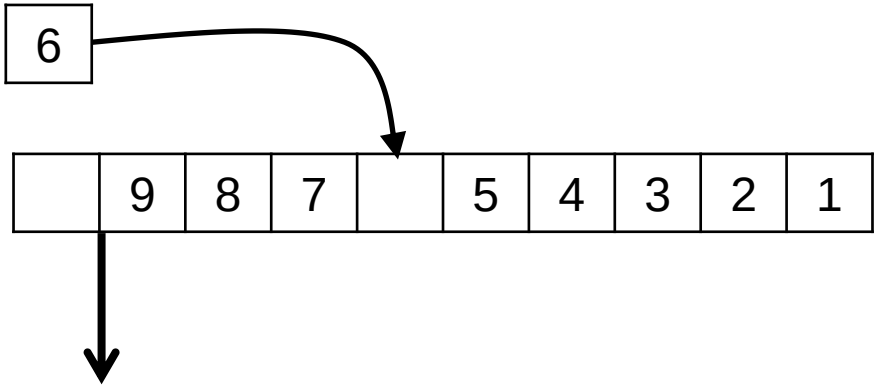
Puffer des Empfängers



ACK ack=Segment 5

1. Empfänger hat Segmente 1-9 außer 6 empfangen

2. Er bestätigt kumulativ bis Segment 5



ACK ack=Segment 9

3. Segment 6 wird empfangen

4. Bis Segment 9 wird kumulativ bestätigt

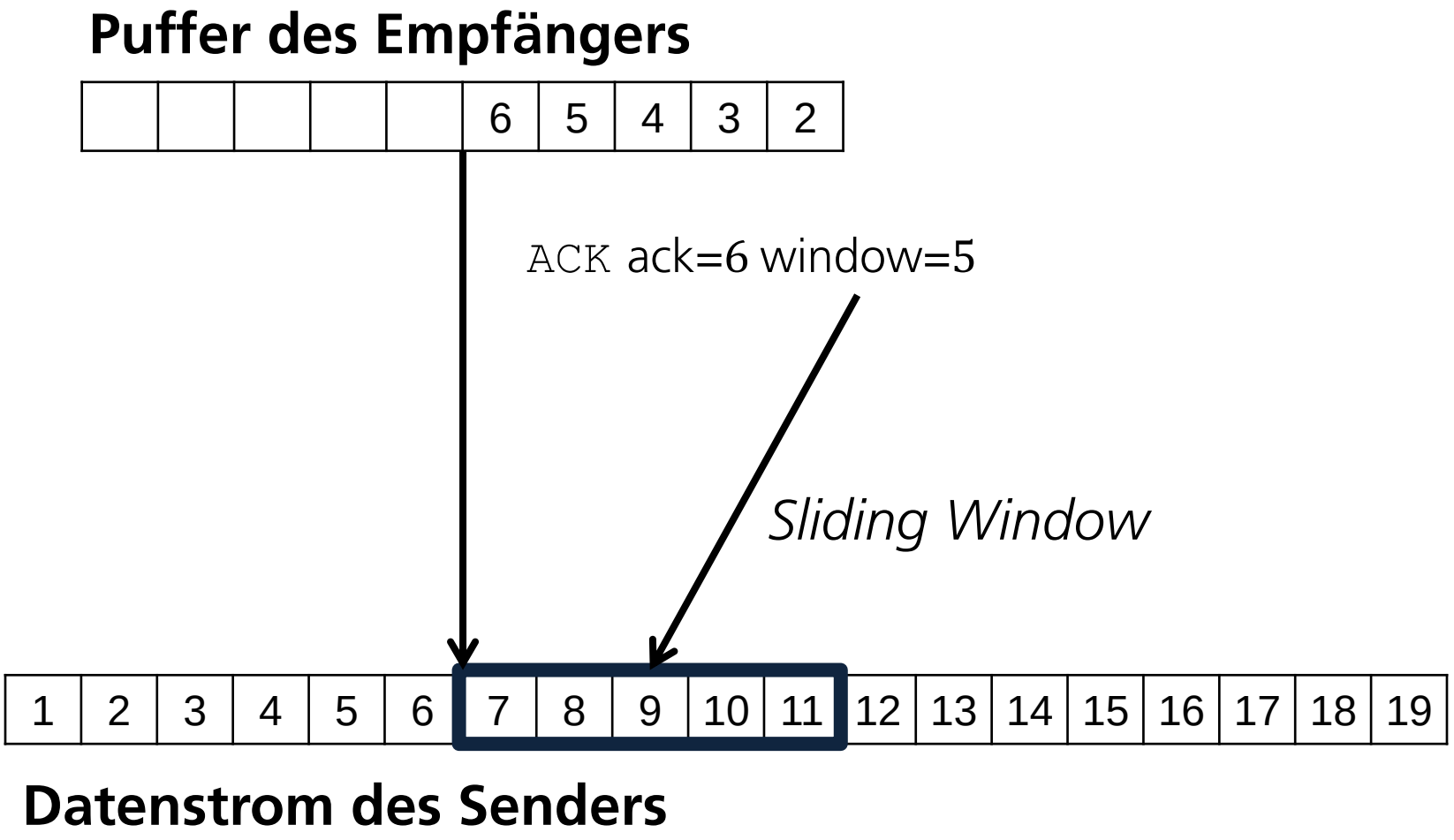
Kumulative Bestätigung

- Kumulative Bestätigung ermöglicht eine erheblich bessere Auslastung des Kanals.
- **Aber:** Wie viele Segmente darf der Sender im Voraus verschicken?
  - Nur so viele, dass die Bestätigungen aller Pakete rechtzeitig ankommen können (hängt von RTO und den Kanaleigenschaften ab).
  - Nur so viele, wie der Empfänger **verarbeiten** oder **zwischenspeichern** kann.
- Eine **Flusskontrolle** wird benötigt! TCP bietet zwei voneinander unabhängige Verfahren: *Sliding Window* und *Congestion Control*.

Flusskontrolle durch Sendefenster

- Der Empfänger kann bei der Bestätigung eines Segmentes angeben, wie viele Daten (in Bytes) er aktuell noch zu empfangen bereit ist, bevor sein Empfangspuffer überläuft. Dieser Wert wird als **Window** bezeichnet.
- Der Sender kann basierend auf dieser Information den nächsten Burst zusammenstellen und senden.
- Der **Empfänger** steuert also den Informationsfluss.
- Da der Sender mit dem vom Empfänger mitgeteilten Sendefenster den zu sendenden Datenstrom überstreicht, spricht man auch von einem **Sliding Window** (Schiebefenster).

Flusskontrolle durch Sendefenster



Überlaststeuerung (Congestion Control)

- **Router** auf Schicht 3 können einige Pakete **zwischenspeichern**, sofern der Kanal, auf dem sie weitergeleitet werden sollen, kurzfristig nicht zur Verfügung steht. Ist dieser Zwischenspeicher voll, werden weitere ankommende Pakete **stillschweigend verworfen** *).
- TCP „erkennt“ diesen Zustand durch erhöhte **Paketverlustraten** (nicht bestätigte Pakete) und reduziert die Senderate.
- Durch **langsames Steigern** der Senderate tastet sich TCP an die Überlastungsgrenze heran.

*) Zu große Puffer führen zu negativen Effekten, siehe *Bufferbloat*

Grundvariante: Slow-Start & Congestion Avoidance

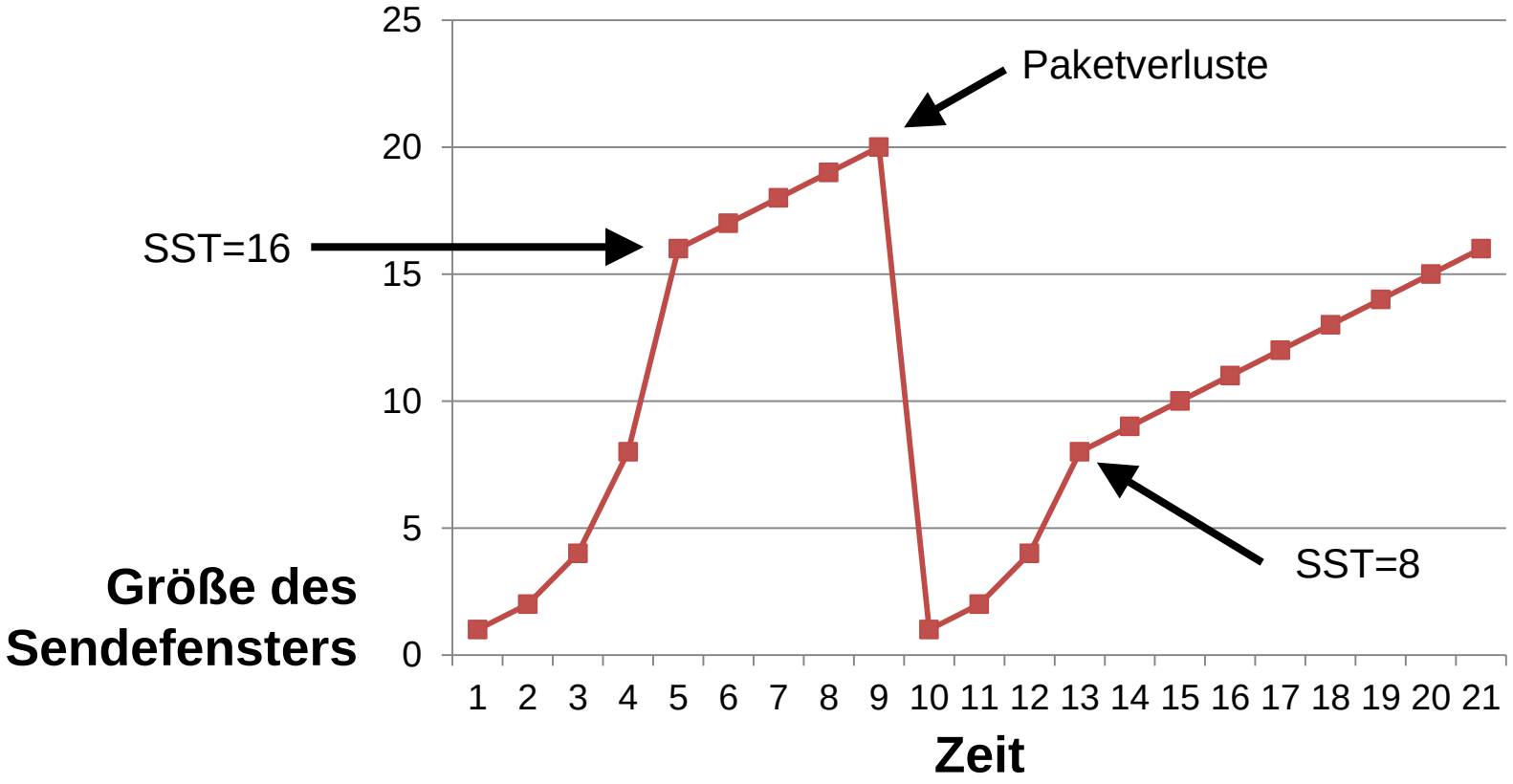
Aufgabe: Der Algorithmus tastet sich schrittweise an die **Größe des Sendefensters** heran, bei der Überlast auftritt (*Congestion Window Size,* CWND).

- Der Startwert von CWND ist mit **einer** (maximalen) **Segmentgröße** (daher *slow-start*) festgelegt.
- Mit jedem **bestätigten** Paket wird CWND um eine Segmentgröße erhöht, wächst also exponentiell[*)].
- Ab einem **Schwellwert** (*slow-start-threshold*, SST) wird CWND nur noch um 1 erhöht. Dies geschieht nur, wenn **alle Pakete** aus einem Fenster **bestätigt** wurden. Diese Phase wird als *Congestion-Avoidance* bezeichnet (Überlastvermeidung).

*) Übungsfrage: Warum exponentiell und nicht linear?

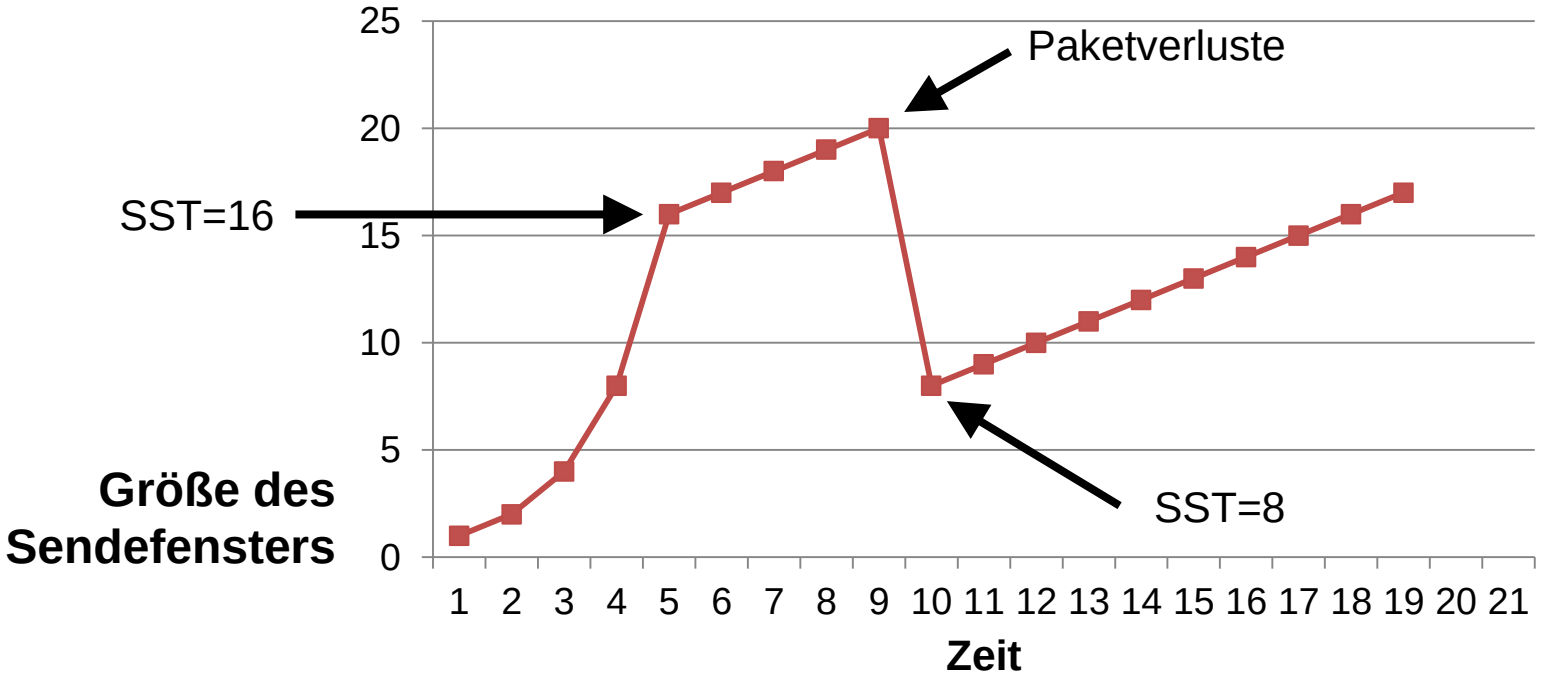
Grundvariante: Slow-Start & Congestion Avoidance

- Treten Paketverluste auf, wird der **Schwellwert** SST **halbiert** und der Algorithmus beginnt von neuem mit CWND=1.



Überlaststeuerung – Optimierung

- Die beschriebene Grundvariante ist durch das Zurücksetzen von CWND auf **eine** Segmentgröße recht ineffizient.
- **Verbesserung:** Rücksetzen auf $SST/2$ und von dort aus direkt *Congestion Avoidance* anwenden (Erhöhen von CWND nur um eine Segmentgröße).



Überlaststeuerung – Hinweise

- *Congestion-Control* und *Sliding-Window* werden oft verwechselt, beschreiben aber voneinander zunächst **unabhängige Aspekte** der Flusssteuerung:
  - *Congestion-Control* erkennt Überlast bei der **Übertragung**.
  - Das *Sliding-Window* vermeidet Überlast beim **Empfänger**. Ist dieses Fenster ausgeschöpft, ohne dass Pakete verloren gehen, greift *Congestion-Control* überhaupt nicht.
- Die Verfahren zur Überlastvermeidung gehen immer davon aus, dass Pakete nicht auf der Leitung, sondern beim **Router** verloren gehen. **Mobile Umgebungen**, in denen der physikalische Kanal erheblichen Störungen unterliegt, sind dadurch nur unzureichend abgebildet. An TCP wird daher immer noch aktiv geforscht.

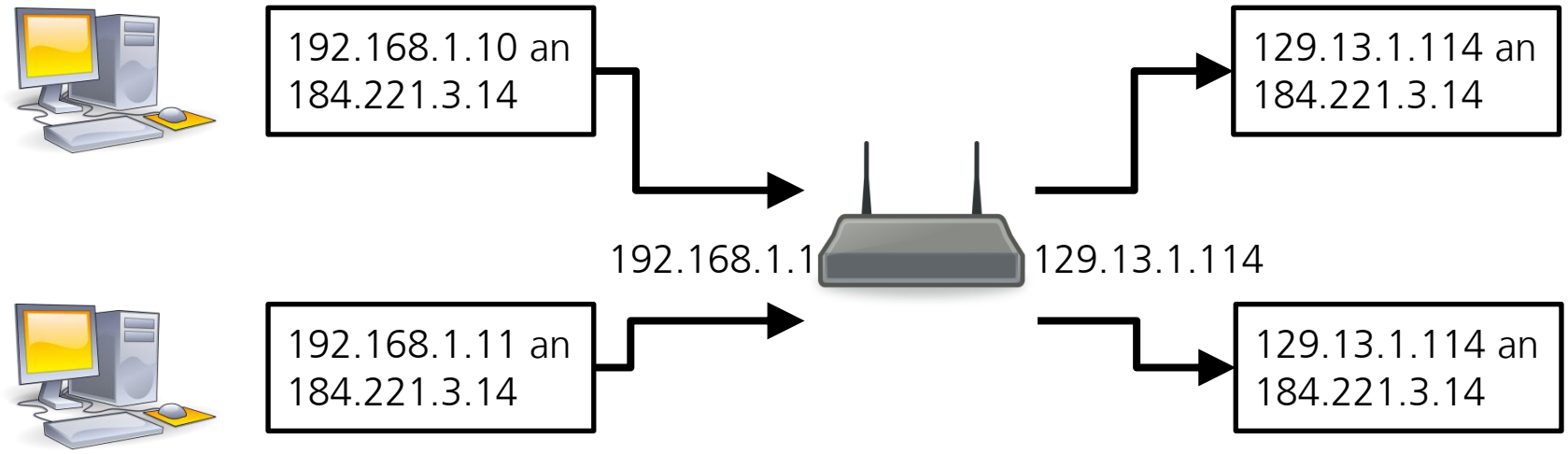
Exkurs: Network-Address-Translation

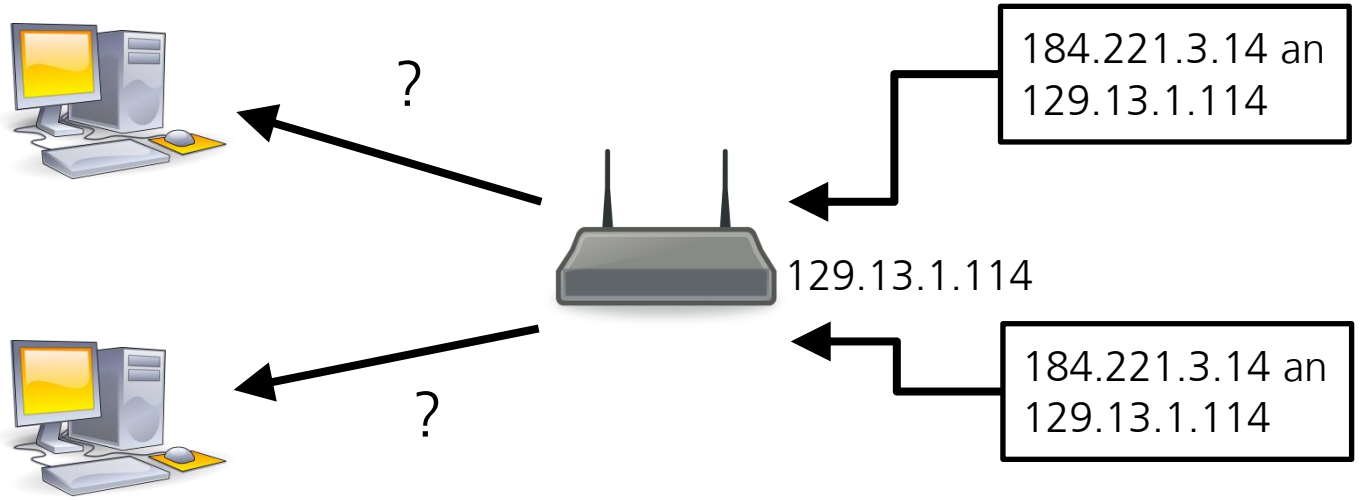
Rückblick auf Schicht 3

Rückblick: Network-Address-Translation

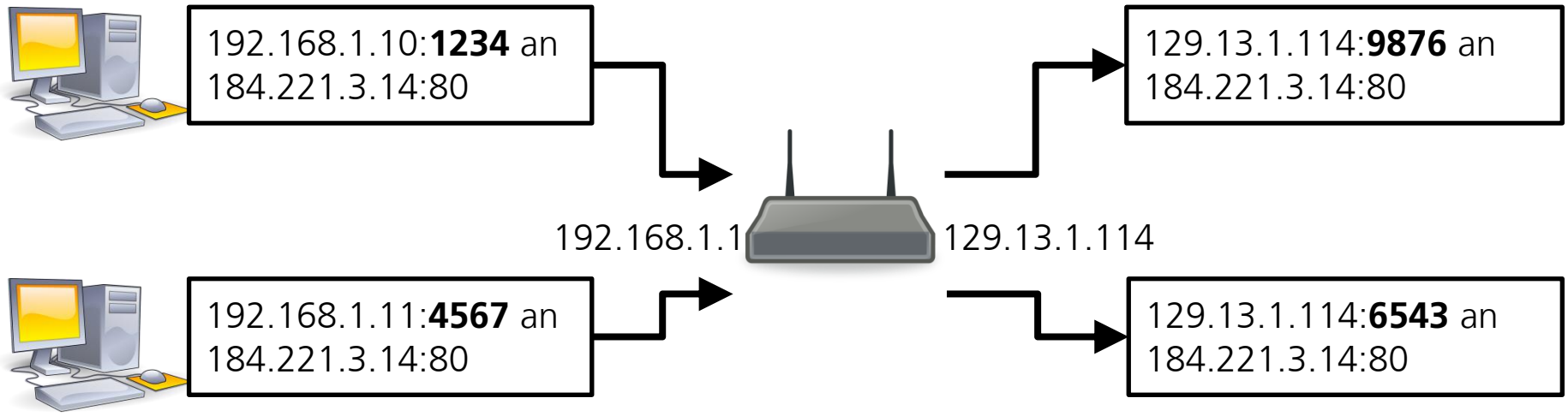
- In einem Netzwerk mit NAT-Router befinden sich typischerweise **mehrere** Geräte, die ggf. parallel Nachrichten an **denselben Zielknoten** verfassen können.
- Bei einer **Rückantwort** muss der NAT-Router erkennen, an welchen der Quellknoten er diese zustellen muss.
- Ohne **Zusatzinformation** ist dies nicht möglich, denn die Antwortnachricht enthält in beiden Fällen denselben Sender und Empfänger.
- **Frage:** Wie kann dieses Problem gelöst werden?
- **Antwort:** Gar nicht, ohne die Grenzen von IP zu verlassen!

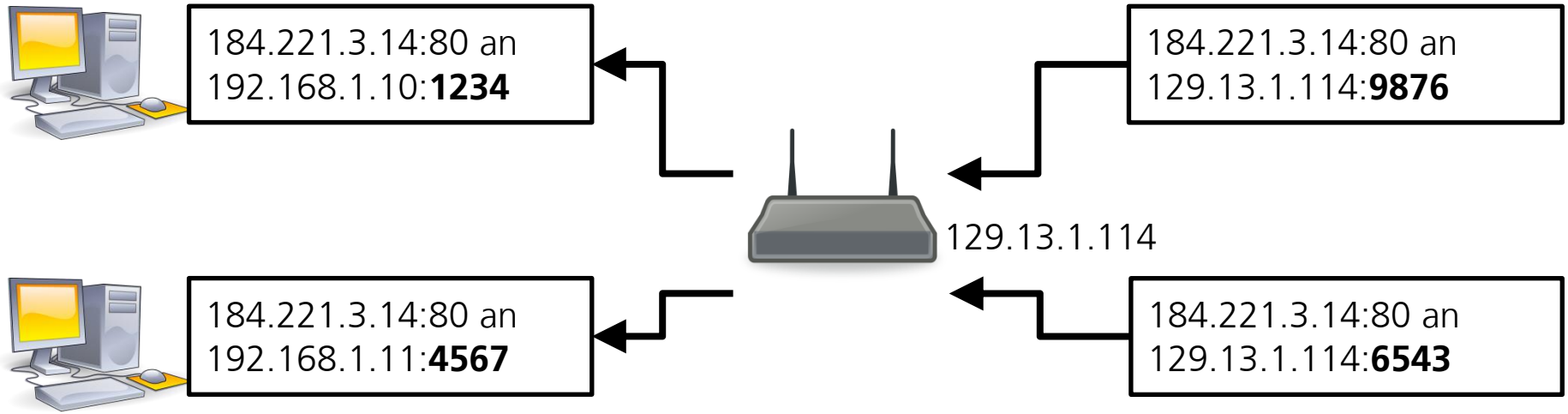
Rückblick: Network-Address-Translation

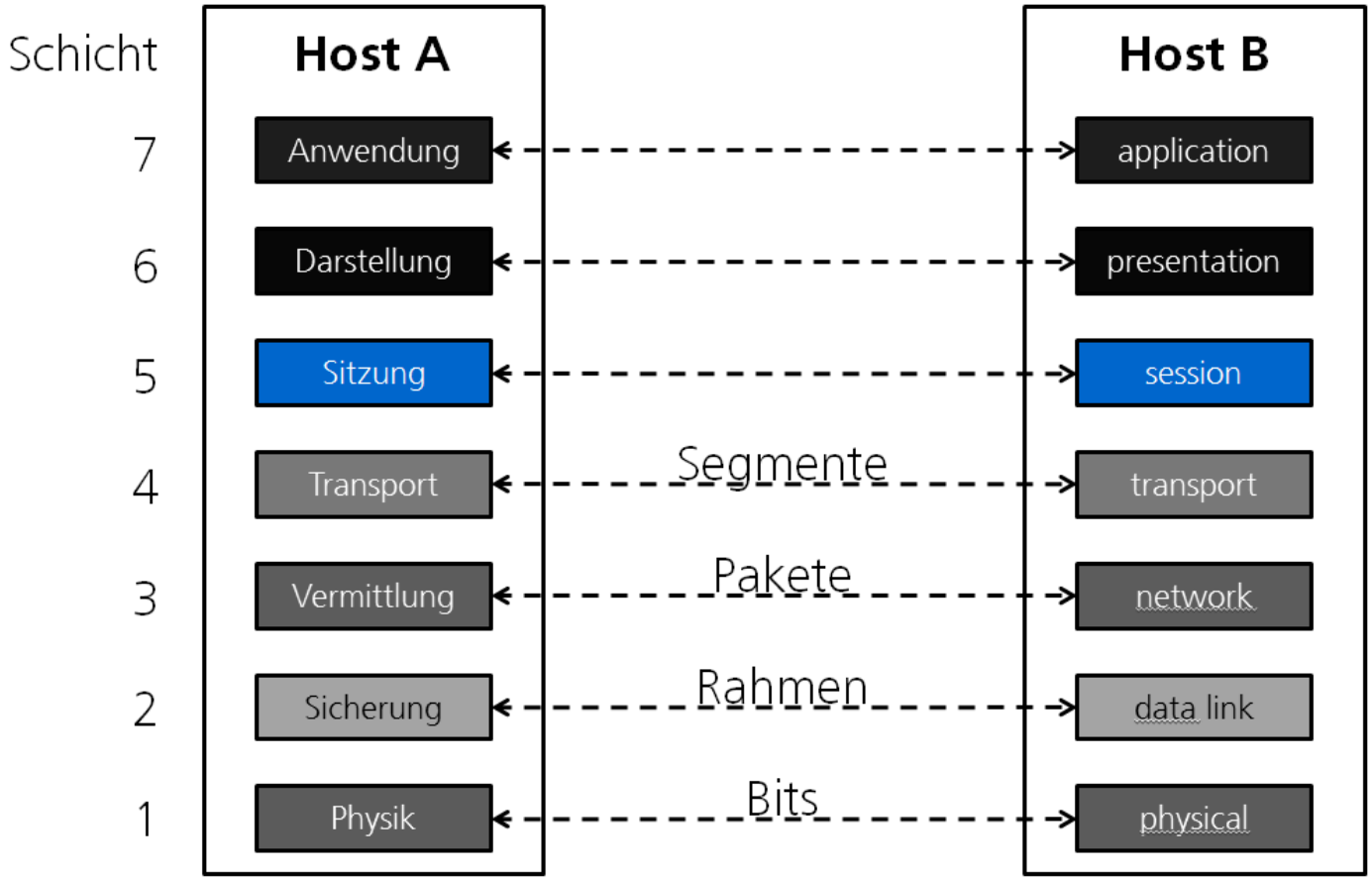




Lösung: Network-Address-Translation mit Ports







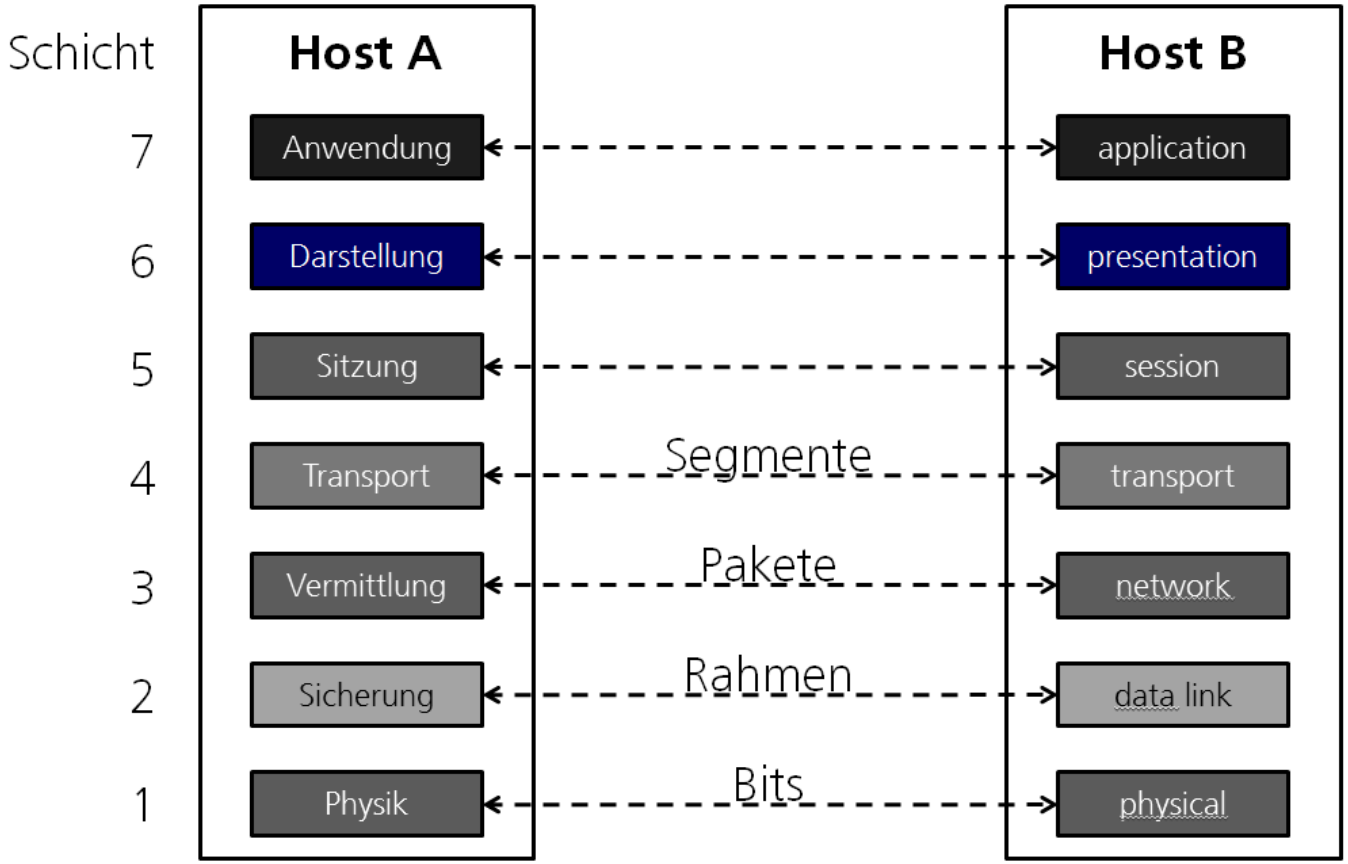
Schicht 5 - Sitzungsschicht

Sitzungen

- Als Sitzung wird ein andauernder, aber nicht zwingend kontinuierlicher Datenaustausch zwischen Client und Server bezeichnet.
- **Beispiel:** In einem Webshop kann der Kunde beim Einkauf zwischenzeitlich Pausen einlegen. Innerhalb der durch den Server vorgegebenen Grenzen kann er den Einkauf fortführen. Der Datenaustausch dauert also an, ist aber nicht kontinuierlich.
- Ein wesentliches Merkmal von Sitzungen ist also, dass sie **zeitlich** über **mehrere**, an sich unabhängige **Verbindungen** der Transportschicht hinweg bestehen können.

Sitzungsschicht

- In der Sitzungssicht werden Maßnahmen getroffen, um eine Sitzung zu beginnen, während ihrer Laufzeit zu verwalten und schließlich zu beenden.
- Durch Speichern an Checkpoints können abgebrochene Sitzungen wiederaufgenommen werden.
- Beispiele für Sitzungsverwaltungen:
  - HTTP Cookies. Durch eine Sitzungskennung (Session ID) kann der Server einen Client „wiedererkennen“ und die Interaktion an geeigneter Stelle fortsetzen.
  - Datenbankmanagementsysteme: Checkpoints stellen gültigen Zustand einer Datenbank sicher, wenn Transaktionen abgebrochen werden.



Schicht 6 - Darstellungsschicht

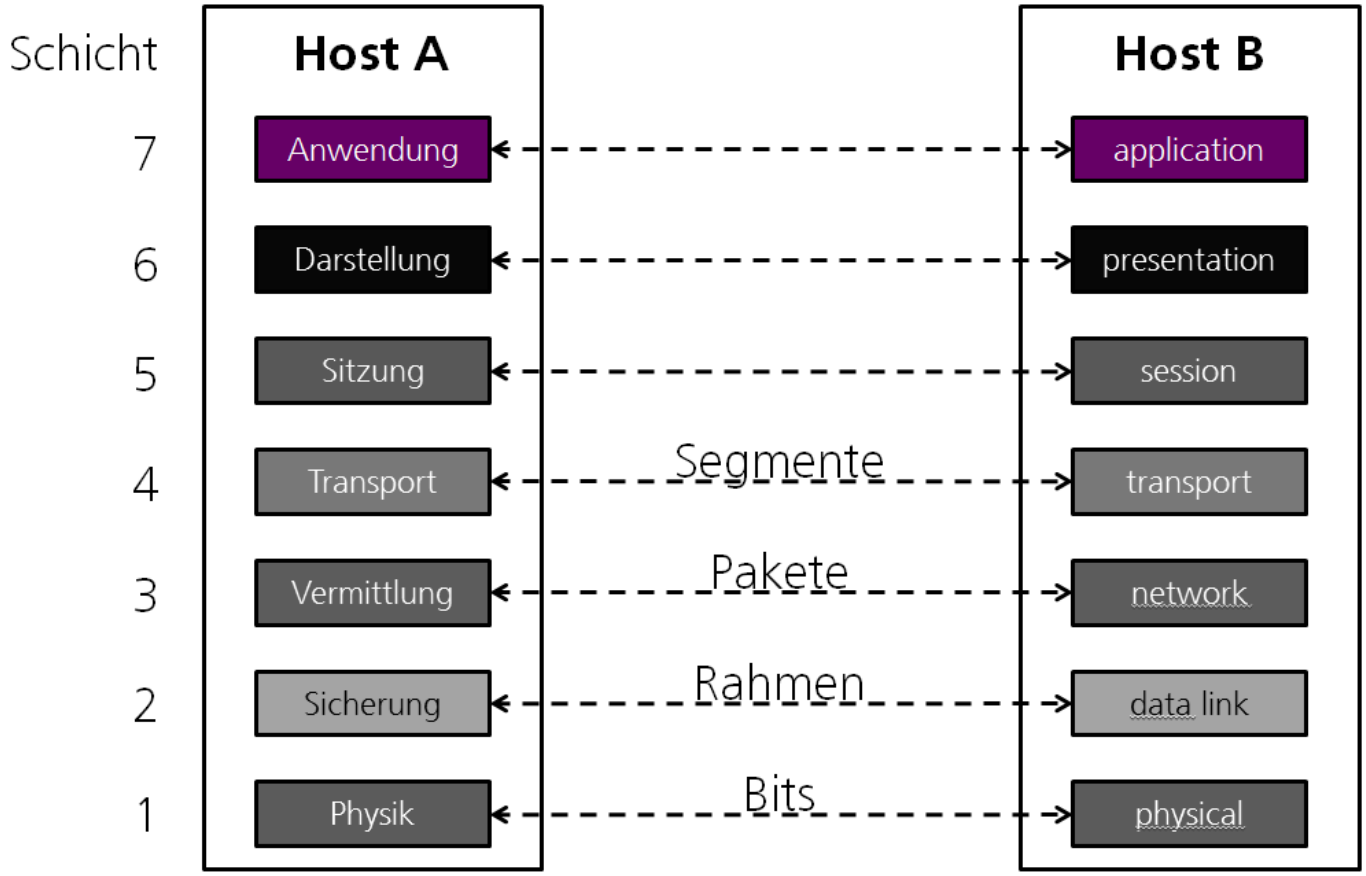
Darstellungsschicht

- Auf der Darstellungsschicht werden empfangene Daten für die Präsentation aufbereitet und zu sendende Daten in für die Übertragung geeignete Formate überführt.
- Außerdem werden der Darstellungssicht auch die Aufgaben der **Verschlüsselung** und **Komprimierung** zugeordnet.
- **Hinweis:** Verschlüsselung existiert auch auf anderen Schichten (bspw. mit IPSec und TLS).

Darstellungsschicht

- **Klassisches Beispiel:** Um einen Zahlenwert > 255 abzulegen, benötigt man mindestens 2 Byte. Festzulegen ist, welches der beiden Byte den höherwertigen Anteil speichert und welches den niederwertigen. Bei der Präsentation (Verarbeitung) eines derart gespeicherten Wertes muss mit beiden Varianten derselbe Wert erscheinen.

- **Etwas moderneres Beispiel:** Videodaten werden auf dem Bildschirm pixelweise dargestellt. Die resultierende Datenmenge (~6MB pro Bild bei Full-HD) muss mittels eines geeigneten Verfahrens für die Übertragung komprimiert werden.



Schicht 7 - Anwendungsschicht

Schicht 7: Anwendungsschicht

- In dieser Schicht werden schließlich die Applikationen eingeordnet. Sie erhalten durch die unteren Schichten Zugang zu Netzwerkdiensten, ohne Details des Übertragung kennen zu müssen.
- Bekannte Protokolle der Anwendungsschicht sind: HTTP, DNS, IMAP, POP3, SMTP, etc.
- Applikationen können eigene Mechanismen zur Ausfallsicherheit, Fehlertoleranz oder Komprimierung bereitstellen. Diese werden nur dann den unteren Schichten des OSI-Modells zugeordnet, wenn sie im Kontext der Datenübertragung relevant sind.

Zusammenfassung

- Die Transportschicht regelt Aufgaben der Übertragungssicherung und Flussteuerung in paketvermittelnden Netzen.
- Die Sitzungsschicht stellt über mehrere Verbindungen hinweg einen gemeinsamen Kontext her.
- Die Darstellungsschicht sorgt für systemunabhängige Präsentation der Daten.
- Die Anwendungsschicht bietet Applikationen die Schnittstelle zur Kommunikation über Netzwerke.