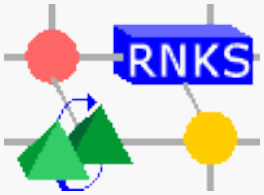


Brandenburgische Technische Universität Cottbus

Lehrstuhl Rechnernetze und
Kommunikationssysteme

Rechnernetze

Eine (kurze) Einführung

Cluj, Wintersemester 2019/20

Prof. Dr.-Ing. habil. Hartmut König

II.5

Protokollfunktionen

(*Fortsetzung*)

König 5

II.5.5

PDU-Kodierung/Dekodierung

König 5.5

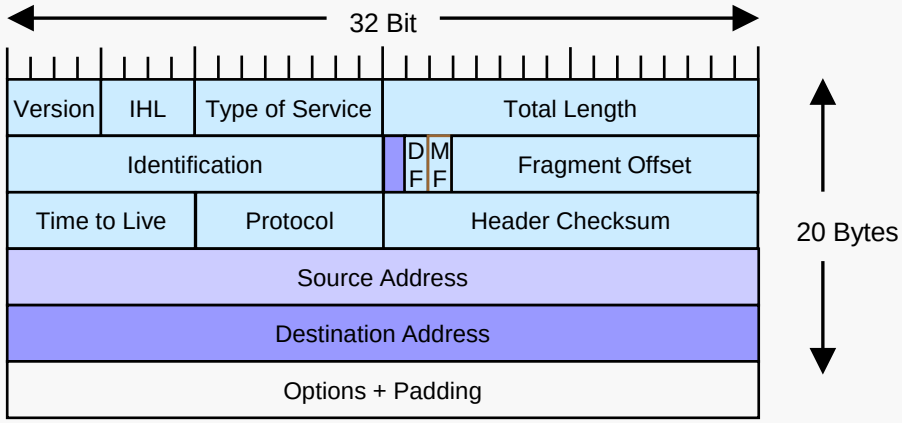
PDU-Kodierung/-Dekodierung

Erzeugen der abgehenden PDUs und Dekodieren sowie Analysieren der eintreffenden PDUs für die weitere Auswertung.

- Lokale Aktion einer Protokollinstanz
  - wird in Protokollbeschreibungen in Regel nur angedeutet

Eine der aufwendigsten Protokollfunktionen in der Implementierung !!!

- Gründe
  - Bit-genaue Positionierung der (Header-) Daten in der PDU (senderseitig)
    - Shift-Operationen
  - Auslesen der Header-Informationen (empfängerseitig)
  - Weiterleitung der PDU/SDU im Protokollstack
    - unter Verwendung von Puffern



IP-Header

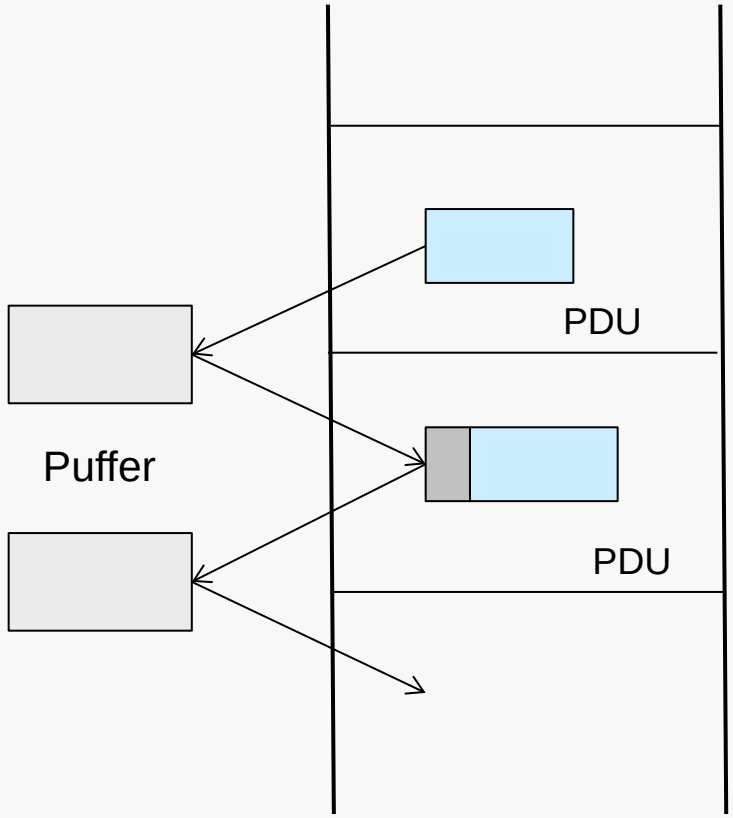
Implementierung der SDU/PDU-Übergabe in einem Protokollstack

- Einfachste Lösung:
  SDU/PDU werden durch Puffer übergeben
  - ↳ erfordert Kopiervorgänge
    - ⟶ ineffizient

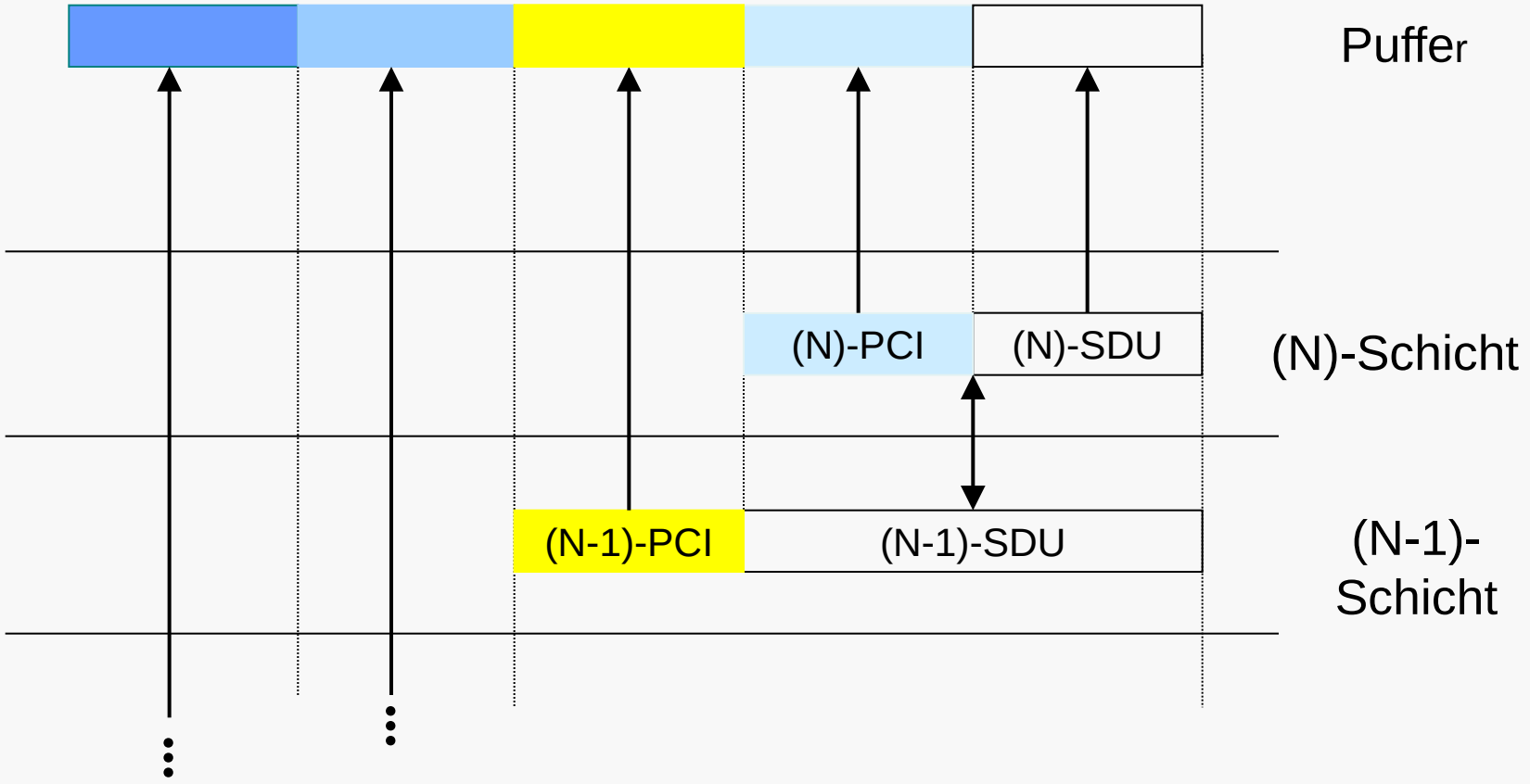
☝ **Goldene Regel der Protokoll-implementierung**:

Minimiere Kopiervorgänge !!!

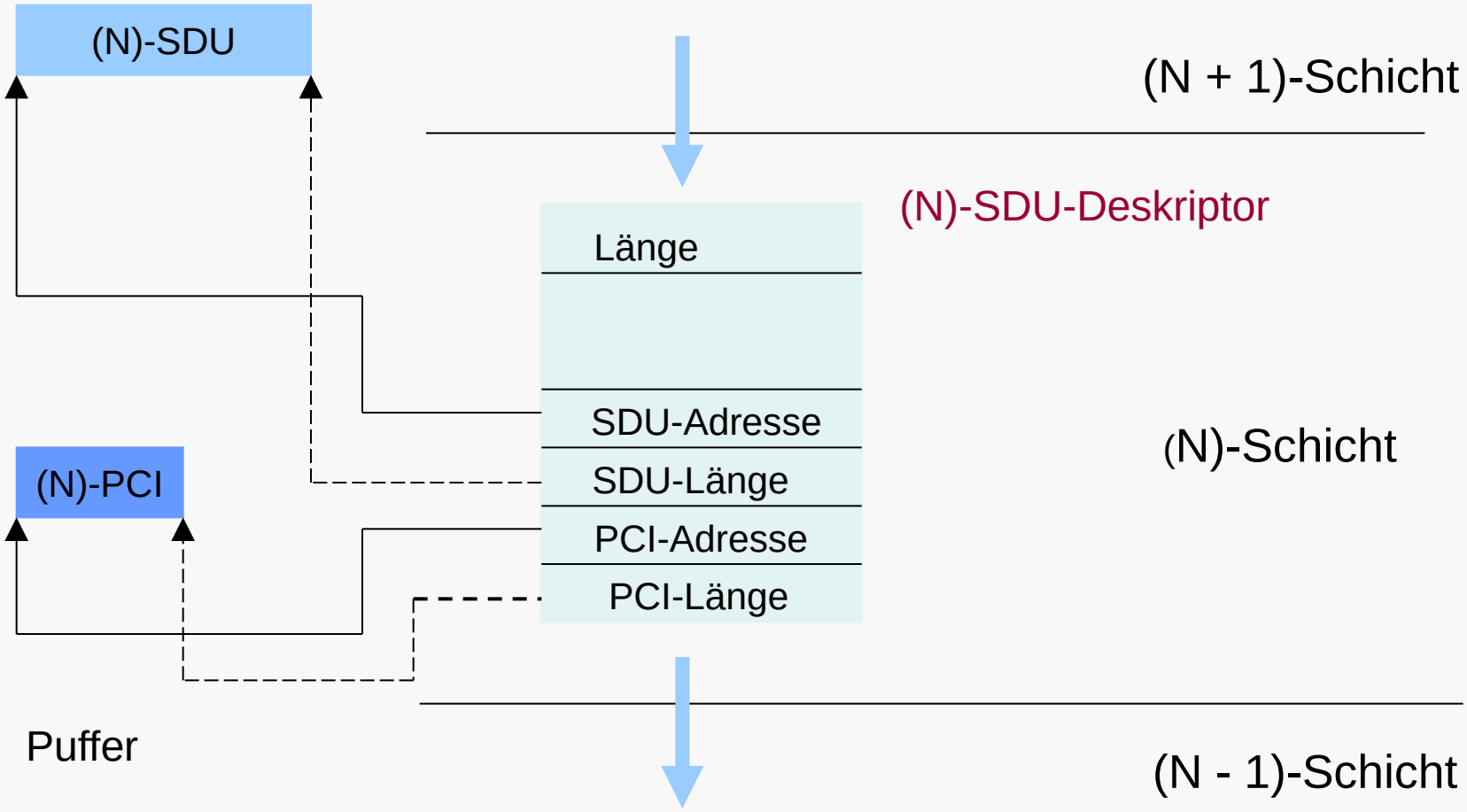
- Ansätze
  - Zentraler PDU-Puffer
    - ↳ Offset-Verfahren
  - Übergabe von Referenzen
    - ↳ Gather/Scatter-Technik



Offset-Verfahren



Gather/Scatter-Verfahren



II.5.6

Anpassen der PDU-Größe

König 5.6

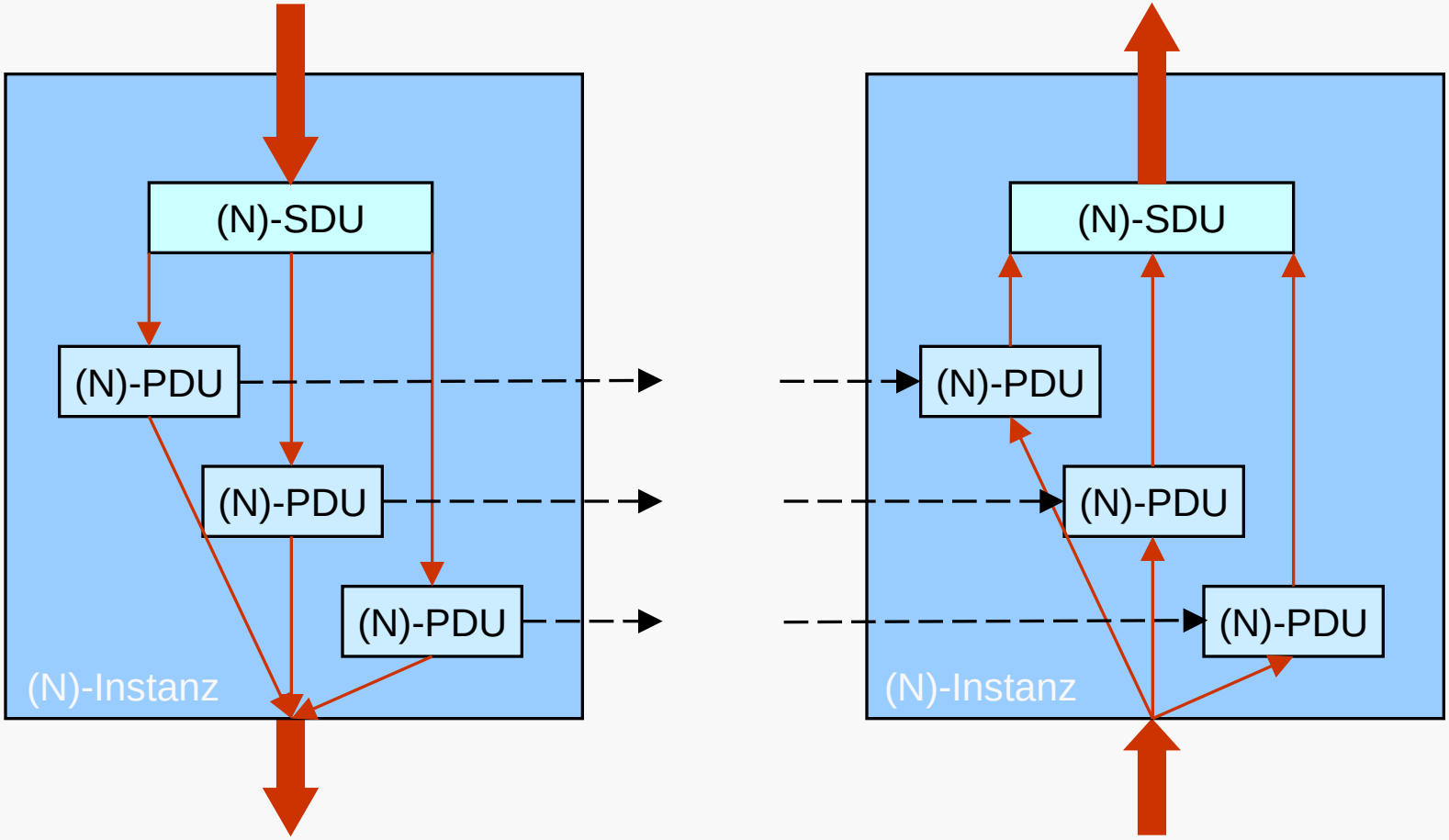
Anpassen der PDU-Größe

Ein wichtiges Problem bei der Abbildung eines (N)-Protokolls auf einen (N-1)-Dienst ist das Anpassen der PDU-Größen.

- Länge der PDUs ist in vielen Protokollen und Kommunikationskanälen begrenzt
  - z. B. IP: MTU – Maximum Transfer Unit
    - PDUs in Protokollen höherer Schichten sind meistens größer

- Verfahren
  - Segmentieren & Zusammensetzen
  - Blocken & Zerlegen

Segmentieren und Zusammensetzen



Segmentieren und Zusammensetzen

Beim Segmentieren werden die (N)-Dienstdateneinheiten in der (N-1)-Schicht in mehrere (N-1)-PDUs zerlegt.

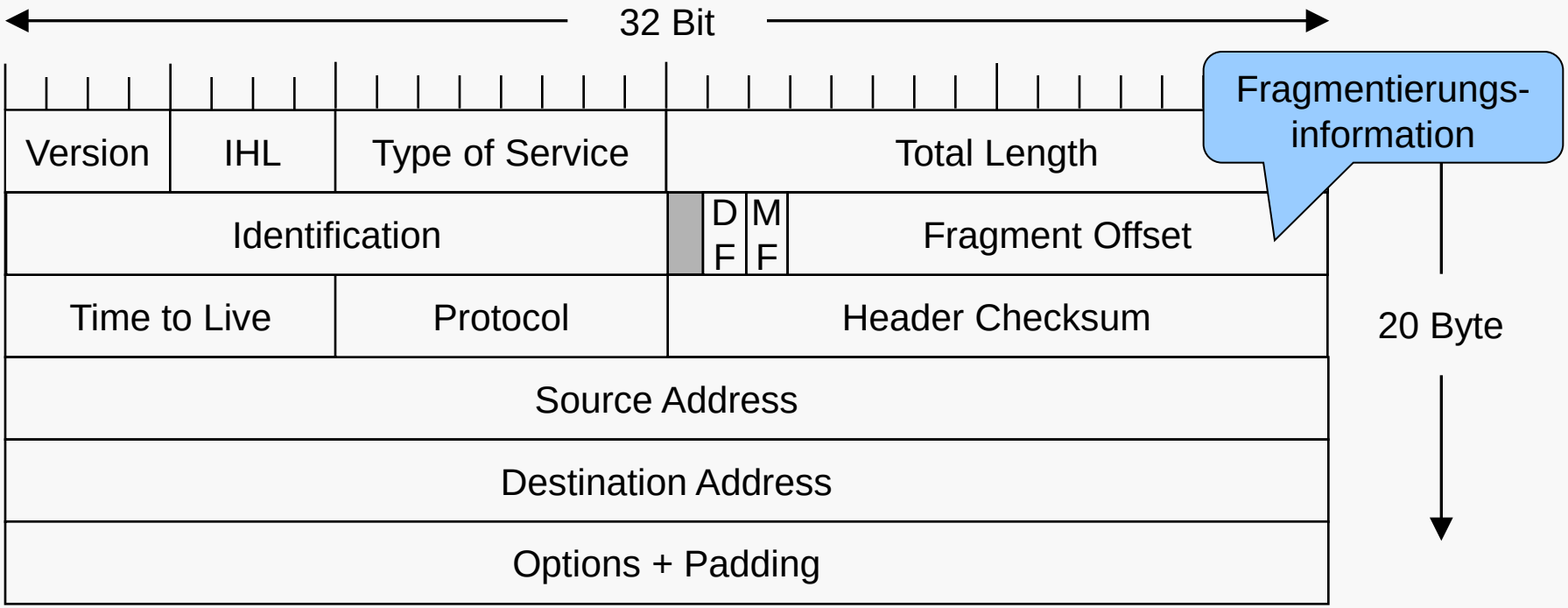
- Segment erhält eigenen PDU-Kopf mit Verweis auf die Position in der ursprünglichen PDU
- (N)-PDUs werden getrennt an die Partner-Instanz übertragen
- Partner-Instanz setzt SDU wieder zusammen

☝ **Segmentieren verletzt nicht das Transparenz-Prinzip !!!**

↳ Zerlegung ohne Bezug auf den Inhalt

☞ **Internet-Terminologie: Fragmentieren** statt Segmentieren !!!

Fragmentierungsparameter im IPv4-Header



Weitere Gründe für das Segmentieren

- Effizienter Fehlerkontrollmechanismen
- In Netzen mit einem gemeinsamen Übertragungsmedium, z. B. in LAN, können gleichmäßigere Zugriffe mit geringeren Verzögerungen zum Medium erzielt werden
- für kleinere PDUs muss weniger Puffer in den Instanzen bereitgestellt werden
- günstigere Möglichkeiten für das Einfügen von Kontroll- und Synchronisationspunkten

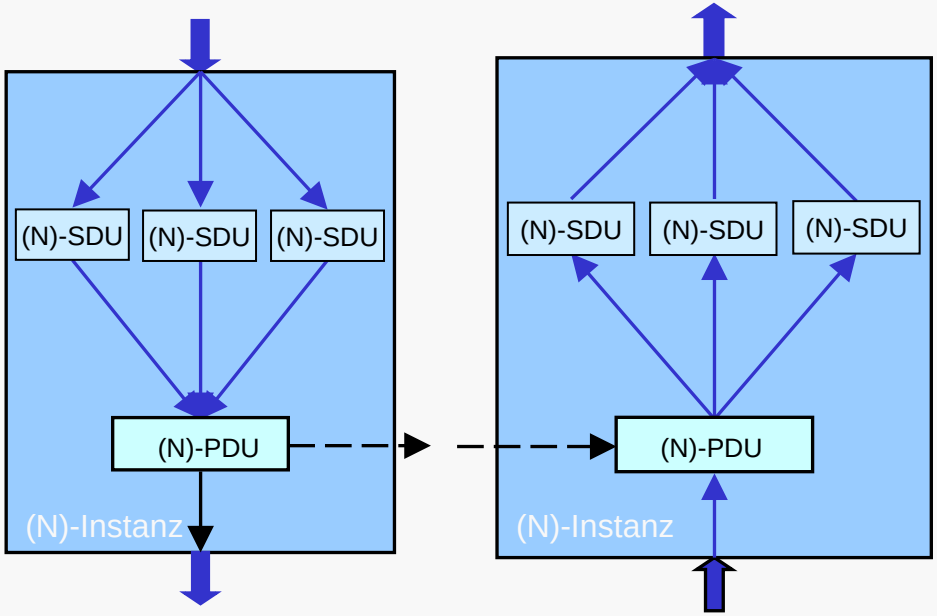
- Nachteile
  - höherer Aufwand
  - Überwachung vieler PDUs (PDU-Verluste)
  - ungünstige Zerlegungen
    - ↳ z. B. 1 Byte Rest

Blocken und Zerlegen

Zusammenfassung mehrerer (N)-SDUs zu einer (N)-PDU

- Vorteilhaft für die Übertragung kleinerer Dateneinheiten
- Reduziert Organisationsaufwand
  - z. B. nur eine Bestätigung

☞ **Vor- und Nachteile komplementär zum Segmentieren !!!**



II.5.7

Vergabe von Sequenznummern

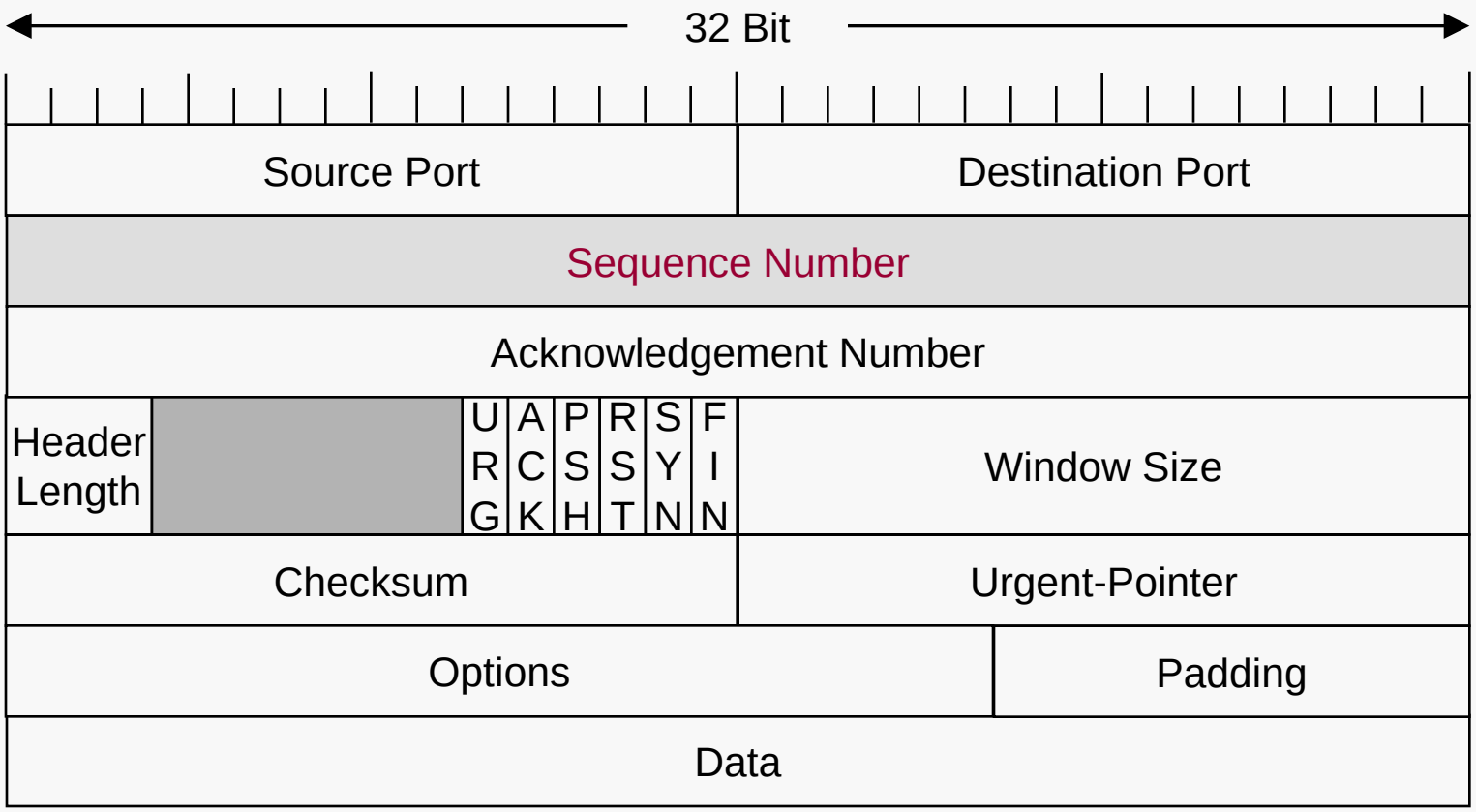
König 5.7
Tanenbaum/Wetherall 6.2.2

Vergabe von Sequenznummern (1)

Um die Reihenfolge der PDUs bei verbindungsorientierter Übertragung zu wahren, werden die Protokolldateneinheiten fortlaufend nummeriert, indem eine Sequenznummer in den PDU-Header eingetragen wird (*sequencing*).

- Verwendung
  - Wiederherstellen der Übertragungsreihenfolge durch die empfangende Instanz
  - Erkennen von PDU-Verlusten und -Dopplungen
  - Bestätigen von Protokolldateneinheiten
    - i. d. R. Bestätigung = Sequenznummer+1
    - Sender wird die Nummer der nächsten erwarteten PDU angezeigt

Sequenznummer im TCP-Segment[1]



1) In TCP heißt eine PDU Segment.

Vergabe von Sequenznummern (2)

- Bereich der Sequenznummern muss nicht notwendigerweise mit Eins beginnen
  - Festlegung des Startwerts während Verbindungsaufbau
    - → z.B. TCP: Zufallswert (empfohlene Implementierung)

- Probleme bei der Nutzung von Sequenznummern
  - Überlauf des Sequenznummernbereichs
  - Zuweisung von „alten“ PDUs zu neuen Verbindungen

A) Sequenznummernüberlauf

Überlauf des Sequenznummernbereichs (1)

Für die Sequenznummer kann nur ein endliche Zahl von Bits im PDU-Kopf zugewiesen werden. Deshalb ist der Bereich der Sequenznummern, der zugewiesen werden kann, endlich.

- Wie mit dem Überlauf umgehen?

- Lösung: Fortlaufende Vergabe modulo des Maximalwerts
  - Wert darf nicht zu klein gewählt werden
    - zu kleiner Wert kann zur Folge haben, dass neue Sequenznummern mit ausstehenden Protokolldateneinheiten korrelieren
  - TCP verwendet als Obergrenze den Wert 2^{32}

Überlauf des Sequenznummernbereichs (2)

- Problem: Hochleistungsprotokolle

  → Verkürzung der Wiederverwendungszeit

  - Übertragungsrate Wiederverwendung

64 Kbit/s:	nach 6,2 Tagen
100 Mbit/s:	340 s
1 GBit/s:	34 s

- Mögliche Lösungen

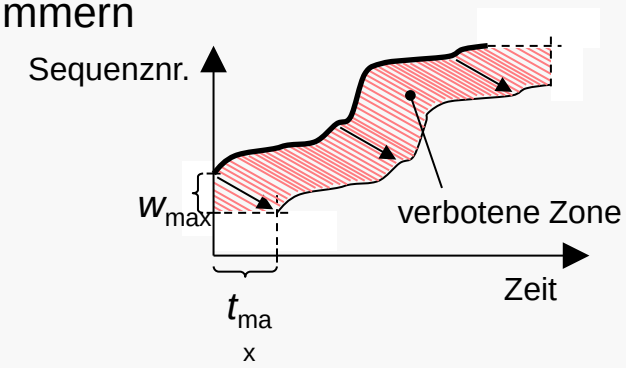
  - Verkürzung der Lebensdauern der PDUs
  - Erweiterung des Sequenznummernbereichs
  - zusätzlich Verwendung von Zeitstempeln (→ bevorzugt)

B) Wiederholte Verwendung von Sequenznummern

Wiederholte Verwendung von Sequenznummern

Durch die zyklische Wiederverwendung von Sequenznummern könnte es (nacheinander) unterschiedliche Verbindungen geben, die die gleichen Sequenznummernbereiche nutzen.

- Problem: (verspätete) PDUs einer älteren Verbindung könnten einer neuen Verbindung zugeordnet werden
- Lösungsvarianten
  - Buchhaltung über ausgesendete Sequenznummern
    - Verbotene Zone
      →  Tanenbaum/Wetherall 6.2.2
    - kaum mehr eingesetzt in der Praxis
  - Definierte Wartezeiten
    - Stillhalte-Zeiten



Stillhaltezeiten

Stillhaltezeiten bedeuten, dass nach dem Abbau der Verbindung bzw. nach ihrem Zusammenbruch die Verbindungsreferenzen eine bestimmte Zeitdauer nicht genutzt werden dürfen. Alle alten PDUs und mögliche ACKs müssen aus dem Netz verschwunden sein.

- OSI-Protokolle
  - Einfrieren der Verbindungsreferenz

    → Protokollfunktion Einfrieren

- TCP
  - Ports sind erst nach einer Wartezeit für neue Verbindung verfügbar
    - Dauer: zweifache max. Segmentlaufzeit (*Maximum Segment Lifetime*)[1]
    - Heutige TCP-Impl.: 120 s
  - Schutz vor Sequenznummernumlauf: Timestamp-Option

1) In TCP heißt eine PDU Segment.

II.5.8

Fluss-Steuerung

(*flow control*)

König 5.8.1
Tanenbaum/Wetherall 6.2.4

Fluss-Steuerung

Die **Fluss-Steuerung** reguliert den Datenfluss zwischen der Sender- und der Empfänger-Instanz.

- Ziel:
  - Ausgleich der Verarbeitungsgeschwindigkeiten
  - Schutz vor Überlast infolge unterschiedlicher Speicherkapazitäten

☝ **Kann auch zur Regulierung des Datenaustauschs an der Dienst-schnittstelle genutzt werden !!!**

- Verfahren
  - Fensterbasierte Fluss-Steuerung
  - Ratenbasierte Fluss-Steuerung (→ Ausführliche Behandlung in VL Internet)

Fensterbasierte Fluss-Steuerung

Die **fensterbasierte Fluss-Steuerung** ist eine reine Ende-zu-Ende-Regulierung zwischen der Sender- und der Empfänger-Instanz, die durch die Aufnahmekapazität des Empfängers bestimmt ist. Sie berücksichtigt nicht die nicht die Netzlast.

- bezieht sich nur auf die Erstübertragung der PDUs
  - nicht auf Wiederholungen !!!

- Verfahren
  - Start/Stopp-Mechanismen
  - Kreditverfahren
  - *Sliding Window*-Protokoll

A) Start/Stopp-Mechanismen

Start/Stopp-Mechanismen

Start/Stopp-Mechanismen regulieren den Datenfluss, indem der Empfänger ein Stopp-Signal aussendet, wenn er keine weiteren PDUs aufnehmen kann.

- Analogie: Ampel
- einfachstes Verfahren
- Problem: Stopp-Signal benötigt Übertragungszeit
- Nachteil: - Pufferkapazitäten für Aufnahme bereits in der Übertragung befindlicher PDUs beim Empfänger erforderlich

  - schubartiger Datenfluss möglich (→ Bursts)

B) Kreditverfahren

Kreditverfahren

Bei den Kreditverfahren erteilt der Empfänger dem Sender einen Kredit, in dessen Rahmen er uneingeschränkt PDUs senden kann.

- Wenn der Kredit verbraucht ist, stoppt der Sender
- Im Regelfall: fortlaufende Kreditvergabe, um Sendepausen zu vermeiden
  - Problem: Verlust von Kredit-PDUs

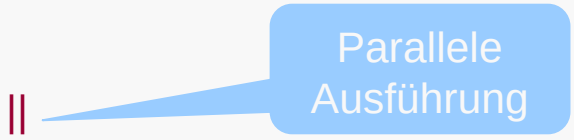
☝ Die Kreditvergabe ist eine reine Sender-Empfänger-Beziehung. Sie berücksichtigt nicht die Belastungssituation im Netz !!!

Beispiel Kreditmechanismus: Senderseite (1)

```
message DT = ...
             UPDATE = record (code: bits; kredit, kredit_nr: integer)
             ACKupdate = record (code: bits; kredit_nr: integer)
            .  .  .
entity sender
signal neuer_kredit
var kredit: integer
begin
   XDATrequ: bilde_DT                          // Protokollteil: SENDEN DT
               if (kredit > 0)
                 DT → empfänger
                 else wait event {             // Warten auf Kredit
                        neuer_kredit: DT → empfänger
                      }

               decr kredit                     // eine Krediteinheit verbraucht
```

Beispiel Kreditmechanismus: Senderseite (2)




```
UPDATE ← empfänger:                    // Protokollteil: EMPFANG KREDIT
        if (UPDATE korrekt){                    // Neue Kredit_PDU?
           kredit :=  UPDATE.kredit
           bilde ACKupdate (incr UPDATE.kredit_nr)
           ACKupdate → empfänger                // Kreditbestätigung
           send neuer_kredit                    // Signal an PT Senden DT

        }
end //sender
```

Beispiel Kreditmechanismus - Empfängerseite (1)

entity empfänger
timer t: 0..? **ms**
var neuer_kredit, kredit_nr: integer **init**(1)
begin
DT ← sender: ... // Protokollteil: EMPFANG DT
||
. . . // Protokollteil: KREDITVERGABE
→ nächste Folie

Parallele Ausführung

Beispiel Kreditmechanismus - Empfängerseite (2)

```
loop{                                              // Protokollteil: KREDITVERGABE
  bestimme kredit(neuer_kredit)
  bilde_UPDATE (neuer_kredit, kredit_nr)
  loop{
    UPDATE → sender                                // neuer Kredit an Sender
    start t
    wait event {                                   // Warten auf Kreditbestätigung
      ACKupdate ← sender:
            kredit_nr := ACKupdate.kredit_nr
            exit  |
      timeout t: skip                              // Wiederholung der Kreditvergabe
    }
  }
 }
end //empfänger
```

C) Sliding Window Protocol

Sliding Window Protocol

Das ***Sliding Window* Protocol** ist die am häufigsten verwendete Variante des Kredit-Verfahrens. Es wird in der Regel nicht als singuläres Protokoll eingesetzt, sondern als Teilprotokoll im Kontext anderer Protokolle, z. B. in TCP.

- Kreditvergabe durch ein „gleitendes Fenster“
  - u. a. mittels Sequenznummern
  - Fenstergröße gibt den Kreditrahmen vor
  - Max. Fenstergröße kann in Protokollspezifikation festgelegt oder beim Verbindungsaufbau ausgehandelt werden

Sliding Window Protocol - Beispiel

- Beispiel
  - Fenstergröße 3, kumulative Bestätigungen
  - PDU-Verluste und Änderungen der Übertragungsreihenfolge nicht berücksichtigt