

**Codetransparenz** bedeutet, dass es für eine Anwendung transparent bleibt, welche Mechanismen und Codes zur Übertragung verwendet werden. So wird z.B. durch Bitstopfen auf der Sicherungsschicht erreicht, dass die Anwendung die Existenz einer ausgezeichneten Bitfolge als Anfangs- und Endflag bei der Codierung ihrer Daten nicht berücksichtigen muss.

**Piggyback Acknowledgement** verwendet für das Senden einer Bestätigung keine separate Dateneinheit, sondern benutzt eine Dateneinheit mit, die selber Nutzdaten enthält.

### 3 Vermittlungsschicht (Network Layer)

#### 3.1 Vermittlungsarten

**Leitungsvermittlung** durchgehender, nichtspeichernder Übertragungskanal zwischen den Endsystemen; Bitfolgen reihenfolgetreu; Vermittlung in den Zwischensystemen erfordert keine zusätzliche Kontrollinformation zur Adressierung; starre Leitungszuordnung; nur verbindungsorientierter Dienst; Beispiel: ISDN

**Speichervermittlung** Zwischensysteme verfügen über Speicher; flexible Ressourcenzuordnung und Fehlerbehandlung; verbindungsloser Dienst möglich; komplex; Verzögerung, Reihenfolgevertauschung möglich;

**Nachrichtenvermittlung**

**Paketvermittlung** Vermittlung aufgrund von Vermittlungsinformation in den Dateneinheiten

**Datagramme**

**virtuelle Verbindung**

#### 3.2 Integrated Services Digital Network

**Kanäle.**

- 1 D-Kanal zur Signalisierung, 16 kbit/s
- 2 B-Kanäle zur Sprach-/Datenübertragung, 64 kbit/s

#### 3.3 Internet Protocol

**Dienst.** IP stellt einen unzuverlässigen und verbindungslosen Dienst zur Verfügung.

**Time To Live** ist ein Feld im Header einer IP-Dateneinheit. Dieser Wert wird vom Sender gesetzt und gibt die maximale Anzahl an Hops an, die das Paket durchlaufen darf. In jedem Router, den das Paket durchläuft, wird der Wert um 1 dekrementiert. Pakete mit einem TTL-Wert von 0 werden gelöscht und es wird eine ICMP-Nachricht an den Absender geschickt. Durch TTL wird die Wegewahl unterstützt und es werden umherirrende Pakete vermieden.

### 3.4 Segmentierung und Reassemblierung

**Motivation.** Existieren auf einem Pfad durch das Netzwerk unterschiedliche Maximum Transfer Units, so müssen eventuell Dateneinheiten beim Übergang segmentiert und später reassembliert werden. Die Reassemblierung sollte nicht schon in zwischenliegenden Routern erfolgen, da sich Routen während der Übertragungszeit einer fragmentierten Einheit ändern können oder einzelne Fragmente verlorengehen können.

### 3.5 Zustandsübergangsdiagramm

*Stimulus; Reaktion*

**Dienstprimitive beim Initiator.**

**TConReq** Anforderung eines Verbindungsaufbaus

**TConConf** Verbindungsaufbau erfolgreich

**TDisInd** Beantworter lehnt Verbindungsaufbau ab

**TPAboInd1** Abbruch durch Übertragungsabschnitt

**Dienstprimitive beim Beantworter.**

**TConRsp** Bestätigung des Verbindungsaufbauwunschs

**TDisReq** Ablehnung des Verbindungsaufbauwunschs

**TConInd** Mitteilung des Verbindungsaufbauwunschs

**TPAboInd2** Abbruch durch Übertragungsabschnitt

**Protokollinstanz.**

**Dienstspezifikation.** Die Dienstspezifikation beschreibt das an der Dienstschnittstelle von außen betrachtete Verhalten des Dienstes.

**Protokollspezifikation.** Die Protokollspezifikation legt durch Regeln und Formate das Kommunikationsverhalten zweier Protokollinstanzen fest.

### 3.6 Routing

**statisches Routing vs adaptives/dynamisches Routing.** Bei statischen Routing-Algorithmen ist die Routing-Tabelle für längere Zeit fixiert. Adaptives oder dynamisches Routing basiert dagegen darauf, dass die Routing-Tabellen laufend angepasst werden.

#### 3.6.1 Distanz-Vektor-Routing

**Prinzip.** Beim Distanz-Vektor-Routing kennt kein Knoten die komplette Route von der Quelle zur Senke. Der Knoten kennt nur die Distanz zu allen anderen Knoten.

**Distanz-Vektor-Tabelle.** Die Distanz-Vektor-Tabelle enthält eine Zeile für jedes mögliche Ziel und eine Spalte für jeden direkten Nachbarn (Next Hop). Knoten  $X$  möchte eine Nachricht über den direkten Nachbarn  $Z$  an  $Y$  verschicken.

$$X \rightarrow Z \rightarrow Y$$

Dann ist der Eintrag in der Distanz-Vektor-Tabelle von  $X$  gegeben durch die Summe aus Kantengewicht zwischen  $X$  und  $Z$  und dem minimalen Eintrag in der  $Y$ -Zeile der Tabelle von  $Z$ :

$$D^X(Y, Z) = c(X, Z) + \min_v \{D^Z(Y, v)\}$$

**Algorithmus.** Zunächst kennt jeder Knoten nur die Distanzen zu seinen direkten Nachbarn, d.h. die Einträge auf der Diagonalen der Tabelle. Diese schickt er an alle direkten Nachbarn, die ihre eigenen Distanzen entsprechend aktualisieren. Solange ein Knoten nach einer Aktualisierung neue kürzeste Wege kennt, schickt er sie an seine Nachbarn.

---

**Algorithm 1:** Distanz-Vektor-Algorithmus für Knoten  $s$

---

### 3.6.2 Link State Routing

**Prinzip.** Die Routing-Tabelle jedes Routers enthält die kürzesten Wege zu allen anderen Routern. Zur Berechnung der kürzesten Wege kann z.B. Dijkstras Algorithmus eingesetzt werden. Dadurch, dass jeder Knoten Informationen über seine adjazenten Kanten broadcastet, kann der einzelne Knoten einen Graphen des gesamten Netzes konstruieren und die kürzesten Wege berechnen.

---

**Algorithm 2:** Dijkstras Algorithmus

---

**Input:** Graph  $G = (V, E)$ , Startknoten  $s$ , symmetrische, positive  
Kantengewichtsfunktion  $c : E \rightarrow \mathbb{R}^+$

```

1   $N \leftarrow \{s\}$ 
2  for  $u \in V \setminus \{s\}$  do
3      if  $\{s, u\} \in E$  then
4           $d(u) \leftarrow c(s, u)$ 
5           $p(u) \leftarrow s$ 
6      else
7           $d(u) \leftarrow \infty$ 
8      while  $N \neq V$  do
9           $N \cup \{u\}, u \notin N \wedge d(u) = \min(d)$ 
10         for  $\{u, v\} \in E$  do
11             if  $d(v) + c(u, v) < d(v)$  then
12                  $d(v) \leftarrow d(v) + c(u, v)$ 
13                  $p(v) \leftarrow u$ 

```

---

**Kostenfunktion.** Die Kostenfunktion der Kanten kann von verschiedenen Faktoren abhängig gemacht werden, z.B.

- physikalisches Medium
- Verkehrsaufkommen
- politische Bestrebungen der Netzbetreiber
- Dienstgütemerkmale, z.B. garantierte Bandbreiten, Latenzzeiten oder Zuverlässigkeitsgarantien

### 3.7 IP-Adressierung

**Subnetzmaske** Die Subnetzmaske zerlegt eine IP-Adresse in Netzwerk-Teil und Endsystem-Teil. Sie ist eine Bitsequenz der Form  $1^m0^n$ . Den Netzwerkteil einer IP-Adresse erhält man durch und-Verknüpfung von IP-Adresse und Subnetzmaske in Binärdarstellung.

$$IP\text{Adresse} \wedge \text{Subnetzmaske} = \text{Netzwerkteil}$$

**Adressbereich.** Zur Darstellung eines Adressbereiches wird die Anzahl der 1er-Bits in der Subnetzmaske an die IP-Adresse angefügt. Eine IP-Adresse  $a$  liegt im Adressbereich  $b/m$ , wenn die ersten  $m$  Bits von  $a$  und  $b$  übereinstimmen.

### 3.8 Address Resolution Protocol

**Aufgabe.** Das Address Resolution Protocol ist für die Zuordnung

$$IP - \text{Adresse} \leftrightarrow MAC - \text{Adresse}$$

zuständig.

**Ablauf.**

1. suche im eigenen ARP-Cache; wenn Eintrag nicht gefunden:
2. sende ARP-Request mit gesuchter IP-Adresse, eigener IP-Adresse und eigener MAC-Adresse an alle (Broadcast)
3. Besitzer der gesuchten IP-Adresse sendet ARP-Reply mit seiner MAC-Adresse zurück
4. lege Wert im ARP-Cache ab

### 3.9 Domain Name System

**DNS-Anfrage.** Eine DNS-Anfrage zur Auflösung eines Domainnamen nach einer IP-Adresse stellt entweder

a) rekursive Anfragen:

- Der Client sucht im lokalen Cache.
- Der Client befragt den Cache des lokalen DNS-Servers.