6 Feldbusprotokolle

6.1 Signalübertragung

- Charakteristika:
  - ⋆ maximale Übertragungsrate in Mbps
  - ⋆ Dämpfung in db pro km
  - ⋆ überbrückbare Entfernung
  - ⋆ Materialeigenschaften (z.B. für Installation)
  - ⋆ Störungsempfindlichkeit
  - ⋆ Kosten, Marktproduktpalette

6.1.1 Übertragungsmedien

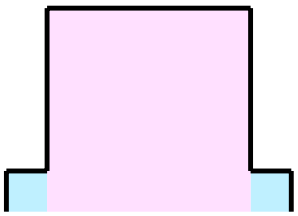
- Einfache (symmetrische) Kupferkabel
  - ⋆ zwei– oder vieradrig
  - ⋆ Gefahr des Übersprechens
    (Abhilfe evtl. Abschirmung)

- Verdrillte Kabel (twisted pair)
  - ⋆ zwei verdrillte Doppeladern mit 20 bis 30 Umschlingungen pro m
  - ⋆ gleichen viele induktiven Störungen aus
  - ⋆ mit Abschirmung STP (shielded TP, teuer), sonst UTP (unshielded TP)
  - ⋆ Übertragungsraten von kbps im km–Bereich bis zu 100 Mbps im Meterbereich
  - ⋆ Anwendung: strukturierte LAN-Verkabelung
- Koaxialkabel
  - ⋆ Konzentrischer Signalleiter im Kabelinnern, dann Dielektrikum, Abschirmung, Isolationsmaterial
  - ⋆ wenig Abstrahlung
  - ⋆ wenig Störungsanfällig
  - ⋆ Dämpfung nimmt mit steigender Frequenz rasch zu
  - ⋆ Anwendung: Ethernet LAN's (50 Ω)
    - ◦ nur ein gemeinsamer Kanal
    - ◦ 10 Mbps auf 500 m
    - ◦ größere Entfernungen durch Verstärkung (Repeater)

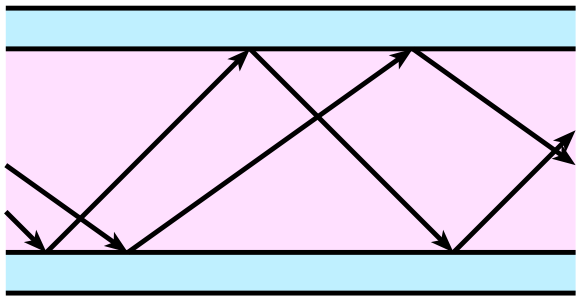
    - Stationsanschluß durch "Transceiver" (Tap– oder Schraubanschluß)
  - ⋆ Breitbandübertragung (75 Ω)
    - mehrere Frequenzbänder (Kanäle)
    - z.B. MAP im Bereich um 192 MHz mit 3 Kanälen $\grave{a}$ 12 MHz Breite

- Lichtwellenleiter
  - ⋆ Übertragung in Glasfasern (Kern mit 10 bis 100 μm Dicke und Glasmantel)
  - ⋆ noch selten Plastikfasern
  - ⋆ monochromatisches Licht
  - ⋆ unempfindlich gegen EMS und Übersprechen
  - ⋆ schwer abhörbar
  - ⋆ geringe Dämpfung (min. 0.2 db/km) zum Vergleich: Fensterglas 50000 dB/km - Stadtluft 10 dB/km
  - ⋆ Geringe Dispersion
  - ⋆ geringes Gewicht
  - ⋆ übliche Wellenlängen im Bereich minimaler Dämpfung, z.B. 850 nm, 1300 nm und 1550 nm

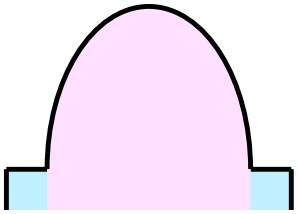
- ⋆ Arten
  - ○ Multimode mit Stufenprofil
    - ▷ Glaskern mit höherem Brechungsindex wie Glasmantel
    - ▷ Lichtführung: Totalreflexion



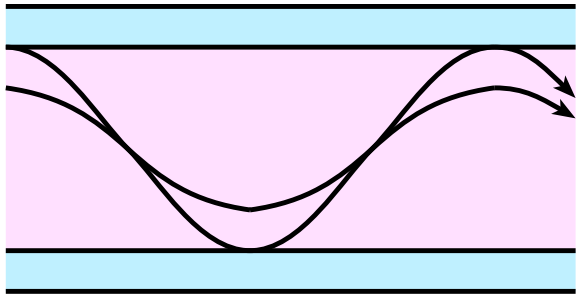
Brechzahl-
profil



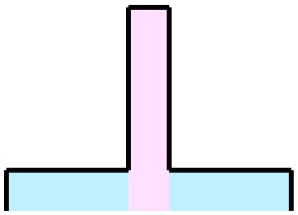
  - ○ Gradientenfaser
    - ▷ Glaskern mit kontinuierlich sich ändernden Brechungsindex
    - ▷ Lichtführung: Brechung



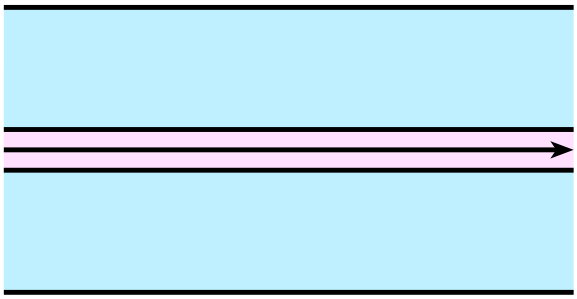
Brechzahl-
profil



- Monomode-Fasern
  - LAN und Post
  - kleinster Glaskern mit höherem Brechungsindex wie Glasmantel
  - Lichtführung: Wellenführung



Brechzahl-
profil



⋆ Charakteristika

	Stufenprofil
Lichtquelle	LED oder Laser
Bandbreite	$\leq$200 MHz/km
Kosten	billig
Datenrate	$\approx$ 100 Mbps
Dämpfung	0.2–50 db/km
Dispersion	50 nsec/km
Kern$\emptyset$	50–200μm
	Gradientenprofil
Lichtquelle	LED oder Laser
Bandbreite	200 MHz/km – 3 GHz/km
Kosten	mittel
Datenrate	$\approx$ 300 Mbps und $\leq$ 20 km
Dämpfung	0.2–15 db/km
Dispersion	1 nsec/km
Kern$\emptyset$	50–100μm
	Monomode
Lichtquelle	Laser
Bandbreite	3GHz/km – 50 GHz/km
Kosten	höher, aber Massenmarkt
Datenrate	$\leq$ 2 Gbps und $\leq$ 80 km
Dämpfung	0.2–2db/km
Dispersion	0.3 nsec/km
Kern$\emptyset$	2–10 μm

6.1.2 Elektromagnetische Störungen (EMS)

- Analogsignale von Sensoren oft im mV–Bereich
- Störspannungen durch
  - ⋆ thermisches Rauschen
  - ⋆ kapazitive und/oder induktive Einkopplungen (durch Nachbarkanal, Schaltvorgänge, Hochfrequenz)
  - ⋆ schlechte Erdung
  - ⋆ schlechte Kontakte (erzeugen mV Spannungsabfall)
- Abhilfen durch
  - ⋆ kurze Kabelwege (Bilden von Unterstationen mit hochwertigem Kabel bzw. Feldbus zum Rechner)
  - ⋆ Potentialtrennung (Optokoppler, transformatorisch)
  - ⋆ Schutz vor Immissionen (Wärme, Kälte, Feuchtigkeit, Staub)

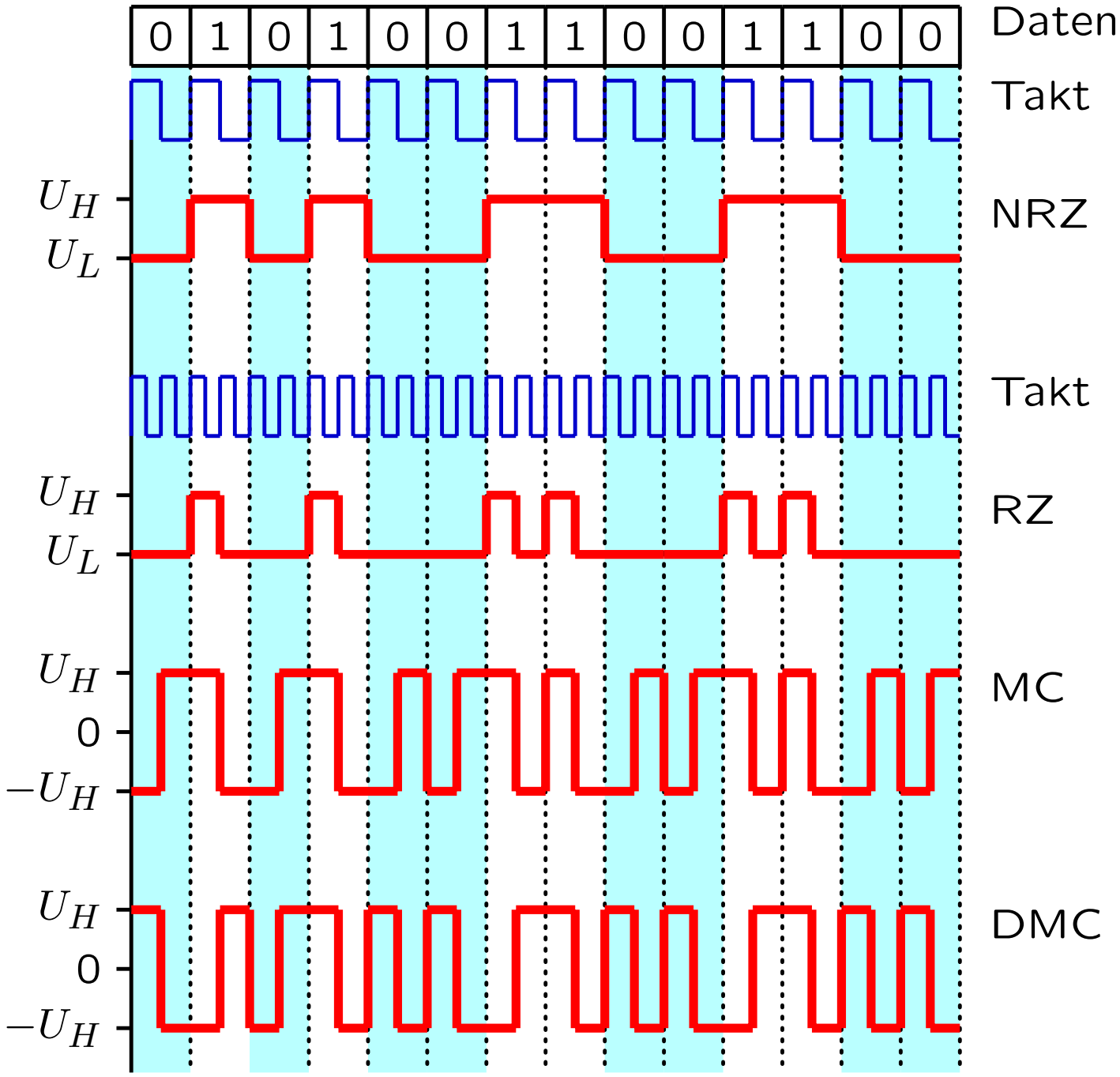
6.1.3 Signalcodierungen

- Ziele
  - ⋆ eindeutige Darstellung der Zustände "0" und "1" und deren Folgen durch elektrische Signale
  - ⋆ Störungsunempfindlich
  - ⋆ Möglichkeit der Erkennung von Fehlern
  - ⋆ möglichst geringer Gleichstromanteil
  - ⋆ bei synchroner Übertragung keine eigene Taktleitung, sondern Ableitung des Sendetaktes aus den empfangenen Signalen (Taktrückgewinnung, selbsttaktend)
- Grundverfahren
  - ⋆ Basisbandübertragung
  - ⋆ modulierte Übertragung mit Trägerfrequenz
- Basisbandübertragung
  - ⋆ Die gesamte Bandbreite des Wegs wird für die Übertragung eines Signals benutzt
  - ⋆ Es wird eine Impulsfolge ohne Modulation mit einer Trägerfrequenz übertragen
  - ⋆ Problem: Impulsverbreiterung durch frequenzabhängige Ausbreitungsgeschwindigkeit und Dämpfung

- modulierte Übertragung mit Trägerfrequenz
  - ⋆ zur Übertragung wird das Signals mit einer Trägerfraquenz moduliert
  - ⋆ gesendet wird $U(t) = A * cos(\omega t + \varphi)$
  - ⋆ Trägerfrequenz ω
  - ⋆ drei Varianten der Modulation
    - ◦ Amplitudenmodulation
      (amplitude shift keying ASK)
      z.B. A = 0 (1)
      bedeutet Binärzeichen 0 (1)
    - ◦ Frequenzmodulation
      (frequency shift keying FSK)
      z.B. ω = 1180 Hz (980 Hz) bedeutet Binärzeichen 0 (1)
    - ◦ Phasenmodulation
      (phase shift keying PSK)
      z.B. keine Änderung von φ bedeutet Binärzeichen 0, ein Phasensprung um 180 Grad bedeutet 1
  - ⋆ mehrere Kanäle, deren Frequenzbänder sich nicht überlappen, können gleichzeitig über ein Medium übertragen werden

- Impulsdarstellung eines Signals mit Takt $\triangle$ t:
  - ⋆ einstufige Übertragungsverfahren
    Codeelemente 0 und 1
    - ◦ unipolar: 0 (1) durch 0 Volt (+U Volt)
    - ◦ polar: 0 (1) durch −U Volt (+U Volt)
    - ◦ bipolar: 1 durch abwechselnd +U, −U Volt
  - ⋆ Mehrstufige Übertragungsverfahren
    Codeelemente haben n >2 Kennzustände
    (n=3 ternär, n=4 quaternär)
  - ⋆ Baudrate = Anzahl Signale (Takte) je sec
  - ⋆ Bitrate = Anzahl übertragene Bits je sec
  - ⋆ bei einstufigen Verfahren:
    Bitrate = Baudrate
- Übliche Codierungen
  - ⋆ NRZ (non return to zero)
    - ◦ 1: U_H in der Taktlänge
    - ◦ 0: U_L in der Taktlänge
    - ◦ Problem: Taktrückgewinnung bei langen Folgen gleicher Zeichen

- ⋆ RZ (return to zero)
  - ◦ 1: U_H in erster Takthälfte, dann U_L
  - ◦ 0: U_L in der Taktlänge
  - ◦ Problem: Taktrückgewinnung bei langen Folgen gleicher Zeichen
- ⋆ Manchester Code MC
  - ◦ Biphase–Code (mind. 1 Übergang je Takt)
  - ◦ 1: Übergang in Taktmitte von U_H nach U_L
  - ◦ 0: Übergang in Taktmitte von U_L nach U_H
  - ◦ Vorteile: kein Gleichstromanteil, selbsttaktend
  - ◦ Nachteil: doppelte Taktfrequenz
- ⋆ Differential Manchester Code DMC
  - ◦ immer Übergang in der Taktmitte für Taktrückgewinnung
  - ◦ 0: Spannungswechsel am Taktanfang
  - ◦ 1: kein Spannungswechsel am Taktanfang
  - ◦ J–Codeverletzung: Kein Übergang am Anfang und kein Übergang in Mitte
  - ◦ K–Codeverletzung: Übergang am Anfang, kein Übergang in Mitte


0 1 0 1 0 0 1 1 0 0 1 1 0 0
Daten
Takt
U_H
U_L
NRZ
Takt
U_H
U_L
RZ
U_H
0
$-U_H$
MC
U_H
0
$-U_H$
DMC

6.1.4 Nachrichtenrahmen

- Nachrichten werden in Rahmen (frames) eingefügt
- neben der Nutzinformation (Daten) gibt es notwendige Verwaltungsinformation, wie
  - ⋆ Absenderadresse
  - ⋆ Empfängeradresse
  - ⋆ Priorität
  - ⋆ Länge Daten
  - ⋆ Sequenznummer
  - ⋆ Quittungen
  - ⋆ Fehlersicherungscode (frame check sequence, FCS)
  - ⋆ Anfangskennung einer Nachricht
  - ⋆ Endekennung einer Nachricht

Vorspann (Kopf)	Daten	Nachspann (Schluß)

Anf.-kennung	Verw.-Inf.	Daten	FCS	Ende-kennung

- asynchrone und synchrone Nachrichtenübertragung unterschieden:
  - ⋆ Asynchroner Start–/Stop–Betrieb
    - ◦ jedes Zeichen (5 bis 7 Bit) synchronisiert sich mit einem vorangestellten Startbit (negative Flanke) und einem nachgestellten Stopbit
    - ◦ Sender und Empfänger benötigen wegen des kurzen Gleichlaufs nur geringe Taktübereinstimmung
  - ⋆ Synchrone Übertragung
    - ◦ die Zeichen einer Nachricht werden unmittelbar nacheinander übertragen
    - ◦ erfordert hohe Zeitstabilität beider Taktgeber bzw. Taktrückgewinnung
    - ◦ ggf. vor Rahmen noch spezielle Steuerzeichen zur Synchronisation (SYN im ISO–7–Bit–Code)

- Code–transparente Datenübertragung
  - ⋆ Ziel:
    Übertragung beliebiger Zeichen (Binärdaten)
  - ⋆ Problem:
    Anfangs– oder Endekennung dürfen in den Daten nicht auftreten
  - ⋆ Lösung bei zeichenorientierter Übertragung
    - ◦ Aussehen Nachricht:

STX	Daten	ETX

      mit STX Start of Text
      ETX End of Text
    - ◦ Darstellung von STX und ETX im Text durch vorausgestelltes Fluchtsymbol DEL (byte–stuffing)
    - ◦ Fluchtsymbol DEL im Text wird DEL DEL
  - ⋆ Lösung bei bitorientierter Übertragung
    - ◦ Aussehen Nachricht (HDLC–Protokoll):

01111110	Daten	01111110

    - ◦ in Daten wird nach 5 aufeinanderfolgenden 1 eine 0 eingeschoben und beim Empfänger wieder entfernt (bit–stuffing)

6.1.5 Erkennen von Übertragungsfehlern

- Einzelfehler, insbesondere aber Bündelfehler
- Beispiel: Bei 4800 bps stört EMS von 10 msec 48 Bits
- Fehlerraten, z.B. Telefon 10^{-5}, LAN 10^{-9}, LWL 10^{-12}
- Methoden zur Fehlerkennung
  - ⋆ redundante Übertragung mit Vergleich der Nachrichten auf Übereinstimmung beim Empfänger
  - ⋆ Sicherungscode
- Sicherungscodes
  - ⋆ Querparität (vertical redundancy check VRC)
    - ◦ Paritätsbit pro 7-Bit-Zeichen oder Wort
    - ◦ Bitfehler ungerader Anzahl erkennbar (quer)

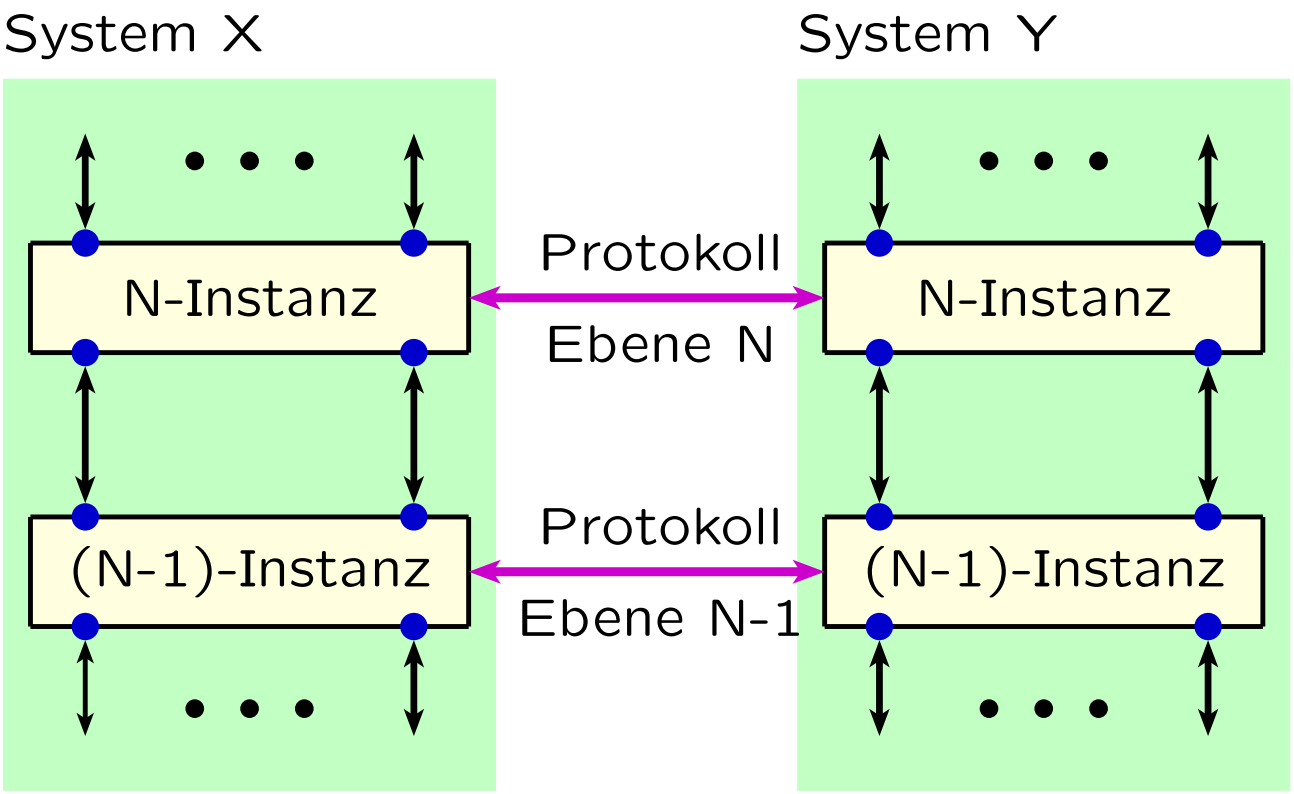
- ⋆ Längsparität (longitudinal redundancy check LRC)
  - ◦ EXOR über alle Bytes bzw. Worte
  - ◦ Bitfehler ungerader Anzahl erkennbar (längs)
- ⋆ Kreuzsicherung
  - ◦ Längs- und Querparität
- ⋆ Zyklischer Sicherungscode (cyclic redundancy check CRC)
  - ◦ Bits einer Nachricht als Koeffizienten eines Polynoms betrachtet
  - ◦ Division durch geeignetes Generatorpolynom $G_r(x)$ des Grads r
  - ◦ Rest wird als Sicherungscode (Länge: r Bit) übertragen
  - ◦ Bündelfehler bis zur Länge r erkennbar
  - ◦ Bündelfehler der Länge größer r mit Wahrscheinlichkeit 2^{-r} nicht erkennbar
  - ◦ heute übliches Verfahren; mit $G_{16}(x) = x^{16} + x^{12} + x^5 + 1$ (bei HDLC)

- Fehlerbehandlung
  - ⋆ Nachrichten sind numeriert
  - ⋆ Quittieren der Nachricht (acknowledge ACK)
  - ⋆ manchmal auch negative Quittung, falls Fehler erkannt
  - ⋆ Ausbleiben der positiven Quittung (Zeitüberschreitung) bedeutet Fehler

    (hervorgerufen auch durch Verlust der Nachricht im Netz oder durch Nichterkennung von Nachrichten wegen Fehlern in Blockkennzeichen)
  - ⋆ erneutes Senden der Nachricht
  - ⋆ damit aber Problem der Erkennung von Nachrichtenduplikaten
  - ⋆ Duplikate anhand Sequenznummer erkennbar (sliding window protocol)

6.2 Protokolle

- Protokolle sind Regeln für den Datenaustausch zwischen Partnern auf derselben Ebene (peer-to-peer)



(Peer-to-Peer) Protokoll

Dienstezugangspunkt (service access point)

Diensteprotokoll, Diensteschnittstelle

- Rechnernetz–Protokolle durch ISO/OSI–Referenzmodell beschrieben
- international genormt
- Schichten (layers)

7 Application	Anwendung
6 Presentation	Darstellung
5 Session	Sitzung
4 Transport	Transport
3 Network	Vermittlung in Netzen
2 Data–Link	Sicherung (Punkt-zu-Punkt-Verbindungen, (Mehrpunktverbindungen)
1 Physical	Physikaklische Verbindung

- Feldbusprotokolle sind Spezialversionen für die Geräte der Feldebene. Sie füllen wegen der geringeren Anforderungen und aus Kostengründen meist nur die Schichten 1,2 und 7 aus.

6.3 Aufgabe und Anforderungen

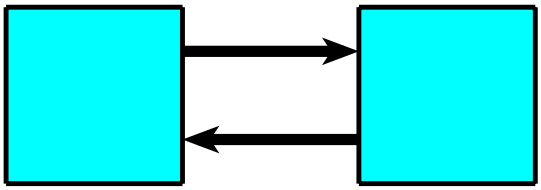
- Feldbus ist Verbindung zwischen Prozeßrechner und den (oft einfachen) Geräten (auf Feldebene)
- viele Geräte am Bus
- Echtzeitbedingungen
  - ⋆ deterministisches Zugriffsverfahren, d.h. projektierbares Zeitverhalten
  - ⋆ Zykluszeiten 1 bis 10 msec bei 40 bis 60 Geräten
  - ⋆ effizientes Protokoll auch bei kurzen Nutzdatensätzen, d.h. Rahmenorganisation kurz
  - ⋆ Prioritäten der Nachrichten (für Alarme)
  - ⋆ Optimierung von zyklischen Übertragungen, da die Regelungstechnik häufig ein mögliches äquidistantes Zeitraster fordert
- weitere Anforderungen
  - ⋆ zuverlässig in rauher Umgebung
  - ⋆ Übertragungsrate muß nicht unbedingt sehr hoch sein (serieller Bus reicht)

  - ⋆ geringe Kosten für Bus
  - ⋆ das Protokoll soll einfach und stabil sein
  - ⋆ einfache, preiswert zu realisierende Schnittstellen zum Bus, da Teil einfacher und preisgünstiger Sensorik
- Vielzahl von Feldbus–Entwicklungen
  - ⋆ Manufacturing Automation Protocol (MAP), USA
  - ⋆ Factory Instrumentation Protocol (FIP), Frankreich
  - ⋆ Process Field Bus (PROFIBUS), BMFT
  - ⋆ PROFIBUS–DP (dezentrale Peripherie)
  - ⋆ BITBUS, Intel
  - ⋆ INTERBUS–S, Phönix Contact
  - ⋆ Controller Area Network (CAN), Bosch
  - ⋆ Serielles Echtzeit Kommunikationssystem (SERCOS), Bosch
  - ⋆ Local Operating Network (LON), Echelon
  - ⋆ Aktor-Sensor-Interface (ASI), VDMA
  - ⋆ European Installation Bus (EIB)

- schon ab 1982 von General Motors das inzwischen genormte Protokoll MAP (manufacturing automation protocol) entwickelt, das alle 7 Schichten des ISO/OSI–Modells umfaßt. Es zeigte sich, daß MAP für die Belange der Feldebene oft zu umfangreich, zu wenig effizient und daher zu teuer ist
- daher bei Feldbusprotokollen oft nur Schicht 1, 2 und 7, restliche Schichten rudimentär

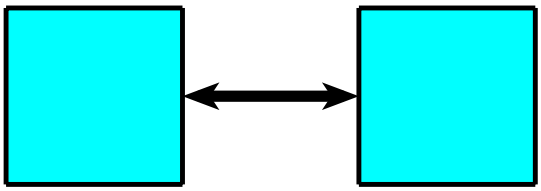
6.4 Verbindungsstrukturen

- Punkt-zu-Punkt-Verbindung, vollduplex



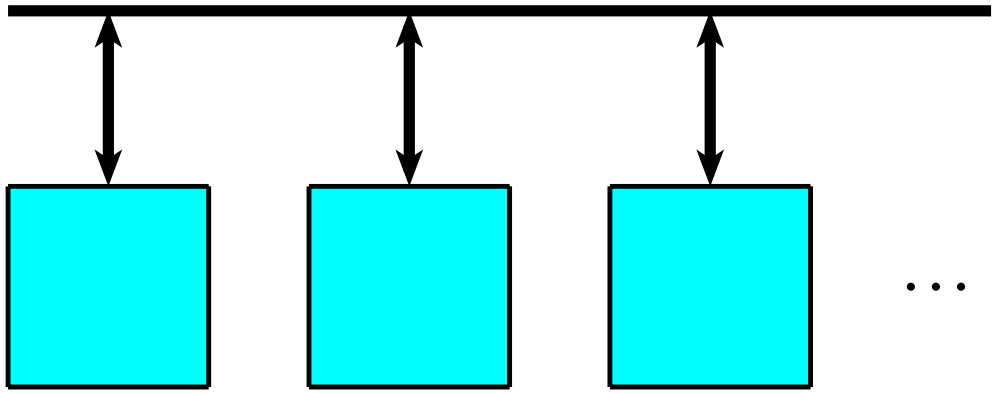
  - ⋆ keine Konflikte

- Punkt-zu-Punkt-Verbindung, halbduplex



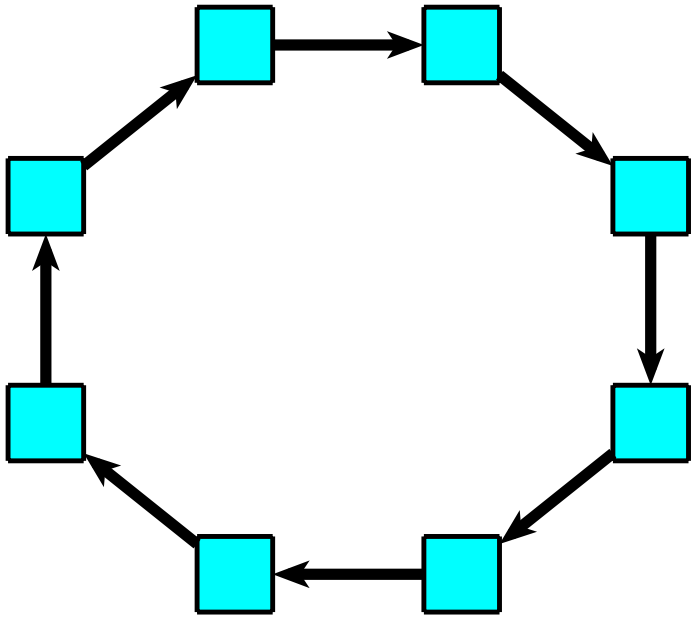
  - ⋆ Konflikt bei gleichzeitigem Schreiben
  - ⋆ gleichzeitige Benutzung des Mediums muß durch eine Zugriffsstrategie (Arbitrierung) vermieden werden

- Bus–Struktur



  - ⋆ Arbitrierung notwendig
  - ⋆ Beispiele: Aloha, CSMA/CD (Ethernet), versch. Reservierungsprotokolle, CAN-Bus
  - ⋆ Ethernet nicht echtzeitfähig, da maximale Wartezeit beliebig hoch im belasteten Fall

- Ringstruktur



  - ⋆ Teilnehmer an geschlossenen Übertragungsring angekoppelt
  - ⋆ Daten durchlaufen den Ring und werden vom Sender wieder entfernt
  - ⋆ Arbitrierung, meist durch Tokenweitergabe, nötig
  - ⋆ Beispiele: Token–Ring, FDDI
  - ⋆ Token–Ring echtzeitfähig, da maximale Wartezeit determiniert

6.5 Buszugang

- Frequenzmultiplex (eigentlich keine Arbitrierung)
  - ⋆ ein Teilnehmer je Kanal
  - ⋆ mehrere Teilnehmer je Kanal
- Zeitmultiplex
  - ⋆ zentrale Arbitrierung
    - ◦ Bus–Arbiter (zentraler Voter)
    - ◦ Daisy-Chain
    - ◦ Master–Slave
  - ⋆ dezentrale Arbitrierung
    - ◦ Token–passing–Verfahren
    - ◦ Speziallösungen, z.B. beim CAN-Bus
  - ⋆ zufälliger Buszugriff
    - ◦ ohne Reservierung (Aloha, CSMA/CA, CSMA/CD)
    - ◦ mit Reservierung (Multi Level Multi Access, Reservation Aloha)

6.6 CAN–Bus

- CAN = Contoller Area Network
- Ziele
  - ⋆ zuverlässiger Bus zur Vernetzung von Sensoren und Aktoren;
  - ⋆ Haupteinsatzgebiet zunächst Kfz
- von Bosch entwickelt
- Lizenzen an INTEL, PHILIPS, MOTOROLA, SIEMENS u.a.
- Norm ISO 11519 und 11898 (Schicht 1 und 2)
- Nutzerorganisation: CAN in Automation CiA jetzt weltweiter Einsatz in Automatisierung

6.6.1 Schicht 1

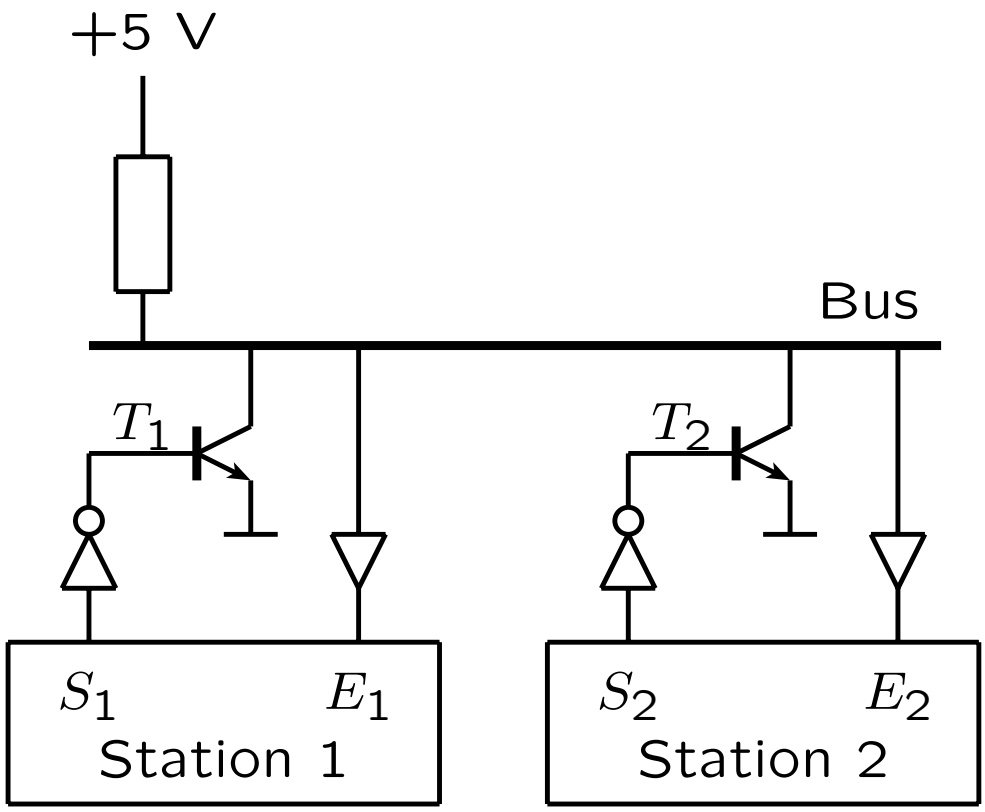
- High–speed:
  125 kbps bis 1 Mbps (max. 40 m)
- Low–speed:
  5 kbps (10 km) bis 125 kbps (530 m)

- Übertragungsmedium nicht festgelegt; meist 2–Drahtleitung (Spannungsdifferenz RS-485), empfohlen ISO/IS 11898
- NRZ–Code, deshalb Bitstuffing
  (nach 5 gleichen Bits wird ein komplementäres Bit eingefügt und auf der Empfangsseite wieder entfernt)
- CSMA/CA– (carrier sense multi access / collision avoidance) Verfahren wird dominantes D–Bit und ein rezessives R–Bit durch entsprechende Buspegel definiert

  D–Bit: logisch "0"
  R–Bit: logisch "1"

- D–Bit überwiegt R–Bit, falls zwei Geräte gleichzeitig senden
- Ein R–Bit–Sender kann somit das gleichzeitige Senden eines D–Bits am Buspegel erkennen
- eine Station i vergleicht empfangenes Bit E_i mit gesendetem Bit S_i und hört auf zu senden, falls $E_i \neq S_i$.

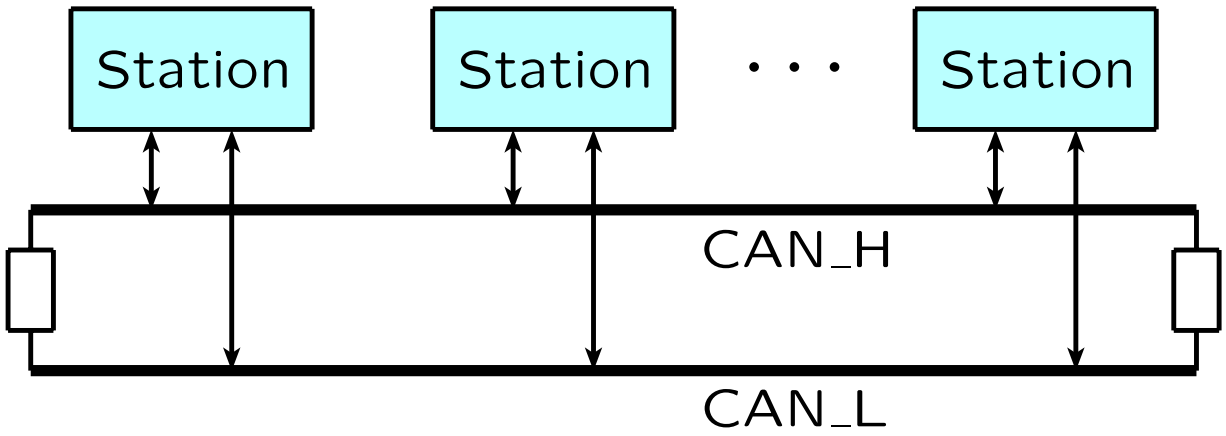
- Realisierung von D– und R–Bit z.B. durch Wired–And–Schaltung



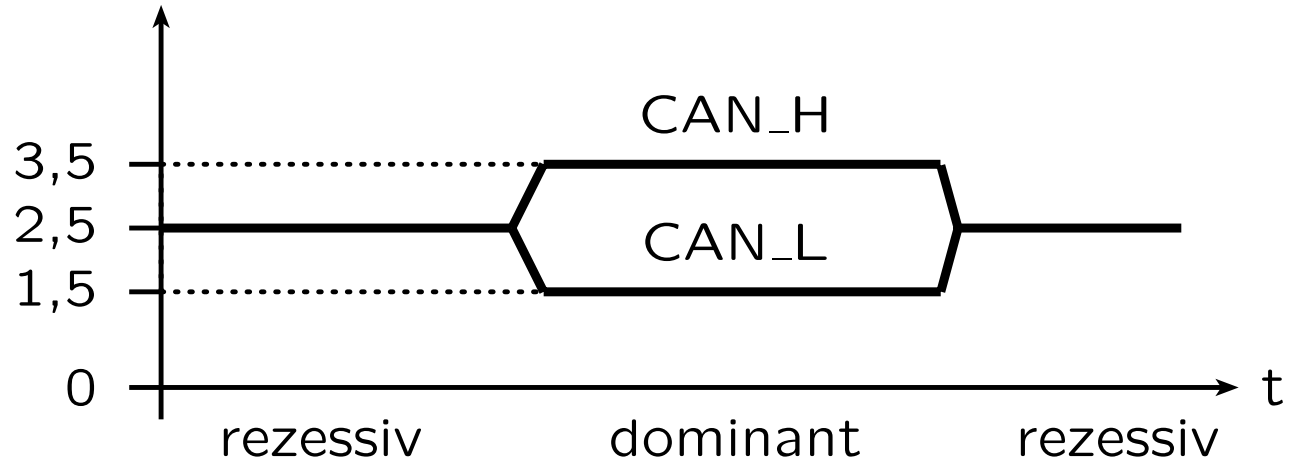
⋆ Transistor T_i leitet, falls $S_i = 0$
$\rightarrow$ Buspegel = 0

⋆ also: D–Bit überwiegt R–Bit

- Busstruktur



- Buspegel nach ISO 11898



6.6.2 Schicht 2

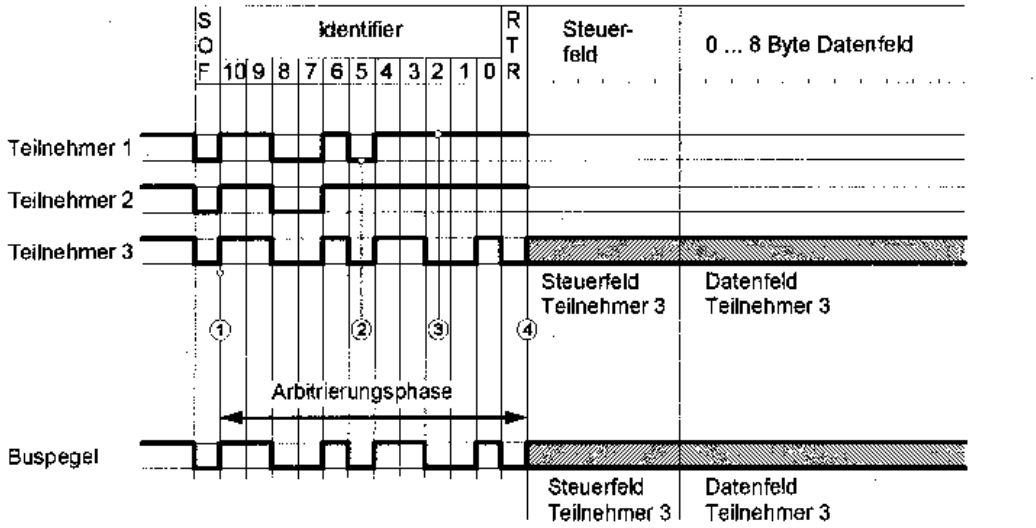
- Nachrichtenrahmen

≥ 3	1	11	1	6	0..64	16	2	7	Bit
IS	SOF	ID	RTR	C	Daten	CRC	ACK	EOF	

IS	Interframe Space mindestens 3 R–Bits zwischen Nachrichten
SOF	Start of Frame (1 D–Bit)
ID	Identifier (11 Bits) davon für Benutzer nur 2032 IDs extended Identifier (29 Bits)
RTR	Remote Transmission Request D–Bit: Es folgen Daten R–Bit: Sendeaufforderung an Empfänger
C	Count, 6 Bit: C[0:5] erste beide Bits sind Null (c[5]=c[4]=0) C[3]...C[0] enthält Länge Datenfeld in Bytes (max. 8)
CRC	Cyclic Redundancy Check 15 Bit Prüfsumme und vorangestellte 0
ACK	Acknowledge (2 Bits: a und dann R–Bit)
EOF	End of Frame: 7 R–Bits

- Maximale Länge einer Nachricht bei 8 Byte Nutzdaten:
  $3 + 1 + 11 + 1 + 6 + 64 + 16 + 2 + 7 = 111 bit$

- daraus maximale Verzögerung bei Sendewunsch einer hochprioren Nachricht
- Busarbitrierung
  - ⋆ eine Station darf senden, falls Bus frei (carrier sense)
  - ⋆ Nachricht beginnt mit Feld zur eindeutigen Busarbitrierung (enthält Nachrichten-ID), falls mehrere Stationen gleichzeitig zu senden anfangen sollten
  - ⋆ es gibt keine gleichzeitigen Sendevorgänge mit gleicher ID
  - ⋆ jede Sendestation hört auf zu senden, wenn auf Bus nicht das gesendete Zeichen erkennbar
  - ⋆ die Station mit der kleinsten Nachrichten-ID setzt sich durch (0 ist dominant)
  - ⋆ also: je kleiner ID desto höher Priorität beim Zugriff
  - ⋆ Übertragung des Siegers wird nicht gestört, da Ausbreitungszeit am Bus (bestimmt durch die Buslänge) viel kleiner als eine Bitdauer



Beispiel eines Arbitrierungsvorgangs im CAN-Protokoll. Die Teilnehmer 1, 2 und 3 beginnen gleichzeitig einen Arbitrierungsversuch (1). Teilnehmer 2 verliert zum Zeitpunkt (2), Teilnehmer 1 zum Zeitpunkt (3) das Buszugriffsrecht. Beide Teilnehmer gehen damit in den Empfangszustand; am Ende der Arbitrierungsphase (4) besitzt nur noch Teilnehmer 3 das Buszugriffsrecht und schaltet seine Nachricht auf den Bus.

- ID ist Objekt–Kennzeichen, durch das der Anwender den Inhalt der Nachricht erkennt
- keine Zuordnung ID zu Station
- Station nimmt nur die für sie bestimmten Nachrichten an (Akzeptanzfilter, Einzelkommunikation oder multicasting)
- Wegen dieser Akzeptanzstrategie sprechen die Entwickler von "objektorientiertem Verfahren" oder "verteiltem Speicher"
- Quittungen

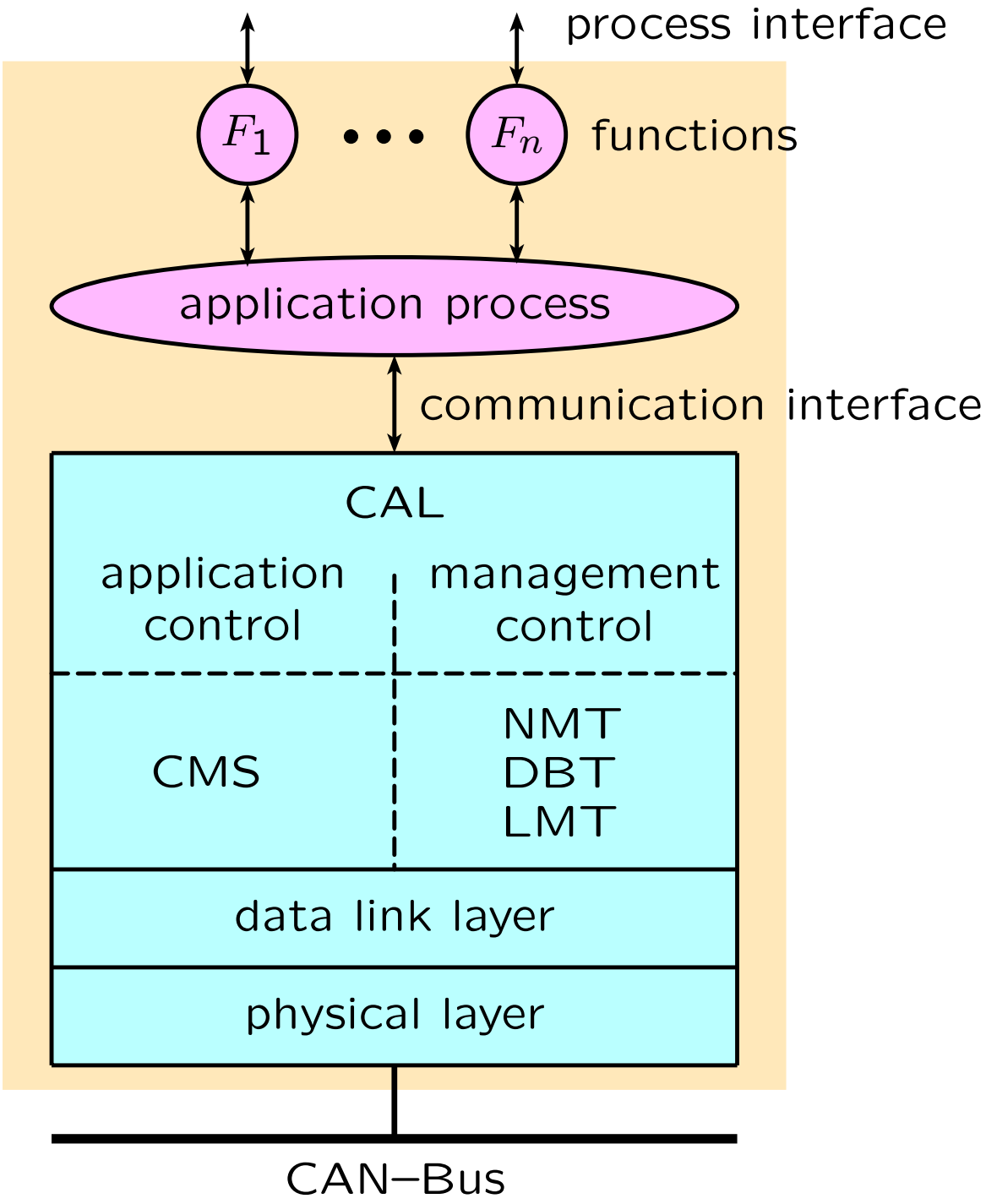
- ⋆ Sender setzt a in ACK auf R
- ⋆ falls der Empfänger Nachricht korrekt erhält (CRC–Test ok), setzt er noch während des Empfangs a auf D
- ⋆ Absender hört beim Senden mit und erkennt so, daß a auf D gesetzt wurde, also eine positive Quittung eines Empfängers vorliegt
- ⋆ Erkennen Stationen einen Fehler (bit–, bitstuff–, CRC–Fehler), dann senden diese im Anschluß an das ACK–Feld eine Folge von dominanten Bits als Fehlerkennzeichen (mindestens 6 aufeinanderfolgende dominante Bits, da diese in gültiger Nachricht wegen Bitstuffing nicht auftreten können). Die Nachricht wird daraufhin erneut gesendet. Durch ein Zählverfahren bemerkt jede Station die Anzahl ihrer Fehler. Beim Überschreiten des Grenzwertes werden Stationen vom Bus abgeschaltet.

- Puffer der Schicht 2
  - ⋆ BASIC CAN
    - ◦ 1 Sendepuffer
    - ◦ 2 Eingangspuffer
    - ◦ Wechselpufferkonzept
  - ⋆ FULL CAN
    - ◦ mehrere Sendepuffer
    - ◦ mehrere Empfangspuffer
    - ◦ alle Puffer je einem festen Identifier zugeordnet
    - ◦ Puffer zu einem ID werden überschrieben, falls neue Nachricht dieser ID eintrifft

6.6.3 Schicht 7

- CAN Application Layer (CAL)
- Erweiterungen
  - ⋆ bestätigter Datentransfer
  - ⋆ Übertragung von Paketen $>$ 8 Bytes
  - ⋆ ID Distribution
  - ⋆ Spezifikation von Standard Device Modules
  - ⋆ Startup
  - ⋆ Knotenüberwachung
- Standards
  - ⋆ CAL/CANopen (CiA)
  - ⋆ Device Net (Allen Bradley)
  - ⋆ SDS (Smart Distribution System), Honeywell Micro Switch

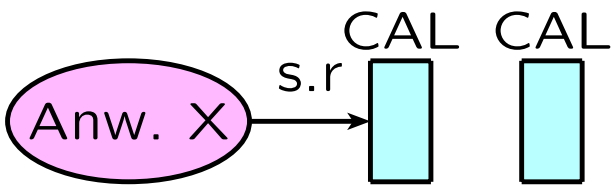
- Struktur eines CAN/CAL–Moduls



- Literatur
  K. Etschberger: CAN: Grundlagen, Protokolle, Bausteine, Anwendungen. Hanser 1994
- Benutzerorganisation CiA legte für CAL standardisierte Objekte (Variable, Ereignisse u.a.) und Dienste fest
  `http://www.can-cia.de/`
- Moduln
  - ⋆ CMS Can–based message specification
    - ◦ Objekte durch Attribute wie Name, Typ, Priorität, minimale Sendewiederholzeit beschrieben
    - ◦ Übliche Dienste:
      Read, Write, Notify, Load usw.
  - ⋆ DBT Distributor
    - ◦ dynamische Zuordnung von CAN–Identifiern zu CMS–Objekten bei der Initialisierung
    - ◦ erleichert Einsatz von Geräten verschiedener Lieferanten
    - ◦ netzweite Konsistenz der ID bei Sendern und Empfängern

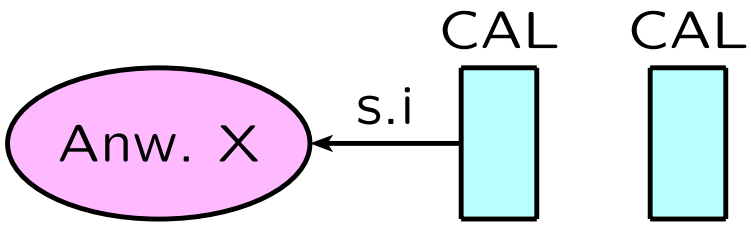
- ⋆ NMT Netzwerk–Management
  - ◦ Initialisierung, Starten und Stoppen von Prozessen auf Knoten
  - ◦ Erkennen von Fehlerzuständen im System
  - ◦ Schreiben/Lesen von Parametern bei den Knoten bei der Initialisierung
- ⋆ LMT Layer Management
  - ◦ Setzen von Zeitparametern in Schicht 2

- Ein Knoten muß für CAL die Masterfunktion übernehmen und einige Identifier hierfür reservieren
- CMS–Dienste
  - ⋆ lokale Dienste
    vom CAL–Modul selbst ausgeführt



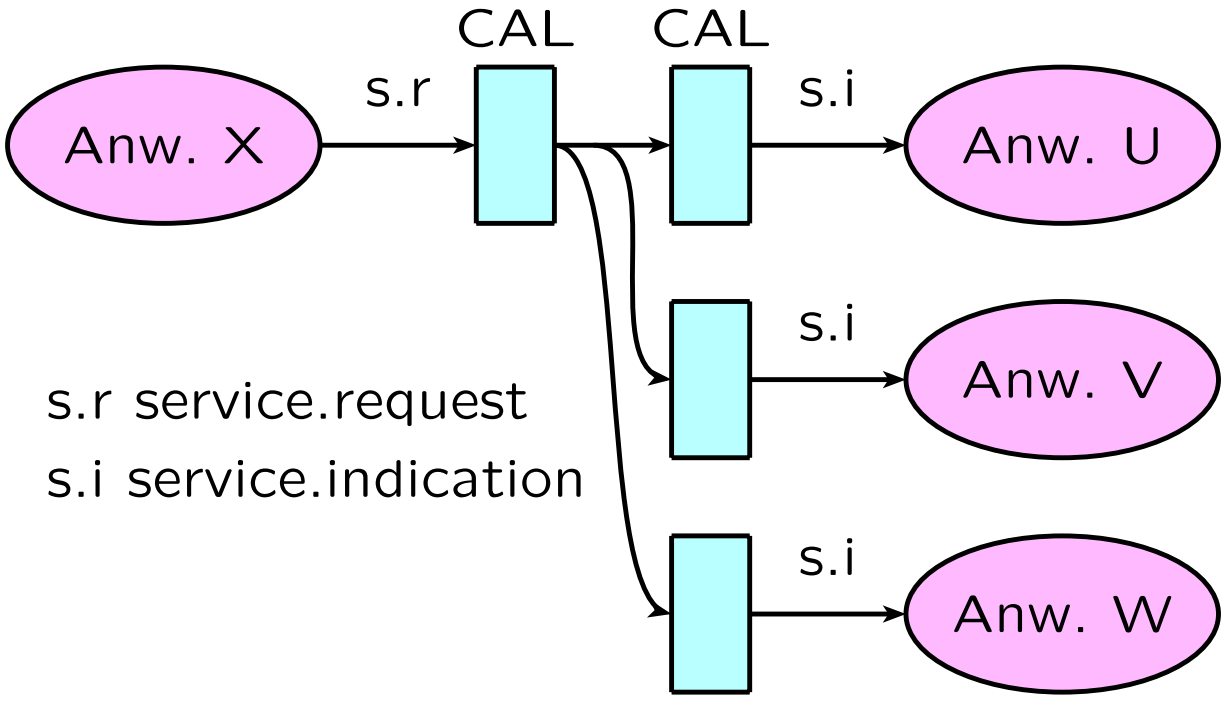
Anw. X = Anwendung X
s.r service.request

⋆ unangeforderte Dienste
lokales CAL zeigt ein von ihm festgestelltes Ereignis an



s.i service.indication

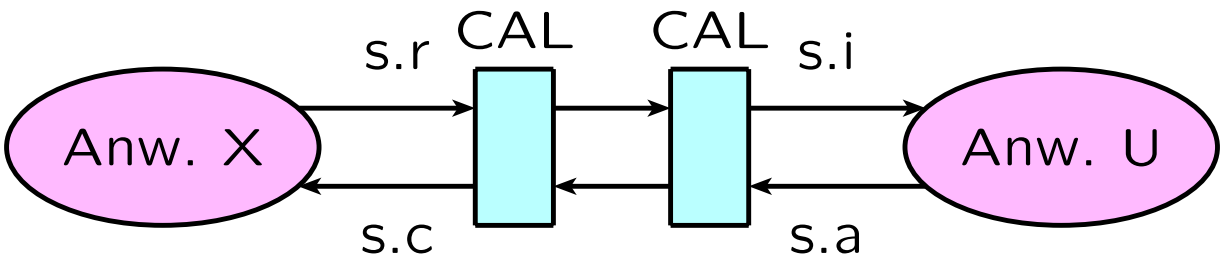
⋆ unbestätigte Dienste
Dienstanforderung über den CAN–Bus verschickt an einen oder mehrere Server



s.r service.request
s.i service.indication

⋆ bestätigte Dienste

Dienst von genau einem Server angefordert; dieser meldet das Ergebnis der Dienstausführung zurück



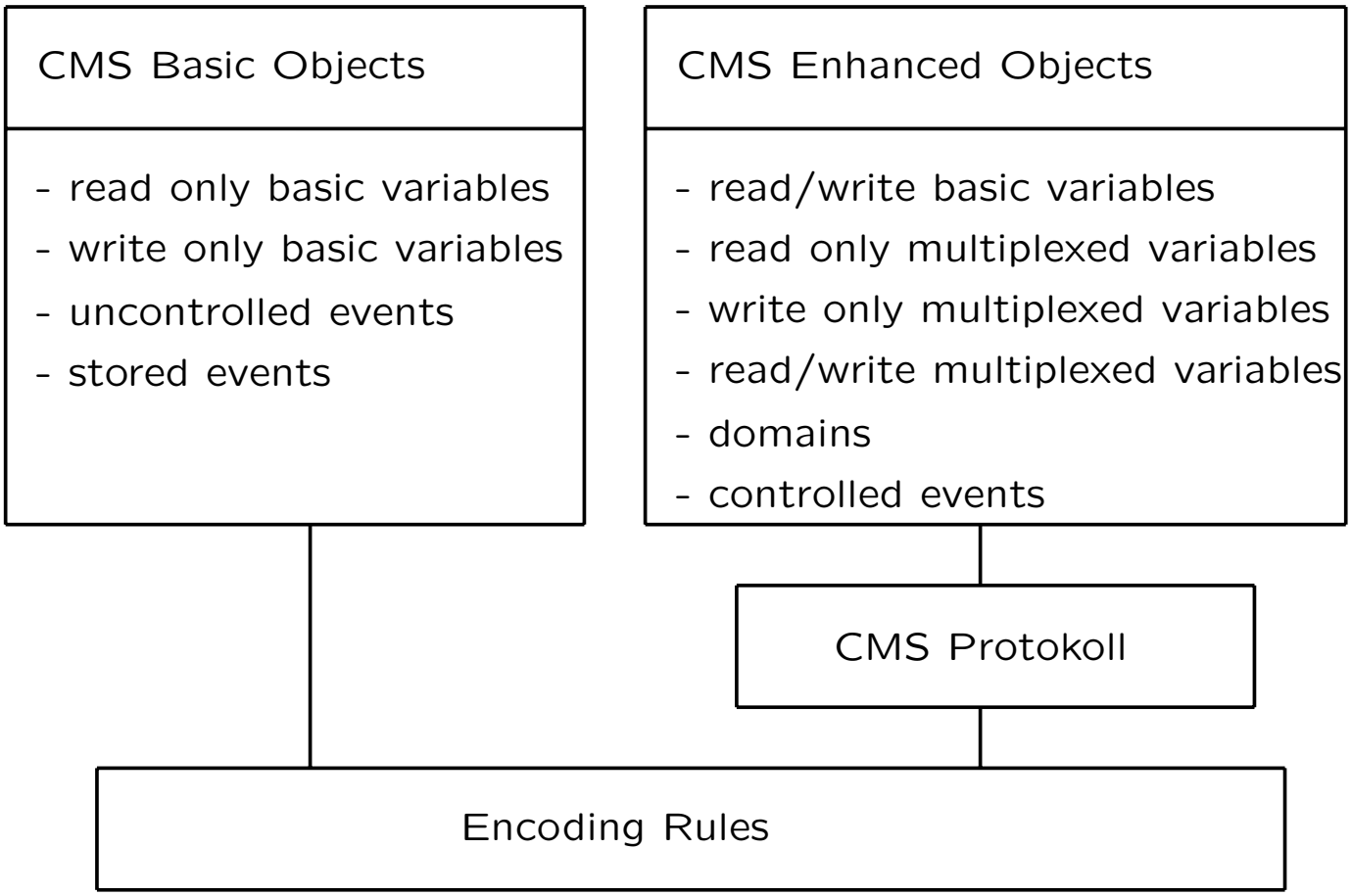
s.r service.request

s.i service.indication

s.a service.response

s.c service.confirmation

CMS–Modell

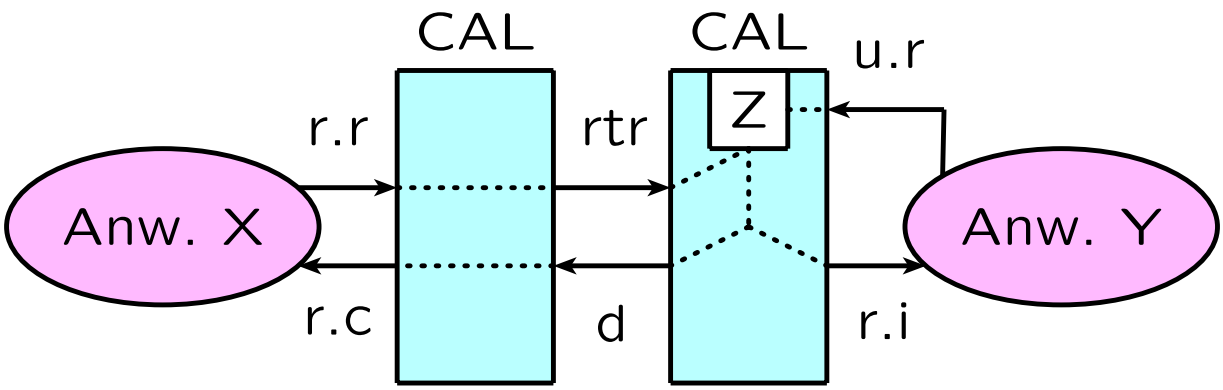

CMS Basic Objects
- read only basic variables
- write only basic variables
- uncontrolled events
- stored events
CMS Enhanced Objects
- read/write basic variables
- read only multiplexed variables
- write only multiplexed variables
- read/write multiplexed variables
- domains
- controlled events
CMS Protokoll
Encoding Rules

- CMS basic objects
  - ⋆ read only variable
    - ◦ kann vom Client nur gelesen werden
    - ◦ read.indication geht an Server-Prozeß
  - ⋆ write only variable
    - ◦ kann vom Client nur geschrieben werden
    - ◦ write.indication geht an Server-Prozeß
  - ⋆ uncontrolled event
    - ◦ Mitteilung, daß ein Ereignis eingetreten, wird spontan von einem Server an alle Clienten (bei Initialisierung definiert) geschickt
  - ⋆ stored event
    - ◦ Ereignis hat Eigenschaften einer read–only–Variable
    - ◦ zusätzlich kann es wie ein uncontrolled event mit dem Dienst "store and notify" vom Server beschrieben werden, wobei ebenfalls eine Nachricht an alle Clienten geschickt (mit dem Ereigniswert)

- Dienste für CMS basic objects
  - ⋆ Identifier der Nachricht bezeichnet eindeutig das basic object
  - ⋆ Dienste

basic object	Dienst	Diensttyp	Anforderer
write-only	write	unbestätigt	Client
read-only	update	lokal	Server
read-only	read	bestätigt	Client
uncon. event	notify	unbestätigt	Server
stored event	store	lokal	Server
stored event	store & notify	unbestätigt	Server
stored event	read	bestätigt	Client

  - ⋆ Zugriff auf die Variablen direkt vom CMS abgehandelt
  - ⋆ Beispiel:
    Übermittlung von Meßergebnissen bzw. Prozeßzuständen über
    Read–Only–Variable Z des Prozesses Y



r.r read.request

rtr remote transmit request

u.r update.request

r.i read.indication

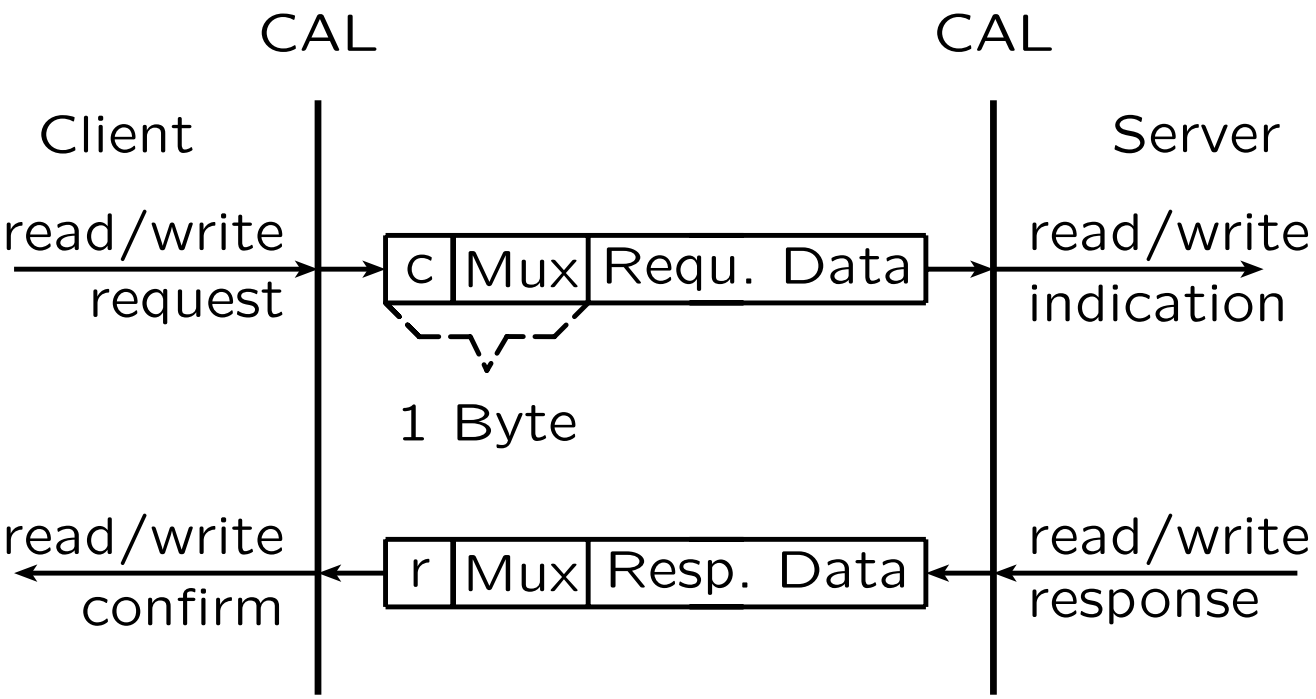
r.c read.confirmation

d CAN-Message with data

Erklärung:
Der Prozeß Y führt eine einfache Variable Z (z.B. die gemessene Temperatur in einem technischen Prozeß). Die Temperatur wird periodisch abgelesen und durch update.request wird die Variable Z aktualisiert. Der Prozeß Y kann die Variable Z jederzeit über einen Dienst read.request lesen. Der Prozeß X wird darüber informiert, daß die Variable gelesen wurde (read.indication).

- CMS enhanced objects (erweiterte Objekte)
  - ⋆ read–write basic variables
    - ◦ 1 Client und 1 Server
    - ◦ in CAN–Nachricht anzugeben, ob lesen oder schreiben
    - ◦ Dienste: read, write
  - ⋆ multiplexed variables
    - ◦ 1 Client und 1 Server
    - ◦ mehrere Variable zu einem CMS–Objekt zusammengefaßt
    - ◦ es gibt Strukturen und Felder
    - ◦ Dienste: read, write
  - ⋆ domains
    - ◦ Datenbereich $>$ 8 Bytes, z.B. für Programme
    - ◦ Dienste (vom Client angefordert): upload, download, initiate, abort
  - ⋆ controlled events
    - ◦ Anzeige des Ereignisses kann freigegeben oder gesperrt werden
    - ◦ es gibt genau einen Server und genau einen Clienten
    - ◦ Anwendung : Synchronisation
    - ◦ Dienste: notify, control

- Beispiel: read–write–Anforderung für Multiplexvariable



Erläuterung:

$$Mux = \begin{cases} 0 & \text{basic variable oder erstes Element einer multiplexed variable} \\ \neq 0 & \text{Auswahlindex multiplexed Variable } (\leq 128 \text{ Elemente}) \end{cases}$$

c: request code (0 write, 1 read)

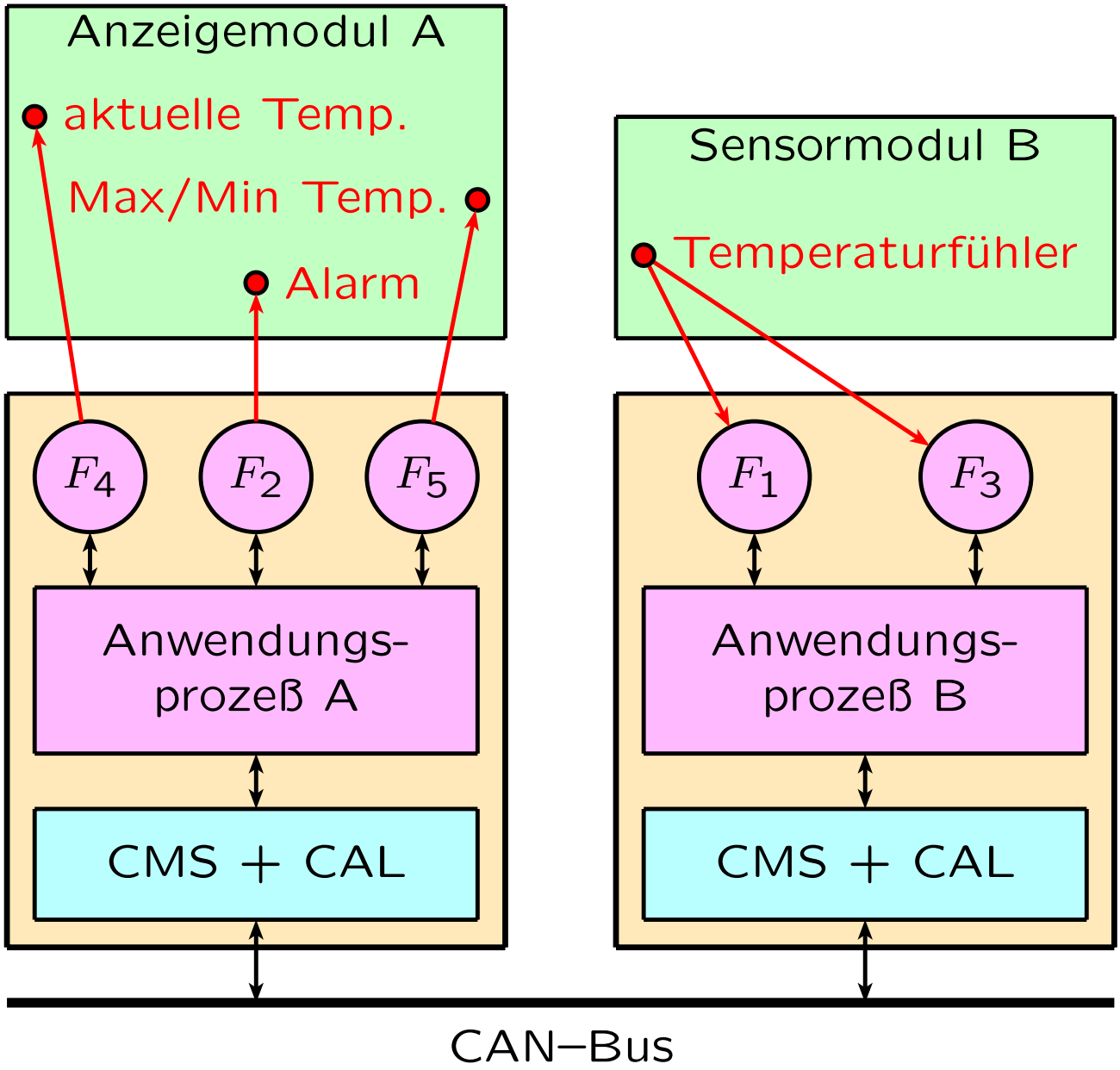
r: result code (0 success, 1 failure)

Request Data: zu schreibende Daten bei c = 0

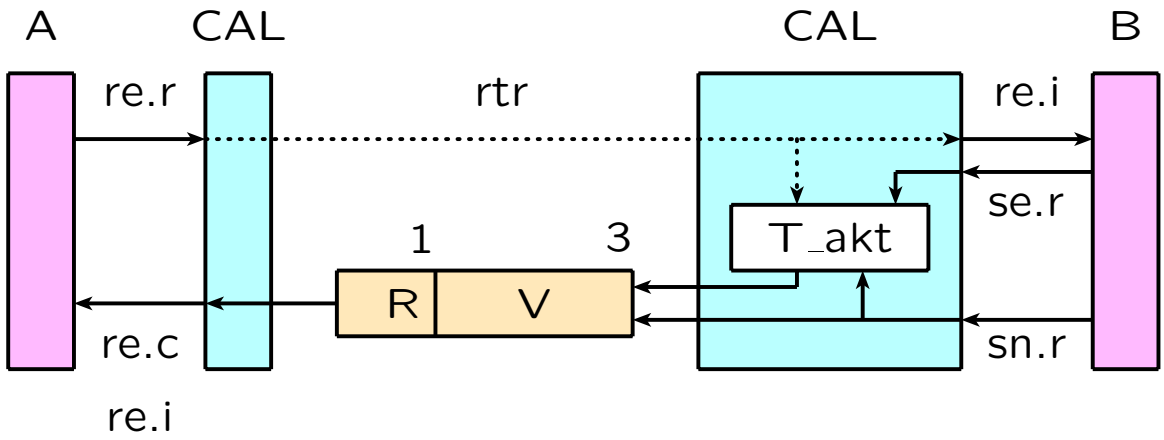
Response Data:

- geschriebene Daten bei c = 0, r = 0
- gelesene Daten bei c = 1, r = 0
- Fehlercode bei r = 1

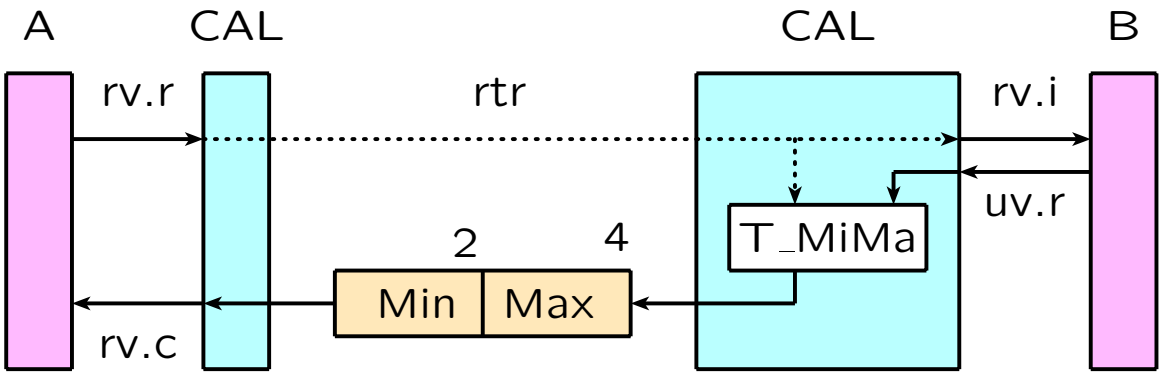
- großes Anwendungsbeispiel (aus Etschberger): Ablesen und überwachen von Temperaturen in einem Sensormodul sowie Anzeige durch einen Anzeigemodul



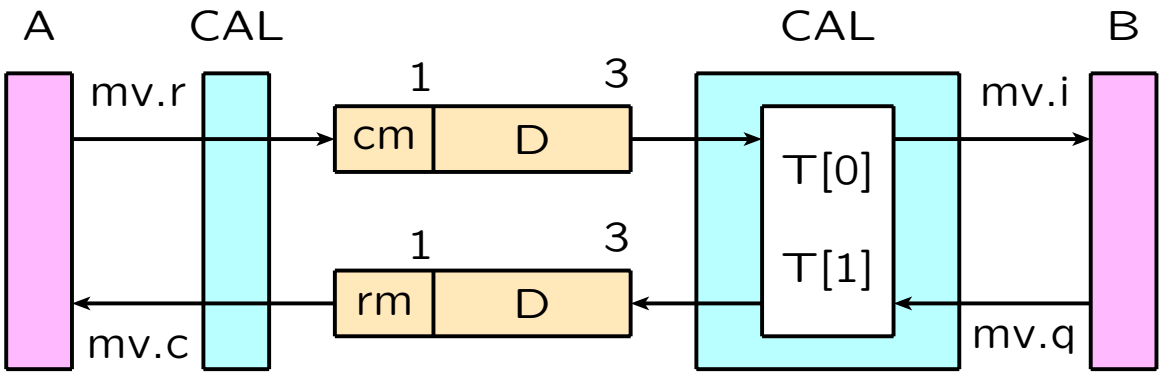
- ⋆ Aufgabe Anzeigemodul A
  - ◦ aktualisiert periodisch Temperaturanzeige (Funktion F4)
  - ◦ gibt Alarm bei Grenzwertüberschreitung (Funktion F2)
  - ◦ aktualisiert periodisch Anzeige der aufgetretenen Temperaturextrema (Funktion F5)
- ⋆ Aufgabe Sensormodul B
  - ◦ Beispiel für ein CMS-Profil
  - ◦ liest periodisch Temperatur und kontrolliert die Grenzwerte (Funktion 1)
  - ◦ bestimmt minimale und maximale aufgetretene Temperaturen (Funktion 3)
- ⋆ weitere Aufgabe von Anwendungsprozeß A
  - ◦ Einstellen der minimalen und maximalen Temperatur im Sensormodul



re.r	read_event.request
rtr	remote transmission request
re.i	read_event.indication
se.r	store_event.request
sn.r	store_and_notify.request
re.c	read_event.confirmation
R	(in/out of) Range
V	Value (Temp. Wert)
T_akt	aktuelle Temperatur
	Stored_Event
	Struktur: R (1 Byte), V (2 Byte)



rv.r	read_variable.request
rtr	remote transmission request
rv.i	read_variable.indication
uv.r	update_variable.request
rv.c	read_variable.confirmation
Min	minimale Temperatur
Max	maximale Temperatur
T_MiMa	minimale und maximale Temperatur Read_Only_Variable Struktur: Min (2 Byte), Max (2 Byte)



mv.r	read/write_mux_variable.request
mv.i	read/write_mux_variable.indication
mv.q	read/write_mux_variable.response
mv.c	read/write_mux_variable.confirmation
cm	c (1 Bit), Mux (3 Bit)
rm	r (1 Bit), Mux (3 Bit)
D	Data
T	Temperatur
	Read_Write_Multiplexed_Variable
T[1]	Obergrenze Temperatur
T[0]	Untergrenze Temperatur

6.7 MAP, Überblick

- Manufacturing Automation Protocol (MAP)
- Entwicklung ab 1982 (General Motors)
- inzwischen genormt
- Probleme
  - ⋆ Breitband–Geräteanschlüsse (10 Mbps) (zu) teuer
  - ⋆ FULL–MAP–Protokoll (zu) umfangreich und (zu) langsam
  - ⋆ kein Massenmarkt, daher Komponenten (zu) teuer
- Schicht 1 (physical))
  - ⋆ Token-Bus (IEEE 802.4)
  - ⋆ FULL–MAP mit Breitbandtechnik (10 Mbps)
  - ⋆ Routers und Gateways zur Kommunikation mit Full–MAP–Segmenten nötig
  - ⋆ auch Basisband-Technik mit 5 Mbps

- Schicht 2 (data link)
  - ⋆ Logical Link Control (LLC, IEEE 802.2)
  - ⋆ 3 Diensttypen
    - ◦ unbestätigte Datagramme (Typ 1)
    - ◦ verbindungsorientierte Nachrichten (Typ 2)
    - ◦ bestätigte Datagramme (Typ 3)
  - ⋆ reduzierte Definitionen für die Steuerebenen (real time segments) wie MINI–MAP und EAP (enhanced architecture protocol)
- Schicht 3 (network)
  - ⋆ End System to Intermediate System Vermittlungsprotokoll (ES-IS, ISO 9542)
  - ⋆ Connectionless-Mode Network Service (CLNS, ISO 8473)
- Schicht 4 (transport)
  - ⋆ Transport Service Class 4 (ISO 8072)
  - ⋆ end-to-end Protokoll
  - ⋆ verbindungsorientiertes Protokoll

- Schicht 5 (session)
  - ⋆ Basic Connection Oriented Session Service Definition and Protocol (ISO 8326, ISO 8327)
- Schicht 6 (presentation)
  - ⋆ abstrakte Syntax und Transfersyntax (ASN.1)
  - ⋆ ISO 8822, ISO 8823
- Schicht 7 (application)
  - ⋆ Aufteilung in 7a und 7b
  - ⋆ Schicht 7a (ACSE)
    - ◦ ACSE Association Control Service Elements
    - ◦ ISO 8649, ISO 8650
    - ◦ allgemeine Dienste zum Verbindungsaufbau und zum Verbindungsabbau: Initiate, Conclude, Abort, Cancel, Reject
    - ◦ Namen/Adressen von Objekten in Schicht 7

- Schicht 7b
  - MMS Manufacturing Message Specification (ISO 9506)
  - FTAM File Transfer Access and Management (ISO 8571)
  - NM Network Managment (ISO 9595)
  - DS Directory Services (ISO 9594)
  - MMS-Zusatzdefinition (bisher kein Standard) zur Einbindung der Dienstaufrufe in C (Parameter, Typen, Anzahlen) existiert
  - Begleitstandards (companion standards) für Roboter, NC-Maschinen und SPS

6.8 MMS bei MAP

- MMS Manufacturing Message Specification
- Client/Server–Konzept
- 16 Objekttypen mit Attributen
- 79 Dienste für Objekte (ohne FTAM)
- Modifikation der Objekte durch den Server über das Netz, durch den Client, durch autonome Übergänge
- Conformance Klassen
  - ⋆ wurden festgelegt, da volles MMS sehr komplex und einfache Geräte nur wenige Funktionen erbringen
  - ⋆ beschreiben die Dienste, die ein Gerät einer Klasse mindestens beherrschen muß
  - ⋆ beim Verbindungsaufbau werden Dienste und Parameter ausgehandelt, im Feldbereich i.a. nur bei System–Initialisierung
  - ⋆ einfachste Geräteklasse hat 5 Dienste

- Objektklasse Virtuelles Gerät (virtual manufacturing device, VMD)
  - ⋆ ein oder mehrere VMD's auf einem realen Gerät realisiert
  - ⋆ VMD's enthalten Objekte und Zustandsinformationen
  - ⋆ Ein VMD hat viele Objektattribute wie z.B.
    - ◦ Steuerprogramm–Name
    - ◦ Hersteller
    - ◦ Modell
    - ◦ Version
    - ◦ Liste akzeptierter Syntax
    - ◦ Status (voll, eingeschränkt)
    - ◦ physikalischer Status (ok, defekt)
    - ◦ Liste von enthaltenen Objektklassen

- Objektklasse Bereich (domain)
  - ⋆ Betriebsmittel
  - ⋆ meist Objekte, die Code und Daten enthalten können (Speicher)
  - ⋆ definieren den lokalen Gültigkeitsbereich der in ihnen enthaltenen Variablen
  - ⋆ Attribute z.B.
    - ◦ Name
    - ◦ Art (löschbar, sharable)
    - ◦ Liste enthaltener Variablen
    - ◦ Liste enthaltener PI
    - ◦ Zustand (loading, complete, in use u.a.)
  - ⋆ Dienste
    - ◦ Definition Bereich
    - ◦ Löschen Bereichen,
    - ◦ Übertragung in Bereich (upload/download)

- Objektklasse Abläufe (program invocation, PI)
  - ⋆ sind die Bearbeiter–Aktivitäten des Servers (tasks, threads)
  - ⋆ PI's auch dynamisch erzeugt und gelöscht
  - ⋆ Attribute z.B.
    - ◦ Name
    - ◦ Art (löschbar, reentrant u.a.)
    - ◦ Zustand (idle,starting, running, stopped, resetting, unrunable u.a.)
    - ◦ Liste benutzter domains
    - ◦ Liste relevanter Ereignisse
  - ⋆ Dienste
    - ◦ Definition PI
    - ◦ Löschen PI
    - ◦ Starten
    - ◦ Halten und Weiterführen (resume) der Programmausführung
    - ◦ Reset– und Kill–Funktionen

- Objektklasse Variable
  - ⋆ Variable sind Objekte, deren Inhalt Werte sind
  - ⋆ es gibt
    - ◦ unnamed variables
    - ◦ named variables
    - ◦ named typed variables
    - ◦ scattered access variables (Gruppe, nur Gesamtzugriff)
  - ⋆ Zugriff durch den Client
  - ⋆ Attribute z.B.
    - ◦ Zugriffsart
    - ◦ Zugriffsrechte
  - ⋆ Dienste z.B.
    - ◦ lesen
    - ◦ schreiben

- Objektklasse Semaphor
  - ⋆ Arten
    - ◦ Token–Semaphor (zählender Semaphor)
      Menge identischer Token, die frei oder belegt sind
    - ◦ Pool–Semaphor (binärer Semaphor)
      Token mit eindeutigem Namen
    - ◦ Semaphor–entry–Objekte
      Listenführung über "frei, belegt, wartend"; auch mit Zeitüberwachung und maximaler Belegungsdauer
  - ⋆ Attribute z.B.
    - ◦ Name, Art, Anzahl, Status (frei/belegt)
    - ◦ Liste Anforderer/Beleger, Prioritäten
    - ◦ Reaktionen bei Zeitüberschreitung oder Verbindungsabbruch
  - ⋆ Dienste z.B.
    - ◦ definieren, löschen
    - ◦ belegen, freigeben
    - ◦ Status abfragen
  - ⋆ Steuerung anderer Dienstanforderungen in Abhängigkeit vom Semaphorzustand

- Objektklasse Ereignis (event)
  - ⋆ Objekte
    - ◦ event condition (Ereignisbedingung)
      - ▷ lokale Bedingungen
      - ▷ Zustandsänderungen
      - ▷ vom Client ausgelöst
    - ◦ event–action (auszuführende Aktion)
    - ◦ Event–enrollment
      (Liste der Clients, die beim Eintreten verständigt werden)
  - ⋆ Dienste z.B.
    - ◦ Definition
    - ◦ Ändern und Löschen von Ereignisbedingungen und Aktionen
    - ◦ auslösen von Ereignissen
    - ◦ Abfrage von Attributen
    - ◦ Benachrichtigung bei Eintreten von Ereignissen

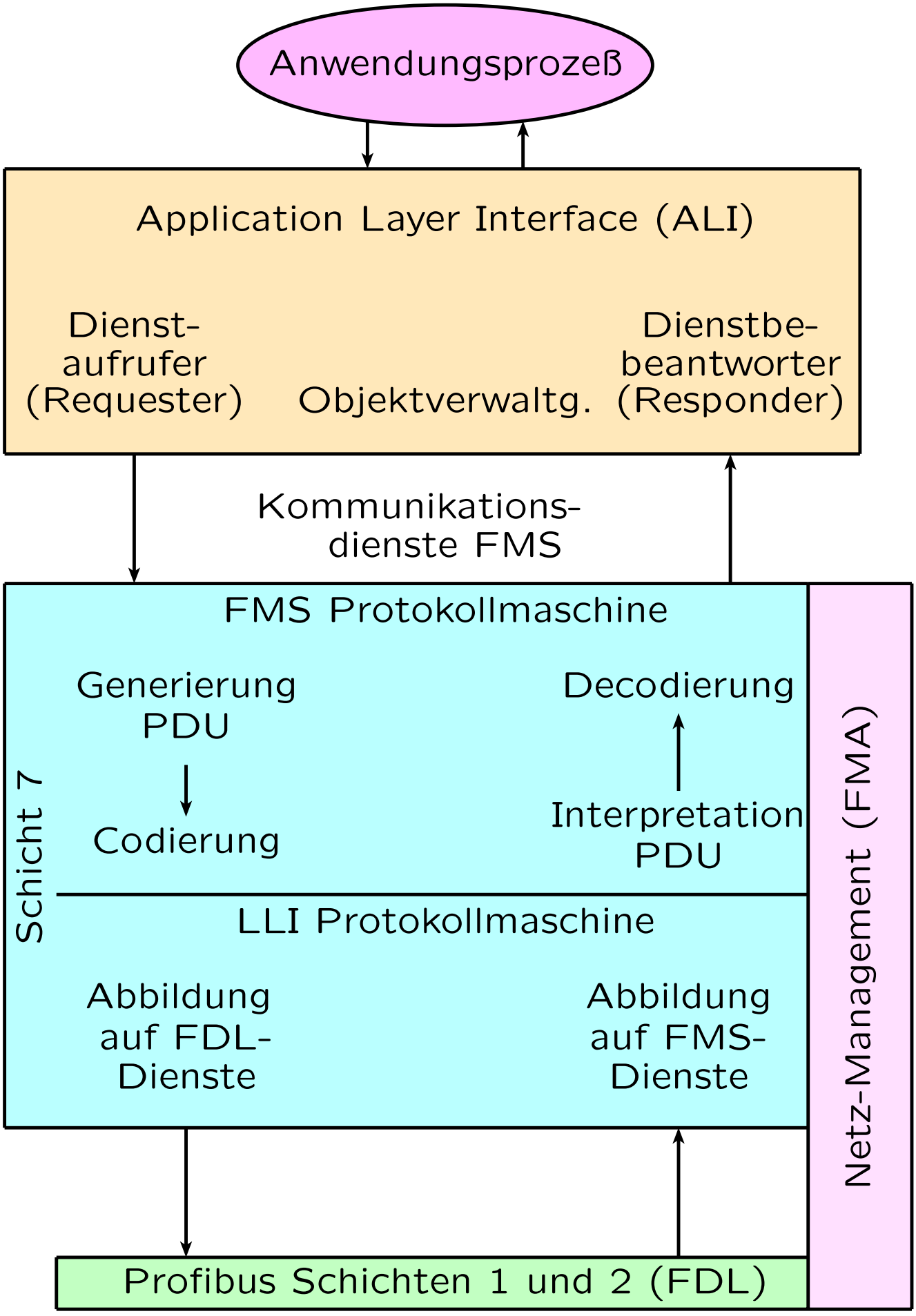
- Objektklasse Journal (journal)
  - ⋆ Unterstützung bei der Erstellung von Logbüchern
  - ⋆ Ordnung Informationen nach Zeitstempel oder Reihenfolge des Eintreffens
  - ⋆ Dienste z.B.
    - ◦ definieren von Journalen
    - ◦ löschen von Journalen
    - ◦ Lesen/Schreiben von Journaleinträgen
    - ◦ Abfragen des Journalzustands

- Robot Control Companion Standards (RCCS)
  - ⋆ ISO 9506
  - ⋆ Definition abstraktes Modell eines Robotersystems
    - ◦ in MMS–Terminologie
    - ◦ als Zwischenschritt noch von MMS unabhängiges Modell mit roboterspezifischen, abstrakten Objekten und Attributen
    - ◦ hieraus dann konkrete MMS–Objekte und deren Namen
  - ⋆ für die verschieden fähigen Robotersysteme wieder "conformance classes" festgelegt
  - ⋆ Robotersysteme als VMD beschrieben mit zusätzlichen Attributen
  - ⋆ die MMS–Dienste greifen auf die roboterspezifischen Objekte zu

- ⋆ jedoch die einzelnen Komponenten ohne Strukturierung
- ⋆ daher ein Vorschlag des deutschen NAM (Normenausschuß Machinenbau im DIN) zu ISO 9506 für eine hierarchische Strukturierung des physikalischen Robotersystems und (neu) die Einbeziehung der Fortbewegung mobiler Roboter
- ⋆ Robotersystem besteht aus:
  - ◦ einem oder mehreren Manipulatoren
  - ◦ Hilfseinrichtungen zur Steuerung des Robotersystems
  - ◦ einem Roboter–Steuerungs–System, das die Manipulatoren und Hilfseinrichtungen (intelligent) betreibt
  - ◦ den Roboter–Anwendungsprogrammen
  - ◦ ggf. dem Fahrwerk zur Mobilität

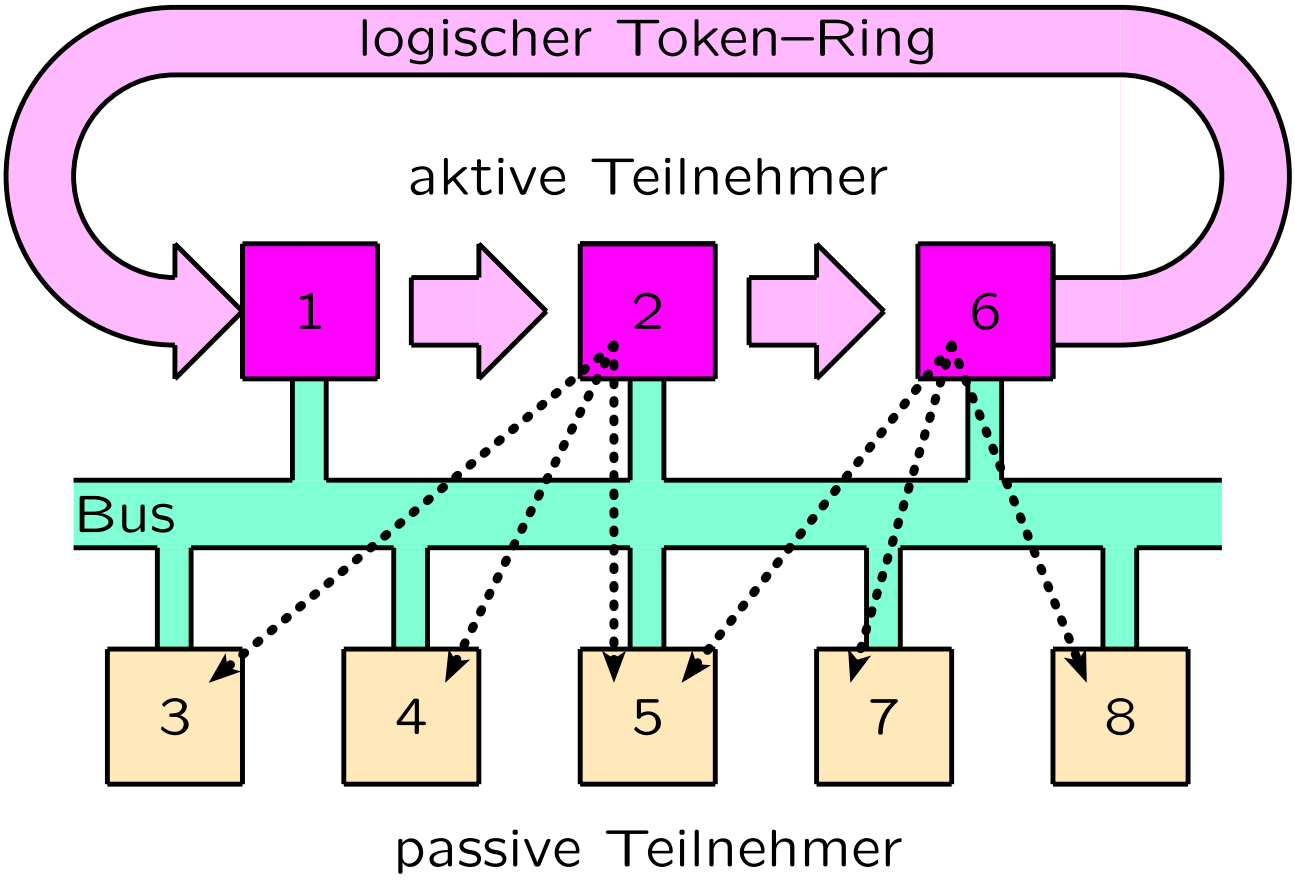
6.9 PROFI–Bus

- Process field bus (DIN 19245)
- deutsche Entwicklung ab 1987
- BMFT (FZI Karlsruhe) und 14 deutsche Industriepartner
- Master– und Slave– Geräte am Bus
- Norm definiert Schicht 1, 2 und 7 (3 bis 6 leer)
  - ⋆ Anwendungsprozesse wenden sich über ALI (application layer interface) an die Schicht 7 (FMS field message specification, an MMS orientiert)
  - ⋆ FMS verkehrt direkt mit Schicht 2 über LLI (lower layer interface)
- da FMS sehr umfangreich, PROFIBUS–DP als reduzierte Version
  - ⋆ Es bleiben die Schichten 1 und 2, an Stelle von FMS treten einige wenige Funktionen und Betriebsarten
  - ⋆ Anwenderprozesse wenden sich über DDLM (direct–data–link–mapper) direkt an Schicht 2


Anwendungsprozeß
Application Layer Interface (ALI)
Dienst-
aufrufer
(Requester)
Objektverwaltg.
Dienstbe-
beantworter
(Responder)
Kommunikations-
dienste FMS
FMS Protokollmaschine
Generierung
PDU
Codierung
Decodierung
Interpretation
PDU
Schicht 7
LLI Protokollmaschine
Abbildung
auf FDL-
Dienste
Abbildung
auf FMS-
Dienste
Netz-Management (FMA)
Profibus Schichten 1 und 2 (FDL)

- Schicht 1
  - ⋆ RS–485 oder LWL
  - ⋆ Übertragung asynchron (NRZ) mit üblichen UART–Bausteinen: 11 Bit für 1 Byte (+ Start-, Paritäts-, Stopbit) UART= universal asynchronous receiver/transmitter
  - ⋆ Bitraten von 9.6 kbps bis 1.5 Mbps in Stufen
  - ⋆ maximal 32 Teilnehmer pro Segment
  - ⋆ mehrere Segmente über Repeater
  - ⋆ maximal 127 logische Teilnehmer
  - ⋆ zwei Prioritätsstufen

- Schicht 2
  - ⋆ Buszugriff
    - ◦ Master–Geräte (aktive Teilnehmer) im (logischen) Token–Bus–Ring
    - ◦ Slave–Geräte (passive Teilnehmer) senden auf Anforderung des Masters in dessen Buszugriffszyklus

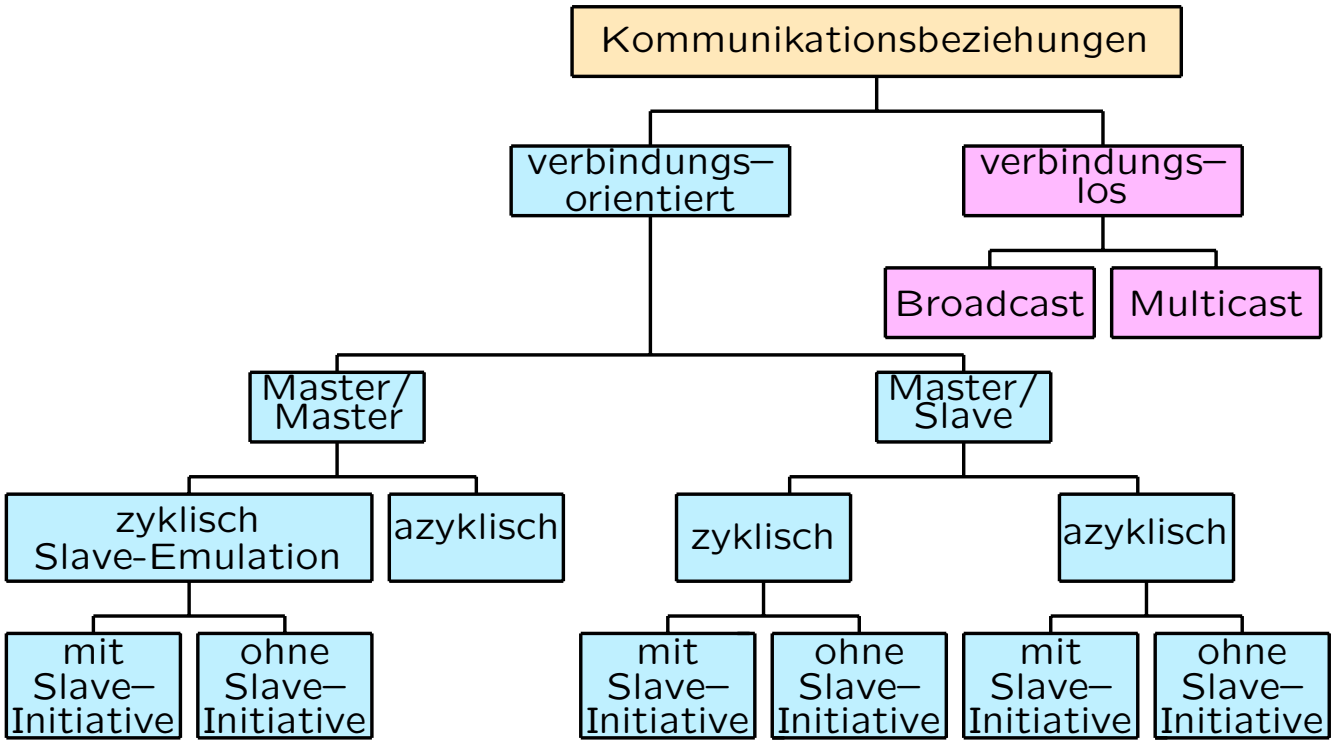


- Nachrichtentypen
  - SDN Send data with no ACK (z.B. Rundruf)
  - Master/Master oder Master/Slave Nachrichten
    - SDA Send data with acknowledge
    - RDR Request data with reply
    - SDR Send and request data
  - Beschleunigung zyklischer Dienste durch "Polling–Liste" in LLI
    - CRDR Cyclic request data with reply
    - CSRD Cyclic send and request data

- Rahmen (Telegramme)
  - Bezeichnungen für Felder

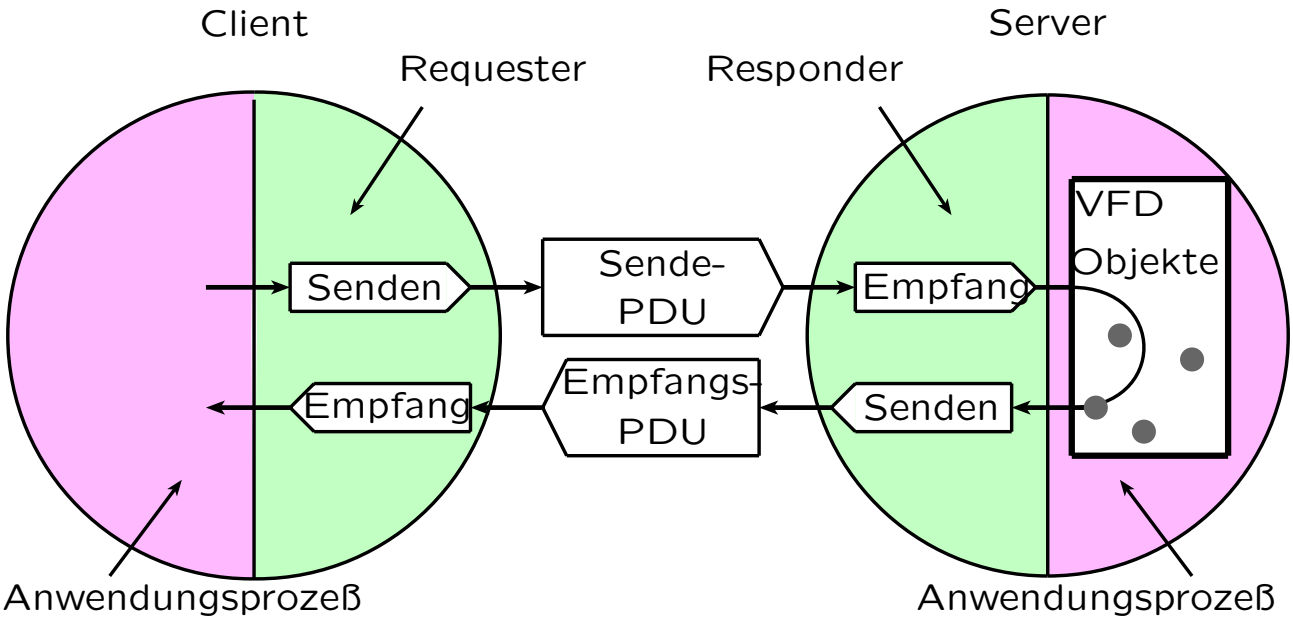
SD	Startzeichen (start delimiter) (vier Varianten SD1 bis SD4)
DA	Empfänger (destination address)
SA	Absender (source address)
ED	Endezeichen (end delimiter)
FC	Steuerzeichen (frame control) Aufruf, Quittung, Antwort, Priorität, Sequenznummer
LE	Länge (length)
FCS	Längsparität (frame–check sequence)

  - Kommando (6 Byte):
    SD1 DA SA FC FCS ED
  - Daten fester Länge (14 Byte):
    SD2 DA SA FC Daten (8 Byte) FCS ED
  - Daten, variabler Länge (max. 246 Byte):
    SD3 LE1 LE2 DA SA FC Daten FCS ED
  - Token: SD4 DA SA
  - Kurzquittung: SC

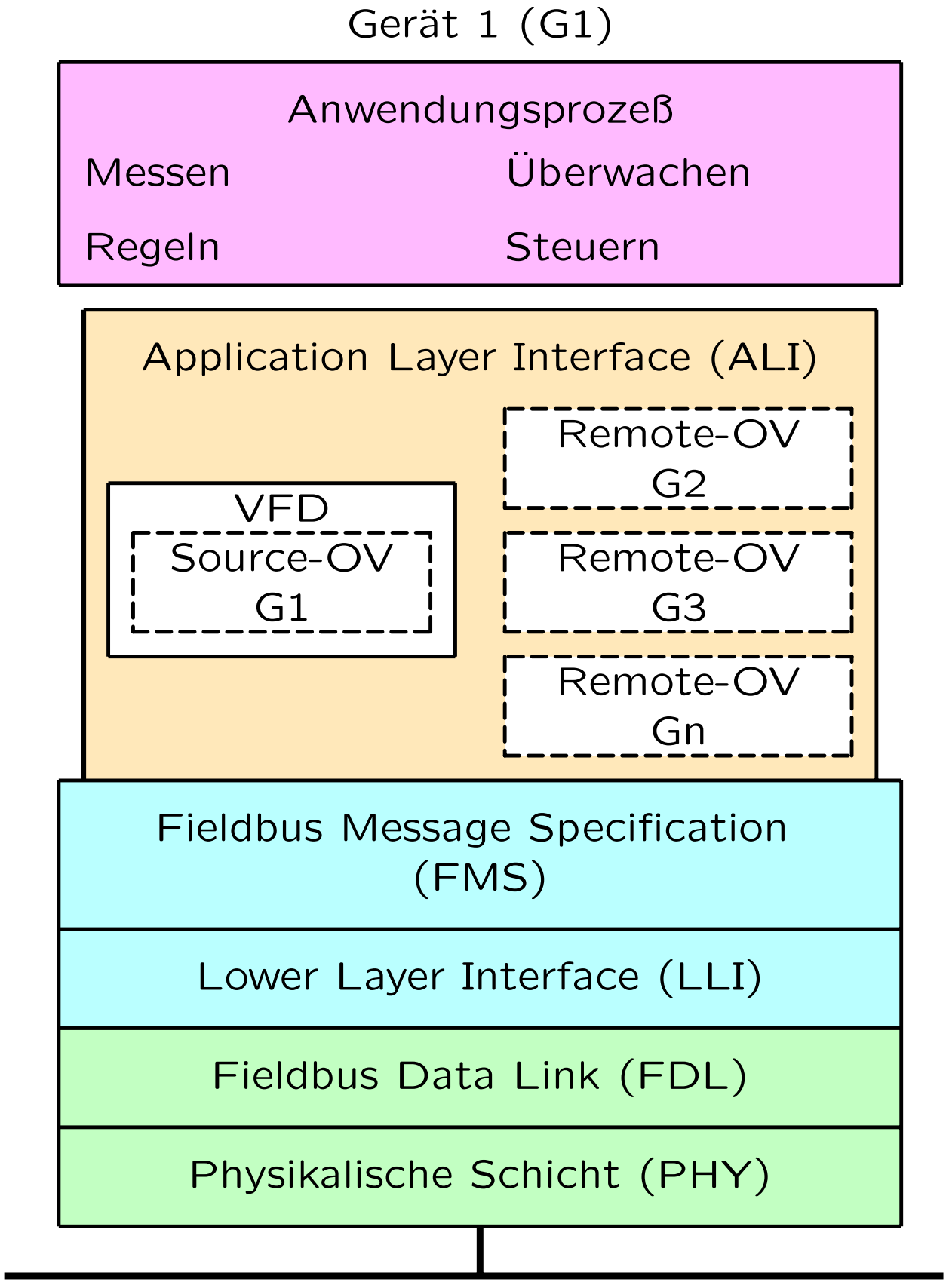

Kommunikationsbeziehungen
verbindungs–orientiert
verbindungs–los
Broadcast
Multicast
Master/Master
Master/Slave
zyklisch Slave-Emulation
azyklisch
zyklisch
azyklisch
mit Slave–Initiative
ohne Slave–Initiative
mit Slave–Initiative
ohne Slave–Initiative
mit Slave–Initiative
ohne Slave–Initiative

- ALI (application layer interface)
  - ⋆ Zur Kommunikation dienen logische Kanalnummern mit Eigenschaften, die bei Konfigurierung festgelegt
  - ⋆ Objekte
    - ◦ Objekte stehen im Anwendungsprozeß und sind über Prozeduranfrage von ALI aus zugänglich
    - ◦ statische Objekte
      - ▷ Variable (simple, array, record, list variable)
      - ▷ Bereich (domain)
      - ▷ Ereignis (event)
    - ◦ dynamische Objekte
      - ▷ Abläufe (program invocations)
    - ◦ Attribute z.B.
      - ▷ Index im Objektverzeichnis OV
      - ▷ Name
      - ▷ max. Größe in Bytes
      - ▷ phys. Adresse
      - ▷ Zustand
      - ▷ freie benutzerspezifische Attribute

- ⋆ Objektverzeichnisse (OV)
  - ◦ ALI enthält die Objektverzeichnisse (OV) der eigenen und ausschnittsweise der fremden Objekte und verwaltet sie
- ⋆ Dienste
  - ◦ Zugriff auf Variable (read/write)
  - ◦ Zugriff auf Bereiche (init, request, terminate up–/down–load)
  - ◦ Programm-Aufruf (create, delete, start, stop, resume, reset, kill)
  - ◦ Ereignisse (notification, alter–event–condition)
  - ◦ VFD-Verwaltung (für virtual field devices)
  - ◦ OV–Verwaltung
  - ◦ Kommunikationsüberwachung
- ⋆ Es werden VFD's (virtual field devices) definiert, die über ihre Objekte angesprochen werden



Virtuelles Feldgeräte–Modell



Schichtenmodell PROFI-Bus

- Anwendungsbeispiel
  - ⋆ periodisches Messen Öltemperatur im Server
  - ⋆ Client liest aktuellen Wert
  - ⋆ Eintrag in OV

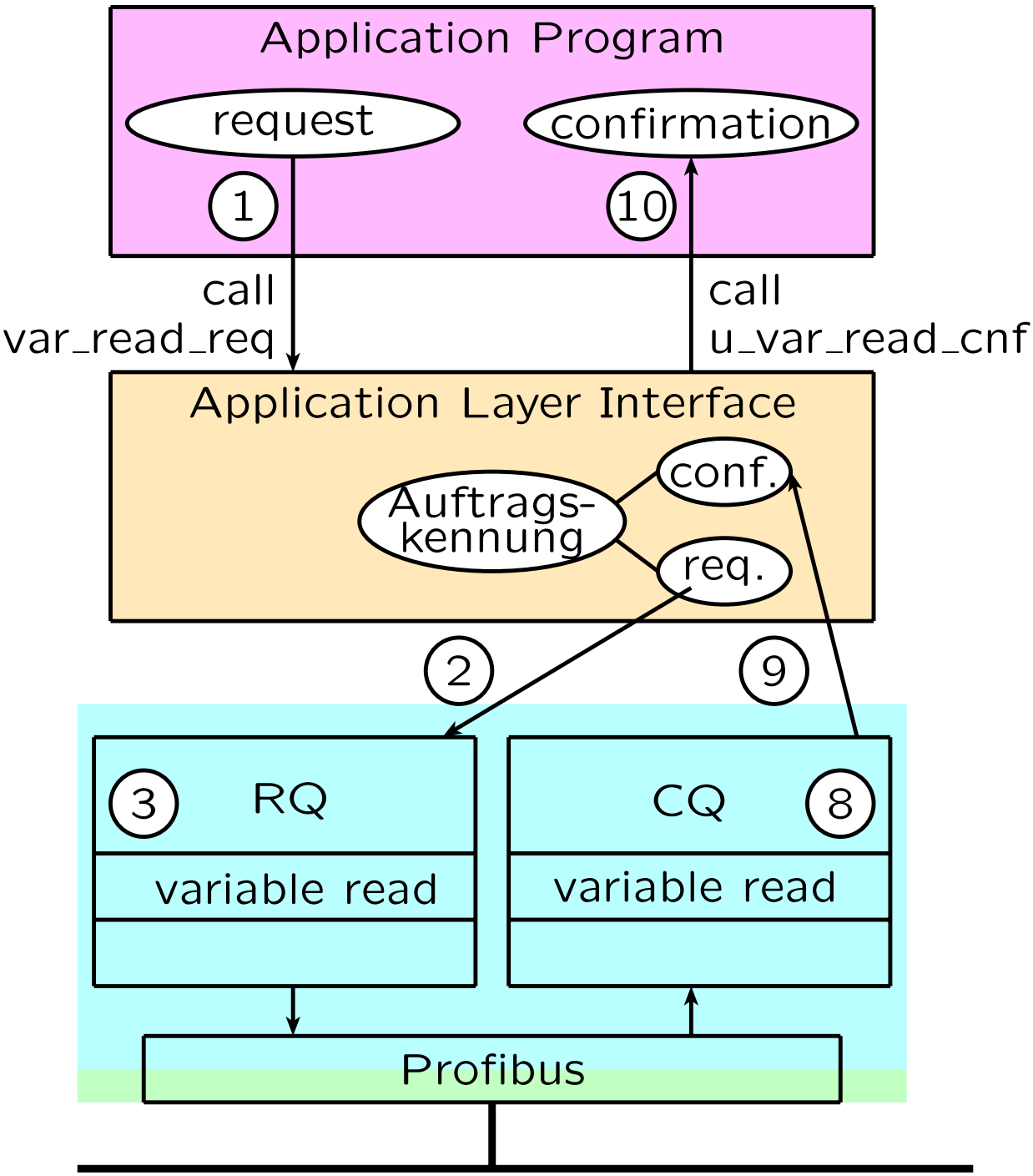
Index	Typ	Länge	Adresse	Symbol	···
...					
112	7	2	6041	Öltemperatur	···
...					

Anmerkungen:

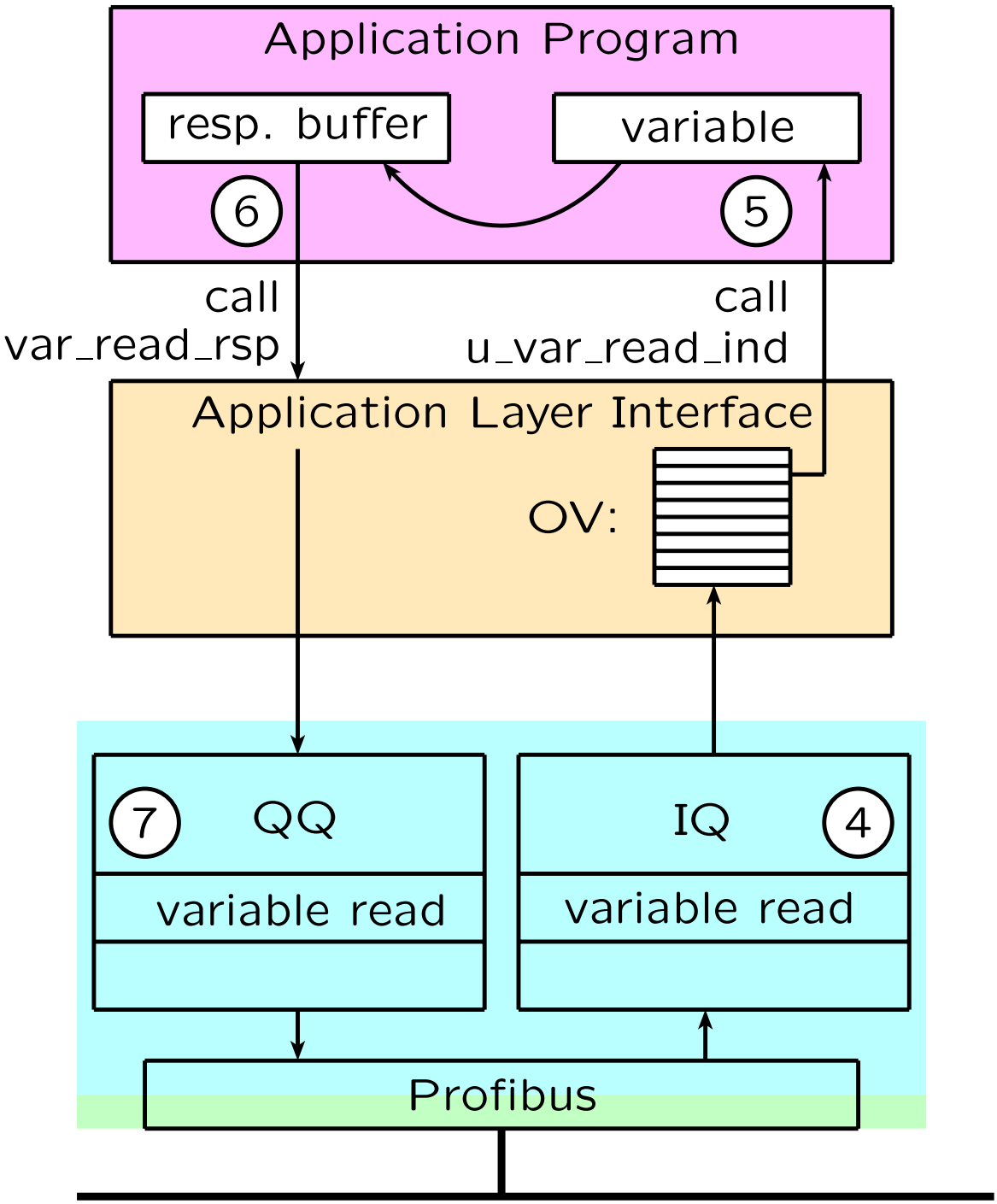
- ◦ Typ 7 bedeutet INTEGER 16
- ◦ weitere Einträge z.B. Maßeinheit

- ⋆ nachfolgend Blockdiagramm für Client und Server
- ⋆ die ausgeführten Schritte beim Lesen durch den Client sind fortlaufend numeriert
- ⋆ Warteschlangen zur besseren Entkopplung der Prozesse

CQ confirmation queue
IQ indication queue
QQ response queue
RQ request queue


Application Program
request
confirmation
1
10
call
var_read_req
call
u_var_read_cnf
Application Layer Interface
Auftrags-
kennung
conf.
req.
2
9
3
RQ
CQ
8
variable read
variable read
Profibus

Client


Application Program
resp. buffer
variable
6
5
call
var_read_rsp
call
u_var_read_ind
Application Layer Interface
OV:
7
QQ
variable read
IQ
4
variable read
Profibus

Server

6.10 SERCOS-Bus

- serielles Echtzeit–Communications–System (SERCOS)
- entwickelt von Bosch und VdW
- Einsatz vor allem für Steuerung von Werkzeugmaschinen-Antrieben und Robotergelenken
- Lichtwellenleiter
- NRZI-Code mit Bitstuffing nach 5 mal "1"
- Ziel:
  präzise äquidistante Zeitraster der Übertragung
- Zeitraster z.B. 62μsec, 125μsec, usw. bis max. 65 msec
- eine Master-Steuerung ist mit (vielen) Slaves (z.B.Antrieben) in einem LWL–Ring verbunden
- bei Fehlern wird nicht wiederholt, sondern auf den nächsten Takt gewartet
- bei Mehrfachfehler: Abschalten

- 3 Typen von Rahmen (”Telegrammen”) zur Aktualisierung der Ist–und Soll–Werte
  - ⋆ Master–Synchronisationstelegramme (SYNCH)
    Master–Station teilt Zeitbasis bezogen auf Zyklusbeginn mit (Rundspruch an alle Slaves)

SYNCH	···	Slave 1	···	Slave n	···	MT

  - ⋆ Antriebs–Telegramme (SLAVE)
    jeder Slave sendet pro Zyklus zu dem ihm zugeteilten Zeitpunkt die angeforderten Daten an den Master

SLAVE	SD	Adr	Daten (z.B. 32 Bit)	CRC	ED

  - ⋆ Master–Datentelegramm (MT)
    nach jedem Antriebs–Telegramm sendet der Master Daten (je 32 Bit) für alle Slaves

MT	SD	Adr	Daten für n Slaves (n∗32 Bit)	CRC	ED

- Start– und Ende–Delimiter SD und ED sind das 01111110-Flag des HDLC–Protokolls
- Interpretation der Daten (z.B.Namen, Funktion) durch Id–Nummern (umfangreicher vorgegebener Katalog) festgelegt (z.B. bedeutet 36 einen Geschwindigkeits–Sollwert)
- viele Nummern frei für spezielle Anwendungen; dies vermindert aber die Portabilität der Programme

6.11 INTERBUS–S

- INTERBUS–S –Protokoll dient zur schnellen zeitäquidistanten Übertragung bei vielen Teilnehmern im Sensor–Aktor–Bereich
- eine Master–Steuerung und n Teilnehmer (Slaves) im Ring angeordnet
- die Daten aller Slaves werden ausgehend vom Master pro Zeitzyklus in einem Rahmen durchgeschoben (wie ein großes Schieberegister); die Slaves erhalten dabei aus dem Puffer des Masters die Daten (Sollwerte) und senden gleichzeitig ihre Ist– und Statuswerte
- die Daten werden in einem Rahmen (Summenrahmenverfahren) gesendet; es gibt keine Adressen, die Position im Rahmen gibt die Teilnehmer–Nummer

Bits	16	16	16	$\cdots$	16	16	16
Frame	loopcheck	T_1	T_2	$\cdots$	T_n	CRC	ED

- Ablauf
  - ⋆ jeder Slave hat ein Schieberegister (1 bis 4 Byte), in das er am Zyklusanfang seine Daten schreibt
  - ⋆ der Master beginnt seinen Puffer zu leeren, jeder Slave schiebt nun die Daten im Ring unbesehen weiter, bis auf die für ihn bestimmten
  - ⋆ letztere bringt er in sein Schieberegister und sendet dafür seine Daten
  - ⋆ am Ende hat jeder Slave in seinem Schieberegister die Daten des Masters und der Master in seinem Puffer die Daten aller Slaves
  - ⋆ beispielsweise Zykluszeit 3 msec bei 48 Slaves mit 16–Bit–Daten
- gelegentliche azyklische Daten (sogenannte Bedarfsdaten wie Parameter, Steueranweisungen) müssen in übergeordnetem Protokoll in 2–Byte–Pakete zerlegt und in freien Zeitschlitzen der Station gesendet werden

- viele Kennzahlen (analog wie bei SERCOS-Bus) erlauben eine einheitliche Interpretation
- Bewertung
  - ⋆ gut geeignet für Sensor–/Aktorebene (wenig Overhead, äquidistant)
  - ⋆ starre Stationszahl: Probleme beim Zu– und Abschalten von Teilnehmern

6.12 Busse zur Gebäude–Leittechnik

- Ziel
  Vernetzung von gemeinsamen und privaten Funktionen in Häusern, Schulen, Banken, Krankenhäusern, Industriegebäuden
- Milliardenmarkt für Vernetzung und Geräteprodukte
- derzeitige Vorschläge ähneln Sensor–Aktor–Bussen
- Beispiele u.a.
  - ⋆ EIB European Installation Bus
    unterstützt z.B. von Bosch, Siemens
    - ◦ twisted pair, powerline
      Funk, Infrarot, EIB.net
    - ◦ ISO/IEC 8802-2 Logical Link Layer
    - ◦ kurze Telegramme mit bis zu 14 Bytes Daten
    - ◦ 2000 Produkte verfügbar
    - ◦ Installationen mit bis zu 60000 devices
  - ⋆ LON Local Operating Network
    spezielle "Neuron"–Chips der Firma Echolon mit 3 CPU's (2 für Protokoll, 1 für Anwendungs–SW)