PROTOKOLLARCHITEKTUREN

Problemstellung:

Wie können (zwei oder mehrere) Knoten eines Netzwerks miteinander kommunizieren?

Lernziele:

- Die Teilnehmer sollen das Zusammenspiel von Service(-Interfaces) und Protokollen erklären können.
- Die Schichten-Strukturierung in Protokoll-Hierarchien und die Auswirkungen auf Netzwerk-Software sollen verstanden werden.

Inhalt:

- Netzwerk-Architektur
- Der Protokollbegriff
- Grundfunktionen in Protokollen
- Protokoll-Hierarchien
- Service-Eigenschaften
- Service-Primitiven
- Das OSI-7-Schichten-Modell
- Das TCP/IP-Referenzmodell
- Vergleich von OSI und TCP/IP Modellen
- Implementierung von Netzwerk-Software

Netzwerk-Architektur

In Netzwerk-Systemen kommunizieren Applikationsprogramme miteinander, wobei der physikalische Austausch von Information über die Hardware (Netzwerk-Adapter) stattfindet.

Zwischen der Applikation und der Hardware gibt es mehrere Abstraktionsebenen. In dem (vereinfachten) Beispiel in Abb. PKA–1 sind die Verbindungsebenen von Host zu Host und von Prozeß zu Prozeß eingezeichnet.

Dabei bedient sich die jeweils höhere Abstraktionsschicht der *Services* der darunter liegenden Schicht.

Zwischen Kommunikationspartnern innerhalb einer Schicht ist die Kommunikation über *Protokolle* definiert.

Diese prinzipiellen Zusammenhänge sind in Abb. PKA–2 dargestellt. Der Rest dieses Kapitels dient zur weiteren Erklärung dessen.

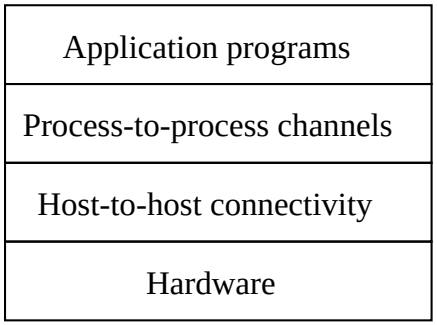


Abb. PKA–1 Ein geschichtetes Netzwerk-System

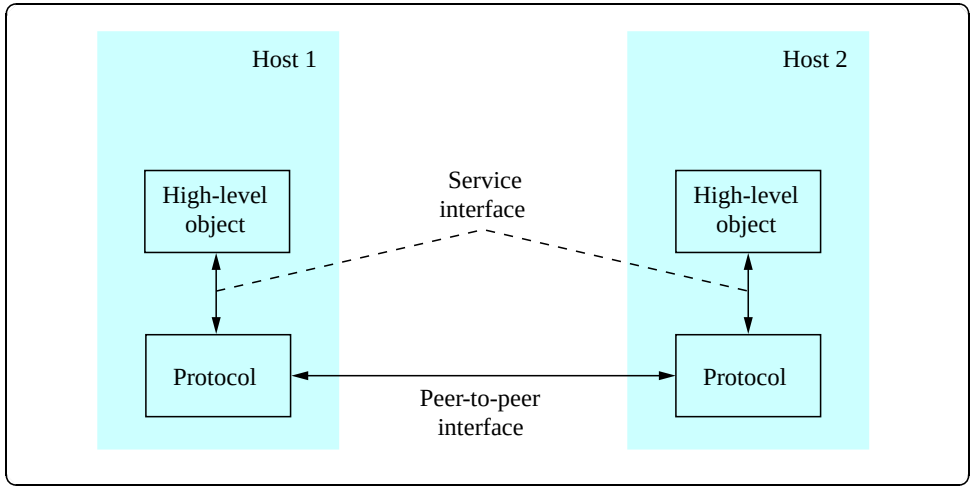


Abb. PKA–2 Service- und Peer-Interfaces

Der Protokollbegriff

Tanenbaum (1996):

> "Basically, a **protocol** is an agreement between the communicating parties on how communication is to proceed. As an analogy, when a woman is introduced to a man, she may choose to stick out her hand. He, in turn, may decide either to shake it or kiss it, depending, for example, on whether she is an American lawyer at a business meeting or a European princess at a formal ball. Violating the protocol will make communication more difficult, if not impossible."

Definitionen

- Kommunikationsvorgang
  - wechselseitiger Daten- bzw. Informationsaustausch zwischen (mindestens) zwei Partnern
  - gegenseitiges Verstehen der Partner ist entscheidend
- informeller Protokollbegriff
  - Protokoll: Menge der Konventionen, die zur Sicherstellung der Verständigung der Partner eingehalten werden müssen
- Kommunikationsprotokoll: Gesamtheit der Regeln, nach denen die Kommunikation zwischen den Partnern abgewickelt wird
  - Festlegung
    - ◇ der Protokollsyntax (wie wird kommuniziert),
    - ◇ der Protokollsemantik (was wird kommuniziert) und
    - ◇ des Protokoll-Timings (wann wird kommuniziert)
- Syntax: Definitionen für
  - Datenformate und
  - Codierungen
- Semantik: Verwaltungsanteile der Kommunikation
  - Spezifikation für
    - ◇ Steuer- und Kontrollinformationen
      - ▷ welche Steuersequenzen (Einzelaktionen) gibt es
      - ▷ wie kann auf eine Einzelaktion reagiert werden
    - ◇ Informationen zur Fehlerbehandlung
  - Abgrenzung der Nutzdaten von den Kontrolldaten
- Timing
  - Zeitliche Reihenfolge der Einzelereignisse
  - Flußkontrolle

Grundsätzliche Eigenschaften von Protokollen

Direkte versus indirekte Kommunikation

- direkte Kommunikation
  - zwischen den Partnern besteht direkte Verbindung
  - keine Beteiligung von Zwischenknoten an der Kommunikation

  Beispiele
  - einzelne Punkt-zu-Punkt Verbindung
  - Mehrpunktverbindungen
    - komplexeres Protokoll wegen der erforderlichen Zugriffskontrolle

- indirekte Kommunikation
  - Beteiligung von Zwischenknoten an der Kommunikation

  Beispiele
  - Kommunikation über ein Vermittlungsnetz
  - Kommunikation zwischen Partnern, die unterschiedlichen Netzen angehören
    - beachte: die Forderung nach Netzstruktur-unabhängigen Protokollen ist nicht immer erfüllbar

Symmetrische versus asymmetrische Protokolle

- symmetrische Protokolle
  - gleichartige Partner kommunizieren

- asymmetrische Protokolle
  - Partner besitzen unterschiedliche logische Ausrichtungen
    - Beispiel: Kommunikation zwischen Benutzer- und File-Server-Prozeß

Offene Kommunikation

- offene Kommunikation
  - uneingeschränkte Kommunikationsmöglichkeit zwischen beliebigen Partnern
  - keine Abhängigkeit von
    - Hardware-Ausstattung
    - Software-Monopolisten (wie z.B. Microsoft)
    - Netzzugehörigkeit
    - Entfernung

- Absicherung einer offenen Kommunikation
  - juristische Offenheit
    - Anschlußmöglichkeit beliebiger Benutzer an Kommunikationssystem
  - Benutzeroffenheit
    - Offenlegung der Schnittstelle zum Kommunikationssystem, d. h. Bereitstellung einer Software-Schnittstelle zwischen Anwenderprozeß und Kommunikationssystem
  - Offenheit der Protokollarchitektur
    - Entwicklung neuer Kommunikationsarchitekturen
    - Integration in das vorhandene Kommunikationssystem

  Grundvoraussetzung: Standardisierung und Normierung der Protokollarchitektur

Monolithische versus strukturierte Protokolle

- monolithischer Protokollaufbau
  - Realisierung der gesamten Kommunikationslogik in einem einzigen Protokoll
  - Probleme (vergl. Entwicklung großer, monolithischer Programmsysteme)
    - hoher Komplexitätsgrad (mit den Folgen)
    - Fehleranfälligkeit
    - Inflexibilität

- strukturierte Protokolle
  - Gliederung des Protokolls in logisch zusammengehörige, handhabbare Einheiten

Grundfunktionen in Protokollen

Fragmentieren und Zusammenfügen

- Datenaustausch zwischen Kommunikationspartnern auf der Anwendungsebene
  - Datenstrom
  - Datenblöcke (Nachrichten)
- Datenübertragung auf der Kommunikationssystemebene
  - Fragmentierung der Daten in kleinere Einheiten (Pakete)

Verbindungsverwaltung

- verbindungsorientierte Kommunikation
  - Verbindungsaufbau
  - Datenübertragung
  - Verbindungsabbau
- Behandlung von Verbindungsunterbrechungen
- Protokollanpassungen auf Teilstrecken einer Verbindung

Synchronisation

- Abstimmung der miteinander kommunizierenden Partner (Protokollmodule)
- Verwendung von Synchronisationsnachrichten

  Berücksichtigung von
  - Zeitverzögerungen und
  - Übertragungsfehlern

Adressierung

- wechselseitige Identifikation der Kommunikationspartner auf den verschiedenen Ebenen
- Verwendung von
  - Namen → welches Objekt
  - Adressen → wo ist das Objekt zu finden
  - Routen → wie ist das Objekt zu erreichen

Fehlerprüfung

- Fault Control
- Entdeckung und Korrektur von Übertragungsfehlern auf Bitebene
- Behandlung des Verlusts bzw. der Verdopplung von Paketen

Flußkontrolle

- Flow Control
- gegenseitige Anpassung der Sende- und Empfangsseite bezüglich der übertragenen Datenmenge
- Überlastung des Empfängers muß vermieden werden

Überlastkontrolle

- Congestion Control
- Schutz des Gesamtnetzes bzw. von Netzteilen vor Überlastung

Betriebsmittelverwaltung

- Prioritätenregelungen
  - Priorisierung von Kontrolldaten gegenüber Nutzdaten
- Zusicherung von Benutzeranforderungen: Garantien für
  - Mindestdurchsatz
  - maximale Wartezeiten
- Vergabe und Kontrolle von Zugriffsrechten auf Netzressourcen
- Gebührenerfassung in öffentlichen Netzen

Multiplexing

- auf physikalischer Ebene
  - TDM-Verfahren
    (Time Division Multiplexing)
    $\Longrightarrow$ "abwechselnd Senden"

  - FDM-Verfahren
    (Frequency Division Multiplexing)
    $\Longrightarrow$ "Senden auf verschiedenen Frequenzen"

- beim Übergang von einer Protokollebene zur nächsten
  - mehrere Verbindungen einer höheren Ebene gehen über eine gemeinsame Verbindung einer niedrigeren Ebene
  - Downward Multiplexing: eine Verbindung einer höheren Ebene wird in mehrere Verbindungen einer niedrigeren Ebene aufgespalten

Protokollhierarchien

- Organisation von Netzwerken (der Kombination von Software und zugrunde liegender Software) in mehreren **Schichten** bzw. **Ebenen**
  - Anzahl, Inhalt und Funktionalität der Schichten hängt vom jeweiligen Modell ab
  - Aufgabe einer Schicht ist es, **Services** höheren Schichten anzubieten und dabei Details der Implementierung bzw. Realisierung der Services zu verbergen.
- Schicht n auf einem Rechner kommuniziert mit Schicht n der Gegenseite mittels dem **Schicht n Protokoll**.
- Die beiden Gegenseiten einer Schicht n werden **Peers** genannt.
- Daten werden nicht direkt von Schicht n zu Schicht n zweier Maschinen transportiert. Dies geschieht ausschließlich im **physikalischen Medium**, der untersten Schicht.
- Zwischen benachbarten Schichten existiert jeweils ein **Interface**. Daten von Schicht n werden jeweils über das Interface zu Schicht $n-1$ bis herunter zum physikalischen Medium gereicht, transportiert und auf der Zielmaschine wieder über die Interfaces bis zur Schicht n geleitet.
- Eine Menge von Schichten und Protokollen wird **Protokollarchitektur** genannt.
  - Eine Protokollarchitektur spezifiziert die jeweiligen Protokolle.
  - **Nicht** Bestandteil einer Protokollarchitektur sind z.B.
    - ◇ Spezifikation der Interfaces
    - ◇ Implementierung der Schichten
- Ein **Protokollstack** bezeichnet die Liste von Protokollen (ein Protokoll pro Schicht), wie sie auf einem System verwendet werden.

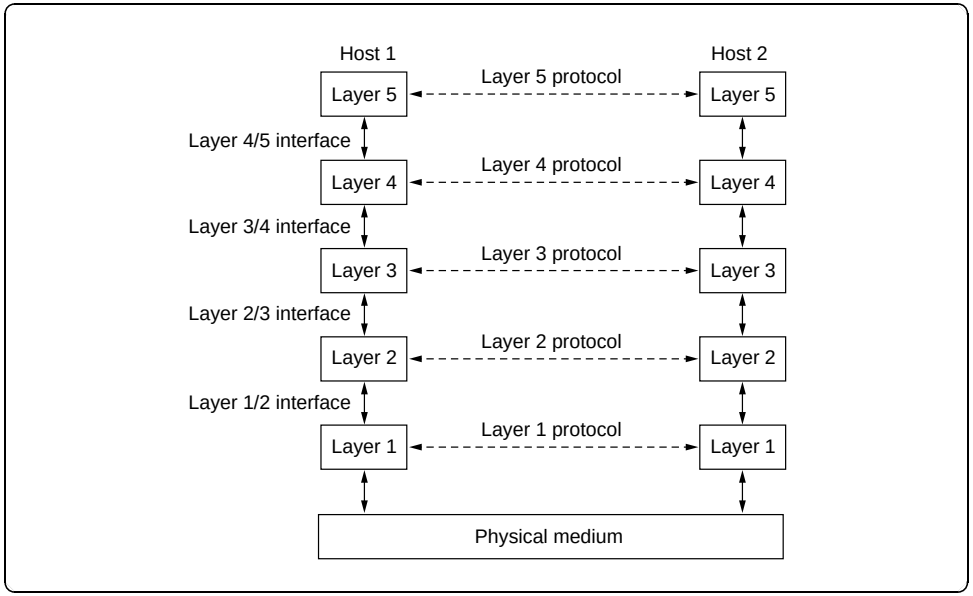


Abb. PKA–3 Schichten, Protokolle und Interfaces

Eine Beispiel-Architektur

- In Abb. PKA–4 möchten zwei Philosophen (Peer-Prozesse der Schicht 3) eine Nachricht über Kleintiere austauschen. Philosoph A spricht Urdu und Englisch, Philosoph B ausschließlich Chinesisch und Französich.
- Da Beide nicht über eine gemeinsame Sprache verfügen, engagieren sie zwei Übersetzer (Peer-Prozesse der Schicht 2). Beide Übersetzer verständigen sich über einer gemeinsame Sprache (z.B. Niederländisch).
- Philosoph A schickt die Nachricht *"I like rabbits"* über die 2/3–Schnittstelle zu seinem Kollegen. Dort kommt sie über die dortige 2/3–Schnittstelle als *"J'aime les lapins"* an.
- Die beiden Übersetzer bedienen sich wiederum zweier Sekretariate. (Peer-Prozesse der Schicht 1) Diese versenden die Nachricht über das physikalische Medium *Fax*.

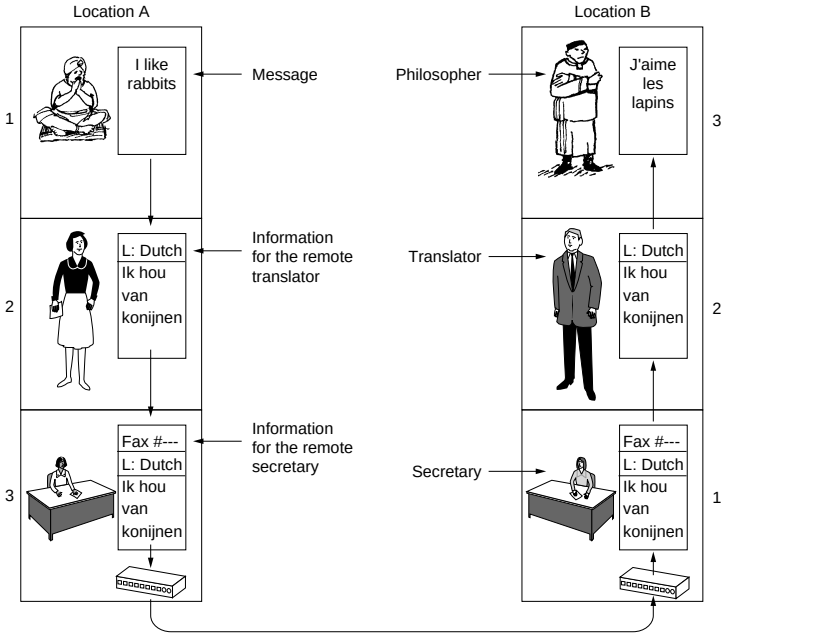


Abb. PKA–4 Die Philosophen-Übersetzer-Sekretariat-Architektur

Die Protokolle der einzelnen Schichten sind völlig unabhängig voneinander:

- Die verwendete Sprache (auf Schicht 2) könnte ebensogut Finnisch sein, so lange die Interfaces zwischen den Schichten 2 und 3 unverändert bleiben.
- Ebenso könnten die Sekretariate auch über Telefon oder email miteinander kommunizieren, ohne die anderen Schichten zu beeinträchtigen. Die anderen Schichten müßten hierzu nicht einmal informiert werden.

Ein "technisches" Beispiel

- In Abb. PKA–5 wird eine Nachricht M von einem Anwendungs–Prozeß in Schicht 5 zu seinem Peer verschickt.

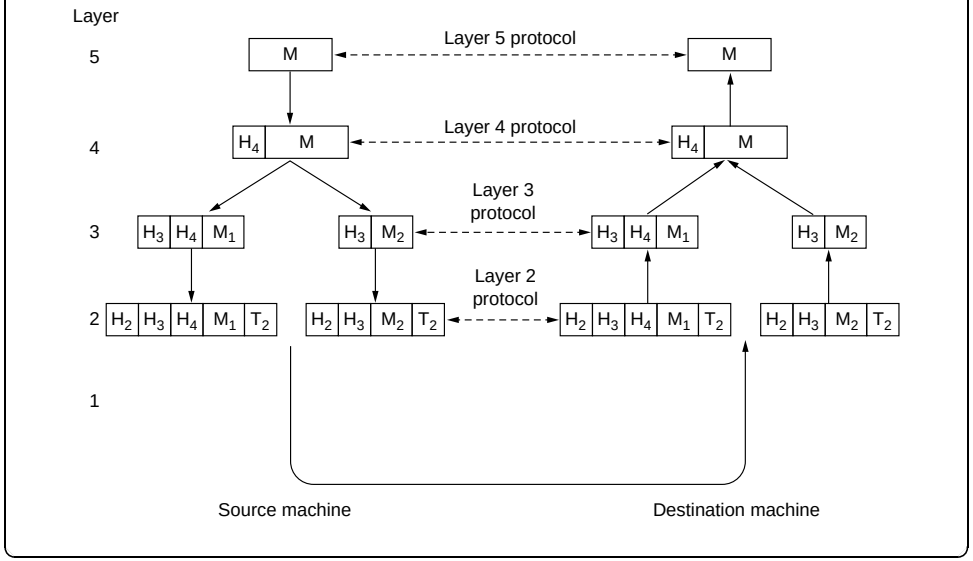


Abb. PKA–5 Informationsfluß in einem System mit 5 Schichten

- Dazu wird die Nachricht an Schicht 4 übergeben. Dort wird ein **Header** vorangestellt, der zur Identifikation der Nachricht im Peer von Schicht 4 dient:
  - Eine Sequenznummer dient zur Einhaltung der Reihenfolge mehrerer Nachrichten.
  - Weitere Header–Informationen können z.B. sein: Länge, Zeitstempel
- In Schicht 3 wird die Nachricht in mehrere Pakete aufgespalten, die jeweils die maximale Paketlänge des Protokolls der Schicht 3 nicht überschreiten. Auch hier wird weitere Header–Information vorangestellt.
- In Schicht 2 wird die Nachricht mit einem weiteren Header und einem **Trailer** versehen. Dieser enthält typischerweise eine Kontrollsumme, mit deren Hilfe Übertragungsfehler aus Schicht 1 erkannt (oder gar korrigiert) werden sollen.
- In Schicht 1 wird die Nachricht als Folge einzelner Bits übertragen.
- Auf der Empfängerseite wird der umgekehrte Prozeß durchgeführt, wobei Header und Trailer jeweils entfernt werden. Beim Übergang von Schicht 3 nach Schicht 4 wird die Nachricht M wieder zu einer Einheit zusammengefügt.
- Die Peer–Prozesse einer Schicht n kommunizieren "horizontal" miteinander. Dafür verfügen sie typischerweise über Funktionen wie:

– *SendToOtherSide*
– *GetFromOtherSide*

Diese Funktionen kommunizieren tatsächlich "vertikal" mit der Schicht $n-1$ und nicht (bzw. nur virtuell) mit ihrem Peer.

Service–Eigenschaften

Verbindsorienterte Services

- Analogie zum Telefon:

  Verbindungsaufbau, Übertragung, Verbindungsabbau

- Der Empfänger erhält alle Informationen (Objekte, Bits, ...) in exakt der Reihenfolge, in der der Sender diese abgeschickt hat.

Verbindslose Services

- Analogie zur Briefzustellung:

  Jede Nachricht ist vollständig adressiert und wird separat zugestellt.

- Die Reihenfolge, in der ein Empfänger Nachrichten erhält, ist unspezifiziert. ("Überholen" von Nachrichten ist möglich.)

Quality of Service

- Zuverlässige Services
  - Information geht niemals verloren.
  - Benötigt Quittungen für Informationen.
  - Zusätzlicher Aufwand und Verzögerungen
- Unzuverlässige Services
  - Information kann verloren gehen.
  - Verzicht auf Quittungen.
  - Reduzierter Aufwand, keine Verzögerungen durch Wiederholungen verlorener Informationen.

	Service	Example
Connection-oriented	Reliable message stream	Sequence of pages
	Reliable byte stream	Remote login
	Unreliable connection	Digitized voice
Connectionless	Unreliable datagram	Electronic junk mail
	Acknowledged datagram	Registered mail
	Request-reply	Database query

Abb. PKA–6 Verschiedene Service–Typen

Ausprägungen der Service-Typen

- vgl. Abb. PKA–6
- zuverlässiger, verbindungsorientierter Nachrichtenstrom
  - ◇ Erhaltung der Nachrichtengrenzen
  - ◇ zwei als Nachrichten der Länge n abgeschickte Nachrichten werden am Ziel auch wieder als zwei Nachrichten empfangen (und nicht z.B. als eine Nachricht der Länge $2 \cdot n$)
  - ◇ Verwendung eines Quittungsmechanismus
- zuverlässiger, verbindungsorientierter Byte-Strom
  - ◇ der Empfänger kann nicht unterscheiden, ob eine Datenmenge der Länge n Bytes z.B. über 2 Nachrichten der Länge $n/2$ Bytes oder n Nachrichten der Länge 1 Byte übertragen wurde
  - ◇ Verwendung eines Quittungsmechanismus
- unzuverlässige, verbindungsorientierte Übertragung
  - ◇ Verfälschung oder Verlust von Teilen der Daten ist tolerierbar
  - ◇ Verzicht auf den Quittungsmechanimus → Effizienzsteigerung
- unzuverlässiger Datagramm-Dienst
  - ◇ keine Verwendung eines Quittungsmechanismus

- zuverlässiger Datagramm-Dienst
  - ◇ Verwendung eines Quittungsmechanismus
- Anfrage-Antwort-Dienst
  - ◇ Verwendung eines Quittungsmechanismus
  - ◇ Quittung enthält Nutzdaten

Service–Primitive

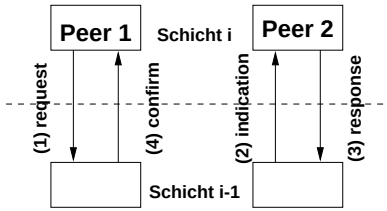
- Service-Primitive (*Operationen*)
  - Schnittstelle eines Services zum Anwender bzw. zu einer Instanz
  - Es können beispielsweise/typischerweise (z.B. im OSI-Modell) vier Klassen von Service-Primitiven unterschieden werden. (vgl. Tab. PKA–1)

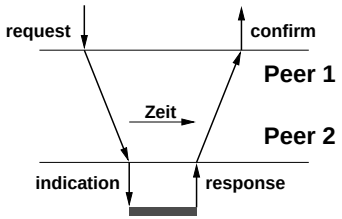
Primitive	Meaning
Request	An entity wants the service to do some work
Indication	An entity is to be informed about an event
Response	An entity wants to respond to an event
Confirm	An entity is to be informed about its request

Tab. PKA–1 Klassen von Service-Primitiven

- ◇ Anforderungs-Primitive (*Request Primitive*)
  - ▷ Anforderung eines Service, z.B.
    - ○ Etablieren einer Verbindung
    - ○ Senden von Daten
- ◇ Anzeige-Primitive (*Indication Primitive*)
  - ▷ Nachdem die Service-Anforderung von der angesprochenen Instanz bearbeitet wurde, informiert sie die Partner-Instanz darüber.
  - ▷ Diese führt dann ihren Teil des Protokolls aus z.B. beim Verbindungsaufbau.

    - ○ CONNECT.request wurde von einer Instanz bearbeitet.
    - ○ Die adressierte Partner-Instanz erhält über CONNECT.indication die Information, daß ein Verbidungsaufbau gewünscht wird.
- ◇ Antwort-Primitive (*Response Primitive*)
  - ▷ Reaktion auf eine Indication-Primitive, z.B. beim Verbindungsaufbau
    - ○ Die angesprochene Partner-Instanz teilt über CONNECT.response mit, ob sie den Verbindungsaufbauwunsch akzeptiert oder ablehnt.
- ◇ Bestätigungs-Primitive (*Confirm Primitive*)
  - ▷ Die Instanz, die über die Request-Primitive eines Service angefordert hat, erhält über die Confirm-Primitive das Ergebnis übermittelt, z.B. beim Verbindungsaufbau.
    - ○ Der Aufruf von CONNECT.confirm liefert zurück, ob die Verbindung auf der Gegenseite akzeptiert wurde oder nicht.





Beispiel eines verbindungsorientierten Service

- Ein einfacher, verbindungsorientierter Service kann z.B. über die folgenden 8 primitiven Operationen definiert werden.
- Die Operationen werden in drei Gruppen gegliedert:
  - Verbindungsaufbau (CONNECT)
  - Datenübertragung (DATA)
  - Verbindungsabbau (DISCONNECT)

1	CONNECT.request	Request a connection to be established
2	CONNECT.indication	Signal the called party
3	CONNECT.response	Used by the callee to accept/reject calls
4	CONNECT.confirm	Tell the caller whether the call was accepted
5	DATA.request	Request that data be sent
6	DATA.indication	Signal the arrival of data
7	DISCONNECT.request	Request that a connection be released
8	DISCONNECT.indication	Signal the peer about the request

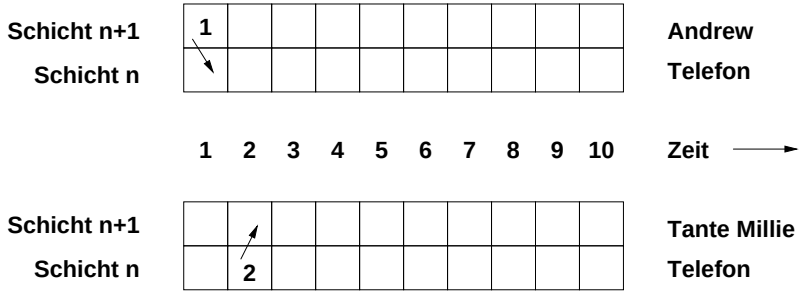
Präsenzübung: *Tea for Two*

Mit Hilfe der auf der vorigen Seite vorgestellten Service-Primitive soll eine Situation modelliert werden, in der Neffe Andrew seine Tante Millie über das Telefon zum Tee einlädt.

Bilden Sie zunächst die folgenden "Operationen" auf die acht Primitive ab:

1	Andrew wählt Tante Millies Telefon-Nummer.	
2	Ihr Telefon klingelt.	
3	Sie nimmt den Hörer ab.	
4	Andrew hört das Ende des Klingelns.	
5	Andrew lädt sie zum Tee ein.	
6	Sie hört seine Einladung.	
7	Sie sagt, sie nimmt die Einladung an.	
8	Andrew hört ihre Antwort.	
9	Andrew legt der Hörer auf die Gabel.	
10	Sie hört das Klicken und legt ebenfalls auf.	

Vervollständigen Sie folgendes Diagramm, in dem Sie die jeweiligen Nummern der Service-Primitive zu den Operationen 1 bis 10 eintragen. Markieren Sie mit Pfeilen, in welche Richtung Information transportiert wird.



Der Zusammenhang zwischen Services und Protokollen

- Services und Protokolle stellen zwei getrennte Konzepte dar.
- Service
  - Menge von Primitiven (Operationen)
  - Spezifikation der Operationen, die eine Schicht ihren Benutzern anbietet
  - keinerlei Festlegung bezüglich der Implementierung
  - ein Service bezieht sich auf eine Schnittstelle zwischen zwei Schichten
    - ◇ untere Schicht: Service-Anbieter
    - ◇ obere Schicht: Service-Nutzer
- Protokoll
  - Menge von Regeln: Festlegung von
    - ◇ Format und
    - ◇ Bedeutung

    der Pakete oder Nachrichten, die zwischen Partner-Instanzen derselben Schicht ausgetauscht werden
  - Verwendung von Protokollen durch die Instanzen zur Implementierung der Services
  - Protokolländerungen sind erlaubt, solange sie nicht die Service-Schnittstelle verändern
  - Protokoll ist für den Service-Nutzer nicht sichtbar
- Analogie aus dem Bereich der Programmiersprachen
  - Service entspricht abstrakten Datentyp
    - ◇ Spezifikation der auf einem Objekt erlaubten Operationen
    - ◇ keinerlei Festlegung bezüglich der Implementierung des Datentyps
  - Protokoll entspricht einer Implementierung des abstrakten Datentyps

Das OSI-7-Schichten-Modell

Enstehungsgeschichte und Zielsetzungen

- 1983: Veröffentlichung der ersten Version "Draft International Standard" (DIS) eines Referenzmodells für eine offene Kommunikation durch ISO
  → *Basic Reference Model for Open Systems Interconnections* (DIS 7498)
- dies ist heute als OSI-Referenzmodell bekannt
  → Reference Model of Open Systems Interconnection
- Zielsetzung
  - Beschreibung eines Rahmens für die Definition und Entwicklung von Protokollen
  - keine Standardisierung von Protokollen
  - keine Definition interner Verhaltensweisen
  - Darstellung der logischen Architektur von Kommunikationssystemen in einheitlicher Form
  - Verbindung der verschiedenen Protokolle und Netzwerktechnologien miteinander, ohne bestimmte Technologien oder Funktionalitäten auszuschließen
  - Netzanbieter können (sollen) sich an diesem Modell orientieren, um eine Zusammenarbeit zu ermöglichen
- Heutige Bedeutung
  - Referenzmodell zur Beschreibung von Rechnernetzen
  - Tatsächliche implementierte Software (TCP/IP etc.) macht keinen Gebrauch davon
- Definition eines Schichten-Modells mit 7 Schichten (vergl. Abb. PKA–7)
  - Schicht 1: Bitübertragungsschicht (*Physical Layer*)
  - Schicht 2: Sicherungsschicht (*Data Link Layer*)
  - Schicht 3: Netzwerkschicht (*Network Layer*, *Vermittlungsschicht*)
  - Schicht 4: Transportschicht (*Transport Layer*)

- Schicht 5: Sitzungsschicht (*Session Layer*, *Kommunikationssteuerungsschicht*)
- Schicht 6: Darstellungsschicht (*Presentation Layer*)
- Schicht 7: Anwendungsschicht (*Application Layer*)

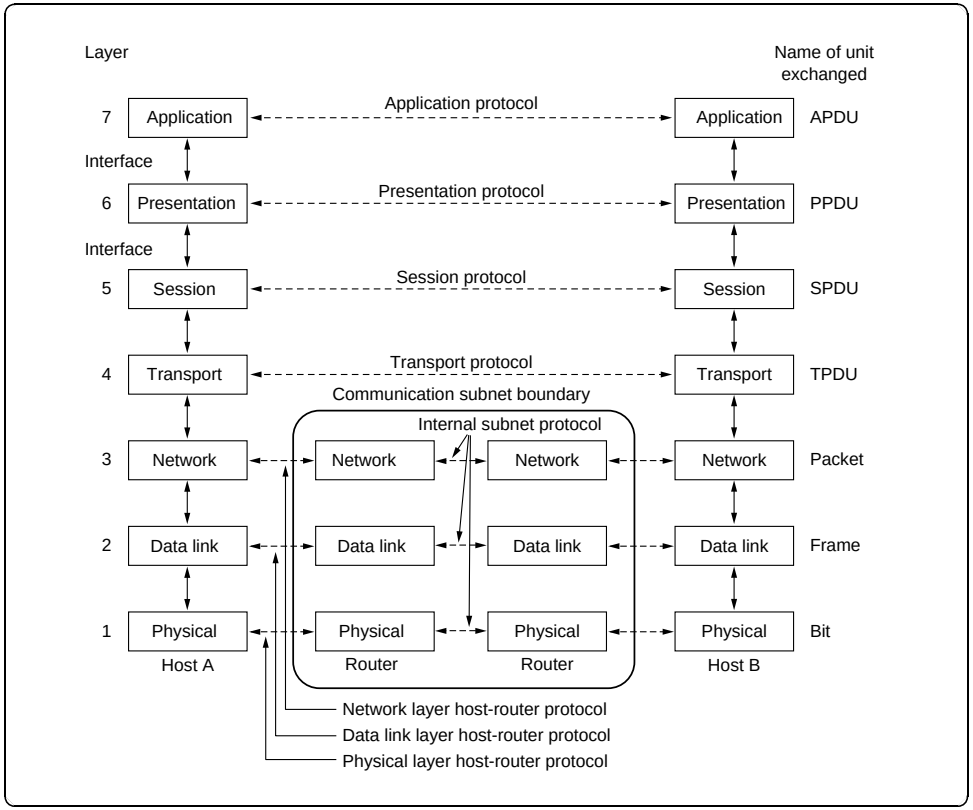


Abb. PKA–7 Struktur des OSI-Modells
(PDU = Protocol Data Unit)

- übergeordnete Funktionalität der Schichten (vgl. Abb. PKA–8)
  - Schicht 1 bis 3: netzorientierte Funktionen
    - ◇ Übertragung
    - ◇ Vermittlung
  - Schicht 5 bis 7: anwendungsorientierte Funktionen
    - ◇ Realisierung höherer Kommunikationsprotokolle
    - ◇ anwendungsspezifische Protokolle – insbesondere innerhalb Schicht 7
  - Schicht 4: Verbund von Endsystemen
    - ◇ Herstellung der Unabhängigkeit der anwendungsorientierten Schichten von den eingesetzten physikalischen Transportnetzen
    - ◇ Einsatz unterschiedlicher Transportnetze für den Verbund von Endsystemen

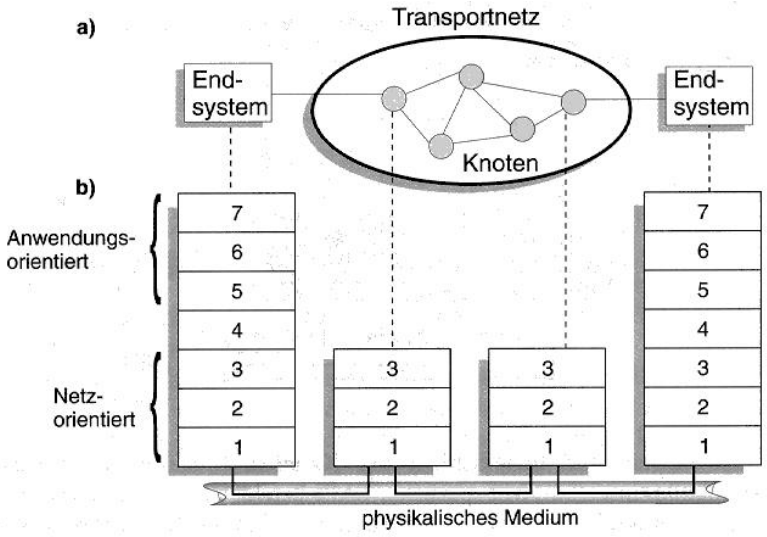


Abb. PKA–8 Kommunikationsnetz
(a) physikalische Struktur
(b) logische Architektur in Form des OSI-Modells

Schicht 1: Bitübertragungsschicht (*Physical Layer*)

- Aufgabe: Regelung des physikalischen Übertragungsablaufes
  - ungesicherte Übertragung von Bitströmen zwischen zwei Stationen
- Funktionen
  - Aktivierung/Deaktivierung der physikalischen Verbindung
  - Bitübertragung
- Grundelemente
  - mechanische Charakteristika
  - elektrische Charakteristika
  - funktionale Charakteristika
  - prozedurale Charakteristika
- Spezifikationen
  - Darstellung von 0 bzw. 1
    - ◇ Spannungsniveaus
    - ◇ Belegungsdauer der Leitung für ein Bit
  - Art der Übertragung
    - ◇ Simplex-Verfahren
    - ◇ Halbduplex-Verfahren
    - ◇ Vollduplex-Verfahren
  - Kabel und Stecker
    - ◇ Zahl der Pins
    - ◇ Pinbelegung
  - Initialisierung und Beendigung eines Übertragungsvorgangs

Schicht 2: Sicherungsschicht (*Data Link Layer*)

- Aufgabe: Absicherung der Übertragung für eine einzelne Teilstrecke
  - Flußkontrolle
  - Fehlerbehandlung
  - Synchronisation

- Funktionsangebot an die Schicht 3
  - Codeunabhängigkeit für die Nutzdaten
  - Unabhängigkeit oder leichte Anpaßbarkeit (der Schicht 3) an verschiedenartige Stationen und Übertragungsstrecken
  - hohe Effizienz bezüglich des Verhältnisses von Nutz- zu Verwaltungsanteilen
  - hohe Fehlersicherheit
  - mögliche Diensttypen
    - ◇ verbindungsunabhängig (nicht quittiert/quittiert)
    - ◇ verbindungsorientiert (nicht quittiert/quittiert)

- Funktionen der Sicherungsschicht im einzelnen
  - Auf- und Abbau der Übertragungsstrecke
  - Flußkontrolle
  - Fehlererkennung
  - Fehlerbeseitigung
  - Folgekontrolle
  - Synchronisation der Rahmen
  - Identifikation der Partner (Adressierung) bei Mehrpunktverbindungen
  - Quittungsmechanismus
  - Überwachung der physikalischen Verbindung

- Kommunikation zwischen zwei nicht direkt verbundenen Knoten
  - Aufteilung der Kommunikation zwischen den Endknoten auf mehrere Übertragungen über Teilstrecken durch das Schicht-3-Protokoll
  - Zustellung der notwendigen Kontroll- und Steuerangaben durch die Sicherungsschicht
    - ◇ Fehlerprüfung
    - ◇ Flußkontrolle
    - ◇ Folgekontrolle

Schicht 3: Netzwerkschicht (*Network Layer*)

- Eigenschaften
  - unterste Schicht, die mit der Ende-zu-Ende Kommunikation befaßt ist
  - in verschiedenen Netzen (Internet, X.25-Netze) stellt sie die oberste Schicht der IMPs dar
    - ◇ IMP: alle Schichten bis zur Netzwerkschicht (incl.)
    - ◇ Host: alle Schichten ab der Transportschicht (aufwärts)

    Folgen:
    - ◇ Dienste der Netzwerkschicht stellen Dienste des Kommunikationssystems (*Subnet*) dar
    - ◇ Dienste der Netzwerkschicht werden zur Schnittstelle zwischen dem Netzanbieter (PTT oder Common Carrier) und den Netzanwendern

- Entwurfsziele
  - Unabhängigkeit der Dienste von der Technologie des Kommunikationsnetzes
  - Abschirmung der Transportschicht von
    - ◇ Anzahl,
    - ◇ Typ und
    - ◇ Topologie

    der vorhandenen Kommunikationsnetze
    - ◇ Netzwerkadressen (an der Schnittstelle zur Transportschicht) sollten ein einheitliches Adressierungsschema verwenden

Schicht 4: Transportschicht (*Transport Layer*)

Bedeutung der Transportschicht

- wichtigste Schicht der OSI-Protokollhierarchie
- viele Applikationen nutzen die Transportschicht direkt
  - zuverlässige Übertragung eines Bitstroms reicht meist aus
  - Beispiel: UNIX-Pipes
- neben dem OSI-Protokoll existiert nur ein weiteres wichtiges Tranportprotokoll: TCP-Protokoll im Internet
- Unterste Schicht der End-To-End Protokolle
- Funktionen
  - Bereitstellung von Service-Primitiven, die unabhängig von denen der Netzwerkschicht sind
  - Kompensation von Fehlern der Netzwerkschicht
    - ◇ Paketverluste
    - ◇ zerstörte Daten

    Beispiel:
    - ◇ Netzwerkverbindung terminiert abrupt während einer langen Datenübertragung
    - ◇ keine Informationen über den Zustand der zuletzt gesendeten Daten

    Reaktion:
    - ◇ Anforderung einer neuen Netzwerkverbindung
    - ◇ Abfrage der Partner-Instanz, welche Daten angekommen sind
    - ◇ Neuübertragung der verlorengegangenen Daten
  - Vorteile für Anwendungsprogramme
    - ◇ Verwendung einer Standardmenge von Service-Primitiven
    - ◇ Unterstützung der Programmportabilität

Transportprotokolle

- Implementierung des Transportdienstes über ein Transportprotokoll zwischen den beiden Transport-Instanzen
- prinzipiell sind die Protokolle für die Schichten 2 bis 4 ähnlich bzgl. der zu behandelnden Fragestellungen:
  - Fehlerkontrolle
  - Reihenfolgeerhaltung
  - Flußkontrolle

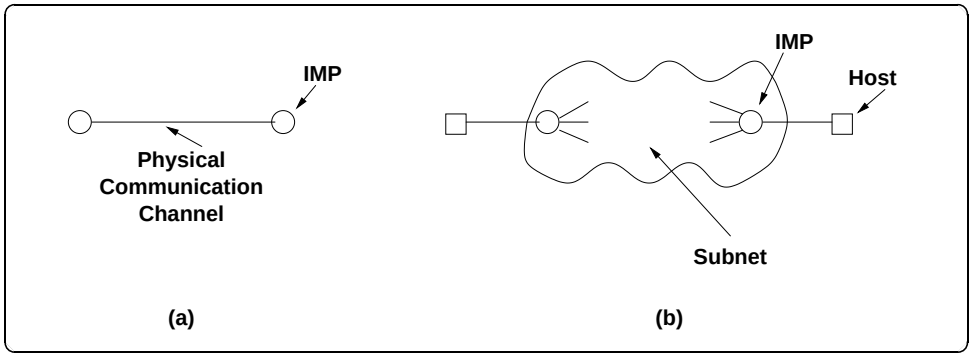


Abb. PKA–9 (a) Umgebung der Sicherungsschicht
(b) Umgebung der Transportschicht
IMP = "Interface Message Processor" (Zwischenknoten)

- entscheidende Unterschiede ergeben sich durch die Umgebung (*Environment*), in denen sie ablaufen (siehe Abb. PKA–9)
  - Sicherungsschicht
    - ◇ zwei IMPs kommunizieren über eine direkte physikalische Leitung
    - ◇ implizite Adressierung der Partner-Instanz über die verwendete Leitung
    - ◇ einfacher Verbindungsaufbau (sofern der Partner-IMP nicht ausgefallen ist)
    - ◇ die Leitung selbst besitzt (praktisch) keine Speicherkapazität
      - ▷ ein gesendeter Rahmen kommt sofort an oder geht verloren
    - ◇ einfache Verbindungsverwaltung ⇒ einfache Pufferverwaltung
  - Transportschicht
    - ◇ die Kommunikation erfolgt über ein Subnetz
    - ◇ explizite Adressierung des Zieles (TSAP) notwendig

- ◇ vergleichsweise aufwendiger Verbindungsaufbau
- ◇ das Subnetz besitzt eine hohe Speicherkapazität
  - ▷ ein Paket kann sehr lange unterwegs sein
  - ▷ in Abhängigkeit des Routing-Verfahrens können Pakete beliebig lange zirkulieren ohne physikalisch verloren zu gehen
- ◇ aufwendige Verbindungsverwaltung
  - ▷ viele Verbindungen liegen vor
  - ▷ die Anzahl der Verbindungen variiert sehr stark
  - ▷ effiziente Pufferverwaltung notwendig

Schicht 5: Sitzungsschicht (*Session Layer*)

- die Notwendigkeit dieser Schicht war (und ist) stark umstritten
- die Sitzungsschicht ist eine *dünne* Schicht
- viele Anwendungen benötigen die Funktionen nicht
  - nach dem Aufbau einer Verbindung können (optional) die meisten Funktionen *abgeschaltet* werden
- Hauptaufgabe
  - Aufbau von Verbindungen (Sitzung, *Session*)
  - Datentransfer in kontrollierter Form

Schicht 6: Darstellungsschicht (*Presentation Layer*)

- Bedeutung der Schicht hat deutlich abgenommen
  - ursprüngliche Funktion:
    - ◇ Datenkonvertierungen, z.B. ASCII nach EBCDIC
  - aktuelle Bedeutung:
    - ◇ Behandlung aller Probleme bezüglich der Datendarstellung
      - ▷ Verschlüsselung
      - ▷ Kompression
      - ▷ Umwandlung der Rechner-internen Datendarstellung in ein für die Übertragung geeignetes Format
      - ▷ Umwandlung der Daten in das interne Format der Zielarchitektur
      - ▷ Übertragung komplexer Datenstrukturen (Objekte)

- Die Aspekte der Darstellungsschicht werden heute praktisch außerhalb von Netzwerksoftware behandelt.
  - Java
  - Multimedia-Datenformate

Schicht 7: Anwendungsschicht (*Application Layer*)

- Realisierung der Anwendungs-Programme
  - Datei-Transfer
  - Terminal-Emulation
  - Electronic Mail ...

Datenübertragung im OSI-Modell

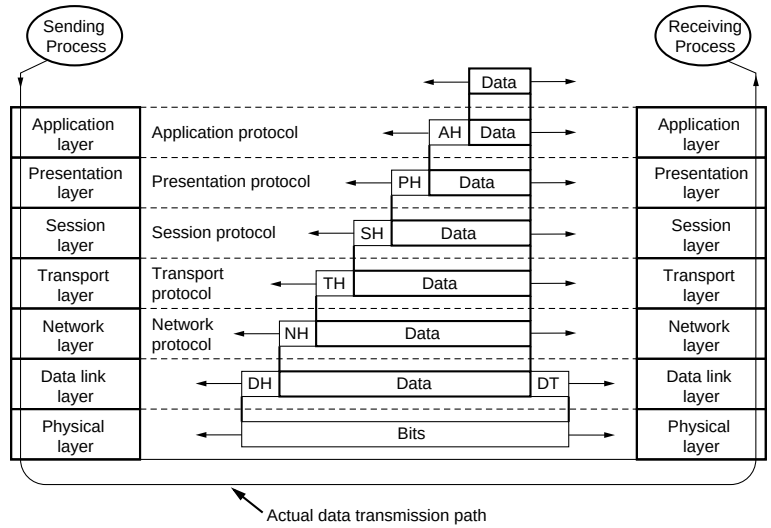


Abb. PKA–10 Datenübertragung im OSI-Modell

Das TCP/IP-Referenz-Modell

- Modell des frühen ARPANET und des heutigen Internet
- Entwicklung vom U.S. Department of Defense (DoD) finanziert
- Bezeichnung über die beiden wichtigsten Protokolle
- Ursprünglich ausgelegt für die Verbindung heterogener Netzwerke, für fehlertolerante Übertragung und für verschiedene Anwendungsgebiete (Datei-Übertragung bis Echtzeit-Sprachübermittlung)

Die Internet-Schicht

- Das TCP/IP Modell realisiert ein Paket-vermittelndes Netzwerk basierend auf einem verbindungslosen **internetwork layer**.
- Die Bezeichnung **Internet-Schicht** reflektiert die Verbindung mehrerer Netzwerke — das weltweite Internet hat seinen Namen auf ähnliche Weise bekommen.
- Für die Internet-Schicht wurde das **Internet Protocol (IP)** entworfen. IP ist vergleichbar mit der Netzwerk-Schicht des OSI-Modells (vgl. Abb. PKA–11).

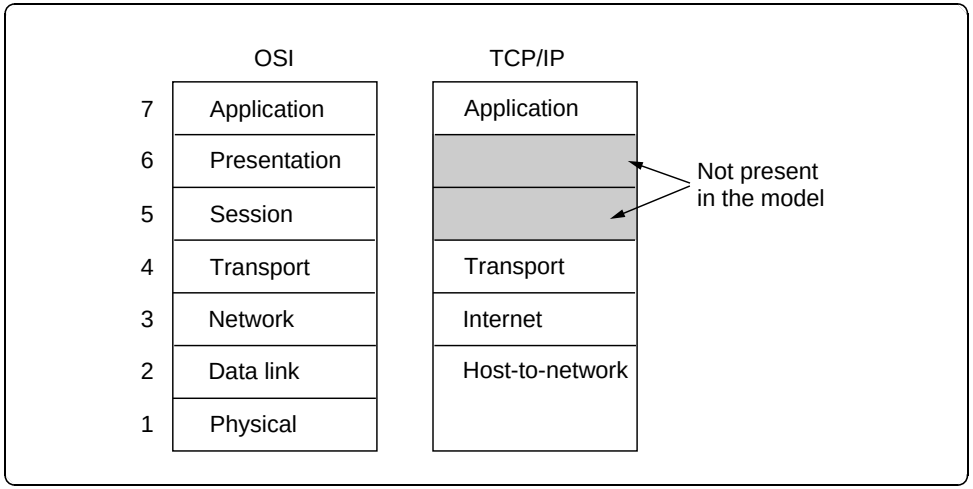


Abb. PKA–11 Das TCP/IP Referenz-Modell

Die Transport-Schicht

- Oberhalb der Internet-Schicht ist die Transport-Schicht angesiedelt.
- Hier sind zwei End-To-End Protokolle definiert:
  - **TCP (Transmission Control Protocol)**
    - ◇ Zuverlässiges, verbindungsorientiertes Protokoll
    - ◇ Fehlerfreie Übertragung von Byte-Sequenzen
    - ◇ Fragmentiert Byte-Sequenzen in einzelne Nachrichten
    - ◇ Flußkontrolle
  - **UDP (User Datagram Protocol)**
    - ◇ Unzuverlässiges, verbindungsloses Protokoll
- Die Relation von IP, TCP und UDP ist in Abb. PKA–12 dargestellt.

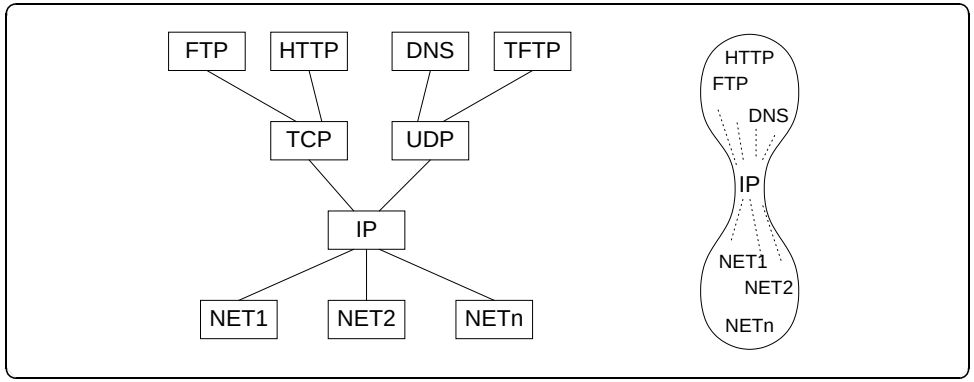


Abb. PKA–12 Links: Einige Protokolle im TCP/IP Modell
Rechts: "Sanduhr-Modell" (Hourglass Design)

Die Anwendungsschicht

- Das TCP/IP Modell enthält weder Sitzungs- noch Darstellungsschicht.
  - (Erfahrung mit dem OSI-Modell zeigt, daß dafür wenig Bedarf besteht.)
- In der Anwendungsschicht sind direkt die "higher-level" Protokolle angesiedelt: (Beispiele aus Abb. PKA–12)
  - **FTP** – Dateiübertragung
  - **HTTP** – World Wide Web
  - **DNS** – Domain Name Service
  - **TFTP** – einfache Dateiübertragung (remote boot)

Die Host-to-Network Schicht

- Protokolle unterhalb der Internet-Schicht sind im TCP/IP Modell nicht definiert.

Vergleich von OSI und TCP/IP Modellen

- Gemeinsamkeiten:
  - Stack unabhängiger Protokolle
  - ähnliche Funktionalität der Schichten
- Unterschiede:
  - OSI definiert die drei Begriffe:
    - *Service* — Funktionalität einer Schicht
    - *Interface* — Zugriff auf die Services einer Schicht
    - *Protocol* — Peer-to-Peer Implementierung der Services einer Schicht
  - TCP/IP trennt diese Begriffe nicht.
    Z.B. bietet die Internet-Schicht lediglich die Services Send-IP-Packet und Receive-IP-Packet.
  - Das OSI Modell wurde entworfen, bevor Implementierungen existierten.
    Folgen:
    - Saubere Strukturierung
    - Unrealistische Annahmen über existierende Netzwerk-Welt
  - Das TCP/IP Modell ist lediglich eine Beschreibung der Implementierung.
    Folge:
    - Das Modell kann nicht für andere Protokoll-Stacks verwendet werden.
  - Verbindungslose vs. verbindungsorientierte Kommunikation:
    - OSI bietet beide Varianten in der Netzwerk-Schicht, jedoch nur verbindungsorientierte Kommunikation in der Transport-Schicht.
    - TCP/IP hat lediglich verbindungslose Kommunikation in der Internet-Schicht, dafür aber beide Varianten in der Transport-Schicht.
      Folge: Applikationen können beide Varianten einsetzen.

Implementierung von Netzwerk-Software

Socket-Schnittstelle

In der TCP/IP Protokollwelt kommunizieren Applikationen miteinander über sog. Sockets (engl. Socket = Steckdose). Stark vereinfacht, besteht die Programmierschnittstelle (Socket-API) aus der folgenden fünf Funktionen.

- `connect`, Verbindungsaufbau aktiv (Client)
- `accept`, Verbindungsaufbau passiv (Server)
- `send`, senden
- `recv`, empfangen
- `close`, Verbindungsabbau

(Vergleiche dies mit den Service-Primitiven der OSI-Welt!)

Prozeßmodell

Abschließend bleibt noch die Frage zu klären, wie die Services der einzelnen Protokoll-Schichten auf die Software-Ausführungseinheiten (Prozesse bzw. Threads) abgebildet werden.

Die o.g. Socket-Operationen gehen von einem einzigen Prozeß (dem der Applikation) aus. Allerdings sind zumindest die folgenden Schichten miteinander in Einklang zu bringen (Beispiel):

- Applikation
- TCP
- IP
- physikalisches Netzwerk

Abb. PKA–13 zeigt zwei prinzipielle Möglichkeiten zur Abbildung der Schichten auf Prozesse auf.

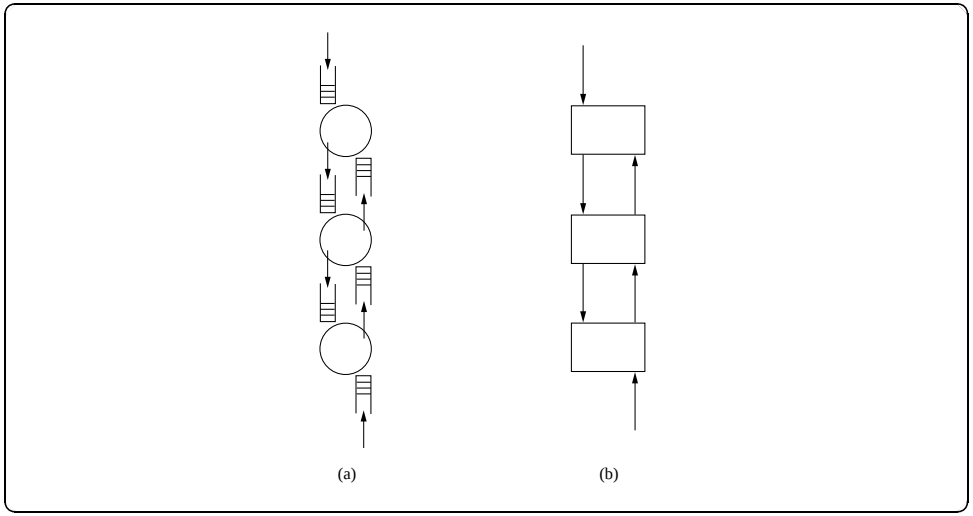


Abb. PKA–13 Alternative Prozeßmodelle
(a): Prozeß per Protokoll
(b): Prozeß per Nachricht

Beim Modell “Prozeß per Protokoll” wird jede beteiligte Schicht in einem eigenen Prozeß (oder Thread) abgearbeitet. Die Nachrichten werden in Message-Queues der jeweiligen Schicht zugestellt. Vorteil einer solchen Implementierung ist die gute Strukturierung: Ein Programm pro Schicht. Wesentlicher Nachteil ist der große Laufzeit-Overhead bedingt durch die Prozeßwechsel. (Auch häufige Thread-Wechsel können die Performance eines schnellen Netzwerks ruinieren.)

Alternativ kann auch das Modell “Prozeß per Nachricht” implementiert werden. Dies geschieht auch in der Praxis. Hier kommunizieren die Schichten untereinander durch Prozeduraufrufe, die wesentlich schneller abgearbeitet werden können als Prozeßwechsel. Beim Senden einer Nachricht ruft die sendende Prozedur der Applikation die Prozeduren der darunterliegenden Schichten auf, bis die Nachricht in der untersten Schicht angekommen ist. Beim Empfangen einer Nachricht wird diese auf der untersten Schicht dem Empfängerprozeß zugestellt; dort wird die Nachricht wiederum per Prozeduraufruf bis zur Applikationsschicht geleitet.

Ganz ohne Prozeßwechsel (Betriebssystem-Aufrufe) kommen sog. User-Level Protokolle aus. Bei diesen wird der Speicherbereich der Netzwerk-Adapter direkt in den Adreßraum der Applikation abgebildet. Somit kann die Applikation direkt über ihre eigenen Datenstrukturen (Variablen) kommunizieren; zeitaufwendige Interaktion mit dem Betriebssystem entfällt. Problematisch ist dabei der Schutz der Applikationen untereinander. Meist wird dies vom Betriebssystem bei der Initialisierung durch geeignete Zuteilung von Netzwerk-Puffern erreicht.

Upcalls

Auch mit dem Modell "Prozeß per Nachricht" lassen sich Prozeßwechsel nicht völlig vermeiden. Dies passiert wenn eine Applikation die `recv` Operation aufruft und zu diesem Zeitpunkt keine Nachricht vorhanden ist. In dieser Situation blockiert (wartet) der Prozeß und eine anderer Prozeß wird der CPU zugeteilt.

Dies ist auf der obersten Schicht (der Socket-API) unvermeidlich. Allerdings kann dieser Effekt zwischen allen anderen Schichten vermieden werden, wenn zum Empfangen ein sog. Upcall-Modell verwendet wird. (Siehe Abb. PKA–14) Hier wird die `recv` Operation durch eine `deliver` Operation ersetzt und die Aufrufrelation umgekehrt: Die untere Schicht ruft eine Prozedur der oberen Schicht auf. (daher der Name)

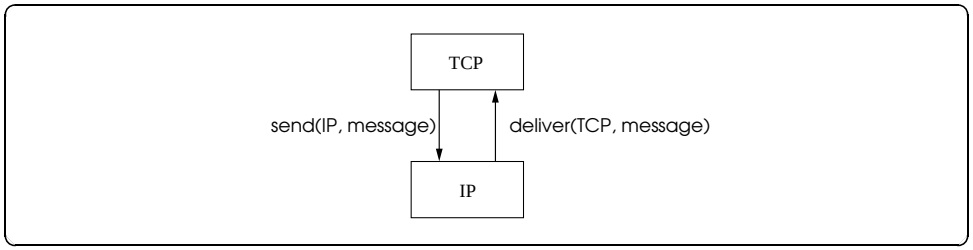


Abb. PKA–14 Protokoll zu Protokoll Interface

Nachrichtenpuffer

Applikationen senden und empfangen Nachrichten über ihre Datenstrukturen (Variablen). Beim Sender wird die Nachricht aus dem Applikationspuffer zumindest einmal in den Netzwerk-Adapter kopiert. Beim Empfangen ist zumindest eine Kopie vom Netzwerk-Adapter in den Applikationspuffer notwendig. Generell bedeutet das Kopieren von Nachrichten jedoch Laufzeitaufwand und sollte deshalb aus Effizienzgründen soweit möglich vermieden werden.

Problematisch sind dabei die Protokoll-Header, die die jeweiligen Schichten der eigentlichen Nachricht hinzufügen. (Vergleiche Abb. PKA–5) In traditionellen Implementierungen bedeutet das Hinzufügen bzw. Entfernen der Header jeweils eine weitere Kopie pro beteiligter Schicht. Effizienter sind sog. "scatter/gather" Interfaces (APIs). Hier werden die Nachrichten zwischen den Schichten nicht als zusammenhängende Speicherbereiche sondern als Vektoren von Startadressen und Längeninformation übergeben. Jede Schicht fügt nur eine weitere Startdresse (und Header-Länge) dem Vektor hinzu. Somit entfällt das Kopieren zwischen den Schichten. Beim Senden auf der untersten Ebene wird dann der Vektor abgearbeitet (gathering). Beim Empfangen wird die Nachricht auf verschiedene Speicherbereiche verteilt (scattering).

Problematisch sind trotz allem noch empfangene Nachrichten für die noch keine `recv` Operation aufgerufen wurde; hier ist die Adresse des Applikationspuffers noch unbekannt. In diesem Fall muß die empfangene Nachricht zunächst doch wiederum in einem Puffer einer Zwischenschicht gespeichert werden. Sowie dann `recv` aufgerufen wird, muß die Nachricht in den Applikationspuffer kopiert werden.

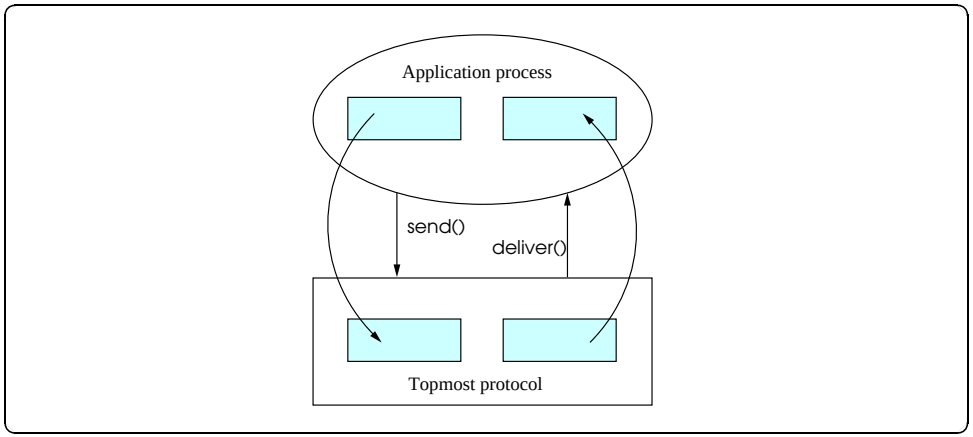


Abb. PKA–15 Kopieren von abgehenden/ankommenden Nachrichten zwischen Applikations-Puffer und Netzwerk-Puffer