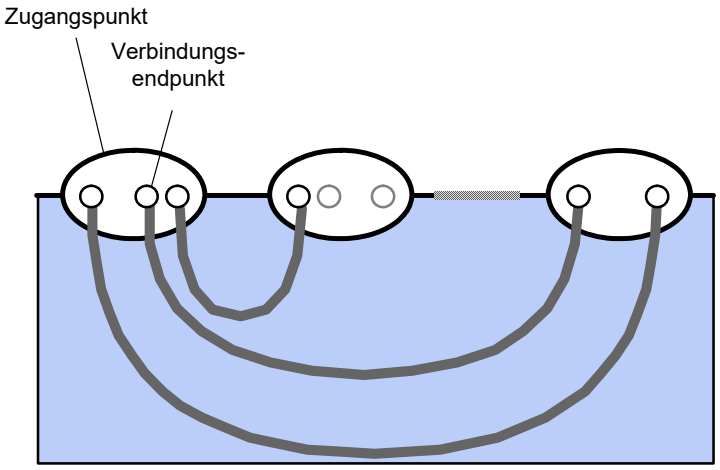
Dienstzugangspunktadressen

- **Adressen sind Verbindungsendpunktidentifikatoren:**
  - Eindeutige Identifikation eines konkreten Dienstzugangspunkts
    - Unterscheidung mehrerer Kommunikationsverbindungen an einem Dienstzugangspunkt
  - Beispiele:
    - Cover Sheet bei Fax
    - Postfachadressen
    - SIM-Karte bei Mobiltelefonen
    - IP-Adressen / Port-Nummern bei TCP/IP



Ein Dienstzugangspunkt ist lediglich eine konkrete Instanz einer Dienstschnittstelle; beispielsweise die Fax-Software, die ein Fax entgegennehmen kann. Teilen sich allerdings mehrere Nutzer ein Fax, kann der Dienstnutzer z.B. durch ein Cover-Sheet mit seinem Namen identifiziert werden.

Bezogen auf das Internet teilen sich mehrere Anwendungen auf einem PC einen Zugangspunkt zum Netz; hier muss eine eindeutige Identifikation einer Anwendung gegenüber dem Dienstzugangspunkt möglich sein (TCP/UDP-Portnummern, siehe Kapitel 5).

Dienstablauf

- **Abfolge von Dienstprimitiven ist nicht zufällig**
  - Abfolge muss spezifiziert werden können
- **Zeitablaufdiagramme (Weg-Zeit-Diagramme)**
  - Pro Dienstzugangspunkt eine Zeitachse (vertikal)
  - Dienstprimitive: Pfeile
  - Weitere Zusammenhänge (neben Abfolge) durch Kommentierungen
    - Kausalitäten
    - Exakte Zeitbedingungen
    - Kommunikationsaktivitäten, z.B. Senden oder Verlust von Paketen
  - Zeitablaufdiagramme können nicht alle Zusammenhänge spezifizieren
- **Weitergehende Spezifikationstechniken**
  - Zustandstabellen und Zustandsübergangsdiagramme (Automaten), Spezifikationssprachen

Bislang wurden lediglich die statischen Beschreibungsaspekte, konkret die Dienstzugangspunkte und die Dienstprimitive, betrachtet. Interessant und komplex wird die Dienstbeschreibung, wenn zusätzlich noch dynamische Abläufe, d.h. das zeitliche Auftreten von Dienstprimitiven an den beteiligten SAPs, betrachtet werden.

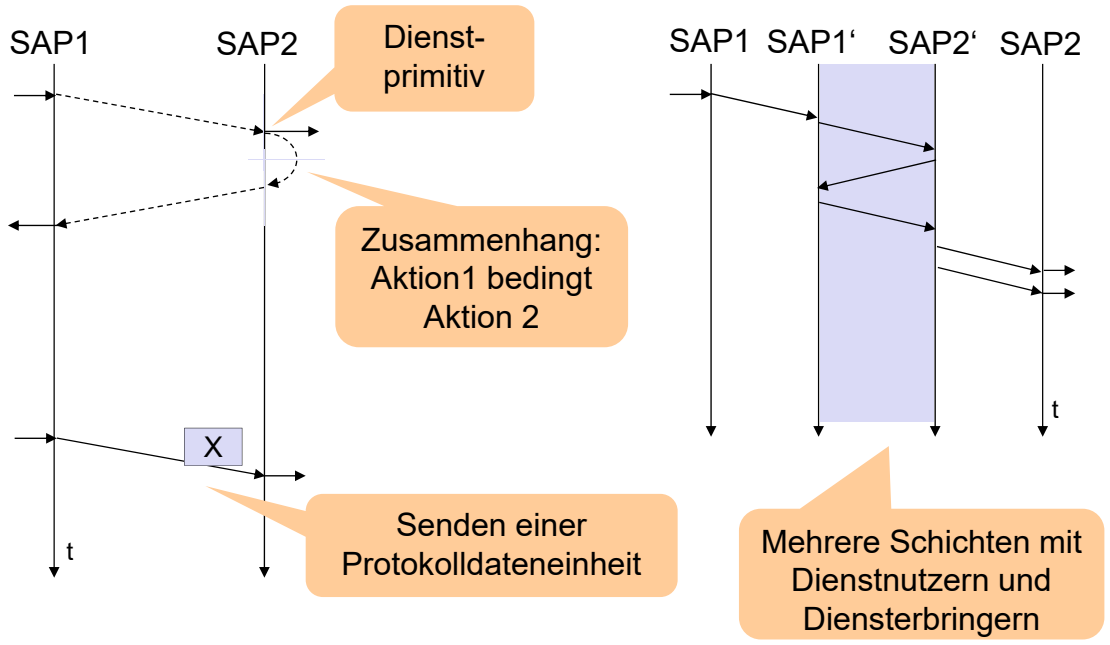
Eine einfache Möglichkeit zur Darstellung von Kommunikationsdiensten mit Betrachtung dynamischer Abläufe sind Weg-Zeit-Diagramme. Diese erlauben es, den Austausch von Dienstprimitiven zwischen zwei Dienstzugangspunkten einfach darzustellen. Solche Darstellungen können dann als Ausgangspunkt für die Implementierung der zugehörigen Protokolle verwendet werden.

Diese Diagramme werden hier knapp eingeführt und ab und zu zur Visualisierung verwendet; sie haben allerdings starke Nachteile, da die Visualisierung komplex wird, sobald ein dritter Dienstzugangspunkt verwendet wird; und auch einfache Verzweigungsmöglichkeiten im Dienstablauf (if-then-else) lassen sich nicht gut darstellen.

Zur tatsächlichen Protokollentwicklung bieten sich stattdessen mächtigere Spezifikationssprachen oder Automaten an – auch wenn die Protokollentwicklung oft auf eine vollständige formale Spezifikation verzichtet.

Dienstablauf

- **Beschreibung durch Weg-Zeit-Diagramme**



Horizontal werden die beteiligten SAPs angegeben, die vertikalen Achsen sind Zeitachsen, an denen die auftretenden Dienstprimitive aufgeführt sind. Die Ursache-Wirkung-Beziehungen zwischen den an getrennten SAPs auftretenden Dienstprimitiven erfolgt durch Pfeile von einer vertikalen Zeitachse zu einer anderen.

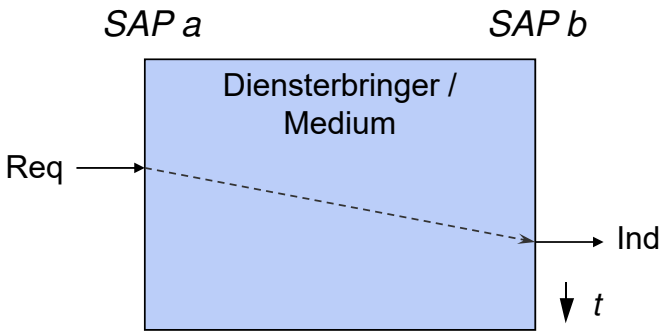
Unterschied der Notation bei den Pfeilen: die gestrichelten Pfeile stellen eine Interaktion der Kommunikationsteilnehmer (mittels Dienstprimitiven) dar, die nicht näher spezifiziert ist. Werden stattdessen (wie unten links dargestellt) durchgezogene Pfeile verwendet, ist uns die Implementierung bekannt (d.h.: wir wissen, welches Protokoll zur Interaktion eingesetzt wird und welche konkreten Nachrichten ausgetauscht werden).

Hinter den gestrichelten Pfeilen könnten sich auch noch komplexere Interaktionen verbergen. Z.B. könnte bei unserer Dienstimplementierung (vorborgen vor dem Dienstnutzer) noch die Nutzung eines weiteren Dienstes nötig werden (siehe rechts auf der Folie) – unser Diensterbringer wäre hier nicht monolithisch implementiert, sondern nutzt selbst einen weiteren Diensterbringer über SAP1‘ zur Erbringung seines Dienstes. Gestrichelte Pfeile sollen also andeuten, dass uns die real notwendigen Zwischeninteraktionen nicht bekannt sind.

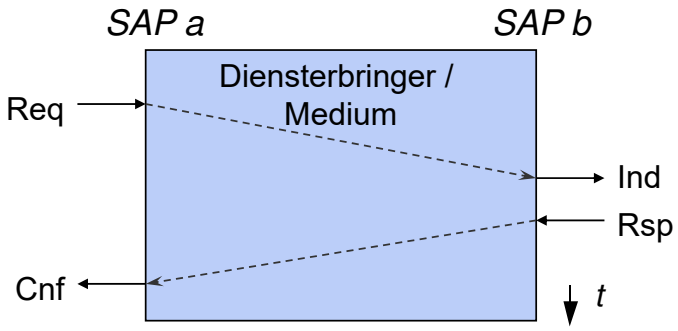
Zu jedem Dienstprimitiv können außerdem Kontrollparameter angegeben werden. Dies sind z.B. Adressen von Sender und Empfänger oder Anforderungen an die Dienstqualität (beispielsweise: schnelle Übertragung notwendig, garantierte maximale Zeit bis zur Auslieferung der Daten).

Dienstbeschreibung durch Weg/Zeit-Diagramme

- **Unbestätigter Dienst**



- **Bestätigter Dienst**



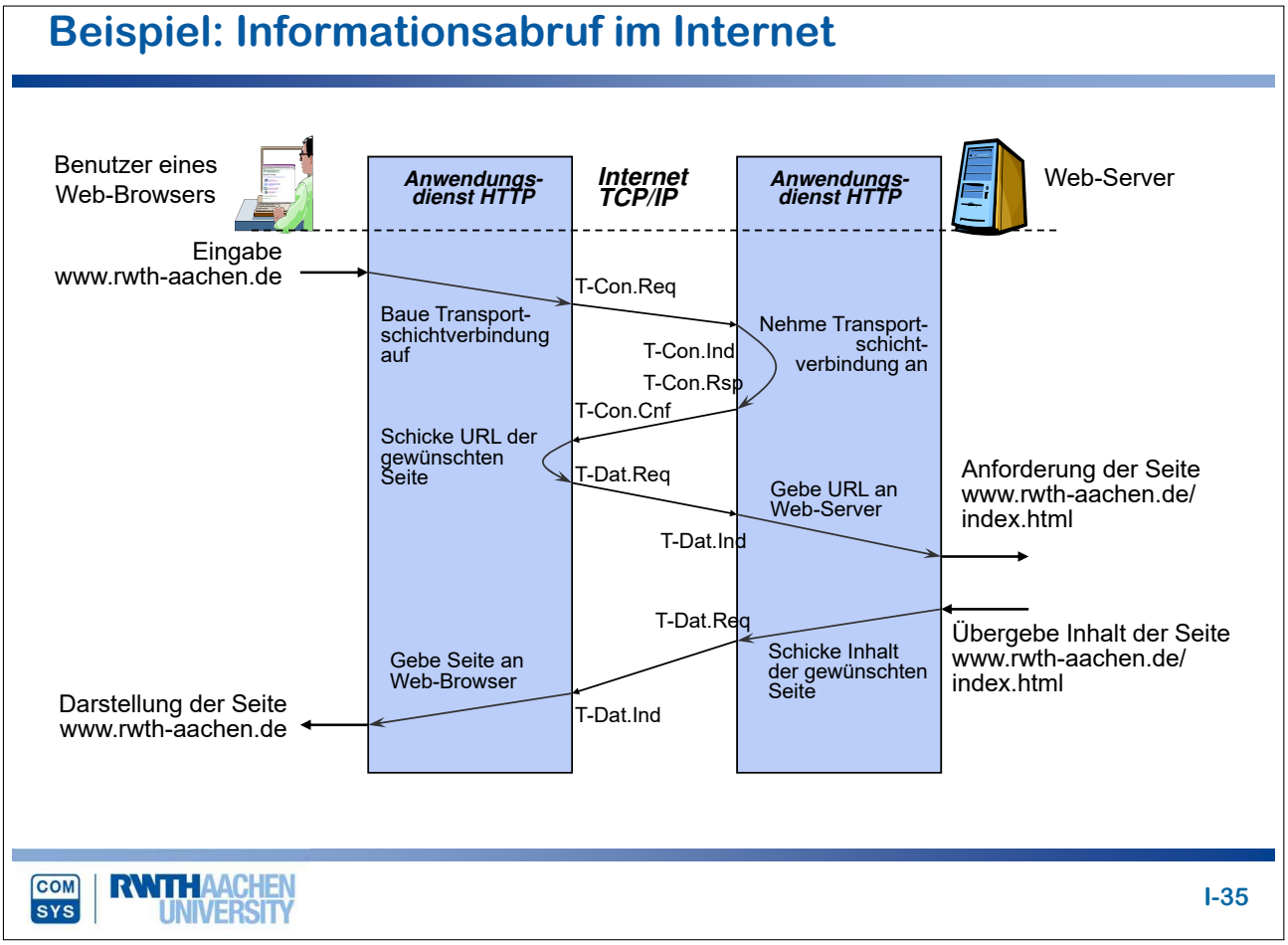
Ein einfaches Beispiel für die Verwendung der Weg-Zeit-Diagramme zur Erläuterung der Dienstbeschreibung ist der Unterschied zwischen unbestätigten und bestätigten Diensten.

Bei Nutzung eines unbestätigten Dienstes übergibt man dem Kommunikationsdienst an seinem SAP die Daten und diese werden – eventuell unter Zuhilfenahme weiterer Dienste – an den SAP des Empfängers ausgeliefert. Der Sender bekommt keine Rückmeldung, ob die Daten tatsächlich ausgeliefert bzw. angenommen wurden. Ob die Daten zuverlässig ausgeliefert werden oder nicht, hängt somit von der konkreten Dienstimplementierung ab (d.h. vom verwendeten Protokoll und den genutzten Diensten tieferer Schichten).

Beispiel: Versand von SMS.

Bei einem bestätigten Dienst hingegen bekommt man eine Rückmeldung über die erfolgreiche Zustellung.

Beispiel: Anzeige, wann eine Nachricht über myENIGMA zugestellt wurde.



Die betrachteten Partner-Protokollinstanzen sind in diesem Beispiel die HTTP-Instanzen im Web-Server und im Web-Browser (Client). HTTP ist ein sogenanntes Anwendungsprotokoll (HyperText Transfer Protocol), das zum Austausch von Dokumenten im WWW entwickelt wurde. Die HTTP-Instanzen bieten kommunizierenden Anwendungen eine Dienstschnittstelle an, stellen also eine Implementierungsmöglichkeit der obersten Schicht eines Kommunikationssystems dar. Sie nutzen Dienste der sogenannten Transportschicht, um Anfrage und Antwort zuverlässig zwischen räumlich verteilten Rechnern zu übertragen.

Der Benutzer des Web-Browsers gibt eine URL ein. Für den Web-Browser nicht sichtbar wird eine Verbindung zum Web-Server aufgebaut, über die die Daten zuverlässig übertragen werden können. Der Verbindungsaufbau ist bestätigt, da die anfragende Instanz wissen muss, ob die Gegenseite die Verbindungsanfrage annimmt und wann sie zum Empfang von Daten bereit ist.

Die Anforderung der gewünschten Webseite hingegen erfolgt unbestätigt. Für die anfragende Instanz ist es nicht wichtig zu wissen, ob und wann die Anfrage auf der Gegenseite angekommen ist, da die unterliegende Schicht die Zustellung garantiert. Ebenso erfolgt die Übertragung der Antwort unbestätigt.

Wann bestätigte und wann unbestätigte Dienste eingesetzt werden, hängt also zum einen von den Anforderungen der Kommunikation selbst ab, zum anderen aber auch von den unterliegenden Diensten.

Dienst und Verbindung

- *Verbindungsloser* **Dienst**
  - Dienstleistung ohne Kontext
    - Jeder Datenaustausch wird isoliert betrachtet, ohne jegliche Berücksichtigung vorhergegangener Kommunikationsvorgänge
  - Kontrollparameter als Teil der übergebenen Dateneinheit
  - Keine feste Verbindung zwischen den Kommunikationspartnern

Die bisher betrachteten Dienstleistungsformen gingen von einfachen Dienstleistungen aus: es wurde kein Kontext unterstellt, d.h. die Dienstleistung wurde vollständig durch das Request-Primitiv und die darin enthaltenen Kontrollparameter definiert.

Diese Art eines einfachen Dienstes bezeichnet man als vorbindungslosen Dienst. Demgegenüber steht der verbindungsorientierte Dienst, der einen Kontext einbezieht.

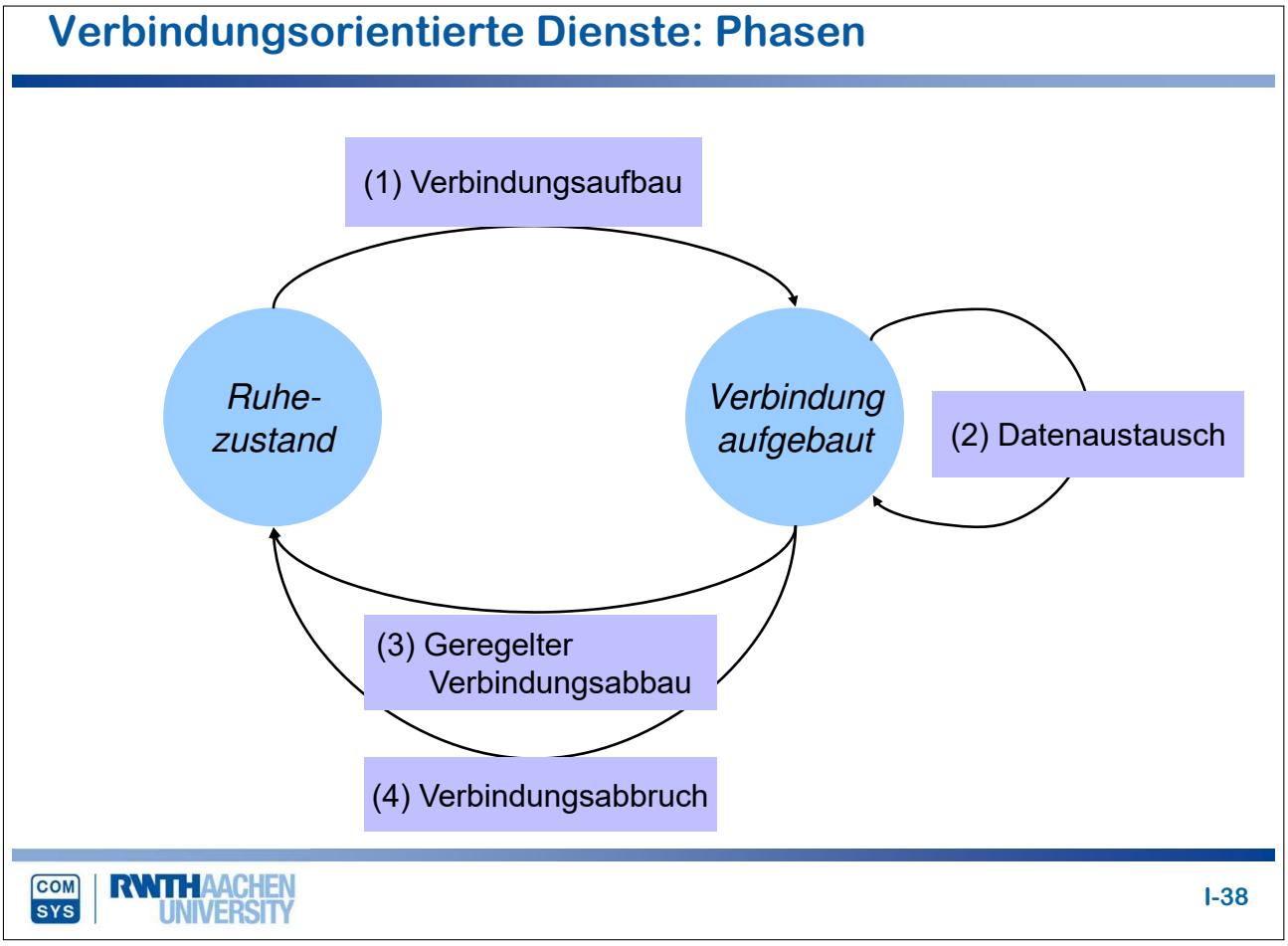
Dienst und Verbindung

- *Verbindungsorientierter* **Dienst**
  - Dienstleistung im Kontext, d.h. Diensterbringung abhängig von früher erbrachten Dienstleistungen
    - Vor dem Datenaustausch zwischen Dienstnutzern: *Verbindungsaufbau* durch die beteiligten Instanzen des Dienstanbieters
      - *Protokollabhängige Aushandlung von Übertragungsparametern*, z.B. Kommunikationspartner, Dienstqualität, Übertragungsweg
    - Datenaustausch innerhalb der Verbindung erfolgt unter Berücksichtigung des aktuellen Verbindungszustands
  - Weniger Kontrolldaten notwendig, aber
    - Zeitaufwand für Herstellung des Kontextes (Verbindungsaufbau)
    - Aufwand für die Verwaltung des Kontextes

Der Kontext bei verbindungsorientierten Diensten entsteht durch früher erbrachte Dienstleistungen. Vor dem Datenaustausch muss bei einem verbindungsorientierten Dienst zu allererst eine Verbindung aufgebaut (d.h.: ein Kontext generiert) werden. Dies geschieht durch einen sogenannten Verbindungsaufbau, bei dem die Dienstnutzer einen Kontext aushandeln, d.h. Kontrollparameter definieren, die auf alle im Rahmen der Verbindung ausgetauschten Dienstprimitive angewendet werden.

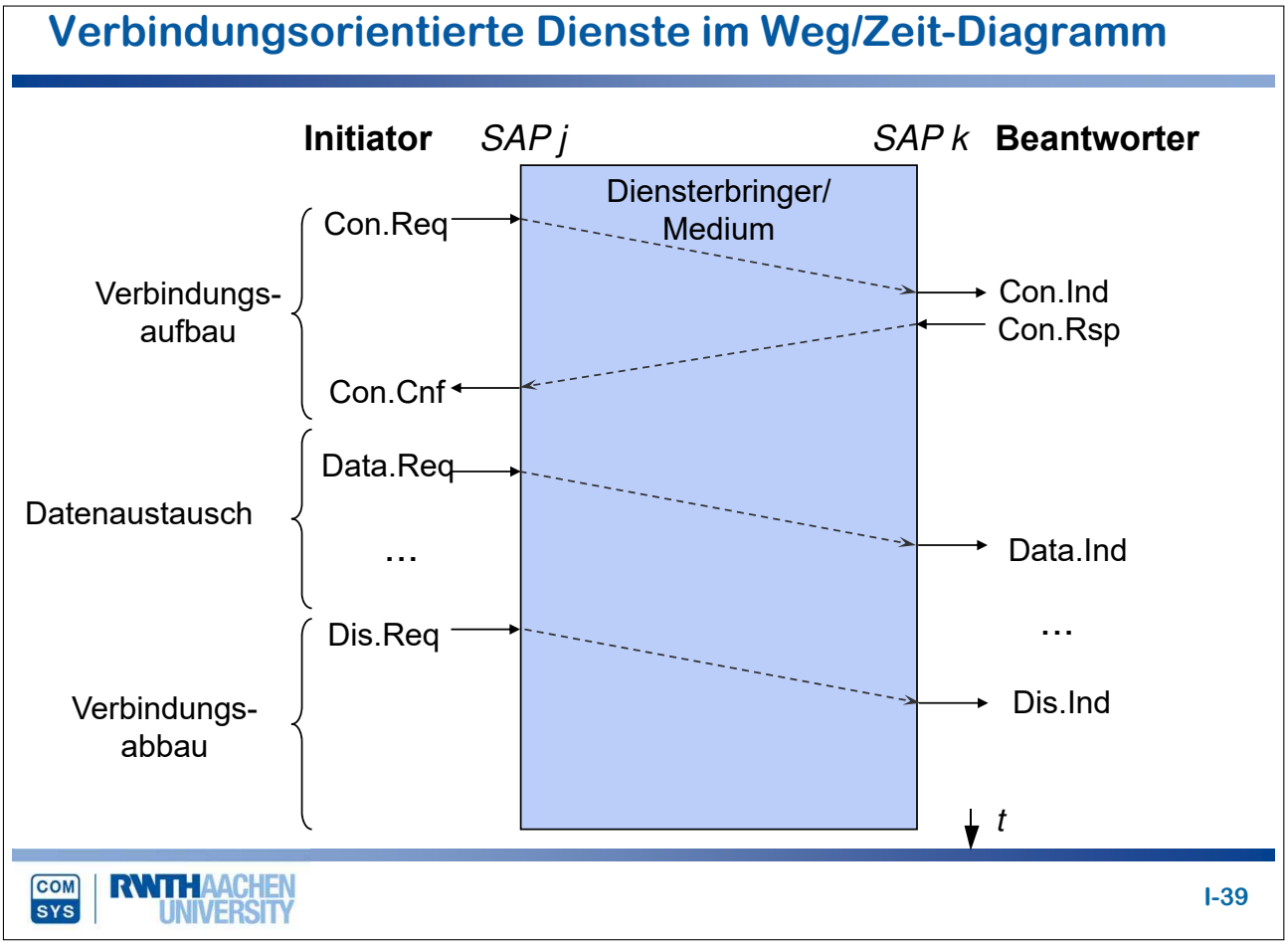
Der Zustand einer Verbindung zum Zeitpunkt *t* setzt sich aus den Parametern zusammen, zusätzlich jedoch auch aus (theoretisch) allen Kommunikationsvorgängen, die von Zeitpunkt 0 (Verbindungsaufbau) bis Zeitpunkt *t* innerhalb dieser Verbindung stattgefunden haben. Wie lange dieses „Gedächtnis“ zurückreicht und wie viele/welche Daten hierbei gespeichert werden (wenn überhaupt), ist jedoch protokollspezifisch. Als Beispiel sei der (später im Detail besprochene) Protokollmechanismus der „Sequenznummern“ erwähnt, der jedem Datenpaket eine spezifische, ganzzahlige Nummer zuordnet. Zusammen mit entsprechenden Bestätigungen von Empfängerseite ermöglicht dies eine Erkennung von Paketverlusten - hierbei speichert der Sender Datenpakete nur solange zwischen, bis eine ordnungsgemäße Bestätigung eintrifft.

Durch Aufbau und Verwaltung eines Kontextes scheint alles komplexer und aufwändiger zu werden. Man hat aber den großen Vorteil, dass man sich die explizite Übertragung einiger Kontrollparameter in jedem Dienstprimitiv sparen kann, weil diese bereits im Kontext beschrieben sind. Zum Beispiel: Beim verbindungsorientierten Dienst kann beim Verbindungsaufbau der Weg durch ein komplexes Netzwerk zwischen den beteiligten Kommunikationspartnern ermittelt und als Kontext aufbewahrt werden. Bei einem verbindungslosen Dienst muss die Wegwahl bei jedem einzelnen Dienstaufruf erfolgen. Ein wesentlicher Vorteil verbindungsorientierter Dienste ist somit, die Tatsache, dass nach einem Verbindungsaufbau nicht mehr jedes Datenpaket einer Verbindung sämtliche (bereits ausgehandelte) Parameter beinhalten muss (z.B. Adresse, Dienstqualität etc.), was eine Reduzierung des Anteils der übertragenen Kontrollinformationen und somit letztendlich auch der Netzlast selbst bedeutet.



Die Klasse der verbindungsorientierten Dienste und das im Bereich der Datenkommunikation zentrale Konzept der Verbindung wird im Folgenden vertieft.

Ein verbindungsorientierter Dienst kann übersichtlich mithilfe eines endlichen Automaten in die vier Betriebsphasen „Verbindungsaufbau“, „Datenaustausch“, „Verbindungsabbau“ und „Verbindungsabbruch“ aufgeteilt werden.



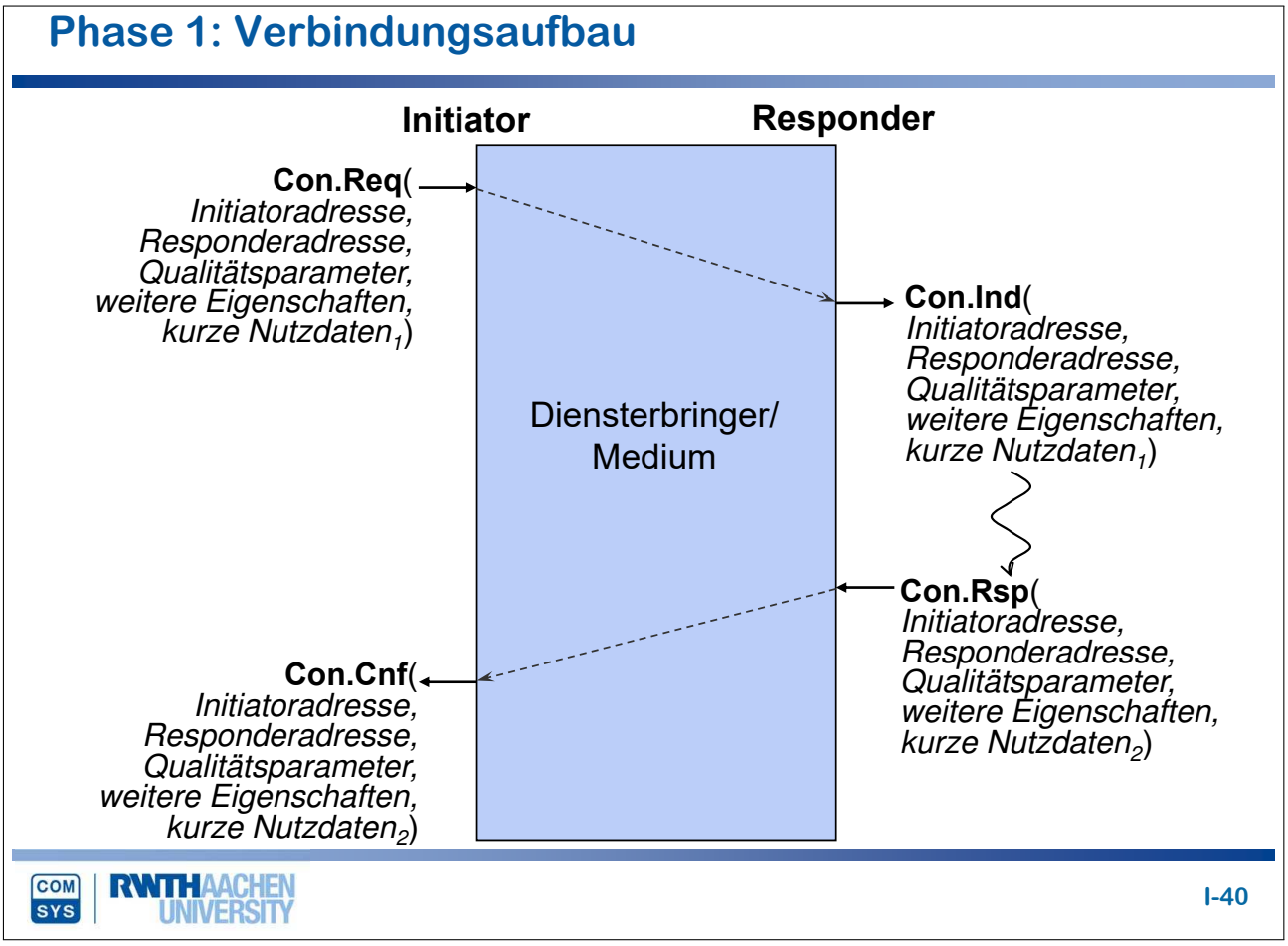
Ein Verbindungsaufbau/-abbau hat im Weg/Zeit-Diagramm folgendes Aussehen:

Phase 1: Verbindungsaufbau durch Con.Req/Ind/Rsp/Cnf

Phase 2: Datenaustausch durch Data.Req/Ind

Phase 3: Verbindungsabbau durch Dis.Req/Ind (unbestätigter Verbindungsabbau)

Üblicherweise ist der Verbindungsaufbau bestätigt, damit ein Dienstnutzer weiß, ab wann die Datenübertragung möglich ist. Datenaustausch und Verbindungsabbau können wie hier unbestätigt, oder auch bestätigt implementiert werden, abhängig von den unterliegenden Diensten.



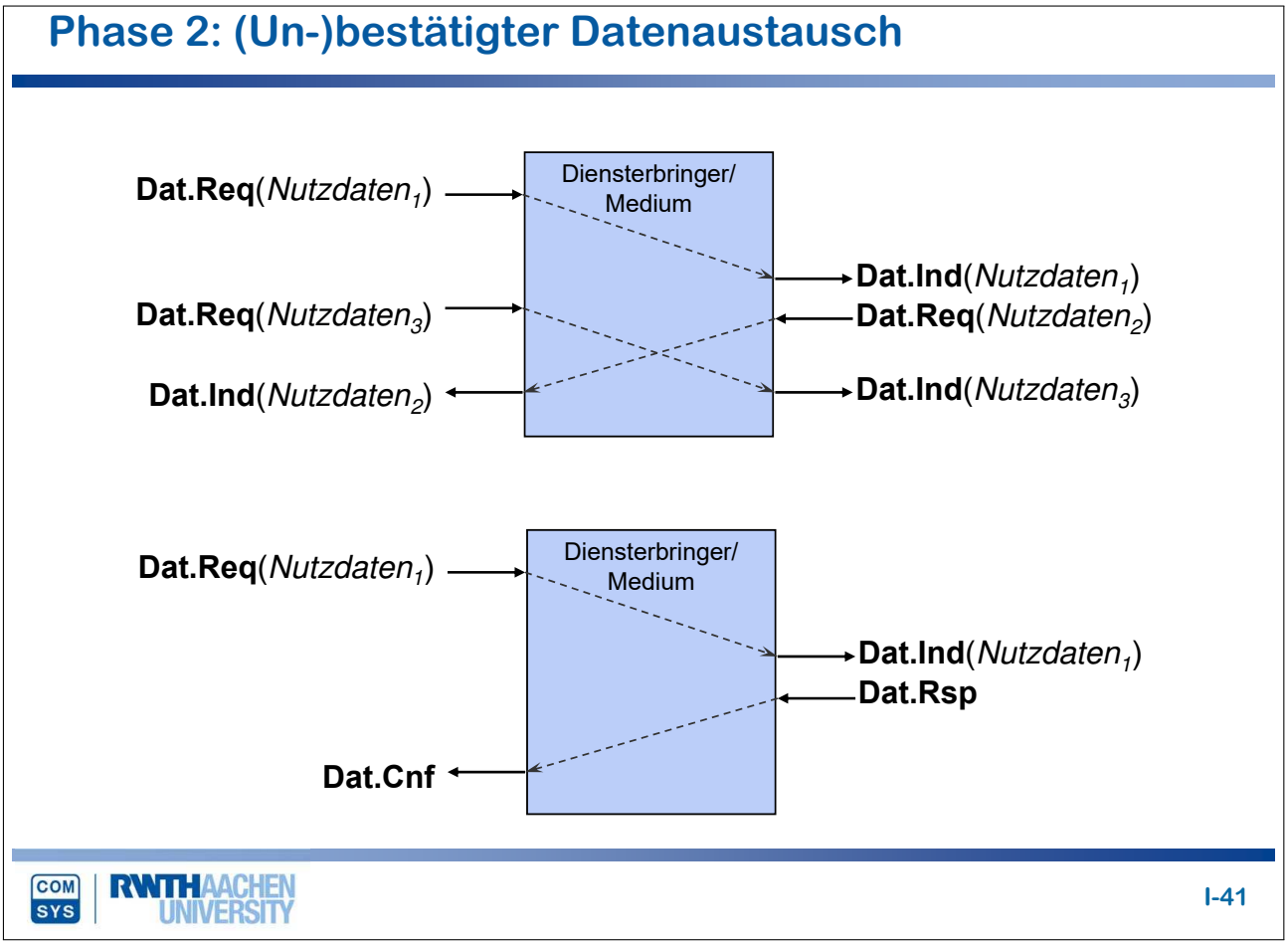
Im Folgenden werden die vier Phasen eines verbindungsorientierten Dienstes genauer behandelt, beginnend mit der Verbindungsaufbauphase. Zunächst sollen die beim Verbindungsaufbau beteiligten Parameter näher betrachtet werden.

Folgende Parameter werden in den Dienstprimitiven zum Verbindungsaufbau genutzt:

- Initiator-/Responderadresse: SAP-Adressen
- Qualitätsparameter: weitere Kontrollparameter. Denkbare Qualitätszusicherungen eines Datenaustauschs sind verlustfreie Übertragung, reihenfolgetreue Auslieferung, hochpriorer Datenaustausch, …
- Kurze Nutzdaten: Dienstnutzer kann während des Verbindungsaufbaus zusätzliche Daten austauschen. Die zusätzlichen Daten stehen i.d.R. in direktem Zusammenhang mit der aufzubauenden Verbindung. Sie können z.B. für die Angabe genutzt werden, für welche überliegenden Protokolle die Verbindung dienen soll (z.B. Transportverbindung zum Zweck des Dateitransfers). Der Responder hingegen kann die kurzen Nutzdaten dazu verwenden, einen Statuscode z.B. für den Grund der Ablehnung einer Verbindung anzugeben.

Ein interessanter Aspekt des Verbindungsaufbaus betrifft die Aushandlung der Qualitätsparameter. Der Initiator gibt gewisse Parameter, die er sich wünscht, vor. Diese werden vom Dienst bzw. vom Responder geprüft und können heruntergestuft werden.

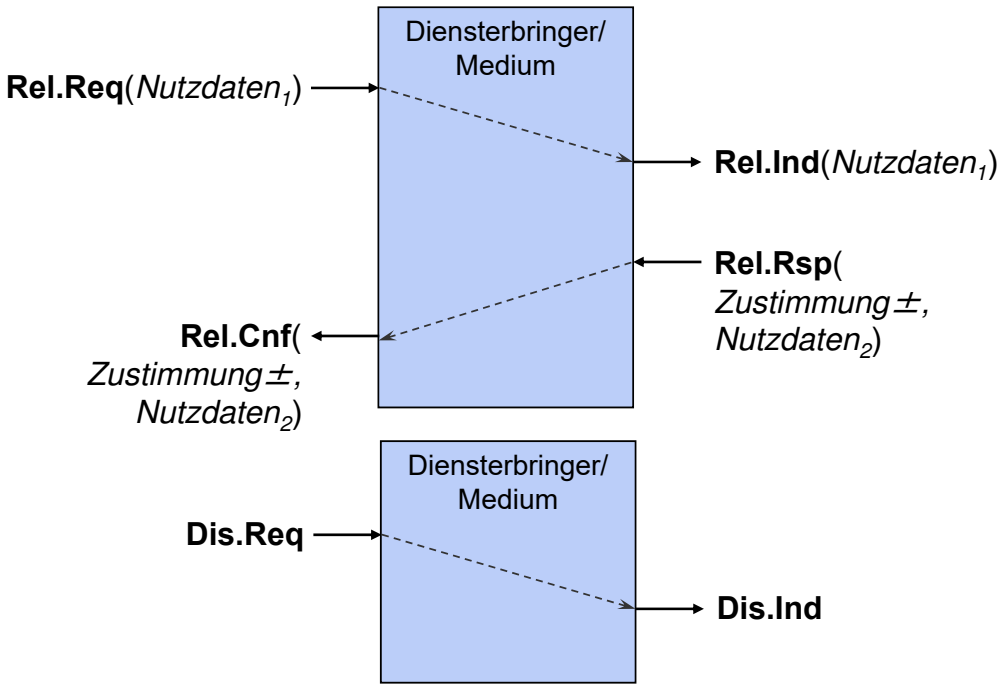
Konkretes Beispiel: Der Initiator wünscht sich einen Datendurchsatz von 200 MBit/s, der Dienst kann ihm aufgrund des zugrundeliegenden Mediums nur 100 MBit/s anbieten und der Responder reduziert letztendlich auf 50 MBit/s.



Die zweite Phase ist der Datenaustausch, also die eigentliche „Produktiv-Phase“, in der die Nutzdaten über die aufgebaute Verbindung ausgetauscht werden. Im Gegensatz zum bestätigten Datenaustausch besitzt der unbestätigte Datenaustausch dabei den Vorteil, dass Sender und Empfänger nur lose synchronisiert sind; das bedeutet, dass sich Sender und Empfänger nur wenig abstimmen müssen und sich entsprechend wenig im Fortschreiten ihrer Arbeit behindern.

Wie die Folie zeigt, beinhalten die beim unbestätigten Datentransfer verwendeten Dienstprimitive Dat.Req und Dat.Ind nur einen Parameter (die unbestätigt zu übertragenden Nutzdaten). Alle weiteren Parameter wurden bereits beim Verbindungsaufbau ausgetauscht/ausgehandelt. (In diesem Beispiel – es können durchaus noch Parameter vorhanden sein.) Die Dienstprimitive Dat.Rsp und Dat.Cnf, die beim verbindungsorientierten Datenaustausch zusätzlich nötig werden, beinhalten gar keine Parameter mehr.

Phase 3: Verbindungsabbau



Der Verbindungsabbau wird durch einen der Dienstnehmer durch die Inanspruchnahme des entsprechenden Dienstprimitivs eingeleitet. Je nachdem, ob es sich um einen bestätigten oder unbestätigten Verbindungsabbau handelt, gibt es unterschiedliche Dienstprimitive:

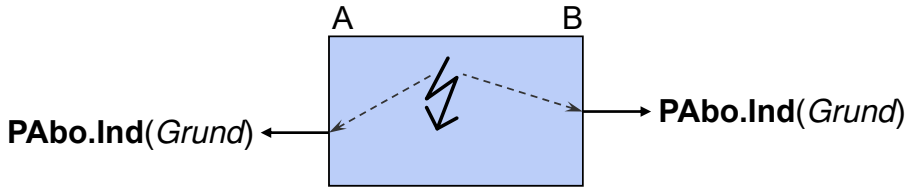
- Release (Rel): bestätigter Verbindungsabbau
- Disconnect (Dis): unbestätigter Verbindungsabbau

Der wesentliche Unterschied zwischen dem bestätigten und dem unbestätigten Verbindungsabbau besteht darin, dass beim bestätigten Verbindungsabbau der zweite Teilnehmer dem Verbindungsabbau widersprechen kann.

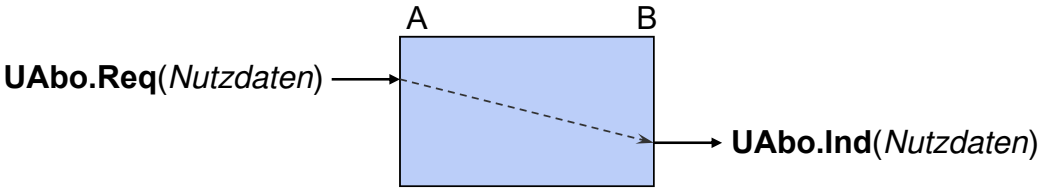
Diese Möglichkeit ist durch den Parameter „Zustimmung+/-“ in Rel.Rsp/Cnf gegeben. In den Nutzdaten könnte der Dienstnutzer dann angeben, warum er es ablehnt, die Verbindung abzubauen.

Phase 4: Verbindungsabbruch

- **Abbruch durch den Diensterbringer (Provider Abort, PAbo)**



- **Nutzerabbruch (User Abort, UAbo)**

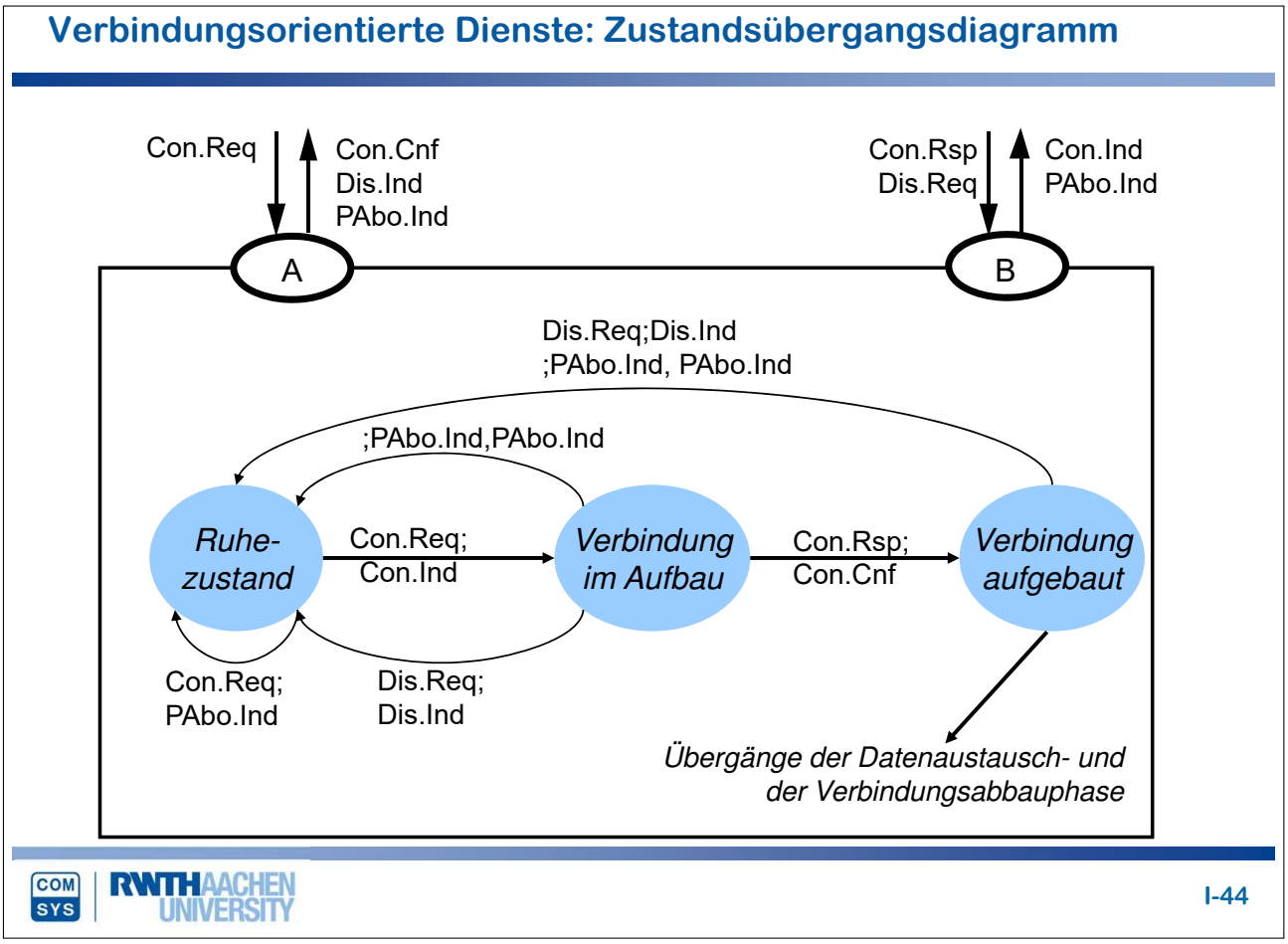


Abschließend soll auch die vierte und letzte Phase des verbindungsorientierten Dienstes behandelt werden, der Verbindungsabbruch. Wie der Begriff „Abbruch“ schon vermuten lässt, handelt es hier um einen ungewünschten Abbau der Verbindung aufgrund einer Ausnahmesituation. Diese Phase kommt also im regulären Betrieb nicht vor.

Die Ausnahmesituation kann in den unterliegenden Schichten aufgetreten sein, wofür das Dienstprimitiv PAbo zur Verfügung steht:

- Provider Abort (PAbo): werden durch den Dienst selbst oder die Diensterbringer tieferer Schichten verursacht
- User Abort (UAbo): Instanzen der über den Dienst kommunizierenden Dienstnutzer oder höhere Schichten

Der Dienst UAbo wird von der Instanz der betrachteten Schicht genutzt, wenn die Ausnahmesituation gerade in dieser Schicht auftritt und statt eines geordneten Verbindungsabbaus ein erzwungener Verbindungsabbruch notwendig ist.



Ein Weg-Zeit-Diagramm beschreibt das zeitliche Verhalten an den beteiligten Dienstzugangspunkten. Dieser von außen beobachtbare Ablauf ist aufgrund der Ursache-Wirkung-Beziehung von Dienstprimitiven gegeben, die sich geeignet durch sogenannte *Zustandsübergangsdiagramme* spezifizieren lässt (die Arbeitsweise eines solchen Diagramms entspricht im wesentlichen dem eines endlichen Automaten). Wie die Abbildung andeutet, wird damit das Verhalten des unterliegenden Mediums (Dienstes) beschrieben. Dieses befindet sich in einem gewissen Zustand – im Beispiel ist zwischen drei unterschiedlichen Zuständen zu unterscheiden. Gewisse eingehende Dienstprimitive werden in jedem Zustand akzeptiert und führen zu Zustandsübergängen: im Zustand „Verbindung in Aufbau“ existieren zu den Primitiven Con.Rsp und Dis.Req Übergänge; außerdem kann hier auch noch ein spontaner Übergang durch einen Abbruch stattfinden. Die Zustandsübergänge werden also durch das Dienstprimitiv ausgelöst, das vor dem „;“ angegeben ist. Nach dem Semikolon sind ein oder mehrere Dienstprimitive angegeben, die an den entsprechenden SAPs auszugeben sind. Die Beschriftung kann also als „Ereignis ; ausgelöste Aktion“ verstanden werden.

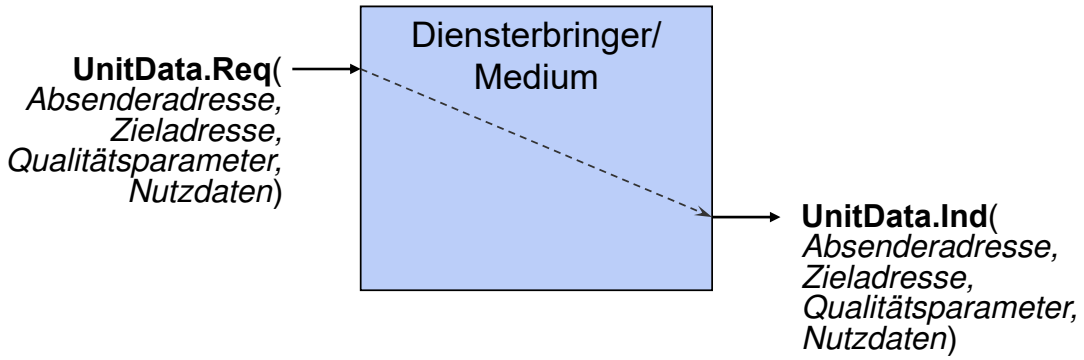
- Syntax: <auslösendes Dienstprimitiv ; resultierendes Dienstprimitiv>, z.B. Con.Req;Con.Ind
- Falls die erste Angabe fehlt, handelt es sich um einen spontanen Übergang.

Ein Weg-Zeit-Diagramm beschreibt einen konkreten Weg durch das Zustandsübergangsdiagramm, durch welches das Verhalten des Mediums dargestellt ist. Das Zustandsübergangsdiagramm verfügt also über größere Ausdrucksstärke.

Der gewünschte Normalweg führt vom Ruhezustand durch Con.Req zum „Verbindung in Aufbau” und durch Con.Rsp zu „Verbindung aufgebaut”. Von hier geht es dann in die diversen anderen Dienstphasen des Datenaustauschs und des Verbindungsabbaus.

Zu bemerken ist noch, dass in obigem Beispiel die Initiative zum Verbindungsaufbau immer vom Dienstnehmer am SAP A ausgehen muss, während der Dienstnehmer am SAP B eine Verbindung nur akzeptieren, ablehnen oder auch abbauen kann. Im Normalfall werden jedoch beiden Dienstnehmern sämtliche Dienstprimitive zur Verfügung stehen (d.h. alle Aktionen können sowohl von der einen wie auch von der anderen Seite ausgelöst werden). SAP A und B beschreiben also keine konkreten Geräte, sondern eher mögliche Rollen bei der Dienstnutzung.

Datagramm-Dienst



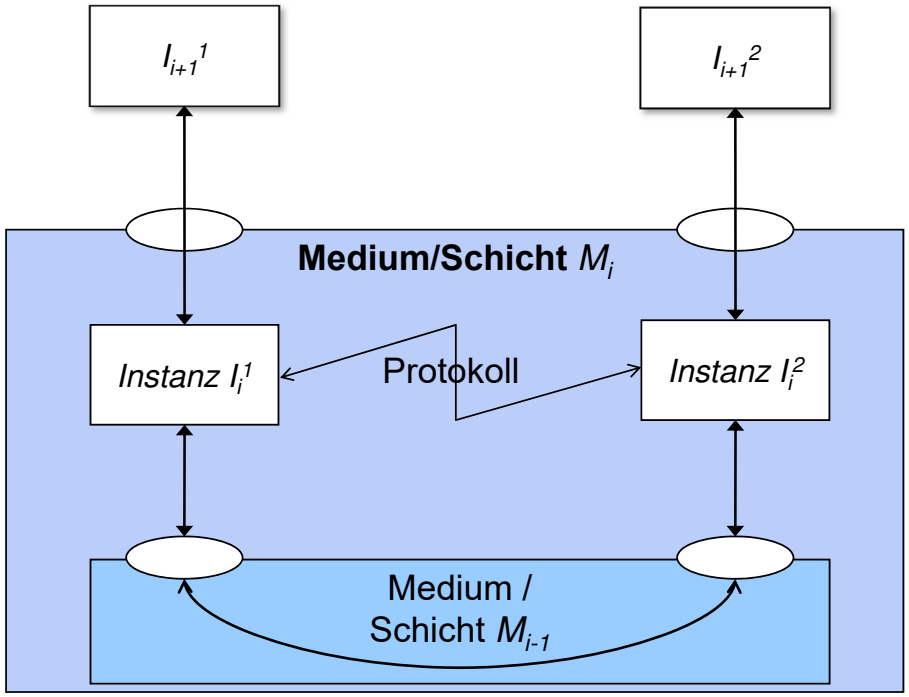
- **Wichtige Spezialkategorie: *Datagramm-Dienst***
  - Verbindungsloser Dienst
  - Unbestätigt und unzuverlässig
    - Unterstützt keine Auslieferungsdisziplin, z.B. keine Garantie für Reihefolgetreue

Wie zuvor gesehen, kann die Datenübertragungsphase bei einem unbestätigt sein, sie kann aber auch bestätigt umgesetzt werden. Bei einem verbindungslosen Dienst verwendet man unbestätigte Dienstprimitive. Dies sagt allerdings noch nichts über die Zuverlässigkeit einer Datenübertragung aus – bei einer Übertragung im Internet kann es z.B. durch Netzüberlastung zu einem Verlust einzelner übertragener Dateneinheiten kommen. Verwendet man einen bestätigten Dienst, erkennt man solche Verluste auf jeden Fall daran, dass keine Bestätigung erfolgt und kann die Anfrage noch einmal stellen. Bei einem unbestätigten Dienst gibt es diese Möglichkeit nicht. Allerdings kann der Dienst intern mit Bestätigungen arbeiten (die dem Dienstnutzer verborgen bleiben), so dass der Dienstnutzer auch bei Verzicht auf Bestätigungen davon ausgehen kann, dass die Daten zuverlässig übertragen werden.

Der Datagramm-Dienst ist ein verbindungsloser Dienst (ohne Bestätigungen), der unzuverlässig arbeitet (also auch intern keine Bestätigungen verwendet). Das einzige Dienstprimitiv, das angeboten wird, ist UnitData, also ein „einzelnes Datum“. In diesem Dienstprimitiv müssen alle für die Übertragung relevanten Kontrollinformationen enthalten sein. Das sind zum Beispiel sämtliche Adressierungsinformation (Absender- und Zieladresse) und die Qualitätsparameter, in denen der Sender seine Wünsche bzgl. der Datenübertragung formuliert. Da der Dienst unbestätigt ist, können die Qualitätsparameter nicht verhandelt werden, d.h. das Medium versucht, den Wünschen des Senders so gut es geht zu entsprechen.

- **Einführung und Begriffe**
  - Was ist Datenkommunikation?
  - Information, Daten, Signale
  - Netze
- **Allgemeine Grundlagen**
  - Dienste
  - Protokolle und Schichten
  - Kommunikationsarchitekturen

Kommunikationsprotokolle



Bislang wurde behandelt, welche Kommunikationsfunktionalität in Form von Diensten an einer Schichtgrenze (der Dienstschnittstelle) angeboten wird. D.h.

- Der Kommunikationsdienst beschreibt, welche Funktionalität (was) angeboten wird.
- Das Kommunikationsprotokoll beschreibt, wie diese Funktionalität erbracht wird.

Zur Untersuchung der Protokolle ist es also notwendig, in die Schicht hineinzuschauen und die darin ‚arbeitenden' Komponenten zu beleuchten. Im Kommunikationsbereich hat sich für diese Komponenten der Begriff der „Instanzen" (engl. *entities*) durchgesetzt.

Ein Protokoll legt das Verhalten einer Instanz (*entity*) fest. Je zwei Partnerinstanzen (*peer entities*) realisieren das Protokoll und erbringen dadurch den entsprechenden Kommunikationsdienst.

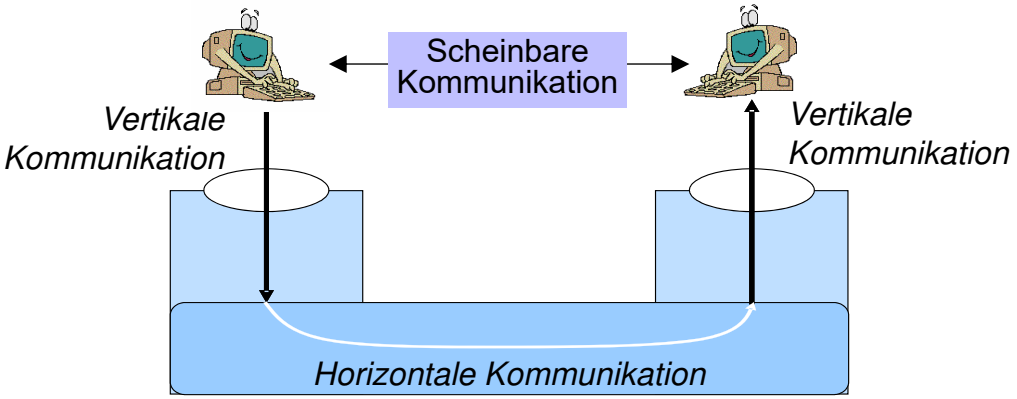
Charakteristisch für ein Kommunikationsprotokoll ist, dass es zwischen zwei Protokollinstanzen abläuft. Im Folgenden soll der oben angedeutete Zusammenhang zwischen Protokoll und Dienst verdeutlicht werden.

Die Partnerinstanzen, zwischen denen das Protokoll abläuft, erbringen einen Dienst des Mediums M_i und setzen dazu auf einem Dienst des Mediums M_{i-1} auf.

In der Abbildung auf der Folie wird gemäß dem zuvor eingeführten Dienstmodell von Medium gesprochen. Dieser Begriff ist gleichbedeutend mit dem Begriff der Schicht oder Ebene. In diesem Beispiel sind die Instanzen*I*(Medium *i*, Nr. 1) und *I*(Medium *i*, Nr. 2) die Partnerinstanzen. Sie erbringen für die darüberliegenden Instanzen Schicht-*i*-Dienste und nutzen die vom darunterliegenden Medium angebotenen Medium-*(i-1)*-Dienste.

Scheinbare & tatsächliche Kommunikation

- **Unterschied: Horizontale und vertikale Kommunikation**
  - Horizontal: Kommunikation zwischen Instanzen der Protokolle
    - Kommunikation findet scheinbar direkt zwischen den Kommunikationspartnern statt
  - Vertikal: tatsächliche Kommunikation innerhalb eines Protokollstapels
    - Tiefer liegenden Schichten erbringen den Kommunikationsdienst

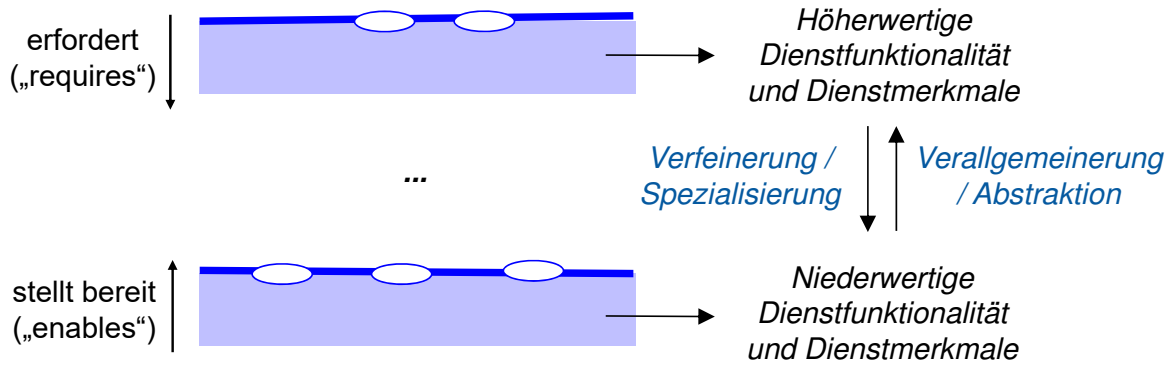


Kommunikationsprotokolle regeln intern innerhalb einer Schicht den Datenaustausch zwischen Instanzen der jeweiligen Schicht. Die Kommunikation der Instanzen über das Protokoll findet scheinbar direkt zwischen den Instanzen statt. Dies wird auch horizontale Kommunikation genannt. Nur auf der untersten Schicht findet horizontale Kommunikation über das physikalische Medium tatsächlich statt; auf allen höheren Ebenen ist die horizontale Kommunikation nur virtuell.

Tatsächlich erbringen aber die tiefer liegenden Schichten den Kommunikationsdienst. Die Protokolldateneinheiten werden an die nächsttiefer liegende Schicht weitergereicht und durch diese zur Protokollinstanz des Kommunikationspartners gebracht. Da die nächsttiefere Schicht dadurch eventuell wieder auf tiefere Schichten zurückgreift, durchlaufen die zu versendenden Daten einen Stapel an Diensten, die durch eigene Protokolle realisiert werden. Man spricht hierbei von einem Protokollstapel (Protokoll-Stack). Durch diesen Protokoll-Stack werden die Daten vertikal weitergereicht.

Geschichtete Architekturen

- **Schichtenmodell eines Kommunikationssystems**
  - *Schichtenstruktur*
    - Bereitstellen von Diensten an der Schnittstelle nach oben
    - Nutzung von Diensten an der Schnittstelle nach unten
  - Grundlegende Umsetzung der Schichtung: *Referenzmodelle*



Kommunikationssysteme sind komplexe Systeme – wie bereits zuvor erwähnt, hilft die Zerlegung der gesamten Kommunikationssysteme in einzelne Module dabei, z.B. Flexibilität und Wartbarkeit eines Kommunikationssystems zu verbessern.

Das übliche Modell dabei ist eine Schichtung der einzelnen Module. Dies erlaubt die Festlegung einer einheitlichen Struktur: das Prinzip der Schichtung erlaubt es, die Interaktionen zwischen den einzelnen Modulen übersichtlich zu halten.

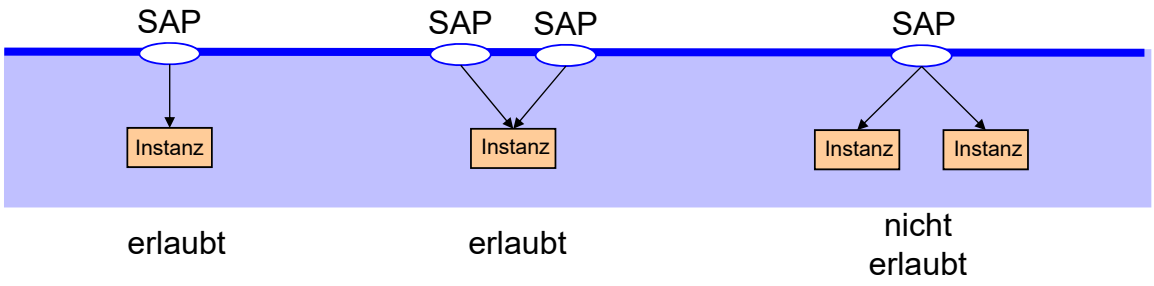
Von oben gesehen bietet ein geschichtetes Kommunikationssystem eine Verfeinerung der Kommunikationsfunktionalität: die Aufgabe einer Schicht kann in Funktionen zerlegt werden, die einen Dienst auf einem bestimmten Abstraktionsniveau erbringen. Diese Funktionen nutzen selbst niederwertige Dienste zur Erledigung ihrer Aufgaben. Von unten her gesehen findet eine Verallgemeinerung statt: die Funktionen einer unteren Schicht können zusammengefasst und so angereichert werden, dass die nächsthöhere Schicht einen höherwertigen Dienst bereitgestellt bekommt.

Eine Schicht wird dabei durch alle Instanzen des Rechnernetzes, welche die geforderte Funktionalität der Schicht realisieren, repräsentiert.

Um weltweit interoperable Protokolle und SAPs zu schaffen, werden Referenzmodelle definiert, die vorgeben, welche Funktionalitäten eine bestimmte Schicht erbringen sollte und wie die Schnittstellen und Protokollinstanzen auszusehen haben.

Gliederung von Kommunikationssoftware

- **Protokolle**
  - Verhaltenskonventionen, die die zeitliche Folge von Kommunikation zwischen den diensterbringenden Instanzen festlegen und das Format (Syntax und Semantik) der auszutauschenden Dateneinheiten bestimmen
    - Austausch von *Protocol Data Units (PDU)*
  - Ein Dienst wird durch Protokollinstanzen des Diensterbringers bereitgestellt und wird über Dienstzugangspunkte (SAPs) realisiert

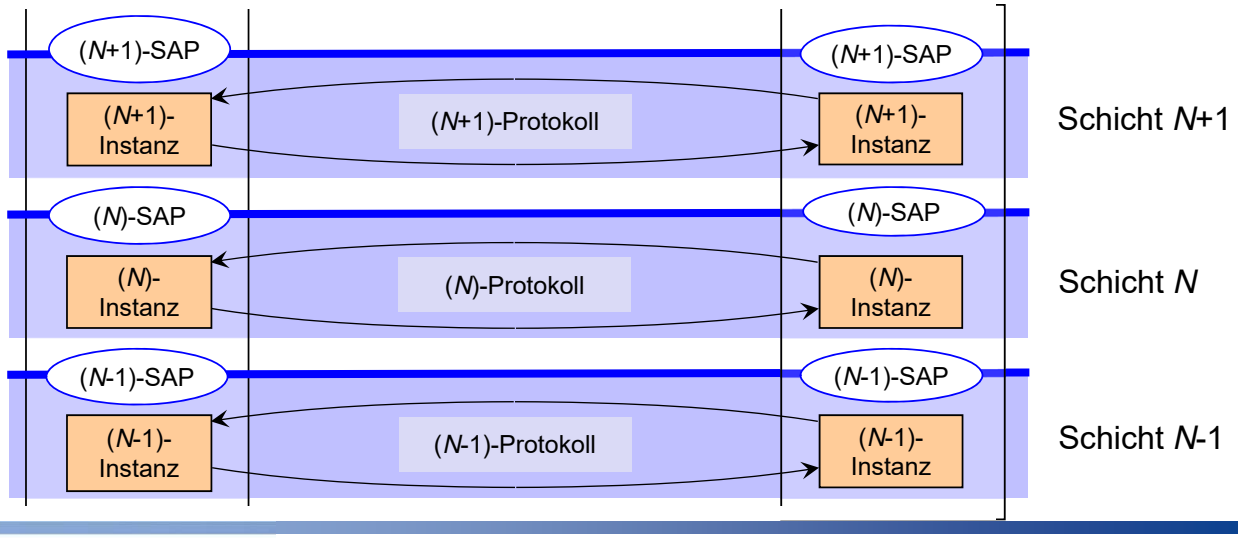


Eine Instanz nimmt die an einem SAP anliegenden Dienstprimitive entgegen, analysiert sie und tritt zur Erbringung des Dienstes mit der Instanz in Wechselwirkung, der der SAP des Kommunikationspartners zugeordnet ist (Partnerinstanz). Diese Wechselwirkung erfolgt durch den Austausch von Nachrichten beziehungsweise Dateneinheiten, den Protokolldateneinheiten (Protocol Data Unit - PDU). Solch eine PDU besteht zum einen aus den auszutauschenden Daten selbst, zum anderen aus Kontrollinformationen, die der Kommunikation der Instanzen selbst zur Erfüllung ihres Dienstes dient.

Protokolldateneinheiten sind gewöhnlich als Bitfolgen mit Signifikanz der einzelnen Positionen kodiert, um ihre eindeutige Interpretation in Rechnern mit unterschiedlichen Datenformaten zu gewährleisten.

Protokolle und Protokollinstanzen

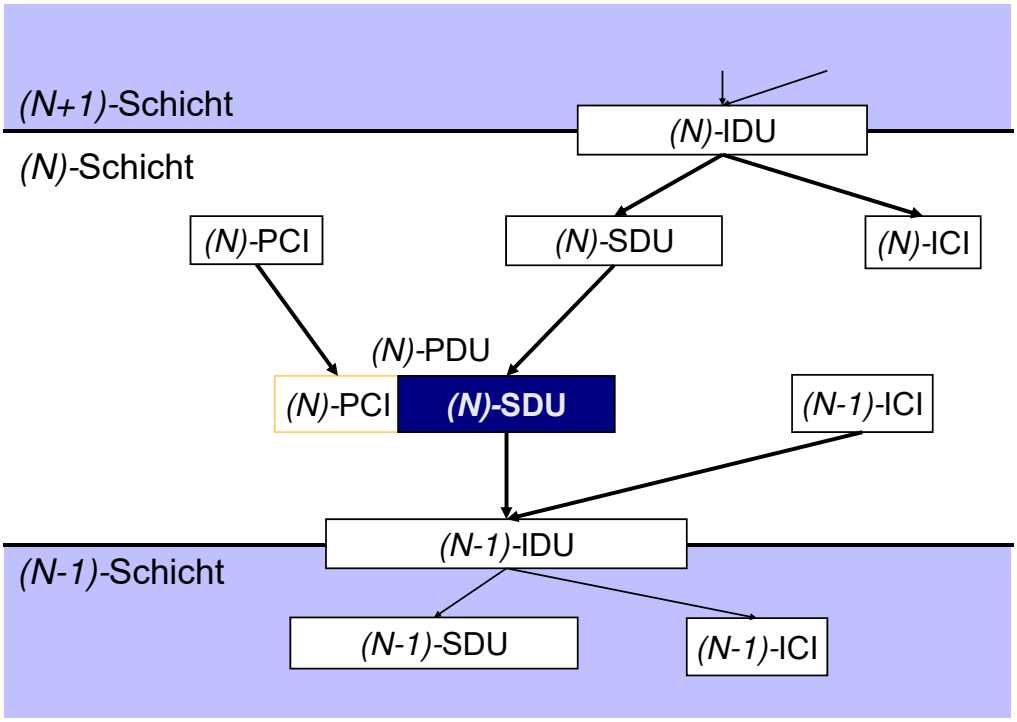
- *Protokolle*: Regeln/Formate für den Datenaustausch zwischen Rechnersystemen
- *Protokollinstanzen*: Protokollimplementierungen in den Rechnersystemen
  - *Schichtung* von Protokollinstanzen



Der Dienst, den eine Schicht erbringt, wird durch das Zusammenwirken von Protokollinstanzen dieser Schicht (knapp auch Schichteninstanzen) auf den beteiligten Rechnersystemen erbracht – eine Schicht zieht sich also über ein gesamtes Kommunikationssystem hinweg. Diese Protokollinstanzen sind konkrete Implementierungen eines Protokolls auf einem Rechnersystem. Das Protokoll enthält alle Regeln, die für die Erbringung des Dienstes auf dieser Schicht erforderlich sind, sowie Formate, in denen die Daten zwischen den Protokollinstanzen ausgetauscht werden. Das Protokoll wird stets nur zwischen den Instanzen auf dieser Schicht abgearbeitet.

Auf einer Schicht existieren meist unterschiedliche Protokolle, die verschiedene Dienste erbringen können – wie beispielsweise ein Protokoll für verbindungsorientierte Kommunikation und ein Protokoll für verbindungslose Kommunikation, welche die vorher bereits dargestellten Dienstprimitive und Regeln zu ihrem Austausch definieren.

Dienst- und Protokolldateneinheiten: Vertikale Kommunikation im Detail



Im sogenannten ISO/OSI-Referenzmodell (welches später genauer erläutert wird) wird das Zusammenspiel von zwei Instanzen an der Dienstschnittstelle in einem Dienstmodell genau festgelegt. Die Zusammenarbeit erfolgt in den folgenden Schritten:

(1) Die *(N+1)*-Instanz übergibt an der Dienstschnittstelle eine *(N)*-Interface Data Unit (IDU).

(2) Die *(N)*-Instanz teilt die *(N)*-IDU in zwei Teile auf:

a) transparent zu übertragende Nutzdaten: *(N)*-SDU (Service Data Unit)

b) Steuerinformationen für die Dienstschnittstelle: *(N)*-ICI (Interface Control Information)

(3) Zur Übertragung der *(N)*-SDU ist gemäß dem vereinbarten Kommunikationsprotokoll eine *(N)*-PCI (Protocol Control Information) zu erzeugen, die gemeinsam mit der *(N)*-SDU die *(N)*-PDU bildet. Im Verlauf der Vorlesung werden die PCI als Header bezeichnet, da sie meist der SDU vorangestellt werden.

Diese (N)-PDU wird transparent zwischen den Protokollinstanzen der Schicht N übertragen.

(4) Zur Übertragung der PDU durch die darunterliegende Schicht ist entsprechende Kontrollinformation für die untere Schnittstelle zu erzeugen, die *(N)*-PDU und diese *(N-1)*-ICI bilden die *(N-1)*-IDU.

Zusammenfassung der Begriffe:

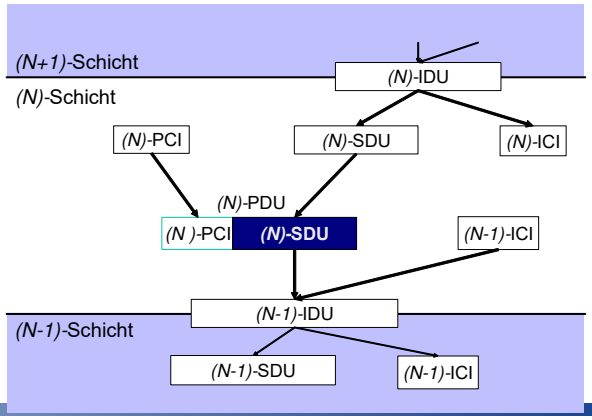
- *(N)*-Schnittstellendateneinheit (IDU): Dies ist die Dateneinheit, die die *(N+1)*-Schicht an die *(N)*-Schicht über den Dienstzugangspunkt (Service Access Point - SAP) übergibt. Sie enthält die Nutzdaten *(N)*-SDU und Schnittstellenkontrollinformationen *(N)*-ICI.
- *(N)*-Schnittstellenkontrollinformationen (ICI): Diese Dateneinheit enthält Informationen über den auszuführenden Dienst, z.B: Parameter des Dienstes, Länge der SDU, ...
- *(N)*-Dienstdateneinheit (SDU): Daten der *(N+1)*-Schicht (*(N+1)*-PDU), die transparent zwischen *(N)*-SAPs übertragen werden.
- *(N)*-Protokollkontrollinformationen (PCI): Informationen, die (über vertikale Kommunikation) zwischen den einzelnen *(N)*-Instanzen ausgetauscht werden, um den Protokollablauf zu steuern.
- *(N)*-Protokolldateneinheit (PDU): Daten, die zwischen *(N)*-Partnerinstanzen ausgetauscht werden. Eine *(N)*-

PDU setzt sich aus *(N)*-PCI und *(N)*-SDU zusammen.

Packets inside the Linux Kernel

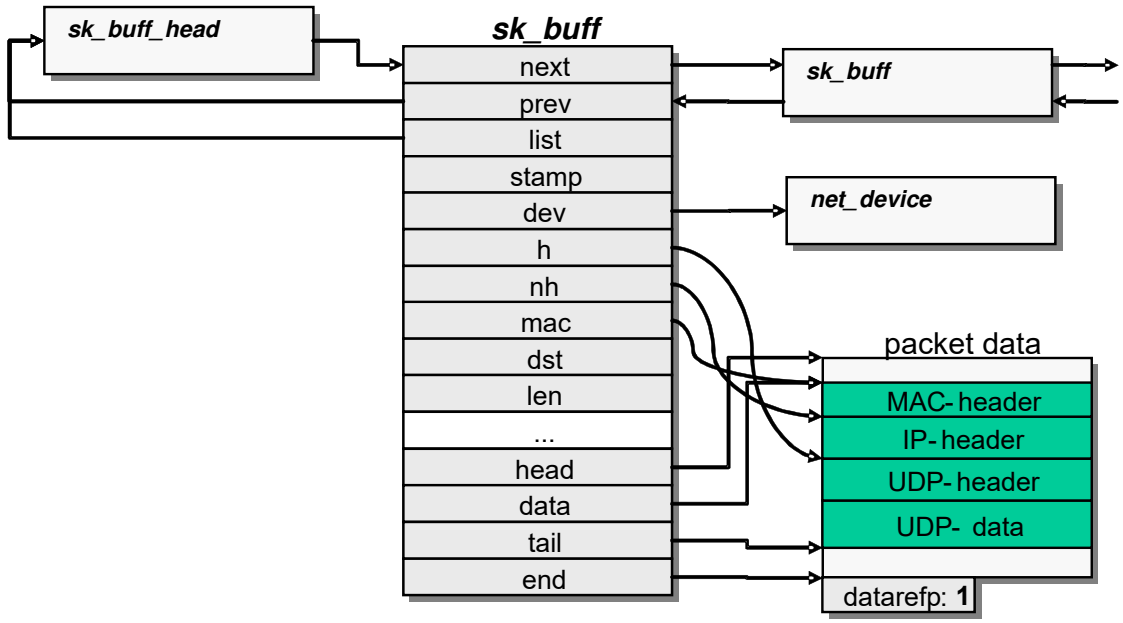
Auszug aus der Vorlesung
Communication Systems Engineering

- **Some questions:**
  - How are packets represented in the Kernel?
  - How to realize SDU, PDU, IDU, ICI, etc.?
  - Which variables do we need?
  - One structure for all protocols
    or individual structures for each protocol?
  - …

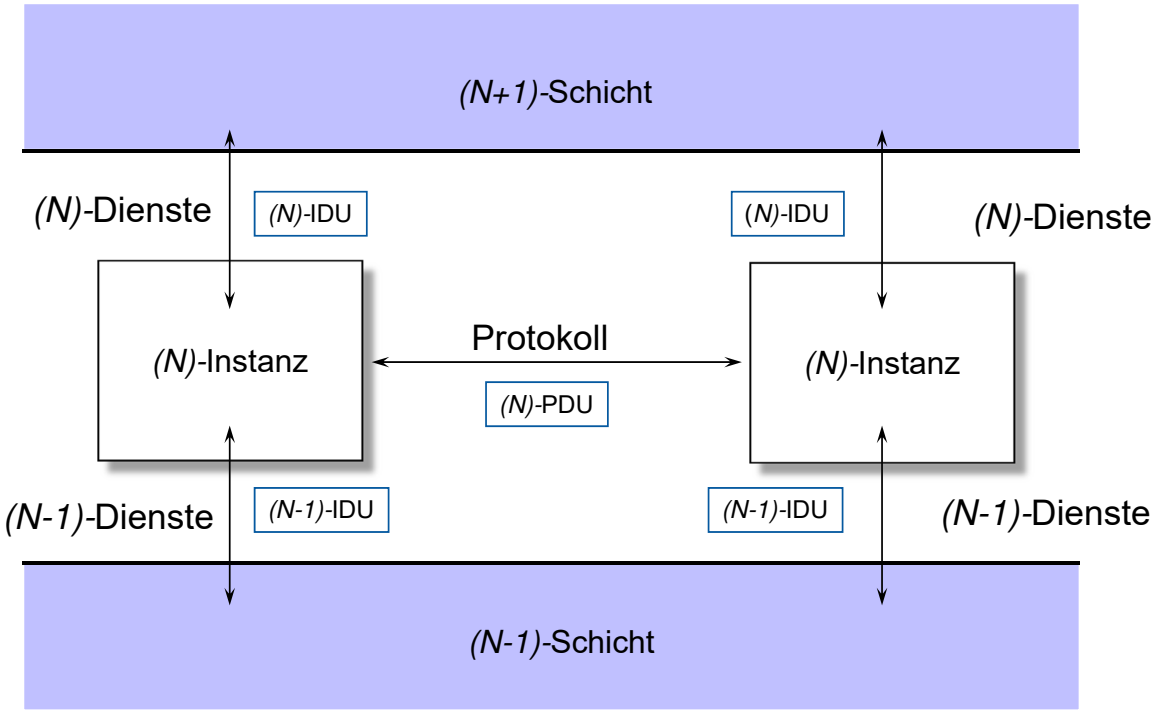


Socket Buffer: Represents Packets in the Linux Kernel

Auszug aus der Vorlesung
Communication Systems Engineering

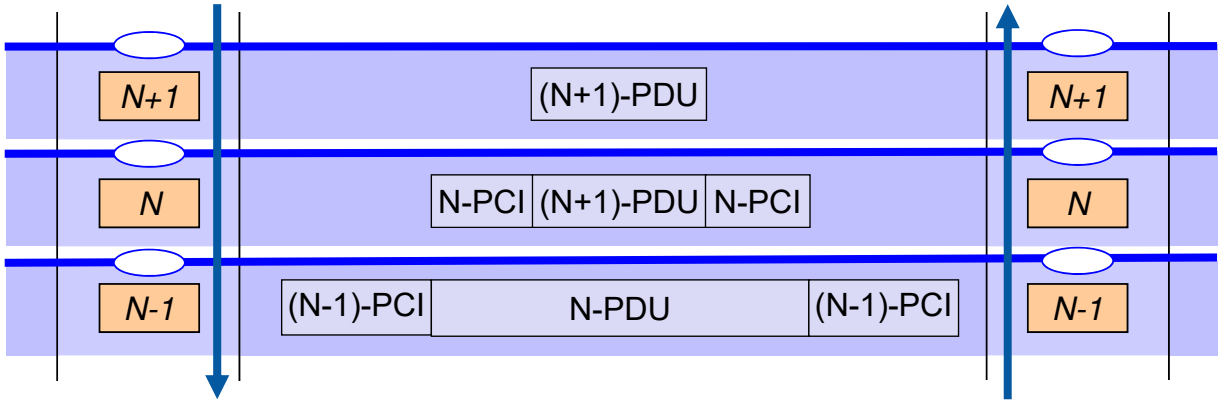


Kommunikation innerhalb/zwischen Systemen



Transparenz der Schichten

- **Transparenz**
  - PDU der Schicht (*N*+1) wird in Schicht *N* als Nutzdaten betrachtet
    - Keine Berücksichtigung des Inhalts, *Transparenz* der Benutzerdaten (SDU)
      - (*N*+1)-PDU == (*N*)-SDU
    - Führt in Schichtenarchitektur in den unteren Schichten zu einer ständigen Vergrößerung des Anteils der PCIs am gesamten Datenumfang



Jede Schicht soll unabhängig von allen anderen implementiert werden können, um Interoperabilität zwischen beliebigen Implementierungen auf allen Schichten zu gewährleisten. Dies hat aber auch einen Nachteil: die PDU einer Schicht, die an die nächsttiefere Schicht weitergereicht wird, wird dort als SDU aufgefasst – als Nutzdateneinheit, die unangetastet bleibt und mit neuen Kontrollinformationen versehen wird. Dadurch besteht eine PDU auf unterster Ebene aus den eigentlichen Nutzdaten selbst plus einer ganzen Kette an PCIs. Die zu übertragene Datenmenge wird umso mehr aufgebläht, je mehr Schichten verwendet werden. Dies kann einen enormen Overhead erzeugen – und das nicht nur bei der Menge der zu übertragenen Informationen, sondern auch bei der Verarbeitung der Daten durch den gesamten Protokoll-Stack.