

# Netzwerke

## Beobachtungen:

- Netzwerke überbrücken räumliche Distanz.
- Datenbanken überbrücken zeitliche Distanz.

## Ziele und Metriken:

- Hohe Zuverlässigkeit (Anzahl verlorener Nachrichten)
- Schnell (Zeit für eine Nachricht)
- Hohe Kapazität (Bandbreite)
- Niedrige Kosten (\$)

## Literatur:

- Tanenbaum: Computer Networks, Prentice Hall 1989.

# Netzwerke und Dienste

Man unterscheidet zwischen dem eigentlichen *Netzwerk* (Karten, Kabel, Satelliten, etc.) und *Diensten* (Software). Weiterhin unterscheidet man zwischen *Datagrammen* und *verbindungsorientierten Diensten*. Manche Netzwerke sind exklusiv für einen Dienst, manche Netzwerke sind für viele Dienste gedacht.

## Beispiele:

<i>Netzwerk</i>	Dienst	Verbindung (J/N)
Fernsprechnet	Telephon	Ja
	FAX	Nein
Gelbe Post	Briefe, Pakete	Nein
Kabelfernsehen	ARD, ZDF, ...	Nein
Autobahn	Reisen	Nein

Weitere Beispiele?

# Netzwerktopologien

## Was es alles gibt:

- Stern (z.B. Host mit Terminals)
- Ring (z.B. Token Ring, FDDI Ring)
- Bus (z.B. Ethernet; Internet Backbone)
- Baum (z.B. Mobilfunknetze)
- Vermaschtes Netz (z.B. das Gehirn – fast)

Topologie beeinträchtigt stark die Zuverlässigkeit, Schnelligkeit, Kapazität und Kosten eines Netzwerkes.

# Trends und Daten

## Stand 1990

	Entfernung	Latenzzeit	Bandbreite	1KB Nachricht
Cluster	100m	0,5 $\mu$ s	1Gbps	10 $\mu$ s
LAN	1km	5 $\mu$ s	10Mbps	1ms
MAN	100km	0,5ms	1Mbps	10ms
WAN	10000km	50ms	50Kbps	210ms

## Stand Heute

Probiert es selber mal mit *ping* aus.

## Prognose 2000

	Entfernung	Latenzzeit	Bandbreite	1KB Nachricht
Cluster	100m	0,5 $\mu$ s	1Gbps	5 $\mu$ s
LAN	1km	5 $\mu$ s	1Gbps	10 $\mu$ s
MAN	100km	0,5ms	100Mbps	0,6ms
WAN	10000km	50ms	100Kbps	50ms

## Fazit

Latenzzeit bestimmt die Kosten für eine Nachricht.

# Das ISO/OSI Modell

**ISO** International Standard Organization

**OSI** Open System Interconnection

## **Idee:**

- Architektur mit sieben Schichten
- Jede Schicht bietet Dienste für die nächste Schicht. Jede Schicht erweitert die Dienste der unteren Schicht. Jede Schicht verkapselt ihre genaue Vorgehensweise.
- Logisch gesehen kommuniziert ein Prozeß nur mit einem Prozeß auf derselben Schicht an einem anderen Rechner. Hierzu definiert die Schicht ein Protokoll.
- Physisch gesehen kommuniziert ein Prozeß nur mit dem entsprechenden Prozeß tiefer auf demselben Rechner.
- Ein Protokoll definiert Format und Bedeutung der Nachrichten innerhalb einer Schicht.

# Die Schichten des ISO/OSI Modells

## **1. Bit-Übertragungsschicht (physical layer)**

- Bit (Signal)-Sequenzen über ein physisches Medium
- Synchronisation
- Medium Access Control
- Beispielmedien: Twisted Pair, Baseband and broadband Coaxial Cable, Glasfaserkabel, Satelliten

## **2. Sicherungsschicht (data link layer)**

- Zusammenfassung der Daten zu Blöcken
- Blocksynchronisation
- Fehlererkennung und Korrektur (Check-Sum Verfahren oder Hamming-Codierung)
- Link Management für verbindungsorientierte Dienste

### **3. Vermittlungsschicht (network layer)**

- Routing (Wegewahl innerhalb eines Netzwerkes)
- Behandlung von Überlastsituationen innerhalb eines Netzwerkes
- Internetworking (Bridges und Gateways zum Übergang zwischen zwei Netzwerken und zur Anpassung von Differenzen in den Protokollen)
- Beispielprotokoll: IP (Internet)

### **4. Transportschicht (transport layer)**

- Aufbau und Erhaltung einer (virtuellen) Verbindung
- vollkommen netzwerkunabhängiger Transportmechanismus
- Adressierung
- Beispielprotokolle: TCP, UDP

### **5. Kommunikationsschicht (session layer)**

- Sitzungen (Sessions), die mehrere (Transport-) Verbindungen umfassen können.
- Beispiele: Remote Procedure Call, rlogin

## **6. Darstellungsschicht (presentation layer)**

- Kryptographie
- Komprimierung

## **7. Anwendungsschicht (application layer)**

- FTP
- Email
- http
- Netzwerkbetriebssysteme (z.B. NFS)

Verteilte Datenbanksysteme setzen oft direkt auf TCP/IP auf.

# Medium Access Control im Physical Layer

## **Motivation:**

- Ein physikalisches Medium verbindet viele unabhängige Stationen miteinander.
- Stationen können gleichzeitig Sendewünsche haben.
- Folge: Kollisionen.

## **Lösungsalternativen**

- Kollisionserkennung;  
Beispiel: CSMA/CD (Ethernet)
- Kollisionsvermeidung;  
Beispiel: Token Passing (Token Ring, Token Bus)

# CSMA/CD

(Carrier Sense Multiple Access/Collision Detection)

- Sendewillige Station hört das Medium ab, um festzustellen, ob die Leitung frei ist.
- Sendet gerade eine andere Station, so wie erwartet.
- Testen zwei sendewillige Stationen gleichzeitig, so kann es zu Kollisionen kommen.
- Kollisionen werden dadurch erkannt, daß die Nachrichten verstümmelt sind. (Gesendetes  $\neq$  Was der Sender auf dem Medium hört)
- Im Kollisionsfall brechen beide (alle) Sender ihre Übertragung ab, jeder wartet ein Random-Zeitintervall ab (jeder Sender würfelt für sich), hört ob das Medium frei ist, und sendet die Daten erneut.

# Bewertung von CSMA/CD

- einfach zu realisieren, kostengünstig
- bei wenigen Stationen sehr effizient  
(Kann jemand die Bandbreite eines CSMA/CD Netzwerks berechnen?)
- keine garantierten Antwortzeiten (nicht realzeitfähig)
- Netz sollte nicht voll ausgelastet werden

## **Beispiel Ethernet**

- Bus
- maximale Auslastung ca. 36%
- es gibt ein Limit für die Anzahl der Rechner
- bei mehr Rechnern braucht man mehrere separate Ethernets mit “Bridges,” die Nachrichten zwischen Rechner unterschiedlicher Netze weiterleiten
- 10 Mbps

## **Beispiel Fast Ethernet**

- Punkt zu Punkt Topologie
- 100 Mbps

# Token Passing Verfahren

## Prinzip

- Netzzugang wird durch ein zirkulierendes *Token* realisiert.
- Nur wer das Token hat, darf senden.
- Zur Fairneß dürfen einzelne Knoten, das Token nur für eine bestimmte Zeit halten.
- Wer senden möchte und das Token hat, bringt seine Nachricht in Umlauf und behält das Token.
- Nachricht wird wie das Token weitergereicht. Empfänger kopiert die Nachricht, wenn er sie kriegt, reicht sie aber auch an den nächsten weiter.
- Wenn Sender seine eigene Nachricht erhält, nimmt er sie vom Netz und gibt das Token weiter.

## Details

- Token Recovery, falls Token verloren geht (Timeout Verfahren)
- Early Token Release, um volle Bandbreite des Netzes auszunutzen

## **Bewertung:**

- fairer Zugang zum Netz; Prioritisierung auch möglich
- garantierter Zugang (realzeitfähig)
- technisch aufwendig und dadurch teuer

## **Beispiele:**

- Token Ring (Überbrücke ausgefallen Sites)
- Token Bus (Aufprägung der logischen Ringstruktur)
- FDDI (doppelter Ring für Normalverkehr und Problemfälle)

# Routing in der Netzwerkschicht

- In den meisten Netzwerktopologien sind verschiedene Wege vom Sender zum Empfänger denkbar
- Knoten führen Wegewahltabellen (routing tables)
- Aufbau Aktualisierung dieser Tabellen
  - statisch:** Vorabberechnung mit Second-Choice Einträgen für Störungen  
reagiert nicht auf aktuelle Netzlast
  - adaptiv:** dynamische Anpassung an Netzlast  
Vorsicht bei verbindungsorientierten Diensten ist die reihenfolgetreue ist nicht gewährleistet.
- in jedem Fall kann z.B. Dijkstra Shortest Path Algorithmus verwendet werden
- Weiterer Freiheitsgrad: direktes vs. indirektes Routing

# Sichere Kommunikation in Transportschicht

## Motivation

- Nachrichten können verloren gehen

## Naiver Ansatz

- Empfänger bestätigt jede einzelne Nachricht.
- Sender schickt nächste Nachricht erst nach Erhalt der Quittung; schickt die Nachricht erneut nach einem Timeout.

## Sliding Window Technik

- Sender vergibt Nachrichten eine Sequenznummer
- Sender schickt mehrere Nachrichten und hält ein Fenster von Nachrichten, die noch nicht bestätigt wurden
- Mit  $x$  bestätigt Empfänger alle Nachrichten mit Sequenznummer  $\leq x$ , so daß Sender sein Fenster aktualisieren kann.
- bei fehlender Bestätigung nach einem Timeout schickt Sender die Nachrichten erneut wie beim naiven Ansatz
- ggf. kann Empfänger gezielt Nachrichten neu anfordern.

# Sicherer Verbindungsaufbau in Transportschicht

## Motivation

- Phantomnachrichten und alte, verzögerte Nachrichten schwirren im Netzwerk herum
- $A$  schickt  $B$  ein Aufbaurequest mit einer Sequenznummer  $x$
- $B$  beantwortet den Aufbaurequest für Verbindung  $x$  und vergibt eine Bestätigungsnummer  $y$
- $A$  bestätigt seinerseits den Aufbau für Verbindung  $x$  mit Bestätigung  $y$
- $A$  schickt Daten an  $B$  im Rahmen dieser Verbindung stets unter Angabe von  $x$  und  $y$

Ein ähnliches Verfahren funktioniert auch zum Verbindungsabbau.

# TCP/IP

## **IP — Internet Protocol für Netzwerkschicht:**

- IP Adressen:
  - 32 bit lang
  - bestehen aus *net id* und *host id*
  - werden zentral vergeben
- Gateways/Router nutzen *net id* Teil einer IP Adresse aus  
Nachteil: mobile Stationen können so nicht unterstützt werden
- Datenformat:
  - Nachrichten bestehen aus Header und den eigentlichen Daten
  - Der Header enthält die IP Adressen des Senders und des Empfängers
  - Header und Daten werden dann noch einmal in einen Frame mit einem Frame Header gepackt.
- Spezielle Features:
  - broadcast innerhalb eines Netzwerks
  - multicast auch Netzwerkübergreifend.

## **TCP — Transmission Control Protocol für Transportschicht:**

- baut Verbindungen auf (3-way Handshake Verfahren)
- realisiert einen Stream von Nachrichten zwischen Sender und Empfänger
- gewährleistet reihenfolgetreue, und sichert zu, daß alle Nachrichten ankommen und keine dupliziert werden. (Vergabe von Sequencenummern für Nachrichten ähnlich wie beim Sliding Window Verfahren.)
- Bestätigungsnachrichten werden so gut wie möglich Huckepack genommen, um Aufwand zu minimieren.
- Verbindungsaufbau teuer: deswegen sollten Anwendungen wie VDBMS, die auf TCP/IP beruhen, Verbindungen zwischen einzelnen Rechner cachen.

## **Alternative UDP — User Datagram Protocol**

- baut keine Verbindung auf (spart diese Kosten)
- garantiert aber auch keine reihenfolgetreue und Verlustfreiheit von Nachrichten
- erweitert IP, damit Nachrichten am Zielrechner zu unterschiedlichen Prozessen geleitet werden kann: Angabe des Ports für jede Nachricht zusätzlich zur IP Adresse.

# ATM — Asynchronous Transfer Mode

- ganz was anderes und cool  
(hat nichts mit IP oder TCP oder so zu tun)
- alle Nachrichten werden 53 byte Pakete (sogenannte ATM Zellen) von ATM-Adapttern zerhackt
- 53 byte = 5 byte Header plus 48 byte Daten
- Grund für fixe Paketgröße:
  - kalkulierbare Verzögerungen
  - 53 byte sind Kompromis für kleine Nachrichten alla Kontrollnachrichten und große Nachrichten alla Multimedia
- virtuelle Verbindungen:
  - festgelegt, auf welchem Weg ATM Zellen vom Sender zum Empfänger geschickt werden
  - Quality of Service Parameter (Übertragungsrate etc.)
- unterscheide zwischen *permanent virtual circuits (PVCs)* (i.e., Standleitungen) und *switched virtual connections (SVCs)* (i.e., Ad-hoc Verbindungen)
- Zellen enthalten keine Adresse sondern virtual channel identifier und virtual path identifier
- Punkt zu Punkt Verbindungen und Multicast möglich