TU Dresden
Fakultät Informatik

Großer Beleg

Thema

Übersicht und Klassifizierung von Automatisierungsnetzen

Vorgelegt von: **Wei, Xiaojun**
Matrikelnr.: 2914327
Studiengang: Informatik
Jahrgang 2002

Betreuer: Dr. rer. nat. D. Gütter
Dr.-Ing. A. Luntovskyy

Verantwortlicher
Hochschullehrer Prof. Dr. habil. A Schill

Beginn der Arbeit: 01.03.2006
Abgabedatum: 31.10.06

TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Systemarchitektur, Professur für Rechnernetze

AUFGABENSTELLUNG FÜR DEN GROSSEN BELEG

Name, Vorname: Wei, Xiaojun
Studiengang: Informatik
Matrikel-Nr.: 2914327
Thema: **„Übersicht und Klassifizierung von Automatisierungsnetzen"**

ZIELSTELLUNG

Innerhalb des CANDY-Projektes sind in den Jahren 2004/2005 Untersuchungen zur Unterstützung des Rechnernetzdesigns entwickelt worden.
Dabei ergab sich, dass im Bereich der Automatisierungsnetze, insbesondere der Gebäudeautomatisierungsnetze, ähnliche Entwurfsprobleme existieren.
Im Rahmen der Belegarbeit soll eine Übersicht über aktuelle Standards auf dem Gebiet industrieller Netze aufgestellt werden, speziell zu Anwendungsgebieten, hardware- und softwaretechnischen Parametern, Infrastrukturanforderungen.
Außerdem soll untersucht werden, wie LAN und Automatisierungsnetze gekoppelt werden können.
Auf Basis dieser Erkenntnisse soll eine Konzeption zur Erweiterung des CANDY-Tools um Komponenten zur Projektierung von Gebäudeautomatisierungsnetzen erarbeitet werden. Dies beinhaltet Vorschläge zur Erweiterung der Netzbeschreibungsspache NDML mit weitgehend einheitlicher Beschreibung von Gebäudegeometrie, Verkabelung usw. Außerdem ist ein Vorschlag zur programmtechnischen Realisierung innerhalb von CANDY zu erarbeiten.

Betreuende HSL: Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill

Betreuer: Dr. rer. nat. Dietbert Gütter
Dr.-Ing. Andriy Luntovskyy

Beginn am: 01. März 2006
Einzureichen am: 31. August 2006

Unterschrift des betreuenden Hochschullehrers

Postadresse (Briefe)
Technische Universität Dresden
01062 Dresden

Postadresse (Pakete u.ä.)
Technische Universität Dresden
Helmholtzstraße 10
01069 Dresden

Besucheradresse
Sekretariat:
Hans-Grundig-Str. 25
Zi. 316 a

Internet
http://www.rn.inf.tu-dresden.de

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabelleverzeichnis

0. Einführung

Automatisierungsanlagen bestehen aus Automationsgeräten (z.B. SPS, DDC; Sensoren, Aktoren, Controlleren, usw), Feldbussen oder Realtime-Ethernet-LAN, einem Managementsystem usw. Die Automatisierung kann man z.B. in der Industrie oder im Gebäudemanagement anwenden. Dies wird „Industrieautomatisierung“ und „Gebäudeautomatisierung“ genannt. Die Automatisierungssysteme können über das Internet mit lokalem Rechnernetz entfernt verwaltet werden. Für die wirtschaftliche und effiziente Systemverwaltung wird insbesondere auf dem Gebiet der Gebäudeautomatisierung „Facility Management (FM)“ entwickelt und eingesetzt. Für Facility Management existiert bisher noch keine einheitliche Definition.
CANDY (Computer Aided Network Design UtilitY) ist ein Netzwerkdesignwerkzeug. In den Jahren 2004/2005 wurden CANDY-Komponenten zur Unterstützung des Rechnernetzdesigns entwickelt. Zur Zeit wird CANDY erweitert, um auch Entwurfsprobleme in der Gebäudeautomatisierung zu lösen.
Der vorliegende Beleg gliedert sich in zwei große Teile:
Im ersten Teil wird die Entwicklungen der Automatisierung vorgestellt. Schwerpunkte sind Erläuterungen zur Gebäudeautomation, z.B. werden im Kapital 3 die wichtigen Komponenten der Gebäudeautomation behandelt. Die einzelnen Komponenten werden aus den Gesichtpunkten von Hardware und Software beschrieben.
Danach werden die Feldbusse schwerpunktmäßig erklärt und miteinander verglichen.
Speziell wird der LON-Bus als ein Kern der Feldbus-Technik in der Gebäudeautomatisierung diskutiert.
Im Kapitel 6.6.2 „Einige mögliche Verbindungsweisen zwischen LON und LAN“ habe ich vier Verbindungensverfahren mit „Router iLON 10/100/1000“, „NSI“, „LON to IP Gateway“ und „OSGi Gateway“ zwischen LON-Busses und LAN-Bus vorgeschlagen. Dafür gibt es abschließend ein Beispiel-Szenario für ein LON-Netzwerk System und das Verbindungsverfahren zu einem Ethernet-LAN.
Im zweiten Teil werden Basiskenntnisse über das Rechnernetzprojektierungstool CANDY und die Rechnernetzbeschreibungsprache NDML gegeben.
In Kapitel 7.3 “Gebäudeautomationsnetz in CANDY” werden die Vorschläge von Gebäudeautomationsnetzdesigen gegeben.
Und in Kapitel 7.4 “Szenariobeschreibung mit NDML” werden ein Beispielszenario durch NDML 3.0 dokumentorientiert und beschrieben. Gegen der früheren Dokumenten, die in Diplomarbeit „Optimierung der Fachsprache NDML mit Abbildung in eine XML-basierte Datenbank“ von Robert Hänel gemacht werden, haben die XML-Dokumente und seine XML-Schemen in meinem Beleg viele Unterschiede.
An Ende des Belegs werden einige Vorschläge zur programmtechnischen Realisierung des Designs von LON-Bussystemen innerhalb von CANDY gegeben.

1. Übersicht von den Automatisierungsnetztechnologien

1.1 Automationstechnik

Die Automatisierungstechnik ist ein Forschungsfachgebiet, das die Konzipierung und Entwicklung von Automaten und anderer automatisch ablaufender Vorgänge umfasst.
Die moderne Automation umfasst zwei Hauptteile: Software und Hardware.
Die Steuerung/Überwachung von technischen Prozessen kontrolliert durch die Software auf den Rechnern ein ganzes Automationssystem.
Die Sensoren sind die Hardware, die die Informationen aus der Umgebung im Automationssystem eingeben und die Aktoren sind die Hardware, die im Gegensatz zu Sensoren die Informationen von dem System ausgeben und z. B. die Motoren der Geräte (Ventile usw.) aktivieren, damit die Automationskontrollierung realisiert wird.
Außerdem kann dieses System durch Software über ein Kommunikationsnetz mit entfernten PCs Daten austauschen. Dadurch kann das Automationssystem von den entfernten PCs gesteuert werden.
Die Struktur dieses softwaregesteuerten Systems wird als Bild 1.1[1] gezeigt:

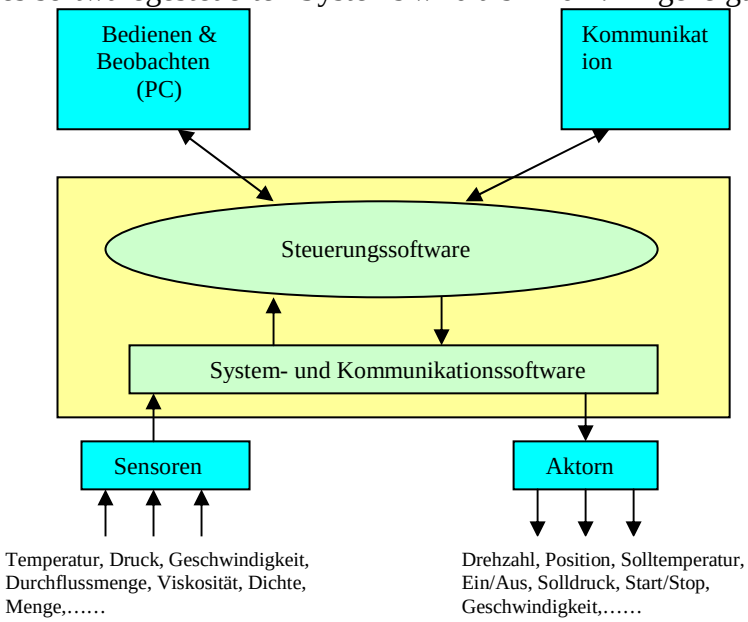


Bild 1.1 Steuerung eines technischen Prozesses durch Software

1.2 zeitliche Skalierung der Automationstechnik

Die Architektur von Kommunikationssystemen (dezentrale Systeme) hat sich in der Historie in Form vier zeitlichen Phasen: “Vorzeit“, „erstes Zeitalter“, „zweites Zeitalter“ und „drittes Zeitalter“ entwickelt.

Vorzeit: Programmierbare Steuerungen

Die softwaregesteuerte Automation entstand in den 60-er Jahren[1], damit werden die Steuerungssysteme nicht mehr direkt an Relais verknüpft, sondern führen durch Einführung einer programmierbaren Maschine die Automation aus. Die **SPS** (speicherprogrammierbare Steuerung), die im Kapitel 3.1.4 „SPS“ detailliert geschrieben wird, wurde geboren. In dieser

Phase war jede SPS ein einzelner isolierter Steuerungspunkt im technischen Prozess. Die Kommunikation und Synchronisation zwischen den selbständig arbeitenden SPS wurde mit digitalen Signalen (oder analogen Signalen) realisiert. Dabei gibt es hunderte digitale Ein-/Ausgänge in der SPS-zu-SPS-Kommunikation.

Erstes Zeitalter: Proprietäre Kommunikation

Im ersten Zeitalter der Industrieautomation (ca. 1975 - 1985)[1] wurde eine **serielle Punkt - zu – Punkt** Verbindung zwischen den SPS oder zwischen den SPS und den übergeordneten Systemen verwendet (Bild 1.2.1). In den SPS-Programmiersprachen wurden **die globalen Beschreibungseinheiten** eingeführt. Diese Einheiten beschreiben die Ereignisse und deren Attribute in den Formen von Bit, Bytes oder Worten, die von allen SPS standardisiert und erkannt werden.

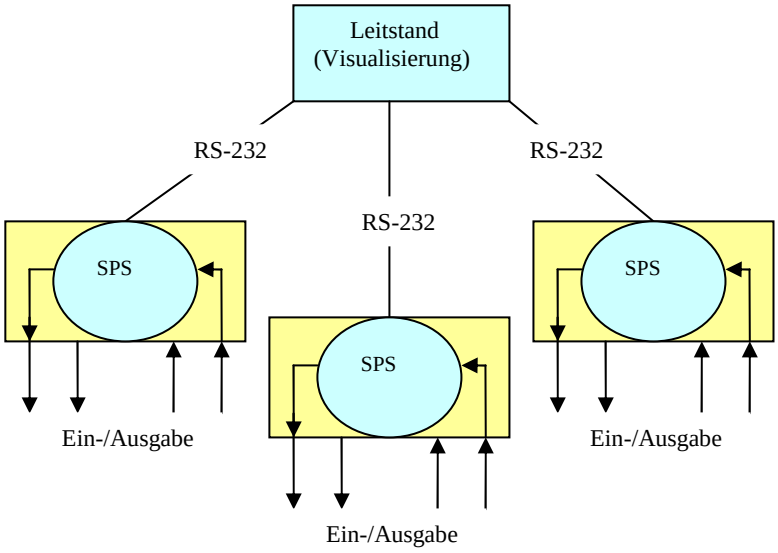


Bild 1.2.1 Proprietäre Kommunikation

Die seriellen Verbindungsverfahren beruhten entweder auf den **RS-232**, den **20 mA-Stromschleifen** oder den **RS-422** Standards[1].

Zweites Zeitalter: Bus-Hierarchien

Im Jahre 1984 (BITBUS durch INTEL)[1] wurde **eine serielle Kommunikation in Echtzeit über ein gemeinsames Kommunikationsmedium (=Bus)** zwischen Steuerung erstmalig vorgestellt. **Die kurzen Antwortzeiten** (**typisch: 2 … 5ms pro Meldung mit 50 Nutzdatenbytes**)[1] erlaubten es, diese Kommunikationsverfahren in schnelleren **(rechtzeitigen)** Prozessen einzusetzen, um Prozessdaten, Synchronisationstelegramme und Zustandsinformationen miteinander auszutauschen.
Es entstand eine Bus-Hierarchie (Bild 1.2.2)[1].

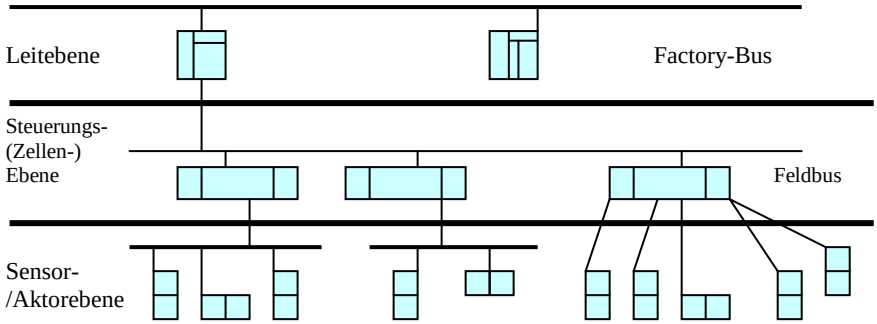


Bild 1.2.2 Bus-Hierarchie

Diese Hierarchie umfasste mindestens die drei Ebenen:

i. Leitebene (= Ebene der übergeordneten Systeme)
ii. Steuerungsebene(= Ebene der Echtzeit-Steuerungen)
iii. Sensor-/Aktorebene (= Ebene der physikalischen Schnittstellen zum Prozess)

Die Softwarearchitektur in diesem Zeitalter ist die Struktur **„Client-/Server“** (Bild 1.2.3[1]). Die Server steuern den Prozess (z.B. Anlagenteilsteuerung, Koordinierung, usw.) und bieten die Prozessinformationen an dem Client „Leitsystem“ und dem Client „Produktionssteuerung“. Das „Leitsystem“ betreut und koordiniert die Server, um die Anlagen im Gesamtsystem zu kontrollieren. Das „Leitsystem“ ist nicht der einzige Client in der Anlage, sondern zwei weitere Clients[1]:

i. eine Visualisierungsstation
ii. ein Qualitätssicherungssystem

Die „Visualisierungsstation“ fordert von den Servern die Zustandsinformationen an und stellt diese bedienerfreulich auf seinem Bildschirm dar.
Der Client „Qualitätssicherungssystem“ holt sich von den Servern die benötigten Informationen, kombiniert und überprüft diese Informationen und archiviert sie anschließend.

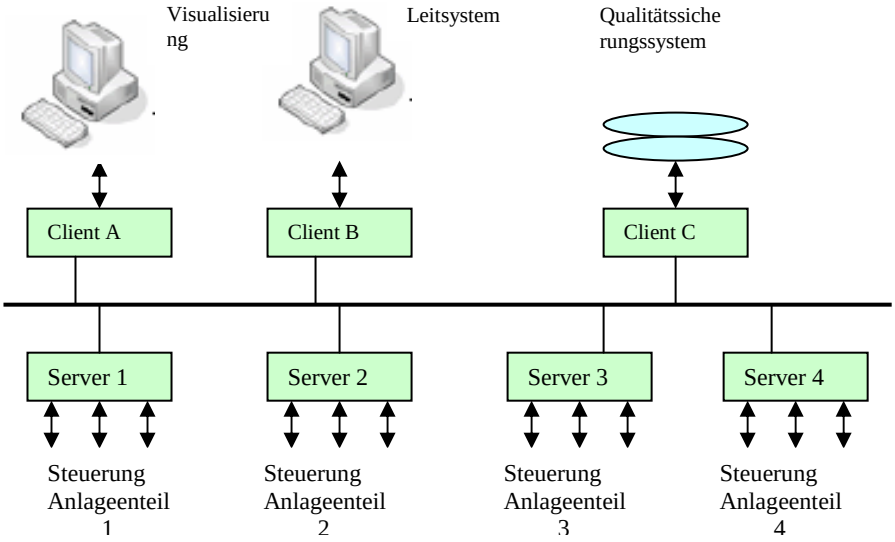


Bild 1.2.3 Client-/Server Softwarearchitektur in der dezentralen Steuerung

In diesem Zeitalter werden die Gebäudeautomatisierungen erstmalig vorgestellt. Dafür wird GLT (Gebäudeleittechnik) entwickelt, die im Kapitel 3.2.1 „GLT“ detailliert beschrieben wird. GLT kombiniert aus den getrennten Clientsystemen „Visualisierungsstation“ und „Qualitätssicherungssystem“ einen komplexeren Client.

Drittes Zeitalter: Systemweite Middleware

Auf der Basis des „Zweiten Zeitalters“ werden zwei grundlegenden Konzepte: Middleware und Komponentenmodelle, in dieser Phase eingeführt.

Middleware

Die Middleware ist eine zusätzliche Systemsoftwareschicht, durch die ein Steuerungsprogramm plattformunabhängig ist. Die Dienste der Middleware können von allen Komponenten im System benutzt werden.
Zur Zeit sind noch keine einheitliche Standards und Produkte für Middleware vorhanden. Die aussichtsreichsten Kandidaten sind **CORBA** (Common Object Request Broker Architecture) und **OMG** (Object Management Group).

Komponentenmodelle

Das Komponentenmodell ist ein Modell, bestehend aus allen Softwarekomponente im System (z.B. Steuerungsprogramme, Visualisierungssysteme, Sensor-/Aktortreiberprogramme, usw.), das standardisiert wird und plattform-/herstellerunabhängig ist.
Mit dem Komponentenmodell können die Komponenten die Nachbildungen von Objekten (Sensoren, Aktoren, Regelungsschlaufen, Visualisierung etc.) und ihre Beziehungen auf dem Bildschirm abstrahieren. Dadurch wird das gesamte Automatisierungssystem modelliert, über Monitore gesteuert, gewartet und erweitet.
Jede Komponente wird über ein Interface, das z.B. durch VB, VC++, Java, C++ entwickelt wird, mit der systemweiten Middleware verbunden.

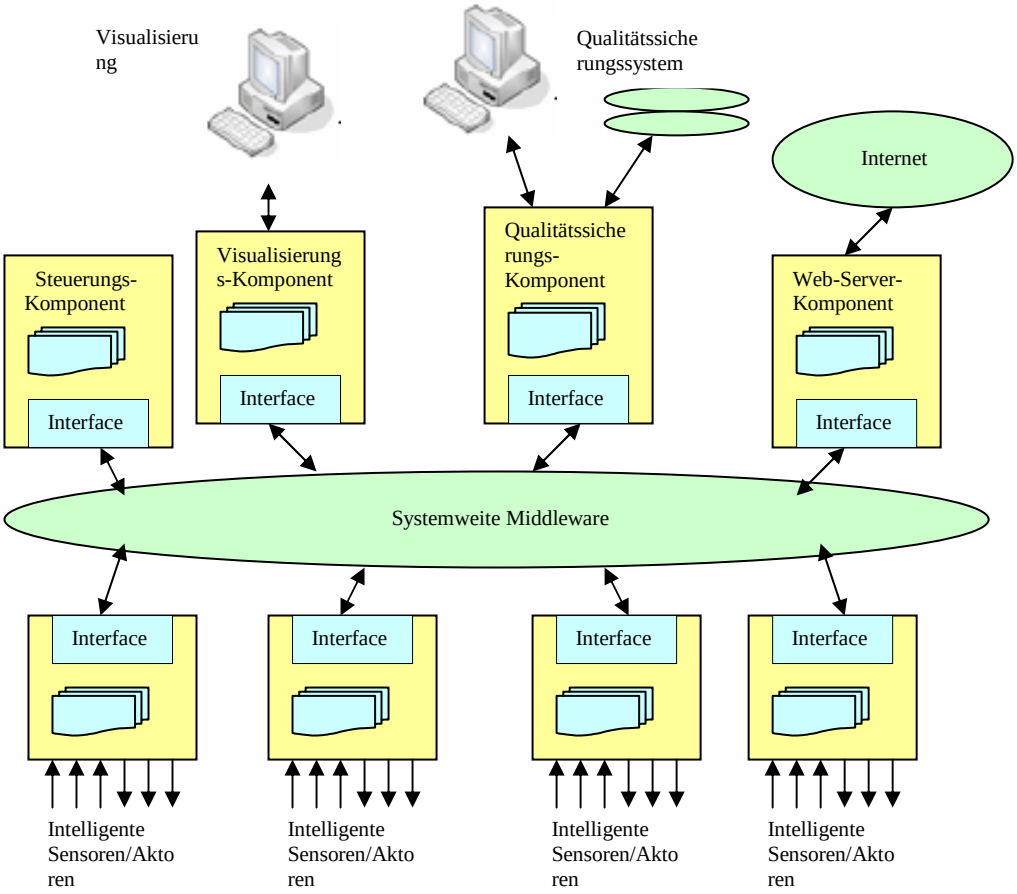


Bild 1.2.4 Automation mit Systemweite Middleware zwischen Komponenten

2. ein Überblick der Gebäudeautomatisierung

Gebäudeautomatisierung ist ein wichtiges Gebiet von den Automatisierungstechnologien. Wie in den Kapitel 1.2.3 und 1.2.4 genannt, befindet sich die Gebäudeautomation als Automationstechnik im zweiten Zeitalter: Bus-Hierarchien oder ist im dritten Zeitalter: Systemweite Middleware einzuordnen.
In diesem Bereich kann man Beleuchtungssysteme, Heizungen, Lüftungsanlagen oder Klimaanlagen nach den Anwenderforderungen (z.B. Beleuchtzeit, Gradation der Helligkeit) zeitgerecht schalten und steuern, um Energien zu sparen. Man steuert zeit- und bedarfsgerecht die Verschattungseinrichtungen in den Jalousien abhängig von Sonnenlicht und Wind, um die Raumumgebung immer optimal zu erhalten. Man kann das Gebäudeautomationstechniksystem im Gebäude einführen, damit die Sicherheit des Gebäudes durch die Überwachung von Fenster-/Türkontakten und von Bewegungsmeldern und das Zutrittskontrollsystem realisiert wird. Die Verbrauchsdaten von Wärmezählern, Wasserzählern, Gaszählern und Stromzählern können über Gebäudeautomationsgeräte erfasst werden. In der Gebäudeautomation werden alle Steuerungsvorgänge im Gebäude zentral erfasst und angezeigt und über das Telefonnetz, das Internet oder die Funk-/Infrarotfernbedienung fernüberwacht und ferngesteuert. Durch diese Kommunikationen kann man auch die Weißware (Haushaltsware, z.B. Mikrowellenherd, Kühlschrank…) mit Notebooks oder Handys im „Outdoor“ entfernt steuern.
Die **Vorteile** der Gebäudeautomation sind:

i. Energieverbrauchsreduzierung durch die optimale Automationsverwaltung
ii. Wohnkomforterhöhung durch intelligente Steuerung: z.B. kann auf einen Tastendruck eine Beleuchtungssituation vordefiniert werden. Damit werden die Leuchten entsprechend der Gewohnheiten von den Bewohnern kontrolliert.
iii. Sicherheit für die Bewohner durch Alarmierung beim Auftreten von kritischen Situationen (z.B. Feuer) und Einbrüche
iv. Zentrale Überwachungsmöglichkeit durch eine Zentrale mit automatischer Alarmweiterleitung an das technische Personal
v. Personalkostenreduzierung mit Facilitymanagement über verteilte entferne Netzwerk

Die **Nachteile** der Gebäudeautomation sind:

i. höhere Anschaffungskosten im Vergleich zur normalen Gebäudeinstallation. Gegenüber der normalen Gebäudeinstallation werden die Kosten nur durch die Energieeinsparungen im Betrieb reduziert, zum anderen sind viele Funktionen von Gebäudegeräte(z.B. DDC, LON-Geräte) sehr viel teurer.
ii. Bei hoher Komplexität ist für den Betrieb der Anlagen qualifiziertes Personal notwendig.
iii. Erhöhte Abhängigkeit von den Anlage-Herstellern (z.B. DDC (Direct Digital Control) Hersteller) der Anlagen. Weil verschiedene Hersteller mit unterschiedlichen Kommunikationsprotokollen (z.B. für unterschiedliche Feldbusse) die vielfältigen Anlagen und Programme herstellen, die auf den Anlagen laufen, muss man am besten die Anlagen eines bestimmten Herstellers benutzen und für nachträgliche Erweiterung die Anlagen des gleichen Herstellers auswählen.

Wie die vorher genannte Automation in „zweitem Zeitalter“ erfasst man in der Gebäudeautomation (GA) die automatisierenden Überwachungs-, Steuerungs-, Regelungs- und Optimierungsfunktionen auf der Basis der Infrastrukturen im Gebäude (z.B. Feldbus-Kabel, Sensor, Aktor, DDC) zusammen. Die Funktionsabläufe werden automatisch nach vorgegebenen Einstellwerten (normale Weise schon von Herstellern dargestellt) durchgeführt. Alle Sensoren, Aktoren, Bedienelemente, Verbraucher und andere technische Einheiten im Gebäude werden miteinander vernetzt. Die Kernrollen in der Gebäudeautomation sind **die dezentrale Steuerungseinheiten (DDC)** und **die dazwischen vernetzten Bussysteme**.
Die Gebäudeautomation wird in drei Ebenen (wie Bild 2.1) unterteilt: **die Feldebene, die Automationsebene und die Managementebene.** Durch die rasante Entwicklung der

Mikroprozessoren in den letzten Jahren wird die bisherige klassische Aufteilung von **Feld-** und **Automationsebene** immer mehr vermischt. D.h. die intelligente Sensoren und Aktoren können auch direkt an die Feldbusse angebunden werden (z.B. LON-Geräte LCN-Geräte, die in Kapitel 5.2.2 „LON-Bus“, 5.2.3 „LCN-Bus“ und Kapitel 6 „Erweiterungen von LON“ erklärt wurden).

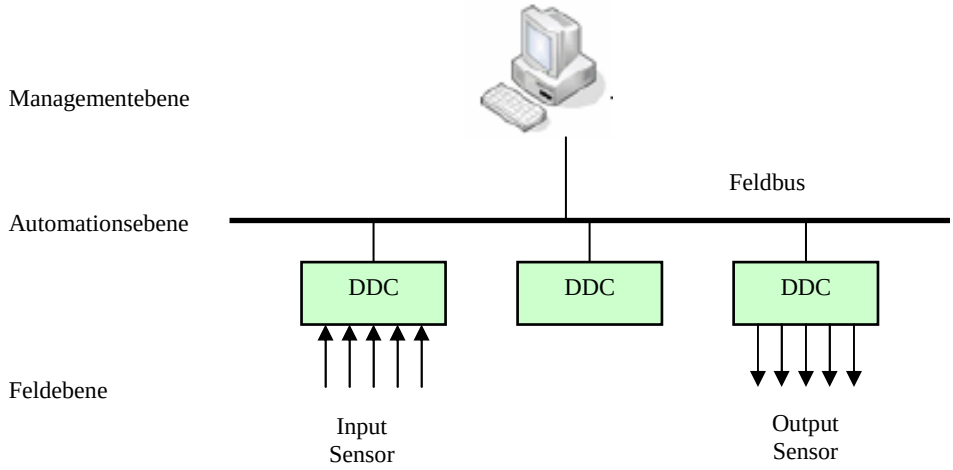


Bild 2.1 Gebäudeautomationsebene

2.1 Gebäudeautomationsebene

Feldebene

In dieser Ebene werden die Sensoren und Aktoren mit den DDCs über oder ohne Stichleitungen verkabelt. **Die Sensoren können in der Regel direkt mit den Eingängen der DDC verbunden werden.** Bei Sensoren werden die Signale, welche die Messwerte von den Umgebungen tragen, über den analogen oder digitalen Eingängen im intelligenten Gebäudeautomationsgerät überträgt. Dagegen wirken sich die Aktoren mit den Sollwerten von den Ausgängen auf den Umgebungen aus. (Diese analogen oder digitalen Ein-/Ausgängen werden im Kapitel 3.1.3 „DDC“ detailliert erklärt.)

Automationsebene

Für den Datenaustausch zwischen den DDCs auf der Automationsebene gibt es viele proprietäre Bussysteme (z.B. LCN, LON…) aus unterschiedlichen Herstellern. Dafür ist Interoperabilität zwischen den Herstellern sehr wichtig.
Die **Interoperabilität** bedeutet hier die Benutzung der DDCs verschiedener Hersteller durch ein offenes System. Sie ist seit Jahren eines der zentralen Themen in der Gebäudeautomation. Aber die Offenheit und Herstellerunabhängigkeit ist nicht wie bei dem Open-Source von LINUX. In der Gebäudeautomation bedeutet die Herstellerunabhängigkeit nur, wenn man ein System installiert, es eine Möglichkeit gibt, die Geräte (DDCs) mehrerer Hersteller ohne größere Probleme miteinander kommunizieren zu lassen. Eigentlich ist die Abhängigkeit von Herstellern noch nicht gelöst.
Aus den Automationsfeldbussen werden insbesondere die herstellerübergreifenden Bussysteme **BACnet** und einfachere **LON (Local Operating Network)** und **Konnex** in der Gebäudeautomation verwendet. **BACnet und LON wird von großen Unternehmen**[2] **eingesetzt,** um die größer Gebäudeanlagen wie z.B.Hotels, Bürohäusern, Kliniken oder Flughäfen mit einem echt offenen Standard zu verwalten. **EIB/KNX** erfüllt **die europäische**

Norm EN 50090 und wird aufgrund der einheitlichen, herstellerübergreifenden Geräte und darauf laufender Software **hauptsächlich im privaten Wohnungsbau**[2] **eingesetzt.**
Außerdem werden **Funksysteme** auch in der Gebäudeautomation eingesetzt, die von vielen Feldbus-Protokollen (z.B. BACnet, LON) unterstützt wird und sehr gut für Nachrüstungen sind, weil keine neue Kabel gebraucht werden.
Als eine Alternative für DDCs werden **SPSs** in der Gebäudeautomatisierung eingesetzt. Eigentlich werden SPSs hauptsächlich in der Industrieautomatisierung anwendet. DDC ist gegenüber SPS vereinfacht und für die Gebäudeautomation spezialisiert. Die Funktionen sind ähnlich wie SPS. Dafür werden die Details in Kapitel 3.1.3 „DDC“ und 3.1.4 „SPS“ erklärt.

Managementebene

In der Managementebene werden die Anlagen überwacht, gesteuert und für den Betreiber visualisiert. Die Funktionen werden von einer speziellen und herstellerabhängigen Software realisiert, die **GLT** (Gebäudeleittechnik) genannt wird. **Die herstellerunabhängigen Schnittstellen** für Managementsysteme sind hier **OPC(im Windows)** und **OSGi(im Java)**, die beide in Kapitel 3.2.2 „OPC“ und Kapitel 3.2.3 „OSGi“ erklärt werden.
Die Software GLT wird von den DDC-Herstellern zusammen mit den DDCs angeboten und die Details werden im Kapitel 3.2.1 „GLT“ beschrieben.
Für Ankopplung der unterschiedlichen herstellerabhängigen Anlagen mit proprietären Bussystemen gibt es eine Möglichkeit, Es ist möglich, **Gateways** einzusetzen.
Für das Facilitymanagement kann das Gebäudeautomatisierungssystem über ein Gateway oder einen Router mit dem Rechnernetz verbunden werden.

2.2 Die Bestandteile beim Aufbau eines Systems zur Gebäudeautomation:

Für den Aufbau eines Gebäudeautomatisierungssystems gibt es viele Auswahlmöglichkeiten mit unterschiedlichen Netzaufbauverfahren und -bestandteilen. Darunter ist nur ein Beispiel zu erklären, welche Bestandteile in der Gebäudeautomation eingesetzt werden können:

i. Schaltschrank
ii. Steuerungseinheiten DDC
iii. Feldgeräte, wie Sensoren und Aktoren
iv. Kabel und Bussysteme
v. Server und Gateways
vi. Gebäudeleitsystem (Software auf dem Leitrechner zur Visualisierung der Systeme)

2.3 Feldbussysteme für Gebäudeautomation:

Zur Zeit gibt es noch keinen einheitlichen Standard für die Feldbussysteme, die in den Gebäudeautomationen verwendet werden. Von verschiedenen Organisationen und Kommissionen werden die unterschiedlichen Standards und Protokolle für Feldbustechniken dargestellt. Die Techniken darf man nebeneinander nicht gleichzeitig in einem Gebäudeautomatisierungsnetzwerksegment einsetzen. Weiter unten sind nur einige Feldbusvarianten aufgelistet. Im Kapital 5 werden diese Feldbusse weiter vorgestellt.

i. Local Operating Network (LON),
ii. Europäischer Installationsbus (EIB), KNX
iii. BACnet,

iv. Controller Area Network (CAN)
v. Profibus,
vi. Interbus,
vii. Local Control Network (LCN)

2.4 Hausautomatisierung

Hier wird zusätzlich bemerkt, dass Hausautomation/Heimautomation ein Teilbereich von Gebäudeautomation ist, die speziell auf die Gegebenheiten privater Wohnhäuser bezogen und an den speziellen Bedürfnissen seiner Bewohner orientiert ist. Viele der in der Gebäudeautomatisierung verwendeten Techniken und Geräte können auch in der Hausautomatisierung verwendet werden. Aber es gibt noch Unterschiede in den Automationen zwischen Gebäudeautomation und Hausautomation. Der Hauptunterschied ist: Bei Gebäudeautomation wird der Schwerpunkt auf effiziente Verwaltung der Infrastrukturen im Gebäude (z.B. Facilitymanagement) gelegt. Bei Hausautomation sind die Schwerpunkte auf Wohnkomfort und Benutzfreundlichkeit des Automationssystems gelegt.

3. Erklärungen von einiger wichtiger Komponenten in Gebäudeautomationstechnik

In diesem Kapitel werden die Grundkomponenten der Gebäudeautomationstechnik in zwei Teile unterteilt.
Ein Teil beschreibt die physikalischen Geräte, die als Knoten oder Unterstationen so genannt werden, z.B. DDC, SPS, und die Slaves von den Knoten. Z.B. werden Sensoren und Aktoren auch in diesem Teil erklärt. In anderem Teil wird die Software erklärt, z.B. GLT, OPC, OSGi… mit der man visuell die physikalische Infrastrukturen überwachen, kontrollieren und mit dem Internet oder Intranet eine entferne Verwaltung realisieren kann.

3.1 Hardware in Gebäudeautomationstechnik

Das Gebäudeautomationssystem besteht aus vielen Hardwarekomponenten. Auf der Feldebene sind die Sensoren und Aktoren, welche Informationen von der Umgebung zu den Steuerknoten ein- und ausgeben. Auf der Automationsebene sind die Knoten wie DDC, SPS… die programmierbaren Geräte für die Automation. DDC wird insbesondere in der Gebäudeautomation angewandt.
SPS ist normal für die Industrieautomation. Im Vergleich mit SPS ist DDC einfacher. SPS kann auch eine Alternative von DDC sein. Für einige Feldbusse gibt es spezielle Geräte. Z.B. wird ein LON-Gerät (dezentral, programmierbar) von LonMark spezialisiert im LON-Bus angewandt. Seine Funktionen sind gleich wie DCC und SPS. Auf der Managementebene ermöglichen PCs das ganz Netzwerk zu verwalten und zu visualisieren. Natürlich gibt es auch Kabel, Transceiver, Ankoppler, usw.
Zu ersten werden in diesem Kapitel die Eingabe- und Ausgabeknoten von Umgebungsinformationen „Sensor“ und „Aktor“ einfach vorgestellt. Danach werden DDC und SPS, die die Kerne vom Gebäudeautomationsnetzwerksystem sind, erklärt.

3.1.1 Sensoren

Ein **Sensor (Gefühl)** oder **(Mess-) Fühler** hat „die Aufgabe, Informationen über den Prozesszustand abzufragen und an die Automatisierungsgeräte weiterzuleiten“[3]. Die Unterschiede zwischen Sensor und Messgerät sind: **Der Sensor** misst die Werte aus den Umweltbedingungen, aber **das Messgerät** misst nicht nur, sondern verarbeitet auch die Messwerte aus den Umweltbedingungen. Hier werden die Beide „Sensor“ genannt.
Sensoren haben folgende Aufgabe:

i. Umwandlung physikalischer Größe (z.B. Wärmestrahlung, Temperatur, Feuchtigkeit, Druck, Schallwechseldruck, Schall, Helligkeit, Magnetismus, Beschleunigung, Kraft) in analoge oder digitale Signale
ii. Bearbeitungen der Informationen vor der Eingabe der Signale im Automatisierungsgerät
iii. Analog-Digital-Umwandlung
iv. Verstärkung der Signale vor der Eingabe der Signale im Automatisierungsgerät

Die Sensoren darf man in zwei Gruppen unterteilen - die aktiven Sensoren und passiven Sensoren:

i. Aktive Sensoren: Geben Signale durch direkte Veränderungen einer Spannung oder eines Stroms in den digitalen Eingängen ab, wenn sich die gemessenen Zuständen ändern. Sie besitzen eigene Energieversorgungen.

ii. Passive Sensoren: Ändern die Größe der elektrischen Signalen (z.B. Widerstand des Dehnungsmessstreifens abhängig von der Dehnung), ohne Berücksichtigung der Beeinflussungen der Energieumwandlung.

In der Automationstechnik dienen Sensoren als Signalgeber. Die Sensoren können nicht nur analoge Signale, sondern auch digitale Signale in Automatisierungsgeräte eingeben. In den letzten Jahren werden die intelligenten Signalverarbeitungen durch die Einbettungen von Mikroprozessoren oder Mikrosystemen im intelligenten Sensor realisiert wie Bild 3.1.1.1.

Bild 3.1.1.1 LS-T01 LON - Temperature Sensor

Dieses Beispiel ist ein intelligenter Temperatur-Sensor mit LonWorks-Protokoll, der ein Produkt von der Firma „CAPELON“ ist. Das Detail des Produkts kann man in „http://www.capelon.se/english/prodlst01.htm“ anfragen.

3.1.2 Aktoren

Die Aktoren beeinflussen die Ausführungselemente (z.B. Motoren, Ventile usw.) des Prozesses. Sie entnehmen die Werte aus den Steuergeräten und wirken damit auf den technischen Prozess ein.

Damit realisieren die Sensoren die folgenden Wirkungen im Prozess:

i. Erzeugung von Linear- und Drehbewegungen mit Motoren
ii. Schaltung von Ventilen, um die Stoffflüssen zu steuern
iii. Ansteuerung von optischen Signalgeber
iv. Ansteuerung von elektronischen Leistungsschaltern, um z.B. Heizung, Kühlschrank, usw. zu kontrollieren.

Die Ausgabe der Signale zur Ansteuerung der Aktoren sind wie Sensoren auch analoge oder digitale Signale.

Gleich als Sensoren werden Mikroprozessor und Mikrosystemen auch in den neuen Aktoren einbettet. Dadurch können die intelligenten Aktoren ohne Kontroller mit den Sensoren im Regelkreis funktionieren.

Bild 3.1.2.1 ILONA-S Intelligent LON Aktor

Dieses Beispiel ist ein intelligenter linearer Temperatur-Aktor mit LonWorks-Protokoll, der ein Produkt von der Firma „Honeywell“ ist. Dieser Aktor kontrolliert die Heizungen mit kaltem

oder warmem Wasser. Das Detail des Produkts kann man in „http://www.lonmark.org/news/in1099_prod.htm“ anfragen.

3.1.3 DDC(Direct-Digital-Control)

Eine DDC (Direct-Digital-Control) ist ein intelligentes Gerät, in dem ein Mikroprozessor eingesetzt wird, der für Steuerungs- und Regelungsaufgaben speziell in der Gebäudeautomatisierung eingesetzt wird. Die DDC ist gegenüber der SPS vereinfacht. Im Unterschied ist das Hauptanwendungsgebiet von der SPS die Industrieautomatisierung.

Hardware

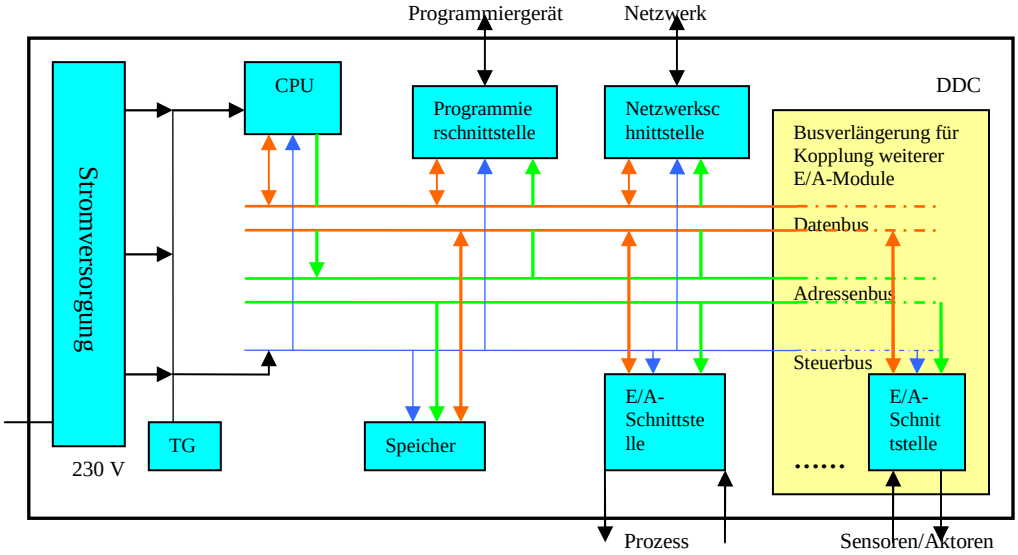


Bild 3.1.3.1 ein logischer Aufbau eines DDC/SPSs

In Bild 3.1.3.1[4] wird ein logischer Aufbau einer DDC gezeigt: Eine DDC besitzt eine CPU, ein internes Bussystem und einen Speicher.
Das interne Bussystem besteht aus Adress-, Daten- und Steuerbus. Über den bidirektionalen Datenbus werden die Daten zwischen CPU und Speicher sowie zwischen Ein-/Ausgabe-Registern durch Beschreiben oder Auslesen ausgetauscht. Die genaue Adressierung der Register oder Speicherzellen erfolgt über den unidirektionalen Adressenbus, der nur von der CPU getrieben wird. Der Steuerbus dient zu der Synchronisation des Datenaustausches.
Im Speicher sind die lokale Aufgaben - das Echtzeit-Betriebssystem, das Anwendungsprogramm, das Prozessabbild (die aktuellen Werte für die Informationspunkte) und die lokale Variablen für die Anwendung. Weiterhin wird auch noch Software zur Kommunikation und zur Programmierung (sind von Herstellern) als Software abgelegt. Diese Software ist an den gewünschten Ablauf von den zu steuernden Maschinen oder Anlagen anzupassen. Damit kann DDC unabhängig von der Steuerungsaufgabe sein.
Hier wird nur eine normale DDC dargestellt, z.B. in dieser DDC gibt es nur eine CPU. Entsprechend verschiedener Feldbus-Protokolle (z.B. EIB, BACnet, usw.) sind die Strukturen von DDCs etwas verändert.
In einigen Feldbussystemen werden DDCs mit anderen Namen genannt, z.B. In LON-Bussystem wird DDC mit drei CPUs als LON-Gerät genannt und in LCN-Bussystem wird DDC mit einer CPU als LCN-Modul genannt.
Die Anzahl der Eingänge und Ausgänge jeder DDC, mit der die Sensoren und Aktoren an DDC angekoppelt werden, ist abhängig von den unterschiedlichen Typen, die von Herstellern angeboten werden.

Die Eingänge und Ausgänge werden für analoge und digitale Signale unterschieden.
Eingänge:

1. Digitale Eingänge zur Erfassung von Schaltvorgängen(z.B. zu Relais) mit einer Spannung von **24V DC** (Direct Current/Gleichspannung).
2. Analoge Eingänge zur Erfassung von Sensorwerten oder externen Steuersignalen mit einer Spannung von **0V** bis zu **10V** oder einem Strom von **0mA** bis zu **20mA**. Damit können verschiedene Sensortypen genutzt werden.

Ausgänge

1. Digitale Ausgänge zum direkten Schalten von **220 V**
2. Analoge Ausgänge mit einer Spannung von **0V** bis zu **10V** Signal, um Ventilen über den Aktoren zu steuern

Darunter gelegenes Beispiel ist eine DDC, die im BACnet-System eingesetzt wird. Ihr Name ist „DDC-Zentrale DDC3500-BACnet“ (Bild 3.1.3.2) und wird von „Kieback & Peter GmbH & Co KG“ hergestellt. Das Detail des Produkts kann man in „http://www.big-eu.org/catalog/show.php?company_id=9&company_name=Kieback & Peter GmbH & Co KG“ anfragen.

Bild 3.1.3.2 DDC 3500 BACnet

Programmierung

Im DDC werden Informationen mit binärer und analoger Eingabe gewonnen, vom Anwendungsprogramm verarbeitet und denn an Ausgänge ausgegeben. Diese Prozesse sind zyklisch vom Betriebssystem organisiert. Wie Bild 3.1.3.3[4], die Informationsverarbeitung im Anwendungsprogramm nutzt grundsätzlich lokale Variablen und muss nur die von den Eingabe-Baugruppen bereitgestellten Datenformate berücksichtigen und die Verarbeitungsergebnisse in dem geforderten Format den Ausgabe-Baugruppen zur Verfügung stellen. Die Verarbeitungsergebnisse werden in den Speicher geschrieben, um durch das Betriebssystem zyklisch die binären und analogen Ausgänge zu setzen. Die Ausgabebaugruppen dienen der Ansteuerung von Schaltung- und Stellungseinrichtungen und sind Komponenten für die Informationsnutzung.

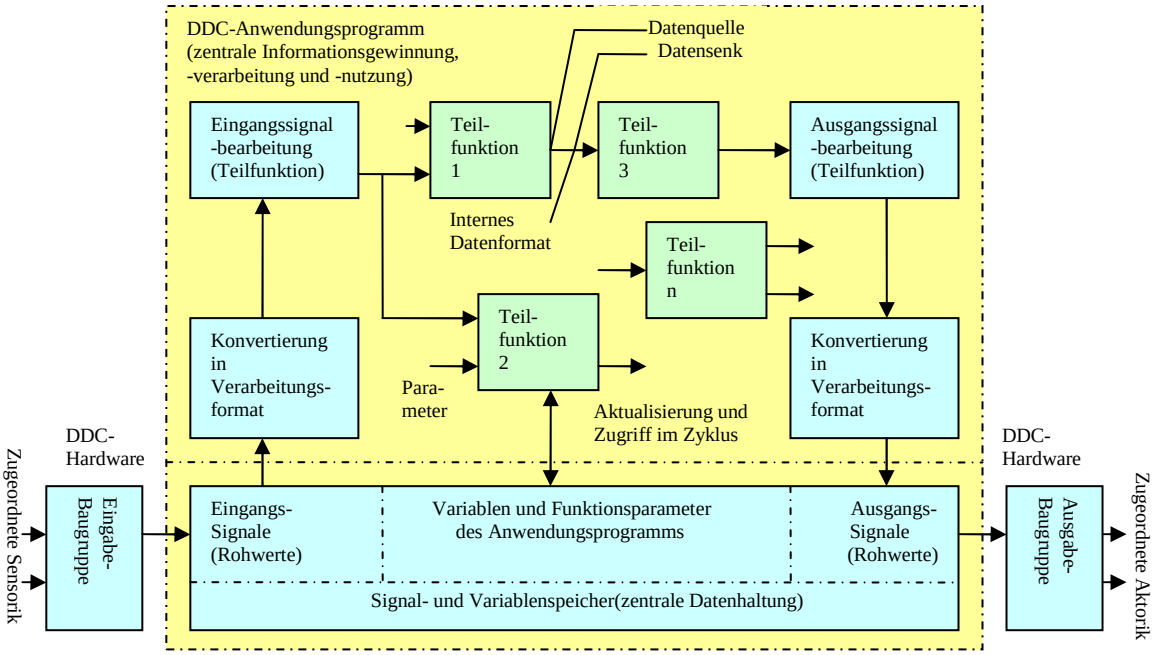


Bild 3.1.3.3 ein Informationsverarbeitungsprozess in einem DDC-Anwendungsprogramm

Die interne Informationsverarbeitung basiert typischerweise auf vom **IEC** (International Electrotechnical Commission) genormten und zusätzlichen herstellerspezifischen Bibliotheksfunktionen, deren Befehlsvorrat für die Realisierung eines breiten Spektrums prozessgebundener Automatisierungsaufgaben ausreichend ist. Das gesamte Programm, das eine ganze Automationsaufgabe auf DDC ist, wird in den zahlreichen Teilfunktionen verteilt. Durch die Verknüpfung einer Datenquelle mit einer Datensenke - beide sind über gleiche Datenformate verbunden – werden die Variablen zwischen den Teilfunktionen bearbeitet und überträgt. Der gesamte zyklische Prozess ist realisiert.
Derartige Programme können in vorkonfigurierten DDCs fest implementiert sein. Der Anwender ordnet nur die Ein- und Ausgänge der Sensoren und Aktoren zu und aktiviert die entsprechend gebäudetechnischen Aufgaben. Durch Parametrieren werden die erforderlichen Parameterwerte in den implementierten Teilfunktionen eingestellt.

Kommunikation

Die DDCs kommunizieren miteinander über den Feld-Bussystemen mit unterschiedlichen Protokollen z.B. LON, BACnet, Profibus und CAN. Wegen der Technikbegrenzungen zwischen den unterschiedenen Feldbussen muss beachtet werden, dass die Kommunikation zwischen den Fabrikaten getestet werden sollte, da die Standards nicht die Interoperabilität garantieren können. Mittels der Gateways ist es dann möglich, die Interoperabilität der proprietären Systeme zu öffnen und verschiedene Fabrikate in einem Gebäude zu realisieren.
Als Nutzerinterface zu den Automationssystem hat sich in der Gebäudeautomation die Gebäudeleittechnik (GLT), die in Kapitel 3.2.1 „GLT“ erklärt wird, etabliert. Dort werden die gesteuerten Prozesse durch entsprechende Visualisierungsmoduln grafisch dargestellt. Für die Anbindung von Gebäudeleittechnik wird OPC als Standard eingeführt.

3.1.4 SPS (Speicherprogrammierbare Steuerung)

Die Funktionen von SPS sind umfangreicher als bei DDC. Das wichtigste Anwendungsgebiet von SPS ist die Industrieautomatisierung. In der Gebäudeautomation ist SPS eine Alternative zu DDC. Hier wird SPS nur grob vorgestellt.
Die Arbeitsweise ist ähnlich wie DDC, die im Kapitel 3.1.3 „DDC“ schon genannt wurden.

Üblicherweise besitzt eine SPS 8 Ein- und Ausgänge[5], häufig ergänzt durch analoge Ein- und Ausgänge. Die sind nicht genug für die Praxis. Man kann die SPS direkt an der SPS mit steckbarem Modul erweitern, die lokalen Erweiterungen genannt werden. Oder durch Master-Slave-Module wird die SPS dezentral erweitet. In dieser Form ist die zentrale SPS ein Master. Die Slave-SPSs werden über ein Bussystem mit dem Master verbunden. Dazu gibt es die Grenzwerte, um die Slaves je Master und die lokale Eweiterungen je SPS anzuzahlen. Die Bussysteme sind häufig herstellerspezifisch. Durch ein Nachdenken von Kabel- und Gerätepreis darf man ein oder beide aus den zwei Weisen auswählen. Darunter ist Bild 3.1.4.1, das die zwei Erweiterungsmöglichkeiten von SPS anzeigt.

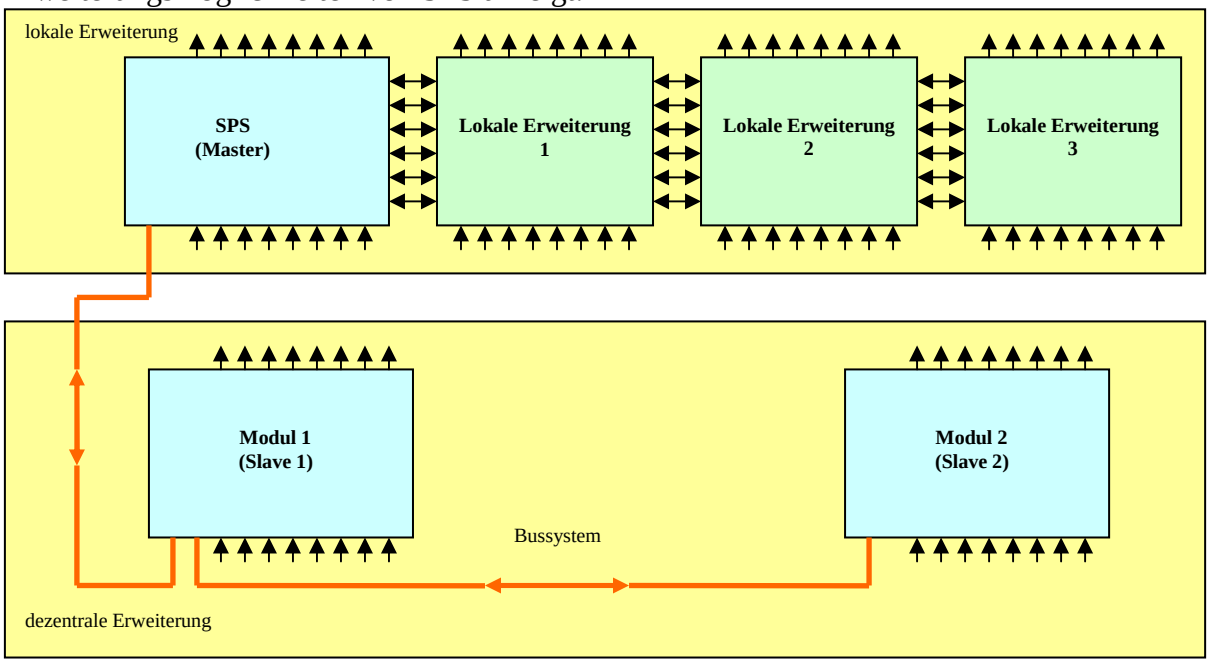


Bild 3.1.4.1 Erweiterung mit SPS

Moderne SPSs sind klassische Microcontroller mit eigener CPU und einer Basis-Software, die ebenfalls die Verknüpfung von E/A-Signalen erlaubten Die Basis-Software besteht aus einem Echtzeitbetriebssystem und SPS-spezifischen "Bausteinen", die SPS-Funktionen werden Zeitfunktionen und Schnittstellen zu Erweiterungsboards realisieren.
Wie Bild 3.1.4.2[5] im RAM wird der Maschinencode, d.h. die Befehle für die zentrale Verarbeitungseinheit (CPU) in der Reihe abgelegt und abgearbeitet. Es ist ein batteriegepufferte 32kByte – RAM. Dieser Vorgang ist ein Zyklus. Das Besondere ist, dass die einzelnen Schritte über die Ein- und Ausgänge außen nicht sofort sichtbar sind, sondern erst am Schluss eines Zyklus wenn alle Resultate und Entscheidung bearbeitet wurden. Deshalb sind manche Fehler in der Programmierung nicht sichtbar und schwer lokalisierbar.
Die Zykluszeit der SPS hängt vom Programm, von dessen Länge und Inhalt ab. Der Programmierer hat die Möglichkeit, die zeitliche Länge zu ermitteln und kann eine maximale Zykluszeit vorgeben.

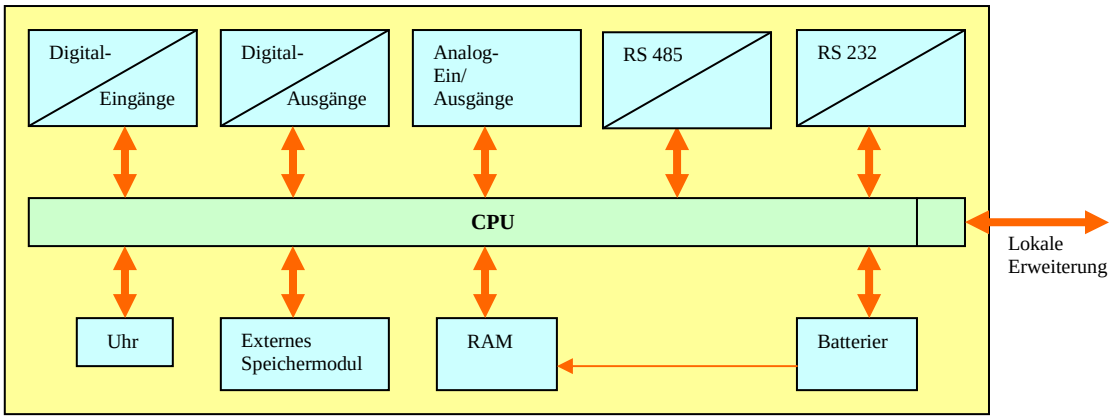


Bild 3.1.4.2 interner Aufbau einer SPS

Da alle Aktoren und Sensoren gleichzeitig die Daten übertragen, ist die Synchronisation eine wichtige Voraussetzung für viele Prozesse. Dafür unterstützt sie eine sekundengenaue Echtzeituhr.
SPS verlangt eine Gleichspannung von 24V: Natürlich ist eine entsprechende Stromversorgung 230V/24V[5] verfügbar.
Digitaleingänge sind gegenüber dem eigentlichen Gerät galvanisch entkoppelt. Sie besitzen eine untereinander gemeinsame Masse und werden gegen eine gemeinsame Spannung von 24V und einen typischen Strom 6mA[5] geschaltet.
Alle **Digitalausgänge** sind gegen die SPS entkoppelt, aber untereinander nicht. Die Versorgungsspannung für die Ausgänge beträgt 24V. Die Ausgänge sind nur mit 0.5A[5] belastbar, der Strom aller Ausgänge darf gleichzeitig 2A[5] nicht übersteigen.
Die **Analogeingänge** stellen 10Bit[5] Wörter mit einem Eingangsbereich von 0 bis 10 V[5] dar. Das bedeutet, der Eingangswert von 0V intern als ein Wort mit der Kombination „00000000 00000000“ und der Wert 10V als „00000011 11111111“ dargestellt wird. Gegen die Digitaleingänge sind diese Eingänge untereinander isoliert. Zu den Analogeingängen gehören auch die Sollwertgeber. Diese Steller können gut für Sollwertvorgaben verwendet werden, wie z.B. Temperaturgrenzen…
Ein **Analogausgang** besitzt einen Bereich von 0 bis 10V[5], stellt 12Bit[5] Wörter. Das Datenwort „00000000 00000000“ stellt 0V dar, „00001111 11111111“ 10V.
Durch **eine standardisierte Schnittstelle vom Typ RS-232**[5] kann SPS mit einem Programmiergerät z. B. PC verbunden werden. Damit kann man programmieren, Änderungen des laufenden Programmes durchführen oder die SPS per Programm dazu bringen, Daten auszugeben und einzulesen. Die SPS kann auch durch Software mit dem Programmiergerät kommunizieren. Die Programmierung geschieht über dieses Programmiergerät, zum Beispiel eine Anwendung unter Windows/Linux auf einem PC oder ein zugeschnittenes System. Die hier bereitgestellte, sogenannte Konfiguration wird beim Programmieren auf die Steuerung geladen. Sie bleibt dort solange im Speicher(RAM), bis sie gelöscht oder überschrieben wird.

3.2 Software in Gebäudeautomationstechnik

Die ganzen Funktionen der Gebäudeautomation werden durch Software-Komponenten realisiert.
GLT(Gebäudeleittechnik) ist ein Interface zwischen Person und Gebäudeautomationstechnik. Die bedient sich einer Visualisierungsmethode zum Systemüberwacher (Person), die alle Gebäudeausrüstungen zu überwacht und steuert. Dazu führt man eine Schnittstelle OPC ein, um GLT an den Automationsinstrumenten anzubinden. OPC ist durch eine Client-Server-Struktur

auf Windowsumgebungen konstruiert. In der Praxis werden die OPC-Anbindungen „BridgeVIEW“ und „FieldPoint“ verwandt, um Clients und Servers zusammenzusetzen.
Es gibt auch eine andere Möglichkeit; durch OSGi kann die entferne Überwachung und Steuerung von der Gebäudeautomation realisiert werden. Gegen OPC läuft OSGi auf der Java Virtual Machine mit einem Gerät - Residential Gateways. Das Gerät kann die intelligenten Endgeräten vernetzen und über Internet oder Intranet die Aufgabe der klassischen Fernsteuerung, Ferndiagnose und -wartung dieser Endgeräte schaffen. Weiterhin wird die Verteilung von Informationen und multimedialen Unterhaltungsinhalten an diese Endgeräte über geeignete Protokolle ermöglicht.
Im Automationsbereich wird OPC mit GLT häufiger als OSGi verwanden.

3.2.1 GLT(Gebäudeleittechnik)

Es gibt zwei verschiedene Bedeutungen von GLT(Gebäudeleittechnik).

i. „**Im weiteren Sinne** zur Bezeichnung der gesamten automatisierungstechnischen Instrumentarisierung mit einem Bezug zur Technischen Gebäudeausrüstung:“[6]
   GLT ist ein Bestandteil der Gebäudeautomation, die sich auf der obersten Ebene „Managementebene“ von der Drei-Ebene-Struktur in Gebäudeautomation (Kapital 2.1) befindet.

ii. „**Im engeren, gebräuchlicheren Sinne** der genutzten Software:“[6]
   Hier ist GLT eine Software, mit der die Infrastrukturen im Gebäude überwacht und gesteuert werden. „**Die Software läuft in der Regel auf einem PC und wird vom Hersteller der Gebäudeautomatisierungstechnik / Direct Digital Control (DDC) geliefert. Es gibt einige wenige herstellerunabhängige GLT-Systeme**.“[6] Die Schnittstelle zwischen GLT und dem Automationsnetzwerk ist OPC (OLE for Process Control), der in Kapitel 3.2.2 „OPC“ detailliert wird.
   Die GLT Software visualisiert die technischen Vorgänge vom Automationsprozess innerhalb des Gebäudes. Die GLT Software sammelt die Daten aus den DDCs im Gebäude über einen Feldbus ein (Feldebene) und stellt graphisch die Daten in einer dem Benutzer verständlichen Art und Weise dar. Die Gebäudeleittechnik dient als Nutzerinterface zur Gebäudeautomationstechnik (Managementebene).

Die Steuerungen der Gebäudeautomation erfolgen durch die DDCs, die direkt die Steuerungs- und Regelungsaufgaben im Bereich der Heizungs-, Lüftungs- und Lichtsteuerungen übernehmen. Über DDCs und Feld-Bus werden die Zustanddaten für die Aktoren und die Aufnahmedaten von den Sensoren in GLT umfasst, z.B:

i. Betriebszustände von Anlagenteilen (Aktor)
   Motoren
   Lüftungsklappen
   Ventile
   Störmeldungen
   Schalterstellungen
ii. Direkte Messwerte (Sensor)
   Temperatur
   Druck
   relative oder absolute Feuchte

Ziele des Einsatzes von GLT sind nicht nur das Gebäudeautomationssystem zu visualisieren und verwalten, sondern auch die Betriebskosten zu reduzieren. Damit ist GLT ein fester Bestandteil des modernen technischen „Facility Managements“.
Für die Echtzeitreagierungsanforderungen in einigen besonderen Situationen z.B. „Brandmelden“ und „Zugangskontrollen“ sind zusätzliche eigenständige Systeme erforderlich.

i. Brandmeldeanlagen / Brandmeldezentralen

Diese Brandmeldezentralen haben die Aufgabe, über eigene Kommunikationswege DDC an GLT anzubinden, um die echtzeitigen Alarme vom Brand zu machen.

ii. Zugangskontrollsysteme

Sind wie bei Brandmeldezentralen auch eigene Systeme zwischen GLT und DDC. So ist es möglich, sofort im GLT-System darauf zu reagieren, wenn sich jemand im sensiblen Bereichen des Zugangs auftritt.

3.2.2 OPC (OLE for Process Control)

OPC (OLE for Process Control) ist eine standardisierte Software-Schnittstelle, die in den Bereichen anwendet wird, in denen Sensoren, Regler und Steuerungen verschiedener Hersteller ein gemeinsames, flexibles Netzwerk bilden und mit einem Rechnernetzwerksystem verbunden sind. OPC ist das Interface zwischen den Gebäudeautomationsinfrastrukturen aus verschiedenen Hersteller und GLT (Facility Management).
Darunter zeigt Bild 3.2.2.1[7] die Einsatzstelle von OPC im Gebäudeautomationssystem:

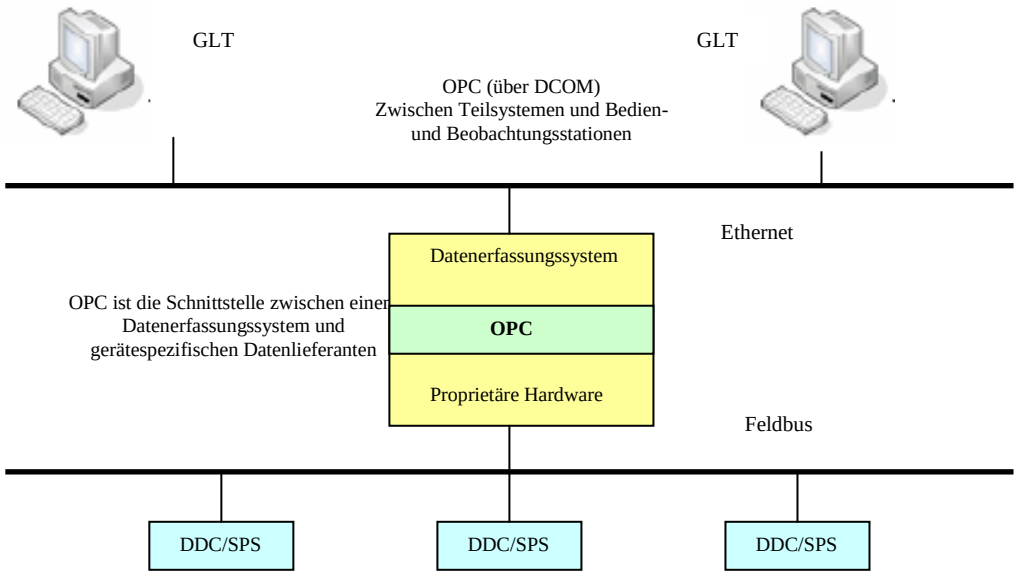


Bild 3.2.2.1 Einsatzbereich von OPC

Jede Automatisierungskomponente von verschiedenen Herstellern bringt ihr eigenes Kommunikationsprofil mit, um Automatisierungshardware mit Software-Applikationen zu verbinden, die aus einzeln übertragenden Softwarepaketen bestehen. Es gibt es viele Treiber von verschiedenen Herstellern. Anstelle dieser Treiber ist für jedes Gerät genau ein OPC-konformer Treiber zu schreiben. Damit kann OPC auf einem Computer die Anwendungsprogramme und die Baugruppentreiber miteinander verbinden.

Funktionsweise

Die OPC-Schnittstelle hat die Struktur von **Client-Server-Hierarchie**. Typische OPC-Server realisieren die Anbindung an die bestehenden Kommunikationssysteme. und OPC-Clients können Bedien- und Beobachtungssysteme, Archivierungssysteme und viele andere Anwender von Prozessdaten sein.
Die Kommunikationswege sind wie Bild 3.2.2.2 dargestellt:

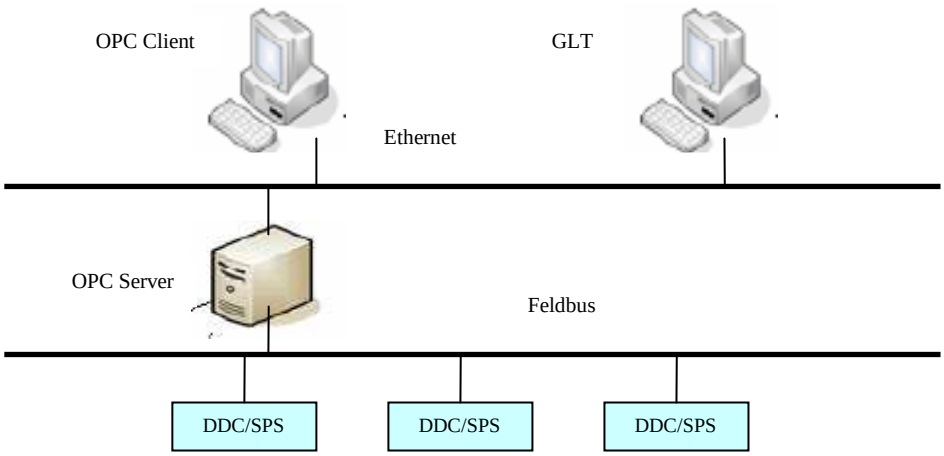


Bild 3.2.2.2 Kommunikationswege mit OPC

Für verschiedene Einsatzgebiete bieten verschiedene Hersteller unterschiedliche OPC-Server mit unterschiedlichen Eigenschaften an. Ein OPC-Client als Nutzer der Dienste ist nicht auf einen Server beschränkt, sondern kann im Rahmen der Leistungsfähigkeit des Systems beliebig viele OPC-Server nutzen, wie Bild 3.2.2.3[7].

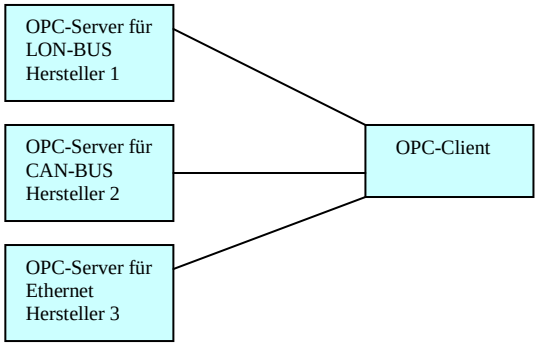


Bild 3.2.2.3 ein OPC-Client gegen beliebigen OPC-Servers

Für die **OPC-Server** werden zur Identifikation des Objekts vom Anbieter eindeutige Namen vergeben. Diese Namen muss ein OPC-Client verwenden, um einen OPC-Server zu spezifizieren. Für den Datenaustausch werden die Schnittstellen und deren Methoden in drei Hierarchische Klassen - OPC-Server, OPC-Group, OPC-Item(Bild 3.2.2.4[7]) verteilt.

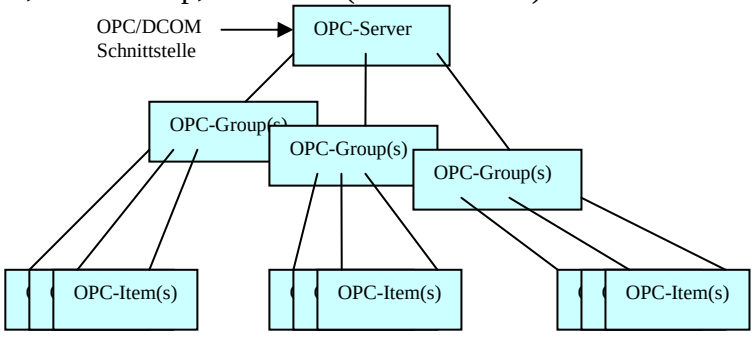


Bild 3.2.2.4 Klassenmodell der OPC-Schnittstelle

Die Klasse **OPC-Server** besitzt verschiedene Attribute und Methoden, die Informationen über den Status, die Version und den Adressraum der verfügbaren Prozessvariablen eines OPC-Server-Objekts liefern. Ein OPC-Server verwaltet die untergeordnete Klasse OPC-Group.
Die Klasse **OPC-Group** strukturiert die vom OPC-Server genutzten Prozessvariablen. Mit Hilfe der Objekte OPC-Group kann ein Client die Einheiten von Prozessvariablen bilden und mit diesen Operationen ausführen. Z.B. können alle Prozessvariablen einer Bildschirmseite eines

Bedien- und Beobachtungssystems in einer Gruppe zusammengefasst werden. Die Klasse definiert auch Methoden, über die die Werte der Prozessvariablen gelesen und geschrieben werden können. Dabei können mehrere Variable in einem Auftrag zusammengefasst und gleichzeitig übergeben werden.
Die Klasse **OPC-Item** repräsentiert für jedes Objekt eine Verbindung zu einer Prozessvariablen. Eine Prozessvariable ist ein Element des Adressraums des OPC-Servers, ein Beispiel ist das Eingabemodul einer speicherprogrammierbaren Steuerung. Ein OPC-Item wird durch seine **Item-ID** identifiziert. Die Item-ID ist ein vom Hersteller des Servers schon festgelegter Name, der innerhalb des Adressraums des Servers eindeutig ist. Mit jedem OPC-Item sind die Eigenschaften der Wert, Qualität und Zeitstempel verbunden. Die Qualität sagt aus, ob der Wert der Variablen sicher ermittelt werden kann (z.B. ob die Kommunikationsverbindung besteht). Der Zeitstempel gibt an, wann der Wert der Prozessvariablen ermittelt wird.
Durch ein Item können beliebige Daten erreicht werden, z.B.:

i. Werte eines Sensors, z.B. Druck, Temperatur oder Durchfluss,
ii. Steuerparameter, z.B. Start, Stop, Öffnen, Schließen,
iii. Statusinformationen von z.B. einem Gerät,
iv. Status der Netzverbindung

Ein **OPC-Client** z.B. ein Bedien- und Beobachtungssystem, kann die Prozessvariablen durch den OPC-Server anmelden. Der OPC-Server überprüft zyklisch den Wert und die Qualität der Prozessvariablen oder aktiviert die Systeme, um die Beobachtung zu machen. Wenn sich der Wert oder die Qualität ändern, ruft der OPC-Server den Client zurück und teilt ihm die veränderte Wert und Zustandsinformationen mit.
Die Verbindung zwischen dem OPC-Server und dem OPC-Client wird über **COM** (Component Object Model) oder **DCOM** (Distributed Component Object Model) von Microsoft realisiert. Für die OPC-Client-Server-Architektur gibt es zwei Arten von COM-Schnittstellen(Bild 3.2.2.5[7]), die sich in der Art des Funktionsaufrufs unterscheiden.

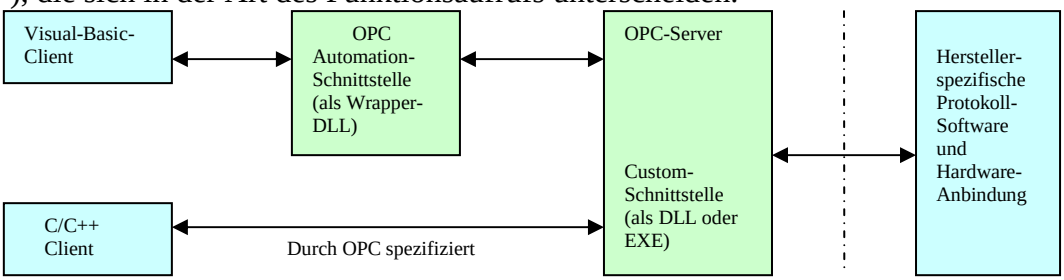


Bild 3.2.2.5 zwei COM-Schnittestellen

Die Custom-Schnittstelle muss von jedem OPC-Server bereitgestellt werden und ist der performante Zugang zu den Prozessvariablen. Die Automation-Schnittstelle basiert auf einem COM Dispatch Interface und ist für interpreter-orientierte Entwicklungsumgebungen wie Microsoft Visual Basic™ oder Borland Delphi™ optimiert.
Client-Applikationen, die auf einer Skript-Sprache wie Visual Basic oder Visual Basic for Applications basieren, müssen die Automation-Schnittstelle benutzen. Und C++ sollte auf der Custom-Schnittstelle aufgesetzt werden.
Über Microsofts-Schnittstellen **DCOM** ist es möglich, dass ein entferner Client auf die Daten auf einem lokalen Server zugreifen und einen verteilten System-Mechanismus realisieren kann. Es ist für OPC-Anwendungen klar, ob die über OPC ausgetauschten Daten von einer Anwendung im eigenen Adressraum, von einem fremden, lokalen Prozess oder auch von einem entfernt über TCP/IP angebundenen Rechner kommen. Damit können die Wege von Datenübertragung und –zugriff optimiert werden. Der OPC-Standard definiert nun bestimmte DCOM-Objekte. D.h. DCOM bedient sich der Schnittstellen, um die OPC-Teilnehmern (über DCOM) miteinander Daten austauschen zu können. Auch kann ein OPC-Server von mehreren Clients angesprochen werden.

Als für die Automatisierungstechnik notwendiger Bestandteil wird das **HMI/SCADA-System BridgeVIEW** oft als eine Software-Entwicklungsumgebung in der Gebäudeautomation eingesetzt. BridgeVIEW besitzt eine Client-Server-Architektur. Wie Bild 3.2.2.6[7], auf der obersten Hierarchiestufe ist die eigentliche Bedienoberfläche (VIs) angesiedelt. Auf der mittleren Ebene läuft die BridgeVIEW-Engine zur I/O-Archivierung und zur Überwachung. Als Schnittstelle zur eigentlichen Prozesshardware fungieren die Device-Server (Geräte-Server). die Device-Server können als unterschiedliche Typen realisiert sein:

i. OPC
ii. VI (Virtuelle Instrumente)
iii. IAK(Industrial Automation Kernel)
iv. DDE

Die Device-Server sind speziell für die verwendete Schnittstelle, z.B. SPS, Datenerfassungs-Hardware, Feldbussysteme, und das verwendete Protokoll ausgelegt. Alle Device-Server sind in einer Standerd-Client/Server-API für BridgeVIEW programmiert.

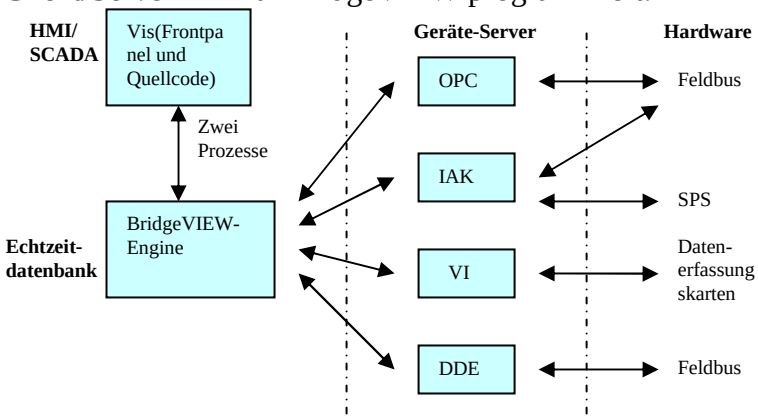


Bild 3.2.2.6 BridgeVIEW-Architektur und Zusammenwirken der einzelnen Einheiten

Fieldpoint ist ein Datenerfassungsmodul für verteilte Anwendungen, das aus einem Standard-PC mit eingebauten Datenerfassungs- und Gerätesteuerungskarten besteht, und wird als eine Domäne von Standard-SPS oder speziellen Anlagen in Teilbereichen der Industrie eingesetzt.
FieldPoint stellt eine Klasse von analogen und digitalen I/O-Modulen zur Verfügung, Alle I/O-Module werden auf ein weiteres Modul „Anschlussklemmenmodul“ aufgesteckt.
Das FieldPoint-System kann auch direkt an ein standardisierte 10-MByte/s-(10BaseT) oder 100-MByte/s (100BaseTX) Ethernet-Netz angeschlossen werden.
Fieldbus-Interface-Modul ist zwischen FieldPoint-I/O-Modulen und Fieldbus gelegen. Es hat standardisierte Funktionsblöcke, die den Interoperabilitätstest der Fieldbus Foundation machen und Analogeingang, Analogausgang, diskreten Eingang, diskreten Ausgang, komplexen und diskreten Ausgang und PID.

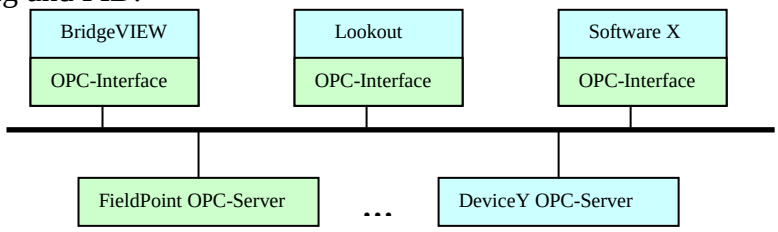


Bild 3.2.2.7 Interoperabilität zwischen den unterschiedlichen OPC-Clients und FieldPoint-Server

Wie Bild 3.2.2.7[7] unterstützt FieldPoint-OPC-Server den vollständigen Funktionsumfang eines OPC-Servers. Somit hat es keine Probleme, an BridgeVIEW, Lookout und LabVIEW sowie auch HMI- und SCADA-Software anzubinden.

3.2.3 OSGi (Open Services Gateway Initiative)

„Die OSGi Alliance (Open Services Gateway Initiative) ist ein **Non-Profit-Industriekonsortium**, **das sich mit einer hardware-unabhängigen Java-Softwareplattform beschäftigt**, auf der Anwendungen und Dienste ausgeführt werden können“[8].
„Gegründet wurde die OSGi Alliance 1999. Ihr gehören (Stand August 2006) 35 Firmen aus recht unterschiedlichen Branchen als Vollmitglied an, wie etwa IBM, Nokia, Motorola, Philips, BenQ, Siemens VDO, Telefonica, BMW, Deutsche Telekom. Zu den Vollmitgliedern kommen noch sogenannte Adopter Associate-Mitglieder hinzu“[8].
„Diese OSGi-Framework ist eine offene, modulare und skalierbare „Service Delivery Plattform“ auf Java-Basis“[8]. Es wird in intelligenten Geräten, die **„Residential Gateways“** genannt werden, einbettet. Damit kann alle Arten von Geräten mit geeigneten Protokollen vernetzt werden, z. B. mit dem Ziel der intelligenten Fernsteuerung von Hausgeräten ("Weiße Ware") zur Hausautomatisierung. Diese realisieren über Internet oder Intranet die Aufgabe der klassischen Fernsteuerung, Ferndiagnose und -wartung dieser Geräte. OSGi wird dementsprechend typischerweise in Fahrzeugen (Telematik), mobilen Endgeräten (Handys, PDAs) und im Bereich der Heimvernetzung (Residential Gateways, Router) oder der Gebäudeverwaltung (Facility Management) angewendet – aber auch in industriellen Automatisierungslösungen oder völlig anders gearteten eingebetteten Systemen.
„Residential Gateway“ ist eine Art erweiterte Router-Funktionalität (vergleichbar mit einem DSL-Router), jedoch als Gateway (Service-Aggregator im kleinen Maßstab) für das gesamte Haus.
Darunter gelegenes Beispiel ist ein residentiales Gateway, das „iPAQ Connection Point“ von „Compaq“ hergestellt wird. Dadurch werden die Hausnetzwerk (Multi-PC, intelligente Geräte) an Internet verbunden. Das Detail des Produkts kann man in „http://www.hometoys.com/htinews/aug01/reviews/ipaq/ipaq.htm“ anfragen

Bild 3.2.3.1 iPAQ Connection Point (ein residentiale Gateway)

Im Bild 3.2.3.2 wird die Vernetzung der verschiedenen Geräte mit unterschiedlichen Protokollen dargestellt.

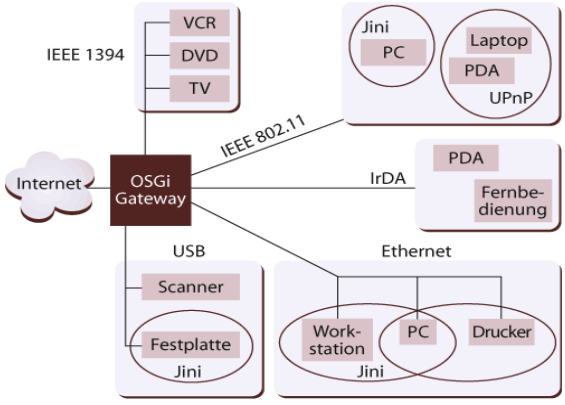


Bild 3.2.3.2 Verbindung über OSGi-Gateway zwischen Internet und Heimnetz

4. Übersicht von den Feldbusse

Feldbus ist ein automatisiertes Kommunikationssystem, die eine Vielzahl von Feldgeräten wie Sensoren, Aktoren mit einem Steuerungsgerät (DDC, SPS, LON-Geräte…) verbinden. Die Feldbusse sind die Schlüsseltechnologie für die Automatisierung und Gebäudeautomatisierung und werden in der Automationsebene eingesetzt, die schon im Kapital 2.2 Automationsebene erklärt wurde.

„Die Feldbustechnik wurde in den 80er Jahren entwickelt, um die bis dahin übliche Parallelverdrahtung binärer Signale sowie die analoge Signalübertragung durch digitale Übertragungstechnik zu ersetzen. Heute sind ca 50 verschiedene Feldbusse, die sich hinsichtlich ihrer technischen Funktionen, Einsatzgebiete und Anwendungshäufigkeit grundsätzlich von einander unterscheiden, wie z.B. PROFIBUS, INTERBUS, ControlNet, Fieldbus Foundation oder CAN. Seit 1999 werden Feldbusse in der **Norm IEC 61158** ("Digital data communication for measurement and control - Fieldbus for use in industrial control systems") weltweit standardisiert.“ [9]

Prinzipbedingt gibt es für die Verbindung zwischen einem Regelungsgerät (DDC, SPS, LON-Geräte…), als ein Master oder Host bezeichnet, und mehreren Sensoren und Aktoren, Slaves genannt, zwei Möglichkeiten von den Verbindungsweisen für ein Feldbussystem auf Sensor/Aktor-Ebene (unterste Kommunikationsebene) wie Bild 4.1[10] :

i. Vom Regelungsgerät aus wird je ein Kabel zu jedem Sensor oder Aktor gezogen (**parallele Verdrahtung**). z.B. SPS mit PP und Parallelverdrahtung(Sternstruktur)
ii. Vom Regelungsgerät aus wird nur ein Kabel gezogen: Das Kabel wird bei jedem Sensor und Aktor vorbeigeführt (**serielle Verdrahtung**). z.B. SPS mit K als Master eines seriellen Zubringer-Busses

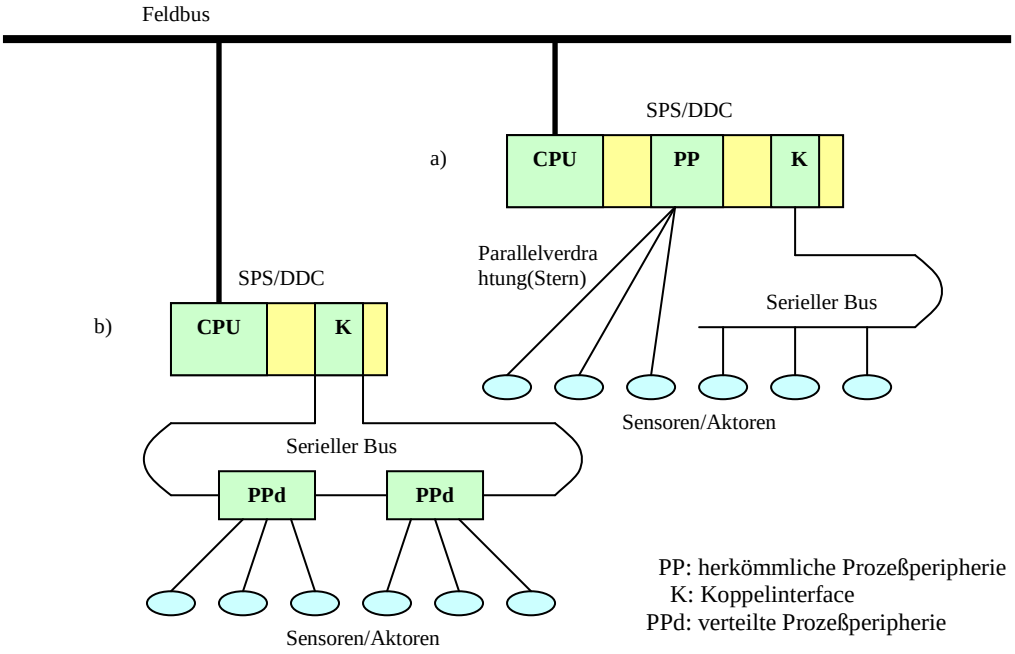


Bild 4.1 Busstrukturen in Sensor/Aktor-Ebene

Eigentlich sind **die Vorteile** der seriellen Verdrahtung im Vergleich zur parallelen Verdrahtung:

i. geringerer Verkabelungsaufwand
ii. Verkürzung der Ausfall- und Wartungszeiten
iii. kurzer Signalwege Auswahlmöglichkeit
iv. gegenüber analogen Werten erhöht sich der Schutz vor Störungen
v. Vereinheitlichung von herstellerabhängigen Geräten und seinen Datenübertragungen über offenen Feldbussen; Erleichterung der Basiskommunikation.

vi. flexible Erweiterungen oder Änderungen und somit Zukunftssicherheit

Die Nachteile der seriellen Verdrahtung im Vergleich zur parallelen Verdrahtung sind:

i. komplexeres System
ii. höherer Preis von Komponenten mit Feldbusfunktionalität
iii. aufwändige Messgeräte
iv. höhere Reaktionszeit
v. Extra Kosten von den Geräten, die viele unterschiedliche Feldbustechniken unterstützen.

4.1 Feldbus-Topologie

Wie das Rechnernetz verknüpft man Rechner, SPS, DDC miteinander unter einer geometrischen Anordnung (Topologie) der Verbindungsleitungen.
Die Topologien im Feldbus-Bereich sind:
Bei Netzen in **Stern-Topologie** (Bild 4.1.1) sind an einen zentralen Teilnehmer alle anderen Teilnehmer mit einer Zweipunktverbindung angeschlossen. In Computernetzen kann es eine spezialisierte Einrichtung sein, zum Beispiel ein Hub oder Switch.

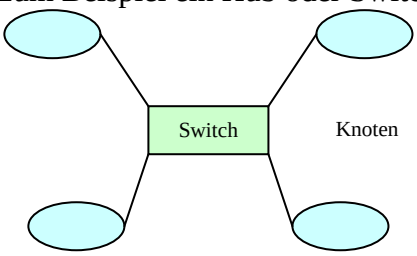


Bild 4.1.1 Stern-Topologie

Bei der Vernetzung in **Ring-Topologie** (Bild 4.1.2) werden jeweils 2 Teilnehmer über Zweipunktverbindungen miteinander verbunden, so dass ein geschlossener Ring entsteht.
Es wird ein Ringleitungsverteiler (MAU=Multi Access Unit) eingesetzt, der verhindert, dass bei einem Ausfall eines Endgerätes das gesamte Netz ausfällt.

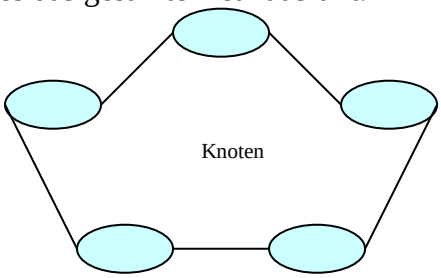


Bild 4.1.2 Ring -Topologie

Eine **Bus-Topologie** (Bild 4.1.3) besteht aus einem Hauptkabel, dem Bus, an das alle Geräte und zwei Endwiderständen angeschlossen sind. Diese Abschlusswiderstände mit dem Leitungswellenwiderstand (Z = 50 Ohm bei Koaxialkabel) dienen zur Verhinderung von Reflexionen. Der Anschluss zwischen den Geräten (also Netzkarten) und Hauptkabel erfolgt über T-Stücke.
Zugriffsverfahren (z.B. CSMA/CD) versuchen zu verhindern, dass sich die Teilnehmer gegenseitig stören. Sie regeln, welcher Teilnehmer die gemeinsame Leitung – den Bus – zu welchem Zeitpunkt zur Verfügung hat.

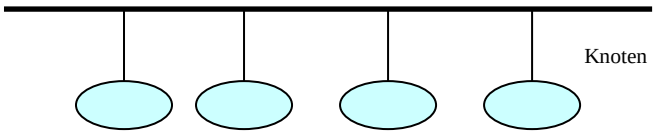


Bild 4.1.3 Bus -Topologie

Die **Baum-Topologie** (Bild 4.1.4) ist eine Netztopologie, bei der mehrere Netze der Sterntopologie hierarchisch miteinander verbunden sind. Hierbei müssen Verbindungen zwischen den Verteilern (Hub, Switch) mittels eines Uplinks hergestellt werden. Häufig wird diese Topologie in großen Gebäuden eingesetzt.

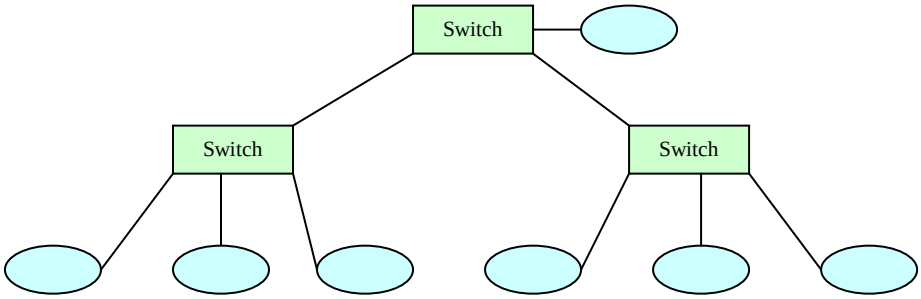


Bild 4.1.4 Baum -Topologie

4.2 Buszugriffsverfahren

Ein Buszugriff heißt, dass zu einem bestimmten Zeitpunkt nur ein Sender auf das gemeinsame Trägermedium zugreift. Entsprechend der verschiedenen Topologien gibt es unterschiedliche Möglichkeiten, den Buszugriff zu regeln. Wie Bild 4.2.1[10] wird das Busverfahren zwischen kontrolliert und zufällig verteilt.

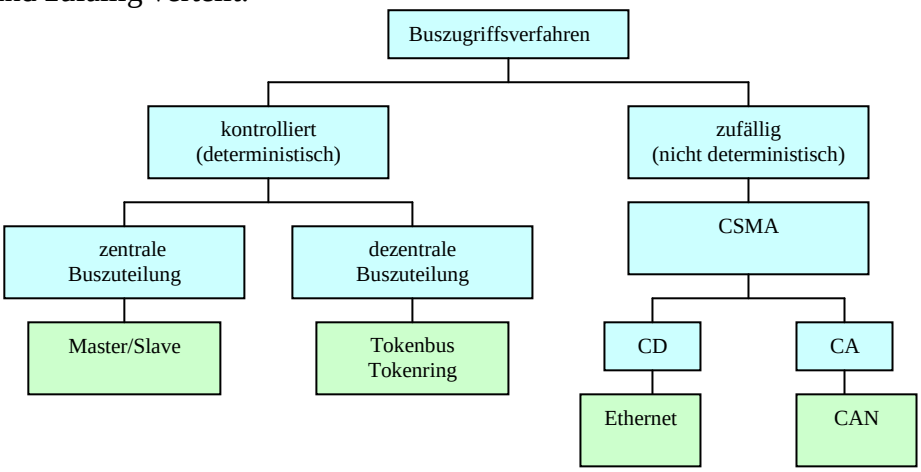


Bild 4.2.1 Übersicht Buszugriffsverfahren

Bei den kontrollierten Buszugriffsverfahren ist der Sender vor dem Sendebeginn eindeutig bestimmt. Diese kann zentral von einer Leitstation(Master/Slave-Verfahren) oder dezentral durch mehrere Steuereinheiten(Tokenbus, Tokenring) vorgenommen werden. Der maximale Zeitraum ist die Datenlänge, die Zeit bis zu der Ende von Datenübertragung. Solche Systeme nennt man **echtzeitfähig**.
Das Master/Slave-Verfahren wird bei z.B. AS-Interface, Interbus und PROFIBUS DP verwendet. Und das Token-Verfahren wird in PROFIBUS DP V1 und PROFIBUS PA eingesetzt.

Bei den zufälligen Buszugriffsverfahren greifen die sendewilligen Teilnehmer nur bei Bedarf auf das Übertragungsmedium zu, das nennt man CS(Carrier Sense). Dabei muss gewährleistet sein, dass das Medium nicht anderweitig von einem anderen Teilnehmer belegt ist. In diesem Fall muss die Sendung auf einen späteren Zeitpunkt verschoben werden, das ist MA(Multiple Access). Damit kann man nicht mehr den maximalen Zeitraum einer Datenübertragung bestimmen. Deshalb ist das zufällige Buszugriffsverfahren **nicht echtzeitfähig**.
In den Feldbussysteme LON, CAN und EIB wird das CSMA-Verfahren verwendet.

4.3 Kommunikationsmodul

Der Kommunikationsmodul des Feldbusses hat nur drei Schichten gegenüber der Grundlage des ISO/OSI-Modells (sieben Schichten) wie Bild 4.3.1.

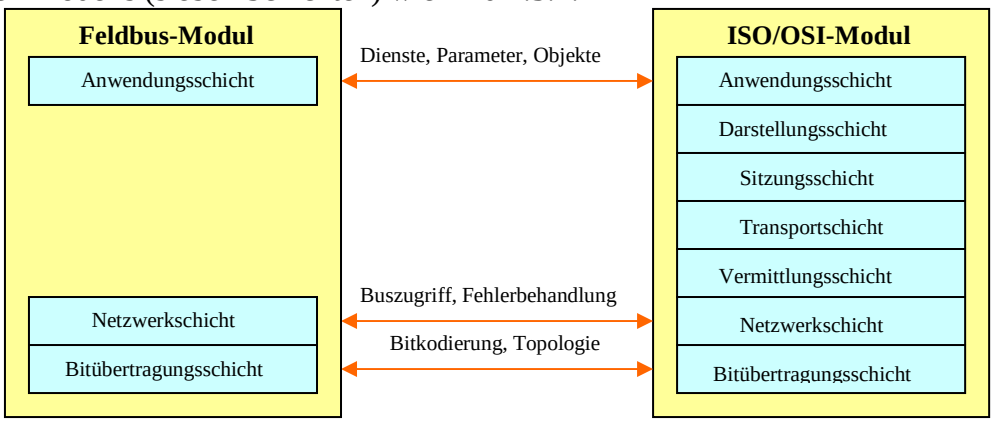


Bild 4.3.1 Felbus-Modul gegen ISO/OSI-Modul

Innerhalb Bitübertragungsschicht (Schicht 1) des Feldbus-Moduls sind physikalische Busankopplungen realisiert. Die Sende- und Empfangssignale werden über entsprechende Verstärkerbausteine geführt, die in SPSs, DDCs oder LON-Geräte kombiniert werden. Nach Busspezifikation bestehen galvanische Trennungen zur Anwendung, die beispielsweise aus Optokopplern und DC/DC-Wandlern für die Spannungsversorgung. Der wesentliche Kostenfaktor in der Schicht 1 ist wenig abhängig von dem Feldbustyp, sondern ist, ob eine galvanische Trennung implementiert werden muss oder nicht. Im industriellen Umfeld wird dies allerdings in der Regel empfohlen.
Innerhalb Netzwerkschicht (Schicht 2) werden die Zugriffsverfahren, Datenrahmen und Datensicherung auf den Prozessoren in SPSs, DDCs oder LON-Geräte realisiert. Die Prozessoren sind mit einer seriellen Schnittstelle ausgestattet, über diese Schnittstelle sie die zu sendenden bzw. empfangenden Bitfolgen mit den Blöcken zur physikalischen Busankopplung austauschen. Als Prozessoren kommen meist Standard-Typen, um sie in allgemeinen Applikationen anzuwenden.
Innerhalb der Anwendungsschicht (Schicht 7) werden Anwenderzugriffe vom Prozessor übernommen. Konkret handelt der Prozessor sich die Umsetzung der in den Feldbusspezifikationen definierten Dienste (z.B. Lesen, Schreiben,…) und Objekte in entsprechende Aktionen, die zur Aussendung und Empfang von Datenrahmen über die Schicht 2 führen.
Die anderen Schichten des ISO/OSI-Modells existieren im Feldbus-Modell nicht. Das heißt, die Funktionen von z.B. Routing, Flusskontrolle, Dialogsteuerung, Transaktionssteuerung, Konvertierung, Komprimierung und Verschlüsselung gibt es im Feldbussystem nicht.

4.4 Telegrammformate

Die Schicht 2 dient der Datenübertragung und –sicherung. Innerhalb dieser Schicht wird ein Satz von Protokollen zur bitseriellen Übertragung von Daten geboten.
HDLC (High Level Datalink Control Procedures)-Datenübertragung läuft synchron oder im Start/Stopp-Betrieb und beinhaltet eine mehrfache Sicherung. Ein Datenpaket enthält eine Adresse, Kontroll und Steuerinformation, die zu übertragenden Daten und Prüfinformation.
HDLC-Rahmen[10] ist:

Flag(Beginn)	Adresse	Kontrolle	Information	FCS(Frame Check Sequence Prüfsumme)	Flag(Ende)
01111110	8 Bits	8 Bits	n Bits	16 oder 32 Bits	01111110

In der HDLC-Norm werden drei Übertragungsklassen definiert: der NRM (Normal Response Mode), der ARM (Asynchronous Response Mode) und der ABM (Asynchronous Balanced Mode). Die werden im **Master/Slave-System** verwendet. Die typische Übertragungsgeschwindigkeiten sind von 9,6kBit/s bis 2MBit/s[10]. SDLC ist ein Protokoll von IBM und enger als HDLC im synchrone Betriebsart.
UART (Universal Asynchronous Receiver and Transmitter) dient einer bitseriellen Datenübertragung über eine Übertragungsstrecke und der Umsetzung der Datenworte zum und vom parallelen Rechnerbus.
UART ist zeichenorientiert im DIN 66 022766 203 und besteht 11 Bits[10].

Startbit	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Parität	Stoppbit
	LSB (niederwertig)							MSB (höchstwertig)		

Die typischen Übertragungsgeschwindigkeiten sind 150Bit/s, 300Bit/s, 600Bit/s, 1200Bit/s, 2400Bit/s, 4800Bit/s, 9600Bit/s, 19200Bit/s, 38400Bit/s und 64kBit/s[10].
Die Datenübertragung des **PROFIBUS** bedient sich auch der UART-Zeichen. Aber es werden nicht einzelne Zeichen übertragen, sondern Zeichentelegramme. Diese Telegramme können reine Informationstelegramme ohne Daten oder Telegramme mit Daten sein.
Die Telegramme ohne Daten haben eine feste Länge von 6 Byte[10]:

Startzeichen	Zieladresse	Quelladresse	Kontrollbytes	FCS	Endzeichen

Die Telegramme mit Daten sind:

Startzeichen	Zieladresse	Quelladresse	Kontrollbytes	8 Datenbytes	FCS	Endzeichen

Die maximale Länge eines **PROFIBUS-Telegramms** sind 255 Bytes[10].
HART wird besonders im z.B. explosionsgefährdeten Bereich bei zwei Systeme **PROFIBUS-PA** und **FOUNATION™** Feldbusse angewendet.
Das Telegrammformat[10] ist:

Präambel	Startzeichen	Adresse	Befehl	Byte-Zähler	Status(nur Slaveantwort)	Daten	Checksumme
3 Byte	1 Byte	1/5 Byte	1 Byte	1 Byte	2 Byte	0…24 Byte	1 Byte

Die Präambel dient der Synchronisation **zwischen Master und Slaves**. Die Adresse hat zwei Formate. Bei Short-Frame-Format werden von dem Adressbyte 4 Bit für die Slave-adresse reserviert. Damit sind nur 15 Teilnehmer adressierbar. Mit den noch verbleibenden 4 Bits wird Master adressiert. Dagegen hat Long-Frame-Format 38 Bit zur Teilnehmeradressierung. Und 2 Bits sind Kontrollbits.
Die **Token-Telegramme**[10] sind:

Startmarke	Zugriffskontrolle	Blockkontrolle	Zieladresse	Quelladresse	Information	Blockprüf	Endemark	Blockstatus
1 Byte	1 Byte	1 Byte	6 Byte	6 Byte	……	4 Byte	1 Byte	1 Byte

4.5 Übertragungsstandards

Für **RS 232-, V.24-Schnittstelle** existieren eine amerikanische Norm RS 232 C = EIA 232, eine internationale Norm CCITT V.24 und eine deutsche Norm DIN 66 020. Bei der RS 232-Schnittstelle handelt sich, um eine Schnittstelle für die Kommunikation zwischen zwei Datengeräten (DEE) oder zwischen einem Datenendgerät und einer Datenübertragungseinrichtung (DÜE). Die RS232 C ist nur für Punkt-zu-Punkt-Verbindungen geeignet.
Die Pegel sind wie folgt definiert:[10]

logische „0“	3V < U < 15V
logische „1“	-15V < U < -3V

Die Leitungslänge hat einen Einfluss auf die maximale Übertragungsrate[10]:

Übertragungsrat	**Leitungslänge**
1200 Baud	900 m
19200 Baud	50 m

RS 485-Schnittstelle ist für Mehrpunktverbindungen geeignet und in der EIA 485 und ISO 8482 beschrieben. An ein RS 485 darf man mehr als 32 Geräte[10] anschließen.
Gegen RS 232 sind die Belastungen von Sender und Empfänger unterschieden. Die Pegel sind wie folgt definiert[10]:

	Sender	**Empfänger**	
logische „0“	1,5V < U < 5V	EIA 485	ISO 8482
		U > 0,2V	U > 0,3V
logische „1“	-5V < U < -1,5V	U < -0,2V	U < -0,3V

Die maximale Übertragungsrate[10]:

Übertragungsrat	**Leitungslänge**
93,75 kBaud	1200 m
500 kBaud	400 m

20mA-Stromschleife ist in der DIN 66 258 Teil 1 beschrieben und ist nur für Zweipunktverbindungen ausgelegt. Die logischen Zustände sind[10]:

logische „0“	0mA < I < 3mA
logische „1“	14mA < I < 20mA

Die maximale Leitungslänge sind 1000m und die maximale Übertragungsrate ist 9600Baud[10].
IEC 61158-2 beschreibt eine Vielzahl unterschiedlicher Bussysteme(z.B. PROFIBUS, InterBus, …). Eine Schnittstelle wird in der IEC 61158-2 mit einem Übertragungsrat von 2,5 MBaud[10]. Und an einzelnem Segment darf man 32 Teilnehmern[10] anschließen.
Die wichtige Faktoren von zwei verschiedenen Busmodule in IEC 61158-2 sind[10]:

	H1-Bus	**H2-Bus**
Ubertragungsrat	31,25 kBaud	1 MBaud
Topologie	Linie/Baum	Linie
Max. Teilnehmerzahl	32	32
Max. Leitungslänge	1900 m	750 m
Max. Stichleitungslänge	120 m	--
Kabelstruktur	Paarweise verdrillt und geschirmt	Ein oder mehrere verdrillte Kabel, gemeinsamer Schirm

4.6 Leitungen und Übertragungsarten

Im Bereich der Feldbusse werden normalerweise Leitungen aus Kupfer verlegt. Lichtwellenleiter finden sich nur in extrem elektromagnetischer verseuchter Umgebung, wegen ihres hohen Aufwands. Für die üblichen Feldbusse mit RS 485-Übertragung verwendet man meist verdrillte, abgeschirmte Kupferleiterpaare (STP), bei kurzen Entfernungen verwendet ungeschirmte, verdrillte Kupferpaare (UTP). Bei höheren Ansprüchen an Störsicherheit und Datenübertragungsrate (Token-Ring, -Bus, CSMA/CD) wählt man Koaxkabel.

Tabelle 4.6.1 Arten von Kupferliter

	Kupferleiter			
	Verdrillt, unabgeschirmt	**Verdrillt, abgeschirmt**	**Koaxkabel**	
	UTP	STP	Yellow Cable	RG 58
Z/Ohm	100-120	100-120	50	50
Bezeichnungen	IBM Type 3, VDE YR	IBM Typ 1, IBM Typ 2, VDE YCY VDE Y(St)Y	10base5	10base2 Cheapernet

4.7 Verbindung von Netzen

Wie bei LAN-Systemen gibt es die Probleme bei physikalischen Netzverbindungen in Feldbussystemen, wie z.B. die Längenbegrenzung der einzelnen elektrischen Schnittstellen von Übertragungsmedien, Adressierung von den Teilnehmern oder unterschiedliche Protokolle zwischen zwei Bussysteme usw.

Ein Repeater(Verstärkerstation) wird in der ersten Schicht des OSI 7-Schicht-Modells gelegt, um zwei Teilnetzwerke von gleicher Art zu verbinden. Wegen der Zeitverzögerung wirkt sich der Repeater bei Master-Slave-System nachteilig aus. Zur Realisierung der Echtzeitforderungen muss die Anzahl der Repeater im Master-Slave-System begrenzt werden. Z.B. Bei AS-Interface dürfen max. 2 Repeater, bei PROFIBUS in Abhängigkeit der Übertragungsrate 4 bis 7 Repeater in Reihe geschaltet werden.

Eine Bridge auf der OSI-Schicht 2 wird zur Verbindung zwei oder mehrer lokaler Netze eingesetzt. Die lokalen Netze müssen mit denselben Protokollen arbeiten, aber die Übertragungsmedien mit unterschiedlichen Übertragungsgeschwindigkeiten oder Topologien können unterschiedlich sein.

Ein Router arbeitet auf Schicht 3 des OSI-Moduls. Durch den Router wird ein optimaler Weg aus vielen Leitwegen für den Transport der Datenpakete von einem Netzwerk zum anderen Netzwerk ausgewählt. Damit ist die Übertragungszeit der Pakete geringer

Ein Gateway übersetzt die unterschiedlichen Kommunikationsprotokolle von zwei oder mehrer unterschiedlichen Bussysteme. Im OSI –Modul ist Gateway von Schicht 4 bis höhere Schicht zu finden. Gateway ist eine Erweiterung von Routern und hat auch die Aufgabe, die Adressen in den verschiedenen Netzen umzusetzen, Paket- und Darstellungsformate umzuwandeln, bei Gateway-Netzwerken die Wegwahl zwischen den Netzen zu verwalten.

4.8 Reaktionszeit der Steuerung

Im Gebäudeautomationssystem gibt es drei Ebenen, die Managementebene, die Automationsebene und die Feldebene. Auf DDC oder LON-Gerät arbeiten die PCs, auf denen die Anwendung-Software läuft. Auf der Automationsebene setzt man SPS, DDC oder LON-Geräte ein und auf der Feldebene nutzt man Sensoren und Aktoren.

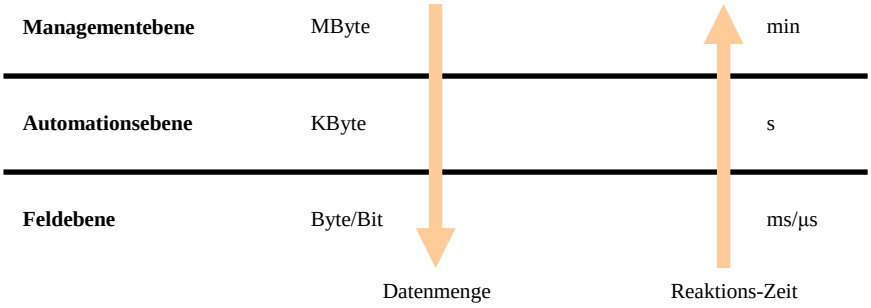


Bild 4.8.1 unterschiedliche Datenübertragungsanforderungen im Gebäudeautomationssystem

Die Datenübertragungsgeschwindigkeiten zwischen diesen Ebenen sind sehr wichtig für die Gebäudeautomationssysteme. Wie in Bild 4.8.1 sind bei unterschiedlichen Infrastrukturen die Geschwindigkeiten in den drei Ebene verschieden. Die Zeit wird in Richtung der unteren Ebenen immer kürzer, und die zu übertragende Datenmenge steigt mit den höheren Ebenen.
Die Abläufe von der Informationsübertragung sind z.B.:

i. Information wird aus der Umgebung über Sensoren in SPS, DDC oder LON-Gerät gegeben und bearbeitet, dann nach Aktoren gesendet.
ii. Oder SPS, DDC oder LON-Gerät werden von PCs mit GLT auf der Automationsebene abgefragt, und danach antworten sie an PCs die Informationen aus Sensoren und Aktoren.

Die Zykluszeit ist abhängig von einem Steueralgorithmus, der Verzögerungszeit der Sensoren und Aktoren, der Übertragungsrate in den Medien und auch von elektrischen Schnittstellen z.B. Router, Gateway…
Für verschiedene Technik, die man in Gebäudeautomation verwendet, sind die Reaktionszeit unterschiedlich.
z.B. bei Sensoren:

i. Nährungssensoren
   - ✓ Induktive und optische: 1 bis 10 ms[3]
   - ✓ Akustische und kapazitive: 10 bis 100 ms[3]
ii. Temperatursensoren: 1 bis 10 s[3]

Tabelle 4.8.1 gibt einen Überblick über die verschiedenen Reaktionszeiten von digitalen und analogen Signale mit der Anwendungshäufigkeit in Prozesssteuerungen. In dieser Tabelle sind die Daten (in Jahr 1997) von der Reaktionszeit und Anwendungshäufigkeit ungefähre Zahlen. Durch die Technikentwicklungen in der Gebäudeautomation werden die Zahlen im Zukunft verändert.

Tabelle 4.8.1 Reaktionszeit der Steuerung

	Digitale Ein-/Aussignal (ca.Anwendungshäufigkeit)	**Analoge Ein-/Aussignal (ca.Anwendungshäufigkeit)**
Alarm	100µs (7%)	100µs (6%)
Antriebsregelung	1ms (18%)	1ms (20%)
Schnelles Regeln	10ms (47%)	10ms (32%)
Bedienen/Visualisieren	100ms (22%)	100ms (31%)
Temperaturregeln	1s (5%)	1s (6%)
Langsames Regeln	10s (1%)	10s (4%)

4.9 Feldbusnormung

Die Feldbusnormung wird international bei der IEC (International Electrotechnical Commission) und in Europa bei der CENELEC und einer weiteren Norm EN behandelt. In Deutschland ist die DKE (Deutsche Elektrotechnische Kommission im DIN und VDE) für die Feldbusnormung zuständig.
Hier kann man die vorhandenen Bussysteme nach Feldbusnormung zuordnen:

Tabelle 4.9.1 Norm des Feldbussystem

Bussystem	**Norm**
Profibus FMS und DP	DIN 19245, EN 50170
Profibus PA	Geplant als Ergänzung zu EN 50170
Interbus, Interbus Loop	DIN 19258; EN 50254 in Vorbereitung(1999)
CAN	ISO 11898
LON	Normentwurf für IEC 62026
AS-Interface	EN 50295, Normentwurf für IEC 62026
Ethernet	IEEE 802.3, ISO 8802/3

4.10 Verbreitete Feldbusse

i. AS-Interface
ii. EIB/KNX(Europäischer Installationsbus)
iii. CAN(Controller Area Network)
iv. LON(Local Operating Network)
v. Interbus
vi. Profibus
vii. LCN(Local Control Network)
viii. BACnet

4.11 Realtime-Ethernet

In letzter Zeit etablieren sich im industriellen Bereich mehrere Kommunikationssysteme auf der Basis von Ethernet mit Echtzeiterweiterungen, die das Potenzial zur künftigen Ablösung der bisherigen Feldbussysteme besitzen:
i. EtherCAT
ii. Ethernet-Powerlink
iii. SERCOS III
iv. PROFINET
v. ETHERNET/IP
vi. VARAN

5. Die einnig wichtige Feldbus-technologien

Seit der Ende der 80er Jahre sind viele Feldbusse im Markt vorgestellt worden, z.B. LON, EIB/KNX, BACnet, CAN, Profibus, Interbus usw. Durch die aktuellen Standards, verwendete Techniken jedes Feldbusses und seine Entwicklungsziele darf man die Feldbusse grob in vier Klassen einordnen.
Die drei Klassen sind:

i. „Absolut dezentrale Systeme“[11]
ii. „Dezentrale, verteilte Konzepte“[11]
iii. „Feldbus als Ersatz der Ein-/Ausgangskarten“[11]
iv. „Zellenbussysteme“[11]

In den folgenden Absätzen werden die Feldbussse jeder Klasse mit einigen Beispielen erklärt.

5.1 Absolut dezentrale Systeme

In den absolut dezentralen Feldbussystemen werden die kleine „teilintelligente“ Sensoren mit den Feldbus-Geräte, indem komplette Chipsätze einsetzt werden, angekoppelt. Die Anschlusstechnik ist durch Aufstecken des Sensors/Aktors auf das Buskabel geschlossen. Und die Versorgungsspannung für die Sensoren wird über das Buskabel mitgeliefert.
Durch die Verwendung des Busse werden die digitale Ein-/Ausgabebaugruppen in den Steuergeräte von eine Anschaltbaugruppe (Kommunikationsprozessor) ersetzt. Damit darf die konventionelle parallele Verdrahtungen nicht mehr benutzt werden.
Darunter sind zwei typische Beispiele von den absolut dezentralen Feldbussystemen.

5.1.1 AS-Interface (Aktor-Sensor-Interface)

AS-Interface ist ein **Master/Slave-System**, der Master ist eine Anschaltbaugruppe in einer SPS.

Bustopologie

Das Aktor-Sensoren-Interface unterstützt eine beliebige Bustopologie. Es kann wie Linienstruktur, Sternstruktur oder Mischungsstruktur sein.
Die maximale Leitungslänge ist **100m** und an jedem Segment darf maximal **31 Slaves**[11] (in neue Version 2.1 ist **62 Teilnehmer**[10]) angekoppelt werden. Wenn das Netz erweitet werden muss, kann man Repeater einsetzen.

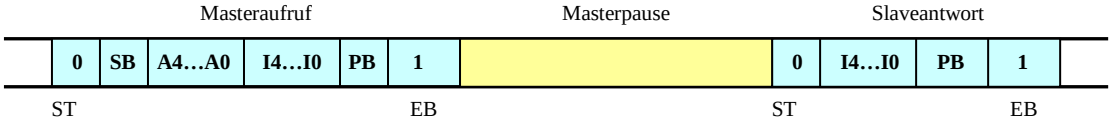
Übertragungsmedien

Bei AS-Interface werden die Datenübertragung und die Versorgungsspannung nur über ein Kabel versorgt.
Als AS-Interface-Leitung werden nicht die übliche Feldbusleitungen (**Twistet Pair**, **LWL**, **Koaxiale Kabel**) verwendet, sondern eine systemspezifische Flachbandleitung. Dieser Kabel bietet für die Slaves den Gleichstromwiderstand an.
Der maximale Strom pro Teilnehmer ist 65mA[11] und der minimale Leitungsquerschnitt ist 1,5mm^2. Die Spannungsversorgung für die Slaves ist 24V(+10/-15%)[11].
Eine separate, weitere Versorgungsspannungseinspeisung ist auch möglich. Es ist für einen deutlich höheren Strombedarf von Aktoren. Diese separate Versorgungsspannung muss getrennt vom AS-Interface-Kabel geführt werden und wird in das Aktor-Modul direkt eingespeist.

Der Übertragungsfrequenzbereich der AS-Interface-Leitung ist 167kHz[11] (Basisbandübertragung).
Weil die Leitung der Versorgungsspannung von Slaves die gleiche Leitung der Datenübertragung ist, ist beim AS-Interface-System eine besondere Modulationsart – **APM** (alternierende Plusmodulation) – eingesetzt. Die zwei Induktivitäten werden im AS-Interface-Netzteil erzeugt. Sie haben die Aufgabe, bei sich schnell ändernden Strömen eine Signalspannung im einzelnen Kabel aufzumodulieren.

Kommunikationsprotokoll

Die Kommunikation zwischen der AS-Interface-Anschaltbaugruppe und den Slaves läuft nach einem **Master/Slave**-Prinzip mit zyklischem Polling ab. Der Master sendet für jeden Slave ein Datentelegramm (Masteraufruf), und der entsprechende Slave sendet eine Slaveantwort zurück. **Das Mastertelegramm beinhaltet 14Bit**[11], **das Slavetelegramm 7Bit**[11], wie Bild 5.1.1.1[11].



ST =Startbit
SB =Steuerbit
A4…A0 =Adresse des Slaves
I4...I0 =Informationen von Master an Slave oder Slave an Master
PB =Paritätsbit
EB =Endebit

Bild 5.1.1.1 Aufbau einer AS-Interface-Nachricht

In Masteraufruf können sowohl E/A-Daten als auch Konfigurationsdaten für den Slave über das Steuerbit (SB) enthalten sein. SB ist 0 für einen Daten-, Parameter- oder Adressaufruf, ist 1 bei einem Kommandoaufruf.
Die Bruttoübertragungsrate von AS-Interface beträgt einschließlich aller funktionsnotwendiger Pausen **167kBit/s**[11]. Pro Teilnehmer werden ca. 150µs[11] vom Master benötigt. Damit kann eine Zykluszeit von 1 ms mit 6 Slaves an Bus angeschlossene Slaves erreicht werden, mit 31 Slaves braucht man 5ms, mit 62 Slaves 10ms.
Mit dem Masteraufruf werden verschiedene Nachrichten zu den einzelnen Slaves gesandt:

i. Die **Datenaustausche** zwischen Master und Slave sind: mit den Adressen wird ein Slave adressiert, der die Daten aus dem Masteraufruf entnimmt und seine eigenen Daten als Antwort an den Master abschickt. Nach einem Reset wird der Slave erst nach einem gültigen Parameteraufruf wieder aktiv.
ii. Der **Parameteraufruf** kann die Parameter, z.B. Messverfahren, Empfindlichkeit vom Slave sein. Erst nach einem gültigen Parameteraufruf wird der Slave auf den Datenaufruf reagieren.
iii. Der **Reset** kann den Slaven in einen definierten Zustand rücksetzen. Es muss bei jeder Mal nach ein Parameteraufruf erfolgen.
iv. Mit dem **Adressieraufruf** wird dem Slave eine neue Adresse zugewiesen. Wenn einem Slave eine neue Adresse zugewiesen wird, muss im Slave erst die aktuelle Adresse gelöscht werden(Betriebsadresse löschen). Die derzeitige Adresse ist 0 und kann die neue Adresse im Slave enthaltet werden.
v. Mit dem **E/A-Konfiguration lesen** wird die aktuelle Konfiguration eines Slaves ausgelesen, die mit der Antwort zum Master zurückgesendet wird.
vi. Um in Verbindung mit der E/A-Konfiguration wird der **ID-Code lesen** implementiert, um Slave zu identifizieren. Der ID-Nummer ist schon vom Hersteller im Slave einbettet.
vii. Mit **Stats lesen** werden die aktuellen Zustände des Slaves vom Master erkannt.

5.1.2 EIB (Europäische Installationsbus)

Der EIB ist ein offenes, umfassendes Bussystem unter dem Standard nach **EN50090**[12]. Das System einschließt **alle Aspekte der Gebäudeautomation**, z.B. die Beleuchtung und Jalousien, bzw. Beschattungseinrichtungen, die Heizung sowie die Schließ- und Alarmanlage. Mittels EIB ist auch die Fernüberwachung und -steuerung eines Gebäudes möglich. Eine Steuerung erfolgt dabei über den Benutzer selbst oder über einen Netzwerkcomputer, ausgerüstet mit entsprechender Software.
Weil EIB das Protokoll in ein umfassendes System der Heim- und Gebäudeelektronik einbettet, ist EIB in erster Linie eine Spezifikation und nicht eine Ausführung. EIB komplettiert sich mit standardisierten Systemkomponenten (z.B. BAU – Bus Access Unit), Netzwerkverwaltung und Interworking-Standards, herstellerunabhängigen Werkzeugen und Programmierschnittstellen für PCs usw..
EIB ist ein offenes Bussystem, weil es von jedem Anwender auf jeder gewählten Chip- oder Prozessorplattform realisiert wird und auch als Netzwerkgrundlage für eigene, individuelle Produkte und BAU-Geräte.

Netzwerktopologie

Der EIB ist ein eindeutiges „peer-to-peer"-Netzwerk mit einer maximalen Kapazität von 65.536 Busgeräten[10]. Das heißt: In einer Einheit von EIB-Netzwerksystem gibt es 15 Bereiche, 16-Bit Systemidentifikation je Bereich und 256 Teilnehmer pro Linie. Somit können bis zu 15 x 16 x 255 +255= **65.536**[10] elektrische Geräte einzeln gesteuert werden. Damit bezeichnet zum Beispiel die Adresse 8.7.233 in Bereich 8, Linie 7, den Teilnehmer 233.
Eventuelle Anschlussbeschränkungen sind abhängig von den Peripheriegegebenheiten (Übertragungsmedien, Transceiver-Typen, Kapazität der Spannungsversorgung) und den Umgebungsbedingungen (elektromagnetische Störungen, …). Und die Installations-, Anschluss- und Produktrichtlinien sollen auch berücksichtigt werden.
Mit Bridges und Routern darf man dieses Netzwerk erweitern.

Übertragungsmedien

Traditional darf EIB-Netzwerksystem die **verdrillte Zweidrahtleitungen** (**Twisted Pair TP**) nutzen. Dieses EIB-Kabel besteht aus vier Adern, wovon jedoch nur zwei verwendet werden. Die Adern des EIB-Kabels haben einen Durchmesser von 0,8 mm. Die EIB-Anlage wird von einer Spannungsversorgung mit 28 V Nennspannung [Gleichspannung] versorgt, z.B. ein „J-Y(St)Y 2x2x0, 8 EIB" mit den Adernfarben: Rot (+), Schwarz (-), Gelb und Weiß (Reserve) (Twisted-Pair-Kabel) verwendet. In diesem Weis ist die maximale Länge für ein physikalisches TP-Segment 1000m[14]. Die höchste Reaktionszeit ist 100ms. Bei dem Fall mit 14 Busteilnehmern ist die schnellste Abrufverfahren dauert 50ms mit 1Byte-Statusinformationen.
Auf der **PL (Power Line)** wird ein neuartiges Modulationsverfahren (Spread Frequency Shift Keying) in EIM eingeführt. Mit einem entsprechenden numerisch angepassten Filter können die zur Verfügung stehenden BAUs eine ausreichende und zuverlässige Datenübermittlung für die Gruppenadressierung auf der Versorgungsleitung sicherstellen. Der Zugriff auf das Übertragungsmedium wird durch eine Präambelsequenz mit den wahlfreien Übertragungslots gesteuert. Die maximale Entfernung zwischen 2 Geräten ist **600m**[10].
EIB RF (Radio Frequency)-Kanäle sind durch unterschiedliche Trägerfrequenzen physikalisch voneinander getrennt. Die Übertragungsentfernung ist ca. **300m**[10].
Neue Medien, wie **EIB Infrarot** und ein schnelles **EIB.NET** (10Mbit Ethernet + Routingfähiges IP) sowie **EIB.MMS** (ADSL basierende MultiMedia Services) sind in der Entwicklung.

Kommunikationsprotokoll

Der Datenaustausch erfolgt über Telegramme. Das ganze funktioniert analog zum Ethernet, das heißt, die einzelnen Busteilnehmer können unabhängig voneinander senden und es kann auch zu Kollisionen kommen. Doch durch das Zugriffsverfahren **CSMA/CA** (**C**arrier **S**ense **M**ultiple **A**cces/**C**ollision **A**voidance) werden Telegrammverluste im Falle der Kollision ausgeschlossen. Der sendende Teilnehmer mit der höheren Priorität sendet im Falle einer Kollision ungehindert sein Telegramm. Der Teilnehmer mit der niedrigeren Priorität zieht sich zurück.
Die EIB PDU (Protocol Data Unit)-Rahmen auf OSI Schicht 2 können eine Länge von maximal **14Bytes**[10] sein (jetzt wird die Möglichkeit einer Erweiterung auf 230Bytes[10] untersucht, wie Bild 5.1.2.1).

Octet 0	1	2	3	4	5	6	7	8	…	N-1	N≤22
Kontroll feld	Quell adresse		Ziel adresse		DAF NPCI: Länge	TPCI	APCI	Daten APCI	Daten		Check Octet

Bild 5.1.2.1 PDU-Rahmenstruktur(N<256) auf OSI-Schicht 2

Der EIB-Bus kommuniziert mit einer Übertragungsrate von **9,6kBit/s**[10]. Im Vergleich zu Computernetzwerken erscheint das langsam, ist jedoch für die Installationstechnik völlig ausreichend (vgl. Computernetzwerke: Diese können derzeit mit maximal 10Gbit/s betrieben werden). Die geringe Datenübertragungsrate kann jedoch bei schlecht programmierten Anlagen zu Problemen führen.

Steuerung und Programmierung

Die Programmierung der Teilnehmer und das Zuweisen der **Gruppenadressen (die Adresse von 16bit)** erfolgt mit einer speziellen, jedoch ebenfalls standardisierten Software, der ETS (EIB-Tool-Software). Die ETS wird von der Dachorganisation EIBA bereitgestellt und sichert die problemlose Zusammenarbeit von Komponenten verschiedener Hersteller (mittlerweile über 100 Hersteller[12] europaweit).
Alle größeren Hersteller für Elektroinstallationsprodukte sowie Heizungsausrüster bieten auch EIB-kompatible Geräte an.
Als Nachfolger für EIB wurde der **KNX** Standard im Jahre 2002[12] von der Konnex Association nach der Norm EN50090 weiter entwickelt. KNX ist abwärtskompatibel zum EIB, so dass bestehende EIB-Anlagen mit KNX-Feldmodulen erweiterbar sind.

Software-Frameworks

i. Windows

OPC - eine herstellerunabhängige standardisierte Software-Schnittstelle für die Windows-Plattform im z.B. EIB- und LON - Systeme. Darauf wird GLT(Gebäudeleittechnik) angelegt, um Bussystem entfernt und visuell zu überwachen. Das wird schon im Kapital 3.2.2 genannt.

ii. Java

OSGi - Middleware-Standard (Java-Framework) für die Einbindung von EHS/CHAIN, EIB, Konnex, LON, etc. in Service-Gateways Das wird schon im Kapital 3.2.3 erklärt.

5.2 Dezentrale, verteilte Konzepte

Für die großflächig verteilte Anlage in Automatisierung verteilt man die gesamte Aufgabe in Teilaufgaben, die in unterschiedlichen Bereichen vorkommen, modular programmiert und in eigenen Intelligentgeräte einbettet werden. Hier gibt es zwei Beispiele: CAN, LON und LCN.

5.2.1 CAN(Controller Area Network)

Der CAN-Bus handelt sich dabei um ein asynchrones, serielles Bussystem, das 1983 von Bosch für die Vernetzung von Steuergeräten im Automobil entwickelt und 1985 zusammen mit Intel vorgestellt wurde, um die Kabelbäume (bis zu 2km pro Fahrzeug[13]) zu reduzieren und dadurch Gewicht zu sparen.

Netzwerktopologie

Wegen des Multi-Master Prinzips, d.h. die Steuergeräte im CAN-Bus hat die gleiche Berichtigung von Informationszugriff gegen des Master-Slave-Moduls, wird das CAN-Netzwerk wird meisten als **Linienstruktur** aufgebaut. Stichleitungen sind in eingeschränktem Umfang zulässig (maximale Länge ist **3m**). Es ist auch ein ringförmiger Bus oder ein sternförmiger Bus möglich.
Die maximale Teilnehmeranzahl hängt von den verwendeten Bustreiberbausteinen (z.B. Transceiver, die eine physikalische Anschaltung an den Bus ist) ab. Es hat die Möglichkeiten von **32**, **64** oder bis zu **110 (mit Einschränkungen bis zu 128) Teilnehmer**[13] pro Leitung. Die Erweiterungsmöglichkeit wird über Repeatern oder Bridges realisiert.
Bei einem Highspeed Bus müssen zusätzlich 2 Abschlusswiderstände von je 120Ohm (zwischen CAN_HIGH und CAN_LOW) an dem jeweiligen Ende verwendet werden.

Übertragungsmedien

Der Bus ist entweder mit Kupferleitungen oder über Glasfaser ausgeführt. **Für Kupferleitungen** werden **nach ISO11898-2 (High-Speed Medium Access Unit) Twisted-Pair-Kabel** mit einem Wellenwiderstand von 108...132Ohm empfohlen. Für spezielle Anwendungen existieren noch **die Standards ISO 115 19-2 (Low-Speed) und ISO DIS 11 992**.
Im Falle von Kupferleitungen arbeitet der CAN-Bus mit Differenzsignalen. Er wird normalerweise mit **3 Leitungen** ausgeführt: **CAN_HIGH**, **CAN_LOW** und **CAN_GND (Masse)**. CAN_LOW enthält den komplementären Pegel von CAN_HIGH gegen Masse. Dadurch können Gleichtaktstörungen unterdrückt werden.
Wegen der drei Leitungen muss es sein, dass ein Bit, nach Zustand, entweder dominant oder rezessiv auf den Busleitungen wirkt. Das dominante Bit überschreibt das rezessives Bit und wird weiter übertragt.
Es wird zwischen einem Highspeed und einem Lowspeed Bus unterschieden.

- ✓ **bei Highspeed 1Mbit/s**
- ✓ **bei Lowspeed 125kBit/s**

Die maximale Leitungslänge beträgt:

- ✓ **bei 1Mbit/s 40m**
- ✓ **bei 500kBit/s 100m**
- ✓ **bei 125kBit/s 500m**

Die Verzögerungszeiten auf der Leitung, des Tranceiver (Sender und Empfänger), des Controllers (Sender und Empfänger) und der gesetzte Abtastzeitpunkt (Sender und Empfänger) müssen auch berücksichtigt werden.

Kommunikationsprotokoll

Die Telegramme in CAN sind als Pakete wie Bild 5.2.1.1 aufgebaut. Dieses Paket besteht aus die Daten wie Messwerte, Stellwerte und die Zustandsinformationen usw.

1	11	1	2	4	0 … 64	16	1	1	7	3
SOF	Identifier	RTR	Reserviert	Datenlänge	DATA	CRC	Slot	Delimiter	EOF	IFS
	Arbitrierung		CTRL				ACK			

Bild 5.2.1.1 CAN-Telegrammaufbau(mit 11Bit-Identifier)

- ✓ SOF(Start of Frame): 1Bit (dominant)
- ✓ Arbitrierungsfeld: einem Identifier(11Bit oder 29+2Bit) + 1Bit RTR(Remote Transmission Request)
- ✓ CTRL(Kontrollfeld): 6Bit (2Bit reserviert + 4Bit Länge der Daten)
- ✓ DATA(Datenfeld): 0-64Bit (in Einheiten von 8Bit)
- ✓ CRC(Prüfsummenfeld): 16Bit (15Bit CRC + 1Bit rezessiven CRC-Delimiter)
- ✓ ACK(Bestätigungsfeld): 2Bit (1Bit ACK-Slot + 1Bit rezessiven ACK-Delimiter)
- ✓ EOF(End of Frame): 7Bit (rezessiv)
- ✓ IFS(Intermission): 3Bit

Der CAN-Bus arbeitet nach dem **CSMA/CA-Verfahren** (unvergleichbar mit CSMA/CD beim Ethernet). Dabei werden Kollisionen beim Buszugriff durch die Arbitrierung oder Bit-Arbitrierung vermieden. Die Daten sind NRZ-L kodiert. Des Weiteren kommt zur Datensicherung das CRC-Verfahren zum Einsatz. Zur fortlaufenden Synchronisierung der Busteilnehmer wird Bitstopfen (bit stuffing) verwendet.

Objektidentifier

Der Objektidentifier ist eine Besonderheit von CAN gegen andere Feldbussysteme.
Der Objektidentifier kennzeichnet nicht das Gerät, sondern den Inhalt der Nachricht. Das bedeutet, dass der Parameter von z.B. Temperatur, Spannung, Druck im Messsystem, ein eigener Identifier zugewiesen sein. Die Empfänger können nach den Identifier entscheiden, ob die Nachricht für sie relevant ist oder nicht. Zudem dient der Objektidentifier auch der Priorisierung der Nachrichten. Die Spezifikation definiert zwei verschiedene Identifier-Formate:

i. 11-bit Identifier, auch "Base frame format" genannt.
ii. 29-bit Identifier, auch "Extended frame format" genannt.

Empfänger oder Sender von Nachrichten können beliebig vielen Identifiern besitzen, aber es darf zu einem Identifier immer nur maximal einen Sender geben.

CANopen/DeviceNet

CANopen und DeviceNet sind auf CAN basierende Schicht-7-Kommunikationsprotokolle, welche hauptsächlich in der Automatisierungstechnik verwendet werden.
Das Verbreitungsgebiet von **CANopen** ist dabei mehr **Europa**. Seit 1995 wird es von der CAN in Automation gepflegt und ist inzwischen als Europäische Norm EN 50325-4 standardisiert.
Es stellt dem Nutzer einen Herstellerunabhängigen Standard zu Verfügung. Dieser Standard definiert Geräteprofile und kommunikationsprofile.
Ein Geräteprofil ist eine Sammlung von Eigenschaften und Funktionen. **Dieses Geräteprofil besteht aus drei Teilen:**

- ✓ „Mandatory Functionality“[10]: Hier müssen alle pflichtige Gerätefunktionen implementiert werden, damit das Gerät einem bestimmten Profil entspricht.
- ✓ „Optional Functionality“[10]: Hier werden nur optionale Gerätefunktionen ausgeführt, die nicht vorhanden sein müssen.

- ✓ „Manufacturer Specific Functionality“[14]: Hier kann jeder Hersteller nach eigenen Vorstellungen Funktionen implementieren.

Die Kommunikationsprofile von CANopen sind:

- ✓ „Service Data Objects“[10]: Sie dienen dem Zugriff auf das Objekt Dictionary. Hier müssen relativ große Datenmengen übertragen werden. Das Zeitverhalten ist unkritisch, daher werden niedrige CAN-Botschaftsprioritäten verwendet.
- ✓ „Process Data Objects“[10]: Hier werden Prozeßdaten übertragen. Die Datenmengen sind klein, dafür sind hohe CAN-Botschaftsprioritäten benutzt.

CANopen kann durch SYNC-Botschaften eine Funktion von Zeitsynchronisation realisieren. Z.B. bei einem System mit mehreren Sensoren erfolgen alle Abtastungen zum gleichen Zeitpunkt, die Übertragung der Daten erfolgt später.

DeviceNet hingegen ist mehr in **Amerika** verbreitet. Es wurde von Allen-Bradley (gehört zu Rockwell Automation) entwickelt und später als offener Standard an die ODVA (Open DeviceNet Vendor Association) übergeben.

DeviceNet wird hauptsächlich für **die Verbindung einfacher Komponenten** wie Sensoren, Variante, Magnetventile, eingesetzt. Es ermöglicht Master-Slave-, Multi-Master- und Peer-to-Peer-Kommunikation. Dabei sind folgende Informationen enthalten:

- ✓ „Geräteadresse“[10]: Dieser Wertidentifiziert den Netzknoten eindeutig.
- ✓ „Klasseberkennung“[10]: Sie identifiziert die Klasse, zu der das angesprochene Objekt zugeordnet wird.
- ✓ „Instanz-Kennung“[10]: Die identifiziert ein Objekt innerhalb einer Klasse.
- ✓ „Attributskennung“[10]: Die identifiziert ein Attribut und einen Parameter eines Objekts
- ✓ „Service Code“[10]: Die zeigt einzelne besondere Funktionen eines Objektes an.

Durch die Definition einheitlicher Gerätemodelle werden alle einfachen Geräte von unterschiedlichen Herstellern untereinander austauschbar. Das heißt man Interoperabilität.

Das im maritimen Bereich sich langsam ausbreitende Protokoll NMEA2000 der NMEA Organisation ist eine Erweiterung von SAE J1939.

5.2.2 LON(Local Operating Network)

LON ist ein Feldbus, welcher vorrangig in der **Gebäudeautomatisierung** eingesetzt wird. Dieser Feldbus wurde von der US-amerikanischen Firma Echelon um das Jahr 1990 entwickelt. Mit ANSI/EIA-709[13] und EIA-852[13] ist LON in den USA bereits ein Kommunikationsstandard. Unter dem Begriff 'Control Network Protocol' wird LON mit der EN14908[13] voraussichtlich 2005 zur Europäischen Norm.

LON-Technik wird im folgenden **Kapital 6** weiter erklärt. Hier ist nur ein Konzept, um LON einfach vorzustellen.

Netzwerktopologie

LON Feldbussystems ist die **dezentrale** Automatisierung. Grundsätzlich ist dieses Feldbussystem ein Multimastersystem. Im LON kommunizieren Geräte (in der Terminologie Knoten bzw. engl. Nodes genannt) über einen Bus miteinander. Natürlich kann die Topologie mit TP auch Ring, Stern, freie Topologie sein. Im LON-System gibt es 255 Teilnetzen und an jedem Teilnetz werden maximal 127 LON-Geräte verbindet. In einem LON-Netzwerk darf max. **32385 Teilnehmern**[4] anschließen werden. Das Netzwerk kann auch über Repeaters, Bridges und Routers erweitern.

Übertragungsmedien

In LON-Bus werden **zahlreiche Medien** für Datenübertragung verwendet, z.B. verdrillte Zweidrahtleitung oder TP, 230-V-AC-Leitung, Infrarot, Lichtwellenleiter, Koaxialkabel, Funkkanal… Die Energieversorgung des Netzwerks ist 42 V DC.
Mit verschiedenen Übertragungsmedien sind die Übertragungsrate, die max. Teilnehmerzahl pro Segment und die weiteste Netzausdehnung unterschiedlich. In Tabelle 5.2.2.1[4] gibt es nur einige Beispiele für die Unterschiede zu erklären.

Tabelle 5.2.2.1 verschiedene Übertragungsfaktoren unterschiedlicher Medien

Medium	**Transceiver**	**Übertragungsrate Kbaud**	**Teilnehmerzahl**	**Topologie**	**Netzausdehnung m**
TP	TPT/XF78	78	64	Linie	2000
	TPT/XF1250	1250	64	Linie	130
	FTT10-A (Power Line)	78	64	Frei	500
				Linie	2700
	LPT-10 (Link Power)	78	128	Frei	500
				Linie	2200
230-V-AC-Netz	PLT-22	5	Anwendungsspezifisch	Stromnetzspezifisch	Stromnetzspezifisch
Funkkanal	300MHz	1,2	Anwendungsspezifisch	Anwendungsspezifisch	Anwendungsspezifisch
	450MHz	4,8			
	900MHz	39			

Kommunikationsprotokoll

Das Kommunikationsprotokoll dieses Feldbusses wird als **LonTalk-Protokoll** bezeichnet. Das LonTalk-Protokoll definiert **die Schichten 2 bis 7** des OSI-Referenz-Modells.
Das LonTalk-Protokoll nutzt das stochastische **CSMA/CD**, wie Ethernet. Es wird jedoch als ein predictiv p-persistent CSMA –optional Prioritätsgesteuert- realisiert[14], um Kollisionen weitestgehend zu vermeiden.
Im LON-Netzwerksystem werden die Daten im Form von **PDU** (Protokoll Data Unit) übertragen. Tabelle 5.2.2.2[11] zeigt eine Abschätzung des Datendurchsatzes auf LON-Netzwerk mit **64Byte** Paketen gewiesen.

Tabelle 5.2.2.2 Datendurchsatz bei LON mit 64Byte Paketen

Bitrate kBit/s	**Pakete/s**
4,9	5
9,8	10
19,5	20
39,1	40
78,1	80
156,3	160
312,5	270
625,0	470
1250,0	560

Aus logischer Sicht kommunizieren die Knoten über Kommunkationsobjekte miteinander, sogenannter NV (Network Variables). Damit Knoten verschiedener Hersteller miteinander kommunizieren können, werden so genannte **SNVTs** (Standard Network Variable Types)

definiert. Das sind Datentypen aus Anwendersicht, z.B. den Typ SNVT_temp_p, welcher eine Temperatur verkörpert. Die Organisation, welche das vorantreibt ist die LonMark Association.

Hardware

Das Kernstück dieses Feldbussystems ist der **Neuron (Chip)**. Der Neuron-Chip enthält **drei 8 Bit Prozessoren (CPUs)**:

- ✓ die Media Access CPU kontrolliert die physikalische Verbindung zum Netzwerk.
- ✓ die Network-CPU ist für die Kodierung und Dekodierung der Netzwerknachrichten verantwortlich.
- ✓ auf der Application CPU läuft die vom Anwender programmierte Software, welche die eigentliche "Intelligenz" eines Knotens repräsentiert.

Jeder Neuron-Chip enthält eine weltweit einmalige, **48Bit lange ID-Nummer**, mit deren Hilfe jeder Bus-Koten im Netz eindeutig identifizierbar ist.

Software

Softwareseitig werden Knoten mit Applikationen für Standardaufgaben oder frei programmierbare Systeme eingesetzt.
Für die Applikationen mit Standardaufgaben werden die Netzwerkschnittstelle und die Aufgaben in LonMark standardisiert und als **Funktionprofile** (functional profiles) bezeichnet. Beispiele sind das Lamp Actuator, das Switch und das Constant Light Controller Objekt.
Freie Programmierung erfolgt z. T. über Neuron C (z.B. mit dem NodeBuilder der Firma Echelon), eine ANSI C Erweiterung oder über graphische Programmierung (z.b. mit dem Programmiersystem IPOCS der Firma SysMik).
Für die Festlegung der Kommunikation zwischen den Geräten (das "Binding"), die Inbetriebnahme und die Verwaltung in LON Netzen werden Netzwerkmanagementtools eingesetzt. Für den physikalischen Zugriff auf die LON Netze werden Netzwerkschnittstellen verschiedener Arten eingesetzt, u.a. PC Einsteckkarten.

Frameworks

i. Windows

OPC - eine herstellerunabhängige standardisierte Software-Schnittstelle für die Windows-Plattform im z.B. EIB- und LON - Systeme. Darauf wird GLT(Gebäudeleittechnik) angelegt, um Bussystem entfernt und visuell zu überwachen. Das wird schon im Kapital 3.2.2 genannt.

ii. Java

OSGi - Middleware-Standard (Java-Framework) für die Einbindung von EHS/CHAIN, EIB, Konnex, LON, etc. in Service-Gateways Das wird schon im Kapital 3.2.3 erklärt.

5.2.3 LCN

LCN (Local Control Network) ist ein universelles Gebäudeautomationssystem, welches vorwiegend in **Wohn- und Zweckbauten** zum Einsatz kommt. Seit seiner erstmaligen Vorstellung an der Industriemesse in Hannover im Jahre 1992 wird LCN in vielen Objekten unterschiedlichster Nutzung und Komplexität eingesetzt. LCN wird im deutschsprachigen Raum als eines der bedeutendsten Bussysteme angesehen.
LCN ist ein proprietäres Gebäudeleitsystem, welches vom deutschen Hard- und Software-Entwicklungs-Unternehmen Issendorff Mikroelektronik GmbH entwickelt wird. LCN unterscheidet sich von den anderen Bussystemen hinsichtlich seines durchgehenden und konsequenten Grundkonzeptes. Seit seiner Entstehung waren der typische Einsatz im Wohn- und

Zweckbau sowie die Einbettung des Systems in die konventionelle Elektroinstallation zentrale Anforderungen an die Funktionsweise und den Aufbau von LCN. Seit dem Start der serienmäßigen Produktion im Jahre 1993 wurde LCN kontinuierlich weiterentwickelt und optimiert. Die LCN-Technologie verfügt auch heute noch über freie Kapazitäten, um neuen und zukünftigen, heute noch nicht absehbaren Anforderungen gerecht zu werden.

LCN-Modul

LCN-Modul ist ein intelligenter und programmierbarer Mikrocomputer. Die LCN-Moduln sind absolut identisch, und werden nach die genau gleiche Art und Weise programmiert. **Das heißt, LCN-Modul kann die komplexen Aufgaben bearbeiten, ohne Verteilungen der Aufgabe in einfacheren Teilaufgaben.** Mit dem LCN-Modul-Konzept unterscheidet sich der LCN-Bus von dem CAN- und dem LON-Bus, welche eine Hierarchie von Teilaufgaben und Programme in übergeordneten SPSs, DDCs und LON-Geräte besitzen.
Die einzelnen LCN-Module unterscheiden sich bezüglich der Bauform und bezüglich der Hardwareausstattung. So gibt es hauptsächlich zwei verschiedene Bauformen der LCN-Module: Für den zentralen Einsatz in der Verteilung ist das „**Hutschienenmodul“**[15]. Das ist eine klassische Bauform. Und für den dezentralen Einsatz in Unterputz- oder Abzweigdosen ist „**Unterputz-Modul (UP)“**[15]. Sie werden in einer tiefen UP-Dose unmittelbar am Einbauorte der Sensoren und Aktoren installiert.
Jedes Modul verfügt bereits über zwei unabhängig voneinander ansteuerbare, dimmbare Aktorausgänge mit 2kVA. Zusätzlich kann jedes Modul mit einer breiten Palette von Ergänzungskomponenten wie Sensoren, Tastern, Aktoren, Kopplern usw. ergänzt werden. Dadurch sind LCN-Module überaus flexibel und für die unterschiedlichsten Anwendungen einsetzbar: Z.B. im autarken Einsatz als leistungsfähige Licht-/Heizungssteuerung eines einzelnen Raumes, oder als Bestandteil eines vernetzten Verbundes von mehreren Modulen für die umfassende Steuerung und Visualisierung von Licht, Rollladen, Heizung, Raumüberwachung im Wohnbau, bis hin zum integrierten Teilnehmer von umfangreichen Großprojekten im industriellen Bereich.

Netzwerktopologie und Übertragungsmedien

LCN ist ein dezentral organisiertes Netzwerk. Mit LCN können beliebige Topologien(Linie, Ring, Stern oder Baum) realisiert werden.
Jeder LCN-Modul hat ein **8Bit** lange Adresse (Modul-ID). Es gibt 256 Adressen[15]. Die Adressen 1…4 werden bei LCN systemintern zur Adressierung des Programmier-PC und des Visualisierungs-PC genutzt. Deshalb für jedes Segment darf man maximal 250 Moduln[15] in beliebiger Struktur (sternförmig, linien- oder baumförmig) direkt miteinander verbinden. Es gibt maximal 120 Segmente[15] miteinander zu verbinden. Damit kann ein LCN-System im Maximalausbau bis zu **30.000 Module**[15] umfassen.
Zur Verkabelung der Bus-Moduln werden **Zweidrahtleitungen nach DIN VDE 250** mit Querschnitten von 1,5mm^2 und 2,5mm^2 eingesetzt. Zusätzlich gibt es noch eine Ader mit max. 30V Spannung zur Datenübertragung, sind die Leitungen 5-adrig. Der maximale Abstand zwischen den zwei Moduln ist **1000m**[15]. Im Segment ist der Abstand **20m**[15].
Natürlich können auch LWL im LCN-Bus eingesetzt werden.

Kommunikationsprotokoll

Da alle Bus-Moduln gleichberechtigt (Mulitimaster-Prinzip) auf den Bus zugreifen Können, erfolgt der Zugriff auf den gemeinsamen Übertragungskanal nach dem **CSMA/CA**-Verfahren. Die Datenübertragungsrate kann max. **9600Bit/s**[15] erreichen.

Zur Datenübertragung im LCN-segmentbus wird das deterministische Verfahren des **Token Passing** genutzt.
Das zwischen Moduln ausgetauschte Telegramm besteht aus der Quell-, Ziel- und Segmentadresse, einem mindestens **24Bit**[15] langen Datenfeld, ein Info-Feld und ein Feld mit einer Prüfsumme, wie Bild 5.2.3.1[15].

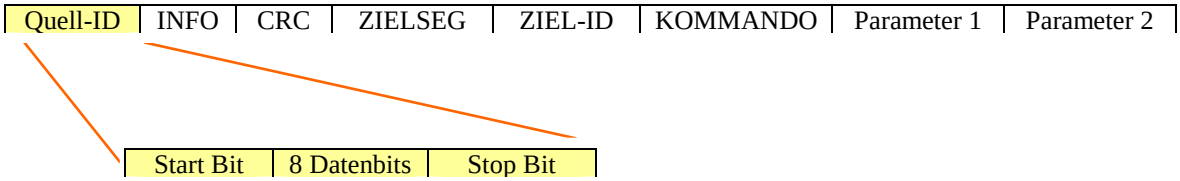


Bild 5.2.3.1 LCN-Telegramm

Programmierung (Software)

„Die Programmierung von LCN erfolgt durch den qualifizierten Elektroinstallateur mit Hilfe der LCN-Programmiersoftware, welche auf einem PC oder Laptop installiert wird.“[16] Der PC wird über den LCN-Koppler an beliebiger Stelle im LCN-Netzwerk angeschlossen. Die Bus-Parametrierung befindet sich nicht nur im Bus, sondern auch als Projekt-Kopie auf dem Rechner. Projekte können deshalb auch unabhängig vom LCN-System geschrieben und anschließend einfach ins LCN-Netzwerk übertragen werden.
Das LCN-System ist abwärtskompatibel: Dies bedeutet, dass die neusten Versionen der LCN-Software und Hardware einwandfrei mit den ältesten LCN-Komponenten zusammenarbeiten.

5.3 Feldbus als Ersatz der Ein-/Ausgangskarten

Um den Verkabelungsaufwand bei der Installation von Peripheriekomponenten zu geringen, gibt es im Feldbussystem einen Prozess, die Ein-/Ausgangskarten für die Steuerungen werden durch E/A-Module ersetzt. Diese Module können die gewünschte Ein-/Ausgangsfunktionalität bieten. In der Klasse geben zwei Feldbussysteme zur Beispiele: Interbus und Profibus DP

5.3.1 Interbus

Interbus ist seit 1987 von Phonenix Contact entwickelt und in 1994 in DIN 19258 genormt. An Anfang 1999 soll EN 50254 in Europa durchgesetzt werden. Das System erfolgt als erstes SPS-Hersteller-unabhängiges System.

Netzwerktopologie

Interbus arbeitet mit einem **Master-Slave-Zugriffverfahren**. Wie Bild 5.3.1.1[11] ist die Bustopologie ein aktiver **Ring**.

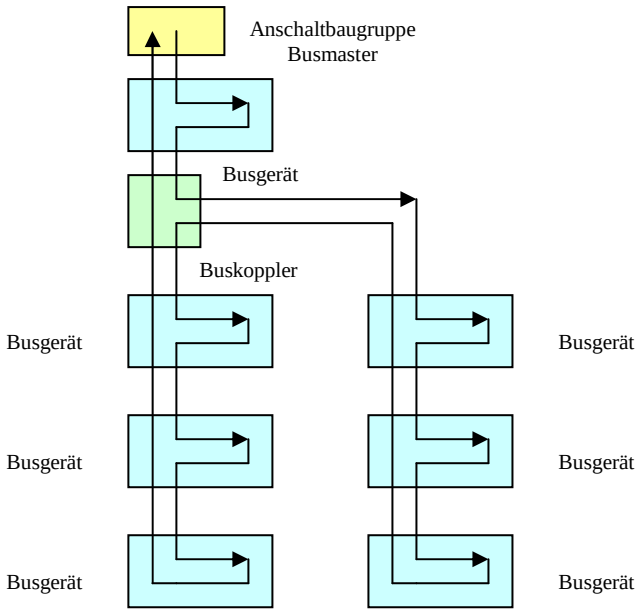


Bild 5.3.1.1 Ringbus bei Interbus

Interbus unterscheidet zwischen Fernbus und Lokalbus. Der Fernbus verbindet maximal **256 Teilnehmer**[11] mit einer Gesamtausdehnung von **13km**[11], der Abstand zwischen den Teilnehmern darf 400m sein und die Datenübertragungsrate ist **500kBit/s**[11]. Im Lokalbus dürfen nur max. **8 Teilnehmer**[11] mit einem Gesamtausdehnung von **10m** anschließen. Die Entfernung zwischen Teilnehmern darf 1,5m[11] nicht überschreiten. Die Übertragungsgeschwindigkeit ist **300kBit/s**[11].
Der Lokalbus startet an einem speziellen Busteilnehmer, einem Buskoppler. In diesem Buskoppler wird eine Schnittstelle für den Lokalbus bereitgestellt. Und der Koppler kann auch ein weiteres Fernbus-Segment angeschlossen werden. Dadurch kann Interbus auch mit Bautopologie verstanden werden.

Übertragungsmedien

Die Verbindungsstruktur im Interbussystem ist **Punkt-zu-Punkt-Struktur**. Als typische Übertragungsmedium wird **ein geschirmtes Rundkabel mit zwei Adernpaaren nach dem Standard RS 485** verwendet.
Neben dem klassen Kupferkabel sind auch Lichtwellenleitern verwendet, um die max. Ausdehnung des Interbus-Netzwerks zu vergrößern und die Störsicherheit zu erhöhen.

Kommunikationsprotokoll

Der Interbus-Protokollstack ist entsprechend dem IOS7OSI-Modul in **drei Schichten** aufgebaut. Die Datenübertragungsverfahren in Interbus-Netzwerk unterscheiden sich stark von anderen Feldbussystemen.
In Interbus existieren zwei unterschiedliche Übertragungszyklen: der Identzyklus und der Nutzdatenzyklus. Die Beide haben ein gleiches Übertragungsschema.
Wenn der Master einen Überblick über die aktuelle Konfiguration des ganzen Bussystems haben will, beginnt er einen **Identzyklus**.
Im Interbus für die in den Sensoren und Aktoren vorkommenden Datenklassen gibt es die zyklische Prozessdaten und die azyklische Parametern. Ein Datenrahmen ist nur **16Bit**[10], damit kann der Master den Busteilnehmern zuordnen.

Im **Nutzdatenzyklus** werden die Ein-/Ausgabendaten für die Endgeräte übertragen. Hier werden die Daten von **zyklischen Prozessdaten** und **azyklische Parametern** unterschieden, wie Bild 5.3.1.2[10].

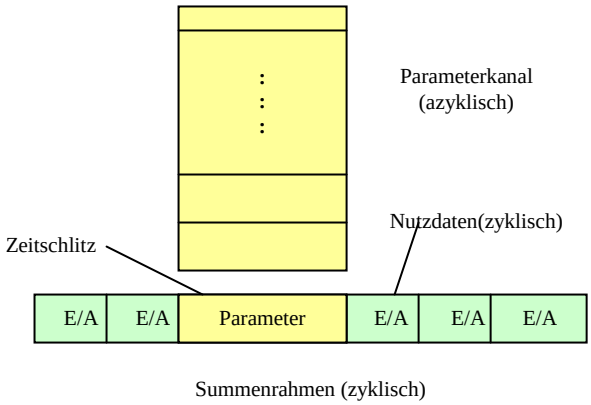


Bild 5.3.1.2 Übertragung der Parameter im Nutzdatenrahmen

Im Prozessdatenkanal werden zyklisch kurze Datenpakete zwischen dem Master und den Endgeräten ausgetauscht. Das Paket ist groß bis zu **64Byte**[10]. Zusätzlich im Telegrammrahmen werden 2 – 16Byte[10] breite Zeitschlitze freigehalten.
In diese Zeitschlitze werden die Parameter aus den Parameterblöcken sequentiell eingefügt. Mit den Parametern werden azyklisch größere Datenmengen zwischen dem Master und den speziellen Endgeräte ausgetauscht.
Umfassung mit Parameter und Prozessdaten ist der Summenrahmen max. **512Byte**[10].
Das Summenübertragungsverfahren wird beim Interbus durch eine **Schieberegisterstrukur** (Bild 5.3.1.3[10]) realisiert. Jeder Interbus-Teilnehmer (Master) besitzt ein Schieberegister. Durch die Schieberegister tauscht der Teilnehmer die Daten mit dem Bus aus. Seine Länge wird durch die Anzahl der Prozessdatenpunkte des Teilnehmers festgelegt.

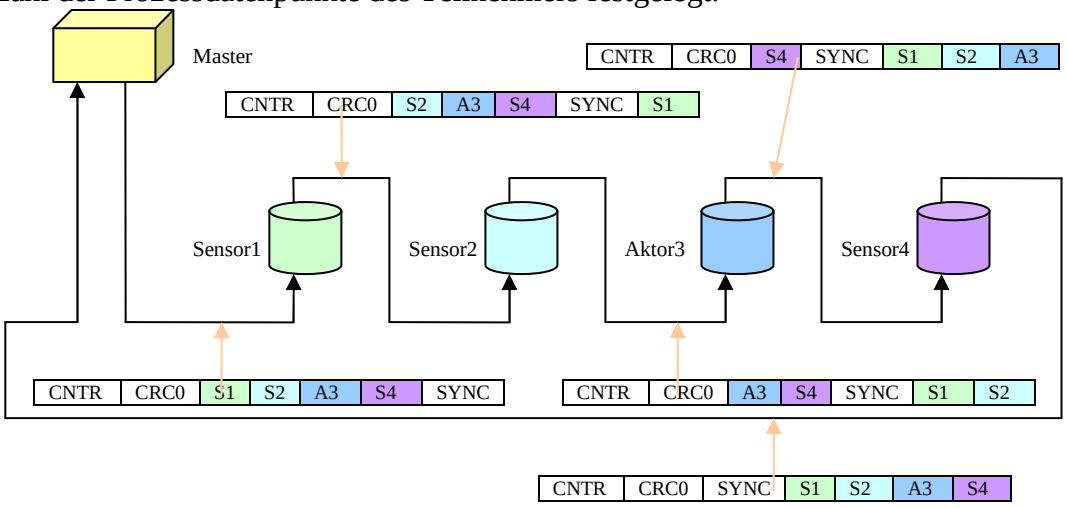


Bild 5.3.1.3 Schiederegister-Struktur

Durch die Aneinanderkopplung aller Teilnehmer ergibt sich ein Schieberegisterring, dessen Länge und Struktur genau dem Aufbau des Nutzdatenfeldes im Summenrahmentelegramm entspricht. Die Prozessdaten werden aus der Peripherie im Ausgabebuffer des Masters hinterlegt. Während diese Datenausgabe durchgeführt wird, erfolgt gleichzeitig der Rückfluss von Prozessinformationen als Eingabedaten in den Eingangsbuffer des Masters. Der Übertragungszyklus erfolgt durch den Master und wird durch die Schieberegister getaktet. Damit können die Sensoren und Aktoren seriell mit dem Master die Daten austauschen.

Jedes zyklische Telegramm beginnt mit einem **16Bit**[10] Loopback-Wort, die als erst Informationen vom Master auf den Ring ausgegeben werden. Sie durchlaufen im Ringsystem und werden als letzte Eingangsinformationen wieder in den Master zurückgelesen.
Nach dem Loopback-Wort folgen die eigentlichen Nutzdateninformationen des Interbus-Systems. Am Ende des Nutzdatenblocks schließt sich die Übertragung einer 16Bit[10] CRC-Check nach CCITT für Datensicherung.
Nach den CRC-Daten folgen noch 16Bit[10], in denen die einzelnen Teilnehmer die fehlerfreie Datenübertragung an den Master zurückbestätigen.
Wie Bild 5.3.1.4[10] für je Informationen, die oben schon genannt werden, gibt es ein Startbit und ein Stopbit, um asynchrones Verfahren zu realisieren.
Für je ganze zyklische Datenblock gibt es ein 3Bit[10] Header, der zusätzliche Informationen wie Zyklusart, Zyklussequenz und Telegrammtyp enthält.

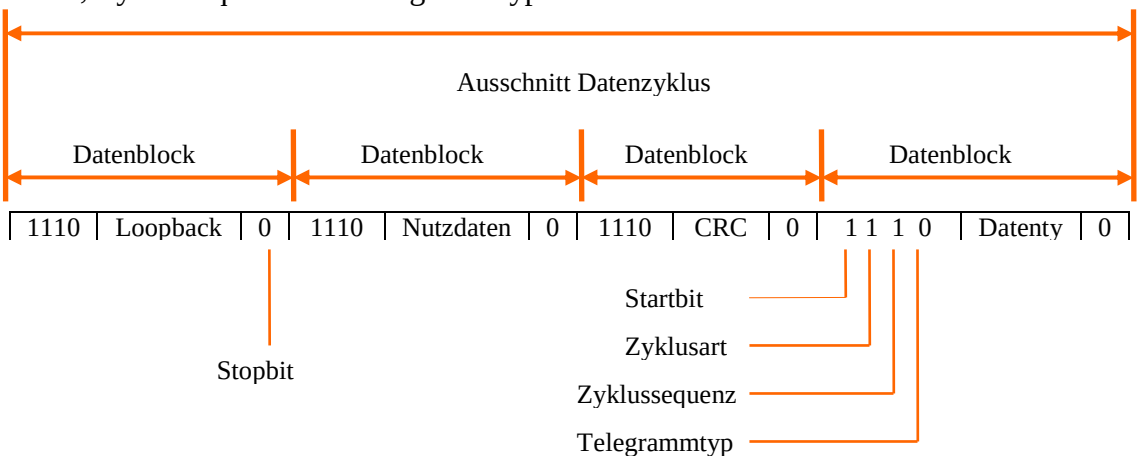


Bild 5.3.1.4 zyklische Datenrahmen von Interbus

5.3.2 Profibus DP

PROFIBUS (Process Field Bus) ist ein Standard für die Feldbus-Kommunikation in der Automatisierungstechnik und wurde anfangs (1989) vom BMFT gefördert. Es werden lila Kabel verwendet. Der PROFIBUS hat in Europa in der Fertigungstechnik einen Marktanteil von mehr als 60 %[17]. Weltweit sind über 10 Millionen[17] Geräte mit PROFIBUS im Einsatz (Stand 2004). Zurzeit gehen Marktanteile in Richtung Industrial Ethernet verloren[22].
Profibus DP (Dezentrale Peripherie) geht aus dem Profibus FMS hervor, weil FMS Nachteile besitzt, wie z.B. komplexe Kommunikationsmöglichkeiten, Unanpassung der untersten Feldebene an Zeitanforderung.
DP steuert die Sensoren und Aktoren durch eine **zentrale Steuerung** in der Fertigungstechnik. Weitere Einsatzgebiete sind die Verbindung von "verteilter Intelligenz", also die Vernetzung von mehreren Steuerungen untereinander (ähnlich PROFIBUS-FMS). Es sind Datenraten bis zu **12 Mbps**[17] auf **verdrillten Zweidrahtleitungen und/oder Lichtwellenleiter** möglich

Netzwerktopologie und -medien

Im Profibus verwendet die Topologie die **Linienstruktur**, d.h. die Teilnehmer sind in einer Reihe an einem Kabel angeschlossen.
Die max. Länge des Kabels ist abhängig von der Übertragungsrate, die Verhältnisse werden in Tabelle 5.3.2.1[11] dargestellt. Zu Erweiterung des Netzwerks kann man Repeater einsetzen.
Im Netzwerk verwendet man meinst nach DIN 19245 Teil 3 eine abgeschirmte, verdrillte Zweidrahtleitung als Übertragungsmedium. Die Schnittstelle ist nach der RS 485 – Norm definiert. Jedes Segment darf max. **32 Geräte** daran anschlossen werden.

Tabelle 5.3.2.1 Zusammenhang zwischen Kabellänge und Übertragungsrate

Übertragungsrate	**Max. Segmentlänge**	**Repeater**	**Max. Buslänge**
9,6kBit/s, 19,2kBit/s, 93,75kBit/s	1200	7	9600m
187,5kBit/s	600m	4	2400m
500kBit/s	200m		800m
1,5MBit/s			
12MBit/s	100m	2	200m

An jedem Ende des Kupferkabels gibt es einen Abschlusswiderstand.
In Profibus-Netzwerk gibt eine Möglichkeit, Lichtwellenleiter zu einsetzen.

Kommunikationsprotokoll

Im Profibussystem ist Master aktiv und sind die Slaves passiv. In diesem System können sowohl Einzelmastersysteme als auch **Multimastersystem** aufgebaut werden, d.h. die Struktur von Profibussystem kann **zentral oder dezentral** sein.
Bei der dezentralen Struktur werden die Mastern voneinander durch einen logischen Tokenring verbunden, wie Bild 5.3.2.1[11].

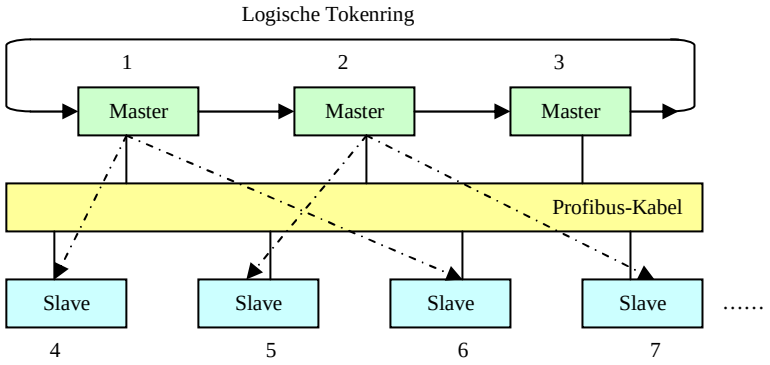


Bild 5.3.2.1 Zugriffsverfahren bei Multimastersystemen

Das Token wird von Master zu Master in der Reihenfolge weitergegeben. Jeder Master fragt zyklisch seine Slaves ab. In Profibus DP bedient je Master 2Byte Ein-/Ausgangsdaten zu den Slaves. Nach Abfrage der Slaves geht das Token an den nächsten Master weiter. Daher mit dem Multimasterverfahren ist die Zykluszeit abhängig von den Zahlen der Slaves je Master, wie Tabelle 5.3.2.2[10].

Tabelle 5.3.2.2 Zykluszeit je Master mit unterschiedlicher Anzahl von Slaves

Anzahl der Slaves	1	5	10	20	30
Zykluszeit(500kBit/s)	2ms	3ms	6ms	10ms	16ms
Zykluszeit(1,5MBit/s)	2ms	2ms	2ms	4ms	6ms
Zykluszeit(12MBit/s)	<1ms				

Innerhalb der Telegramme wird ein genormtes UART-Zeichen mit 11Bit Länge verwendet. Es besteht aus Startbit, 8 Datenbits, Paritätsbit und Stopbit.

UART-Zeichen:

SB	D0 … D7	PB	EB

Es werden vier verschiedene Telegrammtypen verwendet und durch den SD (Startdelimiter) unterschieden:
✓ Paket ohne Daten (L=3)

SD1	DA	SA	FC	FCS	ED

✓ Paket mit Daten und fester Informationsfeldlänge (L=4…249)

SD2	LE	LEr	SD2	DA	SA	FC	Daten	FCS	ED

✓ Paket mit Daten und variabler Informationsfeldlänge (L=11)

SD3	DA	SA	FC	Daten	FCS	ED

✓ Tokenpaket

SD4	DA	SA

SB: Startbit
D0…D7: Datenbits
PB: Paritätsbit
EB: Stopbit
SD1…4: Startbyte, dient zur Unterscheidung verschiedener Telegrammformate
LE: Länge der Nettodaten, (incl. DA,SA,FC,DSAP,SSAP)
LEr: Wiederholung der Nettodaten Länge
DA: Adress des Empfängers
SA: Adress des Senders
FC: Kontrollbyte, kennzeichnet Telegrammtyp
FCS: Prüfbyte, dient zur Kontrolle der richtigen Übertragung der gesendeten Daten
ED: Endbyte, Endbegrenzung des Telegramms

PROFIBUS PA

„PROFIBUS PA (Prozess-Automation) wird zur Kontrolle von Feldgeräten durch ein Prozessleitsystem in der Prozess- und Verfahrenstechnik eingesetzt.“[17]
Dieser Feldbus wird für den explosionsgefährdeten Bereich verwendet. Die Schwerpunkte des Systems sind die Sicherheiten. Dafür werden die Leitungen und die Übertragungsverfahren verändert. **D.h. PROFIBUS PA ist für explosionsgefährdete Bereiche (Ex-Zone 0 und 1) geeignet.** Hier fließt auf den Busleitungen in einem eigenen Stromkreis nur ein schwacher Strom, so dass auch im Störfall keine Funken entstehen können.
Eine **verdrillte Zweidrahtleitung** wird als Übertragungsmedium verwendet und an beiden Enden des Kabels befinden sich Abschlusswiderstände. Die Bustopologie kann bis zu **1900 Meter**[17] lang sein und lässt Abzweigungen zu den Feldgeräten mit maximal **30 Meter** Länge zu. An dem Bus dürfen max. **32 Teilnehmer**[17] anschließen werden. Die Bitrate wird als **31,25 kBit/s**[17] festgelegt. Der Nachteil dieser Variante ist die langsamere Datenübertragungsrate.

5.4 Zellenbussysteme

Zellenbussysteme sind gegen den klassischen Feldbussystemen eine Stufe höher in der Automatisierungsebene angesetzt. Sie werden zur Vernetzung von Steuerungen untereinander und zur Anbindung der Automatisierungsgeräte an Visualisierungssysteme und Datenbanken verwendet. Aufgabe der Zellenbussysteme ist die Übertragung größerer Datenmengen. Bei Zellenbussystemen ist ein deterministischer Betrieb nicht mehrer erforderlich. Für dieses System gibt es ein Beispiel – Profibus FMS.

5.4.1 Profibus FMS

Profibus FMS (Fieldbus Message Specificatin) benutzt das gleiche Übertragungsverfahren wie Profibus DP im Kapital 5.3.2. Eigentlich sind die Kommunikation, das Zugriffverfahren und die Kodierung der einzelnen übertragenen Zeichen aus Profibus DP entwickelt.
Auf der **Schicht 7** (Anwendungsschicht) bietet Profibus FMS eine objektorientierte Kommunikation, damit größere strukturierte Datenmengen zu übertragen.
Die Objekte unterscheiden sich in zwei Gruppen:

i. Statische Objekte
   - Variable
   - Domain
   - Event

ii. Dynamische Objekte
   - Variable-List
   - Programm Invocation

5.5 BACnet

Im Weltweit gibt es zahlreiche unterschiedliche Feldbusse, die von verschiedenen Organisationen und Kommissionen nach verschiedene Protokollen für ungleiche Anwendungsziele entwickelt werden. Viele Feldbusse passen konzeptionell nicht in die bisher vorgestellten vier Klassen. Z.B. BACnet ist eine Feldbus-Technik für die **Gebäudeautomatisierung**, die viel nicht nur andere Feldbus-Techniken, sondern auch die Rechnernetzwerk-Techniken kombiniert.
BACnet (Building Automation and Control Networks) ist ein Netzwerkprotokoll für die Gebäudeautomation. BACnet ist ein Standard der ASHRAE (American Society of Heating, Refrigerating and Air-Conditioning Engineers) und wurde vom ANSI übernommen. Seit 2003 ist BACnet auch ISO-Standard.
Die Entwicklung des BACnet Protokolls begann im Juni 1987, um einen einheitlichen firmenneutralen Standard für die Datenkommunikation in und mit Systemen der Gebäudeautomation bereitzustellen. BACnet wurde im Jahr 1995 ASHRAE/ANSI Standard (135)[18], im Jahr 2003 ISO-Standard (16484-5)[18] und wird in verschiedenen Arbeitsgruppen ständig weiterentwickelt.

Konzept

BACnet gewährleistet Interoperabilität zwischen Geräten verschiedener Hersteller, wenn sich alle am Projekt beteiligten Partner auf bestimmte vom Standard definierte BIBBs einigen. Ein **BIBB (BACnet Interoperability Building Block)** definiert welche Services und Prozeduren auf Server- und Client-Seite unterstützt werden müssen, um eine bestimmte Anforderung des Systems zu realisieren. Das zu einem Gerät gehörende Dokument PICS (Protocol Implementation Conformance Statement) listet alle unterstützten BIBBs, Objekt-Typen, Zeichensätze und Optionen der Kommunikation auf.

Beschreibung

Der Standard definiert eine Reihe von **Diensten (Services)**, die zur Kommunikation zwischen Geräten der Gebäudeautomation verwendet werden. Diese Dienste gliedern sich in verschiedene Gruppen: Gemeinsame Datennutzung, Alarm- und Ereignisverarbeitung, Verarbeitung von Wertänderungen, Geräte- und Netzwerk-Management, usw.

Der Standard definiert verschiedene **Typen von Objekten**: Gerät, Benachrichtigung, Trendaufzeichnung, Kalender und Zeitplan; Eingang, Ausgang und Wert jeweils analog, binär oder mehrstufig; Zählereingang, Programm, Regler, usw.
Der Standard definiert außerdem **Prozeduren** für die Alarmverarbeitung.

4-Schichten-Kommunikation

BACnet definiert eine zusammengefasste 4-Schichten-Kommunikation als Bild 5.5.1.

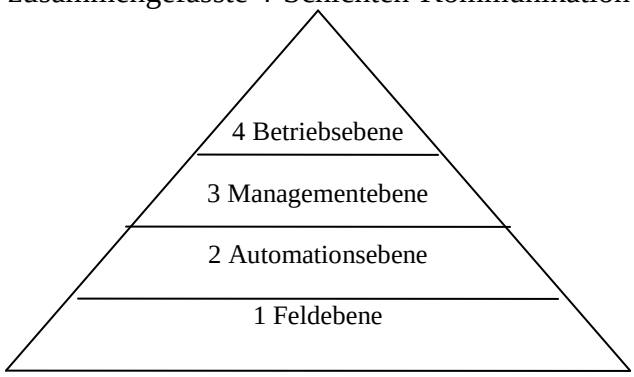


Bild 5.5.1 4-Schichten-Pyramide

Entsprechend der **4-Schichten-Pyramid-Struktur** wird die Techniken, die von BACnet zusammengenutzt werden, in Tabelle 5.5.1 vom Technischen Komitee 247 der CEN geordnet.

Tabelle 5.5.1 Normenvorschläge des CEN (TC247) für die Bussysteme in der GA

Ebene	**Bussystem(Protokoll)**	
	Bezeichnung	**Norm (bereits existierende Norm)**
Betriebs- oder Managementebene	BACnet	EN V 1805-1/2 (ANSI/ASHRAE 135-1995)
Automationsebene	BACnet Auf LonTalk oder Ethernet/IP	EN V 13321-1
	PROFIBUS FMS	EN V 13321-1 (EN 50170, Vol.2)
	EIBnet Auf Ethernet/IP	EN V 13321-2
Feldebene	LonTalk	EN V 13154-2 (ANSI/EIA 709.1)
	KONNEX (BATIBUS; EIB;EHS)	EN V 13154-2 (EIB: DIN VDE 0829)

Die folgende Alternativen sind auch für die Schicht 1 und 2 bietet:
i. ARCNET(Attached Resources Computer Network)
ii. BACnet/IP
iii. PTP (Point-To-Point) über RS-232
iv. MS/TP (Master-Slave/Token-Passing) über RS-485.

5.6 Fazit

In oberen Kapitalen werden 9 Arten von Feldbusse mit den Übertragungstopologien, -medien, Zugriffverfahren usw. schon erklärt.
Wegen der verschiedenen Übertragungstopologien ist das Zugriffverfahren jedes Bussystems für die Datenübertragungen ungleich. Und damit ist die Datenübertragungsrate auch unterschiedlich. In der unteren Tabelle[4] werden die Eigenschaften jedes Zugriffverfahrens und die entsprechende Feldbussysteme, die das Zugriffverfahren anwendet, dargestellt.

Tabelle 5.6.1 wichtige Buszugriffsverfahren und die Techniken benutzte Feldbus

Name	Verfahren	Nachteil	Feldbus
Master-Slave	Ein zentraler Teilnehmer steuert den Buszugriff(Monomaster system)	i. Abfrage der Slaves auch dann, wenn keine neuen Daten vorliegen ii. Hohe Zykluszeit bei vielen Teilnehmern iii. Master-Ausfall führt zu Systemausfall	PROFIBUS-DP AS-Interface Interbus
Tokenumlauf (token passing)	Buszugriffsbereichtigung wird von Teilnehmer zu Teilnehmer in einem logischen Ring weitergegeben(„flying Master“, Multi-Mastersystem)	i. Hohe Tokenumlaufzeit bei vielen Teilnehmern ii. Teilnehmer bekommt token, auch wenn keine neuen Daten vorliegen	PROFIBUS-FMS (Es wird ab 2007 nicht mehr normiert sein)
CSMA	Jeder Teilnehmer versucht unabhängig den Buszugriff(Multi-Mastersystem)	i. Kein deterministisches Verfahren ii. Bei hoher Busbelastung sind Kollisionen und damit Wartezeit wahrscheinlich	EIB CAN LON LCN

Die Tabelle unterteilt sich in mehrere Teiltabellen. Zwischen den 8 Feldbussystemen darf man in der Tabellegruppe 5.6.2(1)(2)(3)[4] einige wichtige Eigenschaften auswählen und vergleichen.

Tabelle 5.6.2 Vergleichug von einigen wichtigen Eingenschaften der Feldbussysteme

(1)

Protokoll	Ursprüngliche Zielstellung
AS-Interface	serielle Verdrahtung von Sensoren und Akoren und Ersetzung der Parallelverkabelung
EIB	Installationsbus(Schalten von Verbrauchern)
CAN	Schneller Aktor-Sensor-Bus für mobile Objekte
LON	Universeller Einsatz(80% aller Kommunikationsaufgaben)
LCN	Raumbezogene Gebäudeautomation
Interbus	Schneller Aktor-Sensor-Bus für Indutrieautomation
Profibus DP	Schneller Aktor-Sensor-Bus(Fertigungsautomation)
Profibus FMS	Leichtungsfähige Kommunikation auf Zellenebene(Fertigungsautomation)

(2)

Protokoll	Merkmale				
	MAC	ÜR(kB/s)	Nutzdaten (Byte) Max.	Teilnehmer pro Segment max.	Leitungslänge pro Segment max.
AS-Interface	Master-Slave	167	4 Bit	62 pro Segment	100m
EIB	CSMA	9,6	14	256 pro Segment 57.600 total	1000m
CAN	CSMA	1000	8	128 pro Segment	500m
LON	CSMA	1250	31	127 pro Segment 32385 total	2700m
LCN	CSMA	9,6	20	250 pro Segment	1000m

				30000 total	
Interbus	Master-Slave	500	512	256 total	400
Profibus DP	Master-Slave	12000	249	32 pro Segment	1200
Profibus FMS	Token oder Master-Slave	500	249	32 pro Segment	1200

(3)

Protokoll	**Merkmale**				
	Verfügbare Medien	**Topologie**	**Interoperabilität**	**Komponenten für Gebäudeautomation**	**Intra-/Internet**
AS-Interface	-	-	-	-	-
EIB	++	+	+++	+++	++
CAN	-	-	-	-	+++
LON	+++	+++	+++	+++	+++
LCN	-	-	+++	++	-
Interbus	-	-	-	-	+++
Profibus DP	-	-	-	-	++
Profibus FMS	-	-	-	-	++

+++: sehr ausgezeichnet - : eingeschränkt

Eigentlich sind nur die Feldbussystemen EIB, LON, LCN und BACnet speziell für die Gebäudeautomationen entwickelt worden. CAN-Bus und Interbus wurden für die Industrieautomation, z.B. Automobilindustrie entwickelt. Profibus ist ein Bussystem für Fertigungs- und Prozessautomatisierung und wird auch in explosionsgefährdeten Bereichen angewendet. Und AS-Interface kann man auch in den explosionsgefährdeten Bereichen anwenden.
Aus Tabelle 5.6.2 kann man sagen, LON-Bus ist das günstigste Bussystem für Gebäudeautomation. Und LON-Bus kann auch eine Infrastruktur auf den zwei untersten Schichten von BACnet-Bus bilden.

6. Erweitungen von LON

LON (Local Operating Network) ist ein herstellerübergreifender Feldbus, der insbesondere im **Facility Management** Verwendung findet. Er wurde für die **Gebäudeautomation** entwickelt, für einfachere Automatisierungsaufgaben. Die angewandte Technik wird von großen Unternehmen bevorzugt im Zweckbau eingesetzt. Ziel ist es, das Management von größeren Gebäudeanlagen, wie z.B. Bürohäusern, Kliniken oder Flughäfen, mit einem dem offenen Standard BACnet zu verbessern.

6.1 Gebäudeautomation mit LonWorks-Technik

Für die gesamte Gebäudeautomation werden in den Feld- und Automationsebenen LON-Geräte in Kommunikations-Netzwerken auf der Grundlage des LonWorks-Protokolls verwendet. Die Managementebene befindet sich im Intranet (basierend auf TCP/IP und HTTP). Über sie wird das System ferngesteuert (siehe Bild 6.1.1[4])

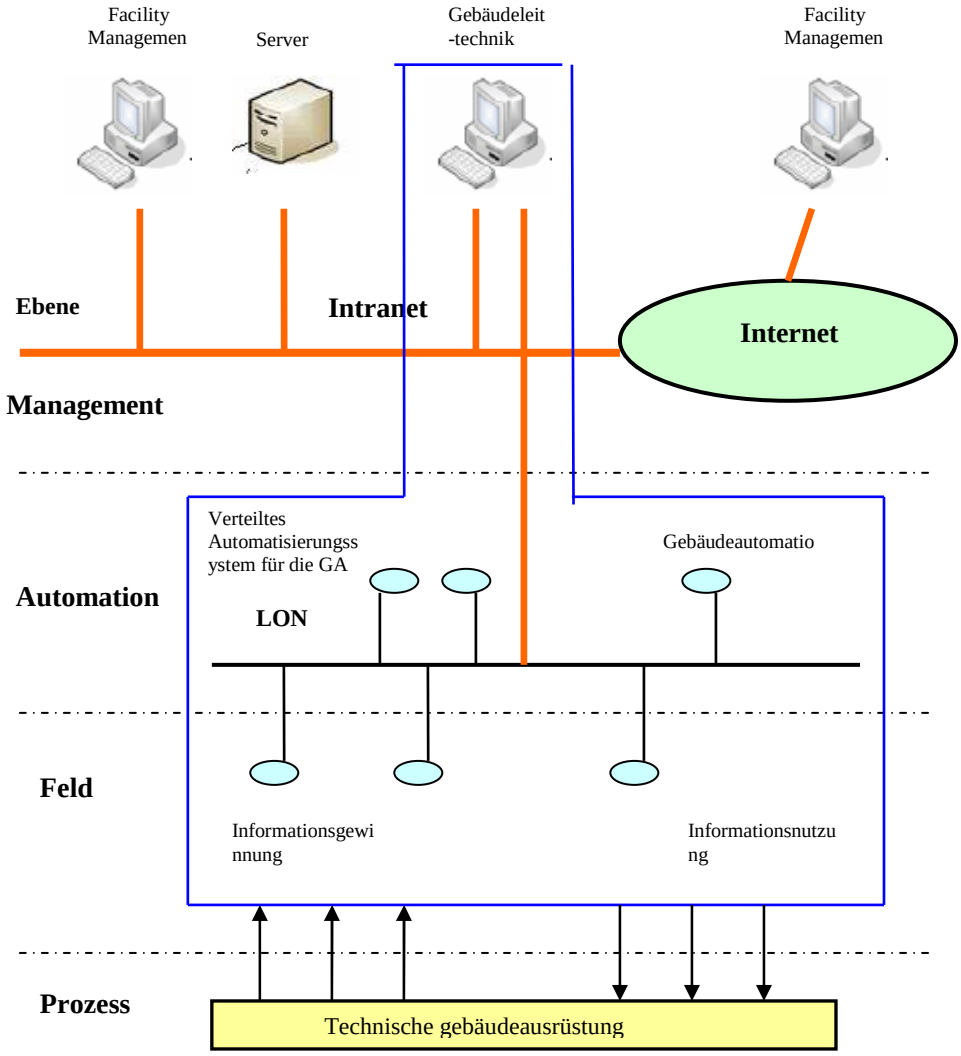
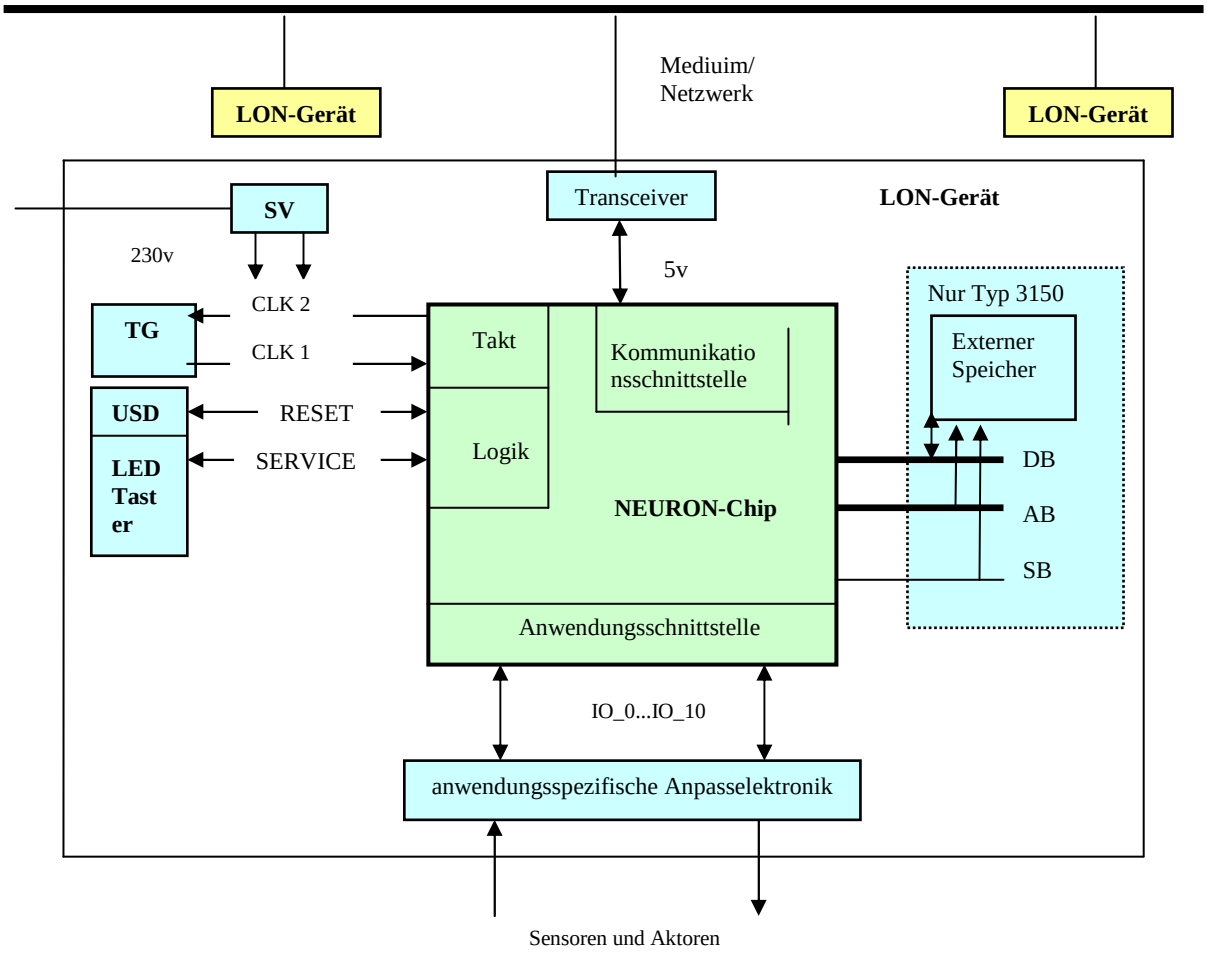


Bild 6.1.1 Einheit von Facilitymanagement und LonWorks-Technik

6.2 LON-Gerät

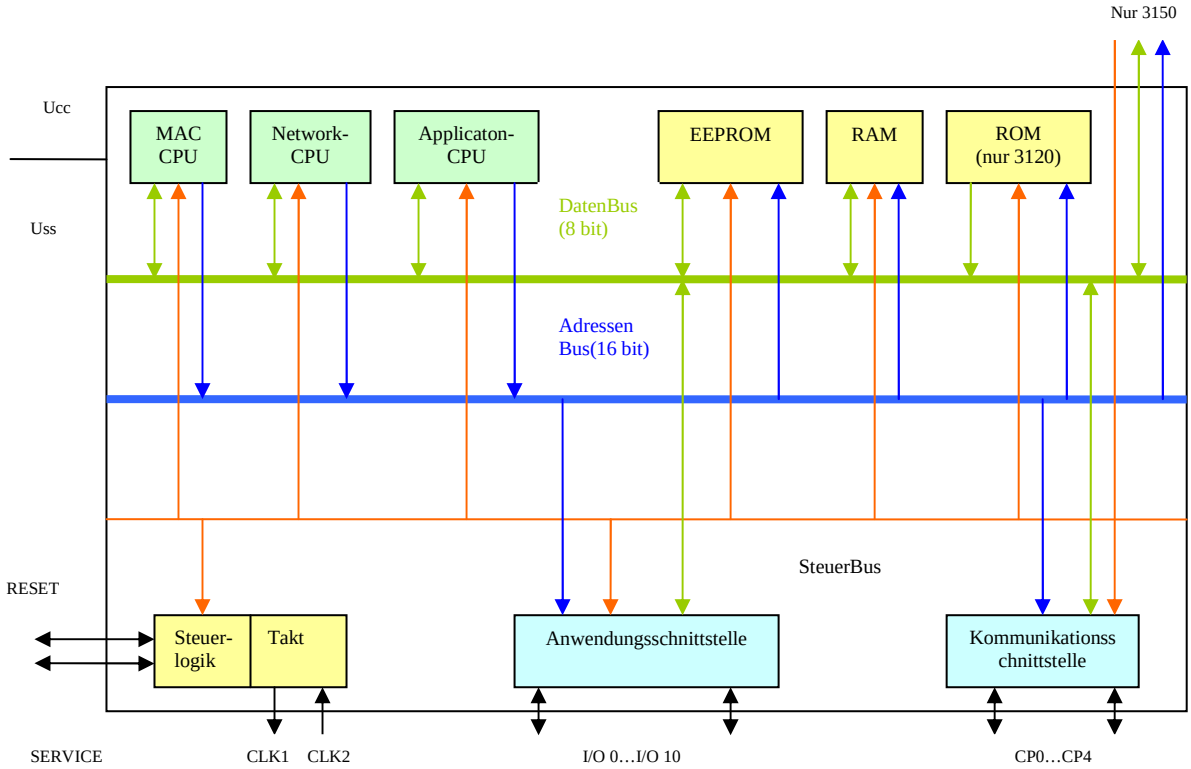
LON-Geräte besitzen ein skalierbares und ein mit **SPS** vergleichbares Leistungsvermögen und können prinzipiell für beliebige Aufgaben in der Gebäudeautomation eingesetzt werden.



SV: Stromversorgung, CLK: Clock, DB: Datenbus, AB: Adressenbus,
SB: Steuerbus, USD: Unterspannungs-Detektion, LED: Light Emitting Diode

Bild 6.2.1 Typischer Aufbau eines LON-Gerätes

Wie in Bild 6.2.1[4] zu sehen, ist der **Neuron-Chip** das dominierende Bauelement.
Der Anschluss der Sensoren/Aktoren und des LON-Geräts, erfolgt über **eine** programmierbare Anwendungsschnittstelle, **anwendungsspezifische Anpasselektronik** genannt.
Die **Kommunikationsschnittstelle** innerhalb des Neuron-Chips steuert einen **Transceiver** (Sende-/Empfangselektronik), mit welchem das Medium an die Informationsübertragung physikalisch angekoppelt wird.
TG (Taktgenerator) und **SV** (Stromversorgung) sind Komponenten des LON-Gerätes. Die externe Energieversorgung erfolgt über einen Netzanschluss oder das Übertragungsmedium (Zweidrahtleitung).
Für auf dem Neuron-Chip laufende Anwendungsprogramme, ist die Möglichkeit der externen Speichererweiterung gegeben.
Die Programmierung und Parametrierung des LON-Gerätes erfolgt über die Kommunikationsschnittstelle oder einen im Gerät eingebetteten, programmierten Speicherchip.
Normalerweise sind die LON-Geräte vorprogrammiert.



MAC: Medium Access Control , CPU: Central Processing Unit

Bild 6.2.2 Hardwarestruktur des Neuron-Chips (Typ 3150)

Der Neuron-Chip stellt eine klassische 8-Bit-Mikrorechnerstruktur mit **drei Prozessoren** dar (siehe Bild 6.2.2[4]).
Eine **MAC-CPU** sowie eine **Network-CPU** realisieren die Kommunikationsaufgaben.
Die **Application-CPU** enthält die lokalen Gerätefunktionen, welche mit Hilfe von Anwendungsprogrammen konfiguriert werden.
Die CPUs besitzen eigene Register (interne Speicher), nutzen jedoch die selbe Verarbeitungseinheit (Anwendungsschnittstelle, Kommunikationsschnittstelle, Steuerlogik, Takt) und den selben Speicher (EEPROM, RAM, ROM) über einen interne Bus (DatenBus, AdresseBus, SteuerBus). Die Zuteilung der Ressourcen auf die CPUs, erfolgt über ein Zeitscheibenverfahren. Auf diese Weise können sie effizient, parallel alle Aufgaben abarbeiten.
Neuron-Chips gibt es in den Varianten: Typ3120, Typ3150. Im Typ3150 sind Speichererweitung und Betriebssystem integriert.
Unter Nutzung der anwendungsspezifischen Anpasselektronik, können LON-Geräte als Sensoren, Steuergeräte oder Aktoren genutzt werden. Die programmierbare Anpasselektronik dient der Signalanpassung und Signalverstärkung. Dafür ist die **Anwendungsschnittstelle** besonders wichtig. Sie stellt die Verbindung vom Neuron-Chip zu Anpasselektronik, Sensoren und/oder Aktor her. Die Anwendungsschnittstelle hat **11 Pins**, von IO_0 bis IO_10. IO_0 bis IO_3 sind mit 20 mA belastbar. IO_4 bis IO_7 dienen der Programmierung. IO_8 bis IO_10 werden für die serielle Kommunikation verwendet.
Einprozessorgeräte können verschiedene Aufgaben nur nacheinander ausführen. Die drei Prozessoren des Neuron-Chips arbeiten hingegen partiell zeitlich parallel zusammen. Auf diese Weise ist es z.B. möglich, wie im Bild 6.2.3[4] zu sehen, Nachrichten in der Hochlaufphase, ohne komplizierte Synchronisationsmechanismen, zu übertragen und weiterzuverarbeiten.
Die Aufgaben werden zyklisch in Schleife abgearbeitet. Die MAC-CPU empfängt Nachrichten und versendet sie an andere Geräte. Die empfangenen Nachrichten werden von der Network-CPU für die Applikation-CPU, welche sie generiert hat, aufgearbeitet und verpackt und anschließend an die MAC-CPU gesandt.

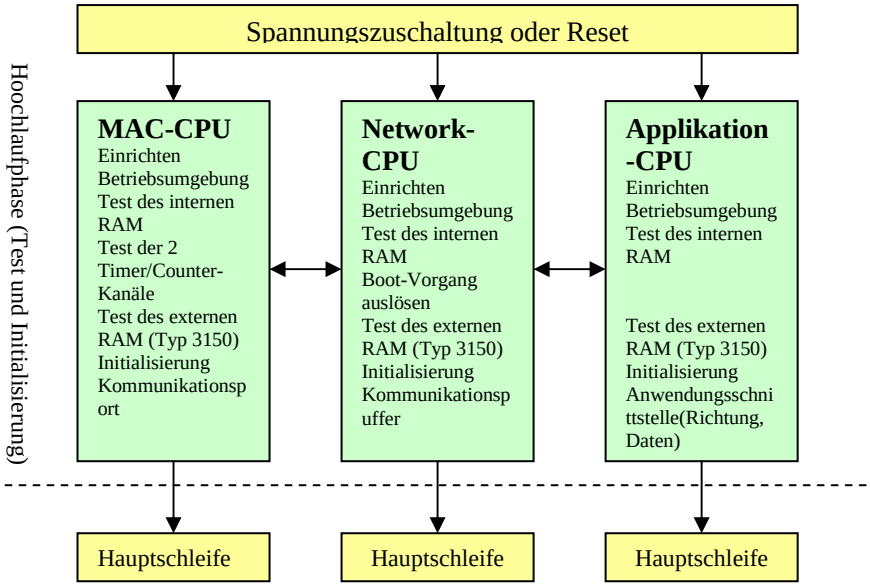


Bild 6.2.3 Aufgaben in der Hochlaufphase der CPUs

Der Informationsaustausch und damit die Entkopplung zwischen den CPUs erfolgen über **spezielle Speicherbereiche (Buffer)** für eingehende und abgehende Nachrichten.

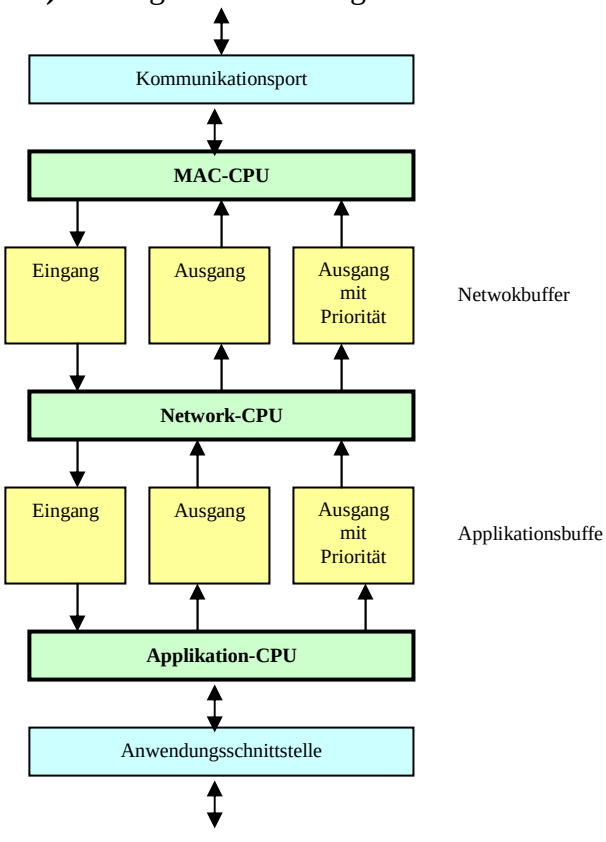


Bild 6.2.4 Inter-CPU-Kommunikation über Shared Memory

Die **Anwendungsprogramme** beinhalten die Intelligenz der LON-Geräte, auf deren Basis unterschiedliche Funktionalitäten mit der selben Hardware realisiert werden können. Dies sogar im Informationsverbund mit anderen LON-Geräten. Die Programmiersprache für die Anwendungsprogramme ist weltweit NEURON C. Diese Programmiersprache ist ähnlich ANSI C, jedoch spezieller und einfacher. Sie ist speziell auf die Hard- und Firmware zugeschnitten.

Alternativ zur NEURON C unterstützen einige Hersteller die Programmierung mit SPS-Fachsprachen, wie z. B. RESPONSE IEC oder WAGO IO PRO 32.
Obwohl ein normales LON-Gerät von komplexen Automationsaufgaben nur einzelne dezentrale Teilfunktionen übernehmen kann, werden auch **halb-zentrale** (viele Funktionen in einem LON-Gerät zusammenfassende) **„intelligente“ LON-Geräte**, z.B. für Raumcontroller, in der LON-basierenden Gebäudeautomation eingesetzt. Zahlreiche Meldungen und Messwerte werden örtlich erfasst und Verbraucher unterschiedlicher Art (Kühlung, Heizung, Jalousie, Beleuchtung) direkt angesteuert. An variierbare (projektierbare) Informationspunkte (binäre und analoge Ein- und Ausgänge) werden „nichtintelligente“ Sensoren und Aktoren mit verschiedenen Anforderungen an die Flexibilität und Leistungsfähigkeit angekoppelt. Dieses Konzept setzt modulare **WAGO-Feldbusgeräte** mit LON-Koppler oder –Controller voraus. Die Modularität wird im WAGO-I/O-System mit Hilfe eines **Klemmbusses** realisiert. Auf den Klemmenträger werden **64 Klemmen**[4] gesteckt. Die Einheit von 2, 4 oder 8 binären und 2 oder 4 analogen Signalen jeder Klemme lässt maximal **500 binäre** bzw. **166 analoge** Datenpunkte[4] für unterschiedliche elektrische Anschlussparameter und Funktionen zu. An die Klemmen werden die Sensoren und Aktoren direkt angeschlossen. Der gesamte Modulträger ist in Unterverteilungen / Zwischendecken befestigt.

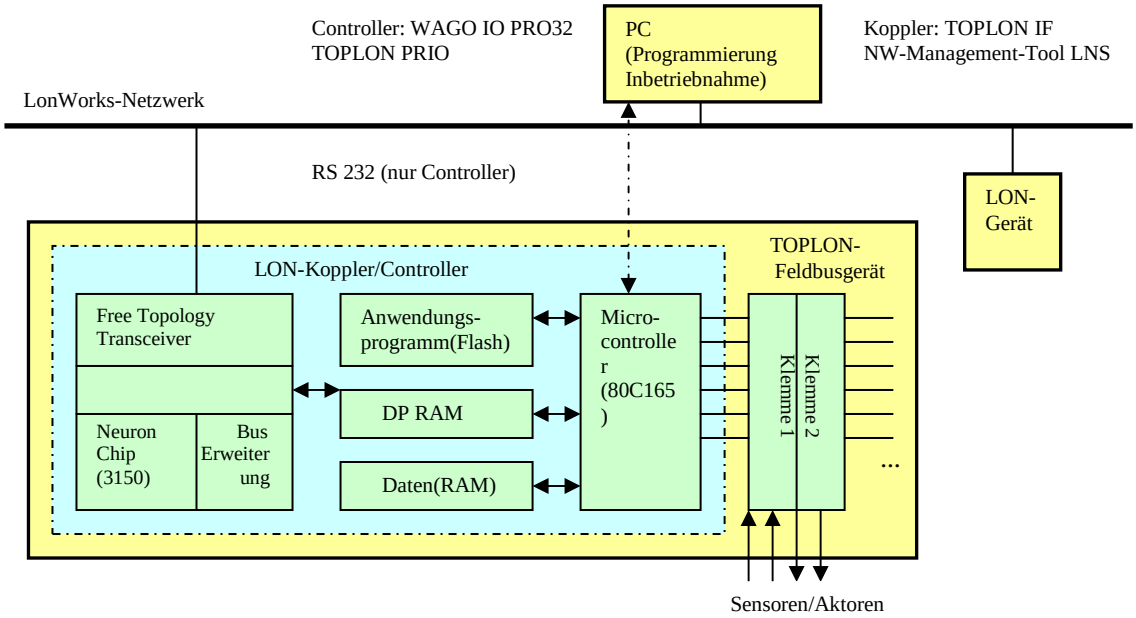


Bild 6.2.5 prinzipieller Aufbau eines TOPLON-Feldbsgerät

Bild 6.2.5[4] zeigt den prinzipiellen Aufbau eines modularen WAGO-Feldbusgerätes für ein LonWorks-Netzwerk. Genannt wird es WAGO TOPLON und ist mit einem vorprogrammierten **LON-Koppler (Typ 750-319)** oder auch mit einem freiprogrammierbaren **LON-Controller (Typ 750-819)** vergleichbar.

Tabelle 6.2.1 unterschiedene Anwendungsprogramme eingebettet im TOPLON-Koppler/-Controller

TOPLON-Feldbusgerät mit	**Anwendungsprogramm im**
LON-Koppler	Neuron-Chip-Programmspeicher (vom Hersteller bereits vorprogrammiert, jedoch eigene Neuron-C-Applikation einprogrammierbar)
LON-Controller	Flash des Microcontrollers (eigene Neuron-C-Applikation)

Die Anwendungsprogramme eines LON-Kopplers laufen als Firmware auf dem Neuron-Chip, der über eigene Speicherressourcen verfügt. Der LON-Koppler muss bis zu 72 elementare, standardisierte, vom Hersteller vorprogrammierte Teilfunktionen der Raumautomation unterstützen können. Mit TOPLON IF (einem LNS-konformen PlugIn – IF für: Installationsfunktionen) kann ein Inbetriebnahme-Management-System die einzelnen Ein- und

Ausgänge der Konfiguration (Anzahl und Typ der erforderlichen Klemmen) festlegen, und damit Teilfunktionen und Netzwerk-Variablen zuordnen.
Eine leistungsfähige Alternative für den LON-Koppler ist der LON-Controller. In diesem wird das Anwendungsprogramm vom Microcontroller abgearbeitet, dessen Speicherressourcen für das Anwendungsprogramm auf 128 KByte ausgebaut sind. Die Programmierung erfolgt mit dem Tool WAGO IO PRO32, ein LNS-konformes PlugIn, welches in der Fachsprache TOPLON-PRIO genannt wird. Die Anwendung wird über eine RS-232-Schnittstelle (nur LON-Controller) in den Programmspeicher geladen. Programme dürfen auch über das LonWorks-Netzwerk gedownloadet werden.
Für Verbindungen der unterschiedlichen Netzwerksegmente oder der Netzwerk mit verschiedenen Protokoll wird ein „sonstigen LON-Gerät“ – **Router -** einführt. Router verpackt **zwei Neuron-Chips** (Bild 6.2.6[4]). Daten werden über parallele Schnittstellen (Input/Output) - Transceiver - ausgetauscht. Telegramme können darin gefiltert bzw. anderweitig bearbeitet werden. Dem folgt das Routing, wonach die Teilnehmer (LON-Geräte) miteinander verbunden sind. Einige LON-Router können auch als Gateway verwendet werden, z.B iLON 10/100/1000, die in Kapital 6.6.2.1 detailliert erklärt werden.

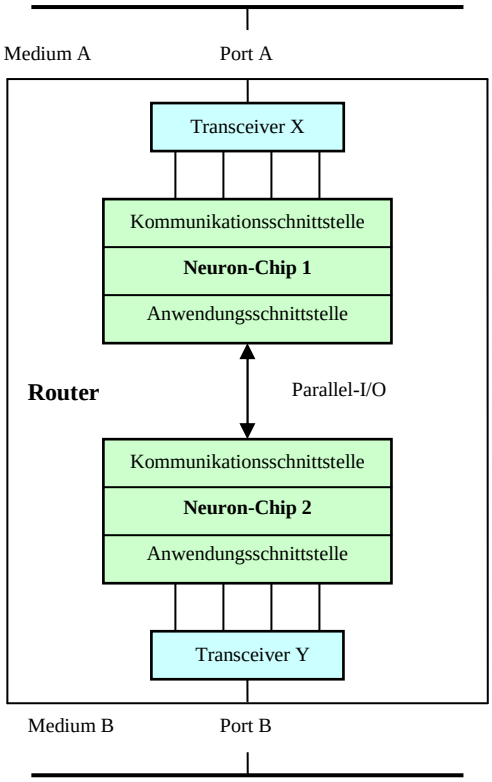
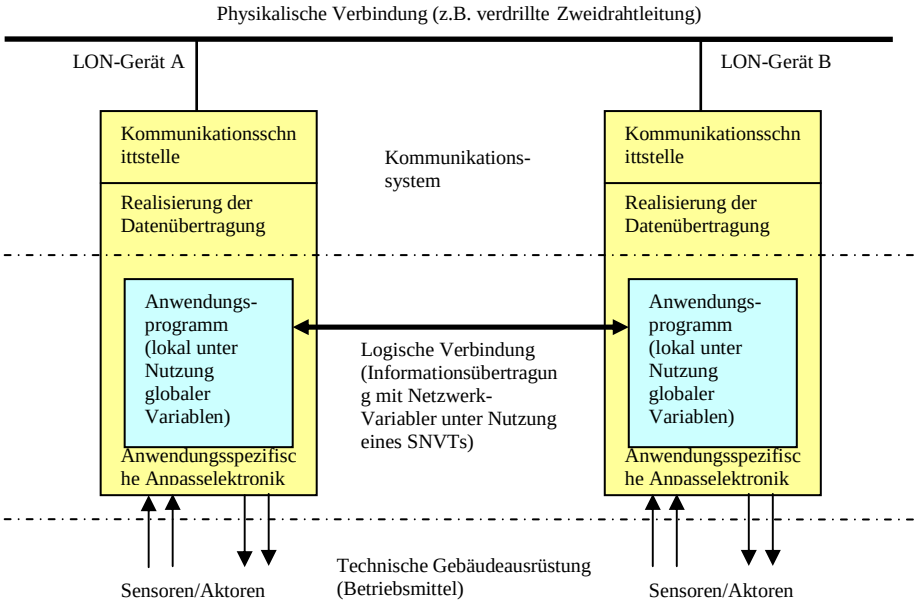


Bild 6.2.6 Der interne Struktur von Router mit zwei Neuron-Chips

6.3 Informationsübertragung zwischen LON-Geräten

Das Kommunikationsprotokoll für den Informationsaustausch zwischen den LON-Geräten wird **LonTalk-Protokoll** genannt. Mehrere entfernte LON-Geräte (Bild 6.3.1[4]) werden **logisch** und **physikalisch** miteinander verkoppelt.
Das Anwendungsprogramm eines LON-Geräts beinhaltet lokale Funktionalitäten, stellt aber auch die logische Verbindung zum Anwendungsprogramm auf anderen LON-Geräten her.
Die physikalische Verkoppelung wird z.B. mit Hilfe von verdrillten Zweidrahtleitungen realisiert.



SNVT: Standard-Netzwerk-Variablen-Typen

Bild 6.3.1 physikalischeDatenübertragung und logische Kommunikationsbeziehung

Für herstellerunabhängige Interoperabilitäten zwischen LON-Geräten wurde die Bezeichnung **SNVT (Standard-Netzwerk-Variablen-Typen)** eingeführt. SNVT sind typgebundene Variablen, die als **Standardobjekte** für Netzwerk-Variablen im LonWorks-Netzwerk verwendet werden und die Kommunikation zwischen LON-Geräten verschiedener Hersteller standardisieren und damit vereinfachen. Auf diese Weise können die Anwendungen auf den verschiedenen Geräten unkompliziert Daten austauschen. Eine logische Kommunikation zwischen Sender und Empfänger.

Die weltweit organisierte **LonMark Interoperability Association** hat eine **SNVT Master List** als Bestandteil der Ressource File erarbeitet. Die SNVT Master List umfasst über **170** SNVTs. Das ist ausreichend für alle gängigen Arten von Kommunikationen in LonWorks-Netzwerken. SNVTs dürfen in **NEURON C** weiterbearbeitet werden.

Für Datenübertragungen zwischen mehreren LON-Geräten müssen, zusätzlich zur übertragenen Information (**Nutzdatum: Wert der Netzwerk-Variablen und Identifikation**) **Zusatzinformationen** im LonWorks-Protokoll definiert werden. Zusatzinformationen können z.B. Telegrammtypen und –formate, Qualität und Typ der Kommunikationsbeziehung, Adressierung, Übertragungssicherheit, Buszuteilung, Übertragungsgeräte sein. Zu diesem Zweck wird die PDU (Protokoll Data Unit), welche aus Zusatzinformationen und Nutzdatum besteht, in LonWorks benutzt.

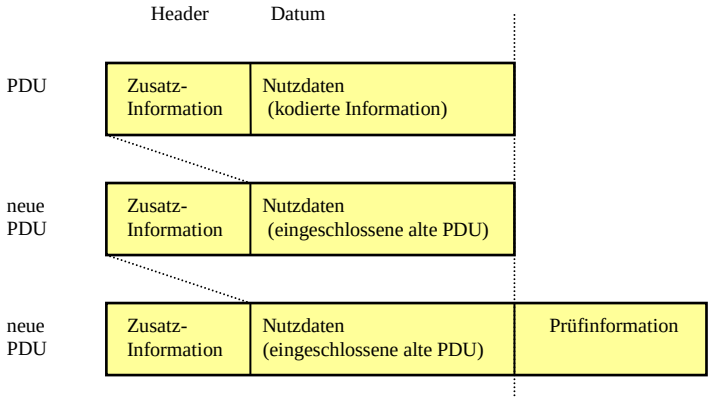


Bild 6.3.2 Bildung von PDU

Für „Freiheit“ der Kommunikationen zwischen den Geräten werden die Zusatzinformationen **flexibel** für die unterschiedlichen Telegrammempfänger zusammengestellt. Die kommunizierenden zwischen zweier LON-Geräte **Bytefolge** – Zusatzinformationen und Nutzdaten - werden als Protokolldateneinheiten **PDU**s genannt. Entsprechend unterschiedlichen Telegrammempfänger wird Zusatzinformation im **Header** oder **Kopf** definiert und damit eine neue PDU gebildet (Bild 6.3.2[4]). Das LonWorks-Protokoll definiert, wie der **Empfänger** Information ver- und entpacken muss bzw. kann.
Normalerweise erfolgt der Informationsaustausch durch die Versendung von **Anwendungstelegrammen**, welche auch als **Implicit Messages** bezeichnet werden. Für besondere Situationen oder Aufgaben, z.B. Netzwerkmanagement, Netzwerkdiagnose, LON-Geräteidentifizierung, etc., werden **Explicit Messages** erzeugt. Die Klassen der Anwendungstelegramme vergleichen einander mit Hilfe des PDU-Headers. Implicit Messages haben einen PDU-Header von **2 Byte**, Explicit Messages einen mit der Größe von **1Byte**, wie Bild 6.3.3[4].

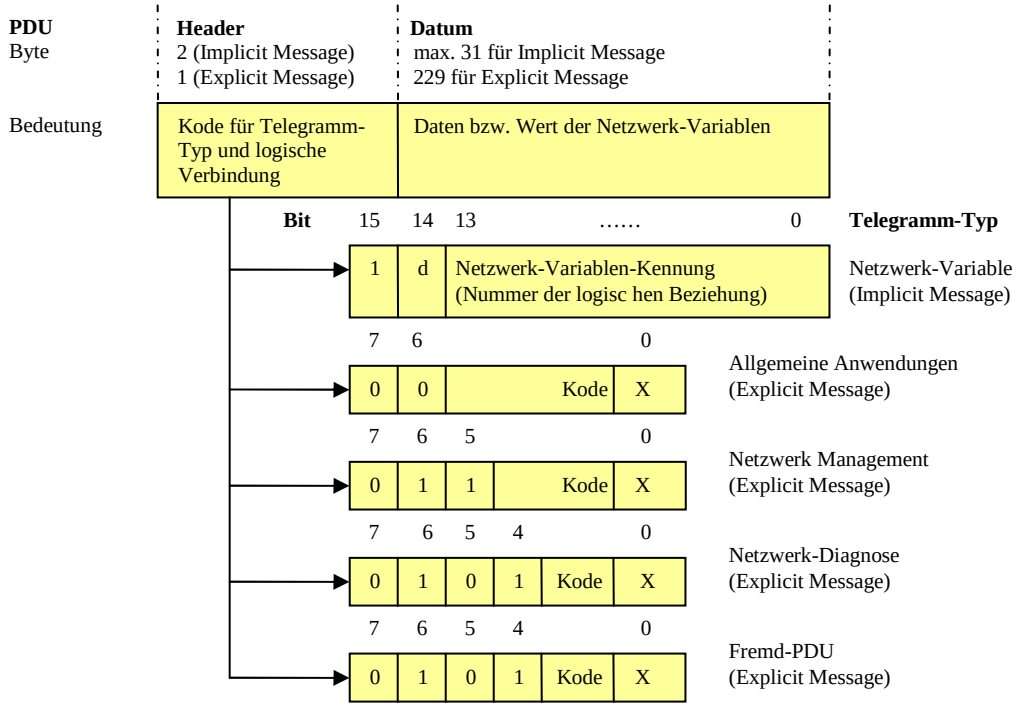


Bild 6.3.3 Kodierung der Anwendungstelegramm-Type

Entsprechend den konkreten Anwendungsbedingungen und gebäudetechnischen Prozesse bedienen die Übertragungsdienste. Ein Anwendungstelegramm wird von einem korrekten Empfänger aufgenommen und bestätigt. Bei einer ausbleibenden Bestätigung (unkorrekter Empfang, Verlust der Bestätigung) muss das gesendete Telegramm wiederholt werden. Dieser Dienst wird als bestätigter (acknowledged Service) Dienst bezeichnet und **ACKD** (acknowledged) abgekürzt. Die unbestätigten Dienste (unacknowledged Service) sind mit **UNACKD** und ohne Wiederholungen von Telegrammen. Die unbestätigten Dienste mit der zeitlichen Abstand zwischen den Wiederholungen sind **UNACKD_RPT** (unacknowledged /Repeated Service). Für die Dienste werden neue Transport-PDU (**TPDU**) wie Bild 6.3.4[4] für Telegrammempfänger kodiert.

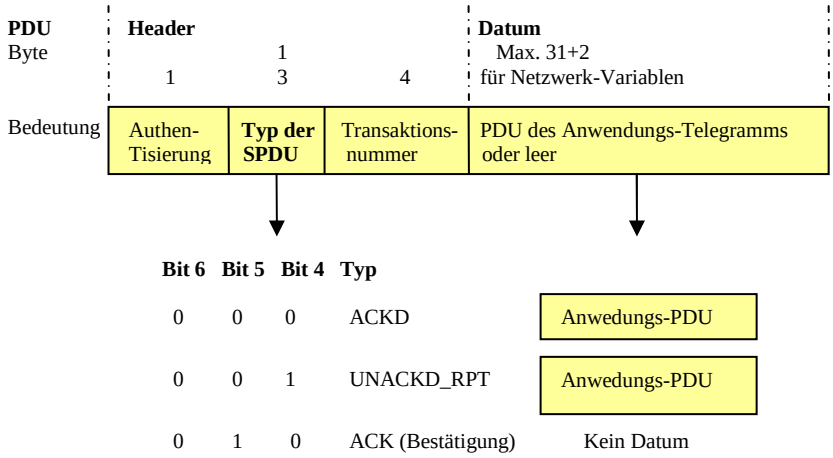


Bild 6.3.4 Transport-PDU

Anwendungstelegramme ohne Bestätigung (UNACKD) haben keinen Header der TPDU. Für weitere „Verpackung“ eines Anwendungstelegramms mit dem unbestätigten Dienst UNACKD kann mit die APDU direkt verwendet werden.
Für die verteilte Kommunikation zwischen LON-Geräten finden, in Explicit Messages, die Befehle „**Request**“ und „**Response**“ (Request-Response-Service/**RESPONSE**) Verwendung. Das angeforderte Gerät ist der **Client**, das antwortende Gerät der **Server** (Bild 6.3.5[4]). Das Request-Response-Prinzip wird auch für die Überwachung von LON-Geräten, die Prozessvisualisierung im GLT (Gebäudeleittechnik), genutzt. In einem solchen Fall sind GLT- und LON-Geräte Clients für Management- und Überwachungsaufgaben. Gesteuert werden sie von LON-Geräten, welche als Server fungieren.

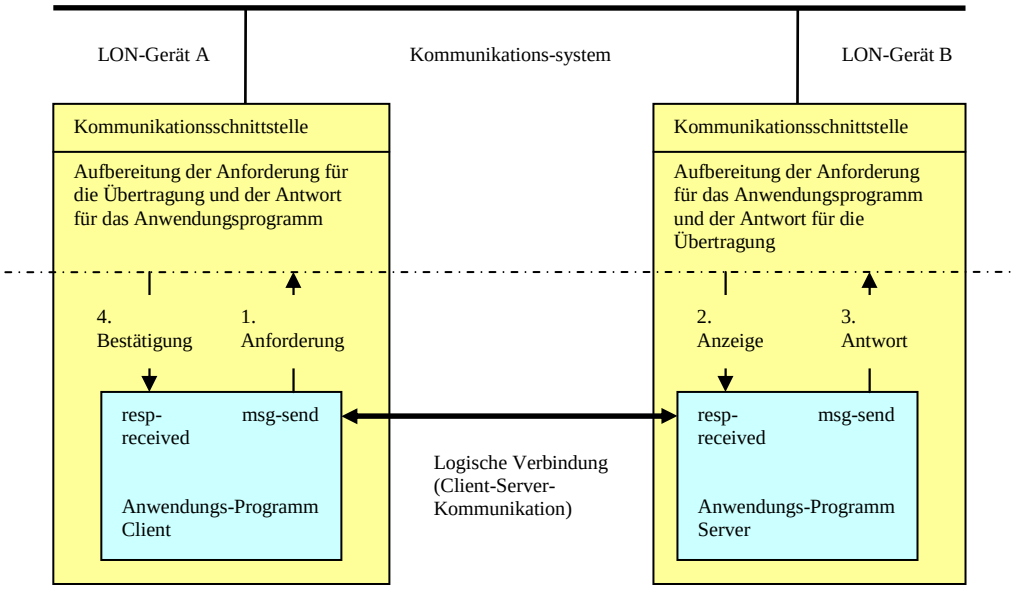


Bild 6.3.5 Client-Server-Dienst in LonWorks

Für Explicit Messages, die den Request-Response-Dienst nutzen, wird eine Sitzungs-PDU (**SPDU**) wie Bild 6.3.6[4] gebildet.

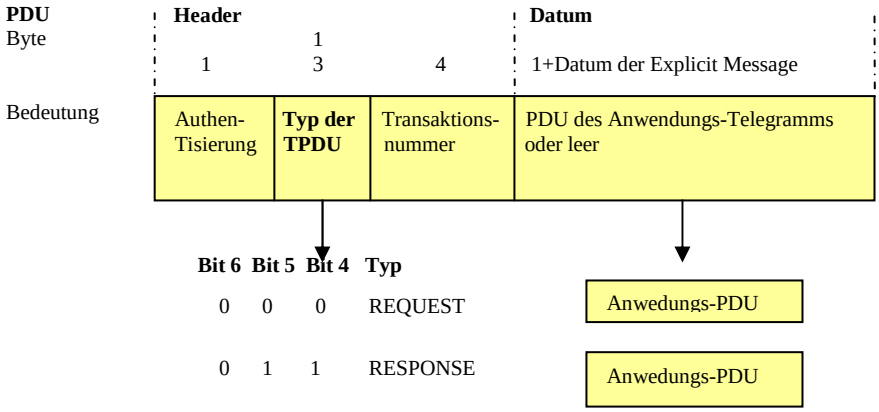


Bild 6.3.6 Sitzungs-PDU

Im LON-Netzwerk ist die **Adressierung** von LON-Geräten sehr wichtig. Das Adresssystem von LonWork ist hierarchisch wie Tabelle 6.3.1[19]. Es fasst Netzwerk-Komponenten mit gemeinsamer Adresse in logische Gruppen zusammen.

Tabelle 6.3.1 hierarchisch Struktur der Adressierung im LonWorks-Netzwerk

Ebene	Erklärung
Domain	Virtuelles Netzwerk, in dem die Kommunikation stattfindet Logische Trennung von Netzwerken, die sich physisch ein Medium teilen Größte Adressierbare Einheit Typischerweise Datenaustausch nur in einer Domain
Teilnetz (Subnet)	Logische Zusammenfassung von Netzwerk-Teilnehmern Es stehen pro Domäne 255 Teilnetz-Adressen zur Verfügung (8-Bit-Adresse)
LON-Gerät (Node)	Kleinste adressierbare physische Komponente des Netzwerkes Pro Teilnetz sind 127 LON-Geräte adressierbar (7-Bit-Adresse)
Neuron-ID	Weltweit einmalige Identifikationsnummer des Neuron-Chips (48-Bit-Adresse, d.h. 6Bytes)
Gruppe	Logische Zuordnung von LON-Geräten zu einer Gruppe (unabhängig von Teilnetzzuordnung oder Medienanbindung) Es sind 256 Gruppen adressierbar (8-Bit-Adresse) Pro Gruppe 63 LON-Geräte (für bestätigten Dienst) Ein LON-Gerät darf gleichzeitig 15 Gruppen zugeordnet werden
Gruppenmitglied	Kleinste adressierbare Komponente einer logischen Gruppe

Die Adressenformate werden in Tabelle 6.3.2[4] gegeben:

Tabelle 6.3.2 Adressenformate für die Empfängeradressierung

Format	Logische Adresskomponenten für Empfänger	Ziel	Länge[1)] [Byte]
#0	Domain, Teilnetz (=0)	Alle LON-Geräte der Domain (Broadcast)	3
#0	Domain, Teilnetz	AlleLON-Geräte des adressierten Teilnetzes (Multicast)	3
#1	Domain, Gruppe	AlleLON-Geräte einer Gruppe (Multicast)	3
#2a	Domain, Teilnetz	Ein logischer Netzwerk-Teilnehmer im Teilnetz (Unicast)	4
#2b	Domain, Teilnetz	LON-Geräte einer Gruppe mit Bestätigung des Telegramms (Multicast)	6
#3	Domain, Neuron-ID	Spezieller (physischer) LON-Geräte mit der Neuron-ID	9

1) zuzüglich bis zu 6 Byte bei Angabe einer Domain-Adresse

Die Adressbytes in Tabelle 6.3.2 sind im Adressenteil des Headers einschliesslich 2 Byte für die Absenderadresse für Teilnetz (Subnet) und LON-Gerät (Node). Damit wird neue PDU mit Netzwerk-PDU (**NPDU**) wie Bild 6.3.7[4] bezeichnet. Die Adressenangabe sind **3** (Adressierung einer logischen Gruppe) bis **15 Byte** (Adressierung über Neuron-ID und **6 Byte** Domain-Adresse) erforderlich.

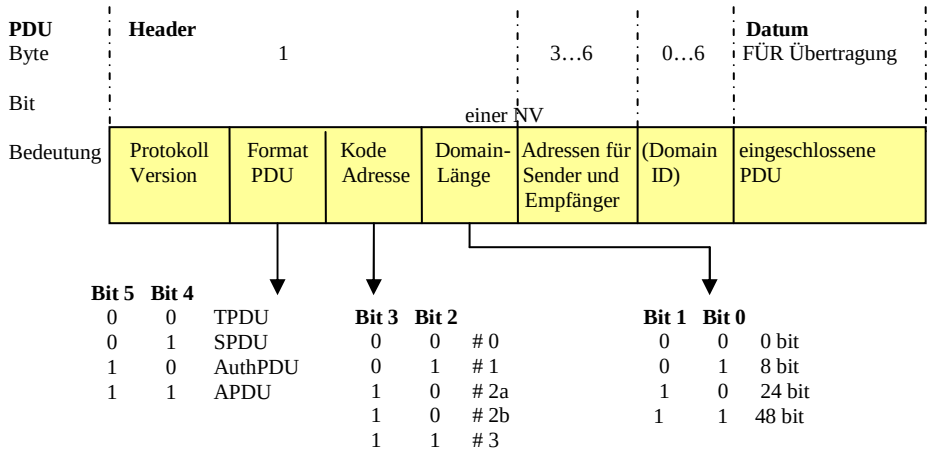


Bild 6.3.7 Netzwerk-PDU

LON-Geräte sind autonome und eigenständige Netzwerkkomponenten. Sie übertragen Informationen, unabhängig voneinander, über ein gemeinsames Medium. Abweichend von anderen Feldbussen, ist das Buszugriffverfahren von LonWorks nicht basierend auf dem Master-Slave-Verfahren sondern auf dem **CSMA** (Carrier Sense Multiple Access)-Verfahren. Für ein deterministisches Buszugriffsverfahren, wie dem Master-Slave-Verfahren, ist der größte Vorteil die große Nähe zur Echtzeit. Eine definierte Wartzeit (eine Obergrenze) kann angegeben werden. Im undeterministischen CSMA-Verfahrens ist dies nicht möglich. Für die Verringerung langer Wartzeiten, kann man die Übertragungsrate erhöhen und die Busbelastung verringern. Darüber hinaus nutzt das LonWorks-Protokoll ein prädiktiv p-persistentes CSMA, um Kollisionen nach Möglichkeit zu vermeiden.
Das Rahmenformat für LonWorks-Telegramm ist wie Bild 6.3.8[4].

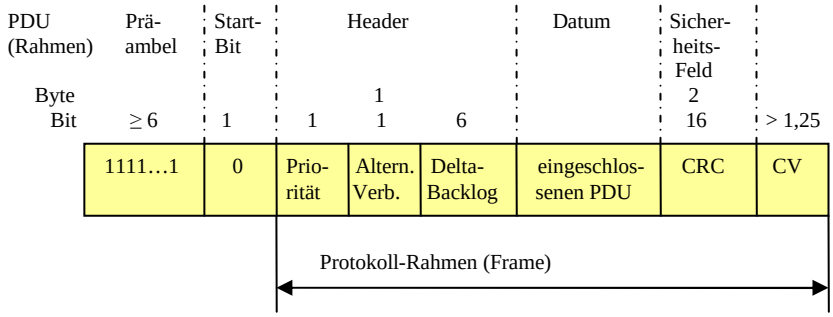


Bild 6.3.8 Rahmenformat für ein LonWorks-Telegramm

6.4 Physikalische Kopplung zwischn LON-Gräten

Empfangsreceiver und Sendetransmitter von LON-Geräten, sind über Schnittstellen miteinander verbunden, welche Transmitter genannt werden, welche in die Geräte eingebettet sind. Normalerweise hat jedes LON-Gerät nur einen Transceiver. Ein Router ist jedoch ein spezifisches LON-Gerät, mit zwei Transceivern.
Das Medium **„Verdrillte-Zweidraht-Leitung“ (TP twisted Pair, STP/UTP)** ist am weitesten verbreitet und wird am häufigsten genutzt, in LON-Netzwerk-Systemen, auf Grund des geringen Materialpreises sowie der Einfachheit, in Bezug auf Verlegung und Anschluss. Mit TP sind alle

Arten von Topologien möglich. Die Signalübertragung erfolgt durch Spannungsimpulse. TP ermöglichen hohe Übertragungsraten und besitzen nur geringe Störempfindlichkeiten (besondern im STP). Auf Grund von Signalreflexionen erfordern TP Leistungsabschlüsse.
Im LON-Netzwerk können allerdings auch Koaxialkabel, Lichtwellenleiter und Radiofrequenzen als Medien benutzt werden. In Koaxial-Leitungen werden Signale als Spannungsimpulse über geschirmte Innenleiter übertragen. Die Übertragungsraten sind hoch und die Störempfindlichkeit gering. Jedoch ist der Materialpreis hoch und die Verlegung sehr aufwendig.
Das Trägermaterial eines Lichtwellenleiters ist aus Glas. LWL sind praktisch nicht störempfindlich. Als Glasfaserkabel können sie über sehr große Entfernungen Daten verlustfrei übertragen. Die Kosten für dieses Medium, dessen Verlegung, Verbindung etc. sind aber deutlich höher als die für Koaxialkabel.
Außerdem kann in LON-Netzwerken Radiofrequenz für die Datenübertragung eingesetzt werden. Dieses Medium realisiert die Datenübertragung durch elektromagnetische (Funk)Wellen. Es ist jedoch sehr störempfindlich, besitzt nur geringe Übertragungsraten (ungeeignet für Echtzeitanforderungen) und benötigt eine komplizierte Netzwerkstruktur. Die Entfernung zw. Sender und Empfänger ist begrenzt und abhängig von der Trägerfrequenz. Dafür ist der Installationsaufwand gering und eine freie Standortwahl möglich.
Alle genannten Medien finden für LAN-Netzwerke Verwendung.
Zusätzlich ist als Medium jede übliche **230-V-Wechsel-Spannungs-Leitung**, via “Power line communication (PLC)” verwendbar. Es ermöglicht die gleichzeitige Nutzung als normales Wechsel-Spannungs-Leitungsnetz und Datenübertragungsleitung. Das Nutzsignal wird durch Aufmodulation der Wechselspannung hinzugefügt. Die Übertragungsraten sind gering, die Entfernungen hoch. Nur bestimmte Modulationsfrequenzen sind zulässig. Hauptproblem ist die starke, netzabhängige Störempfindlichkeit.
Das Medium TP wird für unterschiedliche Transceiver genutzt. In Tabellegruppe(1)(2)6.4.1[4] werden verschiedene Typen dieses Mediums und dazu verfügbare Transceiver mit verschiedenen Faktoren und Kenngrößen aufgelistet.

Tabelle 6.4.1 Medien, Transceiver und Netzwerkparameter

(1)

Medium	Transceiver	Transceivertyp	Übertragungsrat (kB/s)	Teilnehmerzahl pro Segment	Geträte-versorgung
Verdrillte-Zweidraht-Leitung	EIA 485	aktiv	39-625	32	Getrennt
	TPT/XF-78	aktiv	78	64	Getrennt
	TPT/XF-1250	aktiv	1250	64	Getrennt
	FTT 10-A	aktiv	78	64	getrennt
	LPT-10	aktiv	78	128	üder Busleitung

(2)

Medium	Transceiver	Topologie	Netz-ausdehnung	Bemerkung
Verdrillte-Zweidraht-Leitung	EIA 485	Linie	1200	für kleine Systeme
	TPT/XF-78	Linie	2000	transformatorentkoppelt
	TPT/XF-1250	Linie	130	transformatorentkoppelt
	FTT 10-A	Frei	500	i. Transformatorentkoppelt ii. mischbar mit LPT-10
		Linie	2700	
	LPT-10	Frei	500	i. Transformatorentkoppelt ii. mischbar mit FTT 10A
		Linie	2200	

Besonders kostengünstig sind Linienstrukturen mit Stichleitungen, die maximal **3 Meter** lang sind. Bei Nutzung von **FTT-10-Transceivern** sind die TP-Kabellängen von Segmenten mit Bus-Topologien zwischen Mit- und Ohne- Stichleitungen unterschiedlich (siehe Tabelle6.4.2[4]).

Tabelle 6.4.2 Maximale Kabellängen bei Nutzung des FTT-10-Transceivers mit Bus-Topologie

Kabelsort	**Bus-Topologie**	
	Ohne Stichleitungen	**Mit Stichleitungen(max. 3 m)**
¢1,3 mm (TP einfach, Litze)	2700	2200
¢0,8 mm (TP zweifach, abgeschirmt, massiv)	1400	1150
¢0,65mm (TP zweifach, abgeschirmt, massiv)	900	750
¢0.5mm (Normkabel gemäß EN50173, CAT5)	700	575

Das LonWorks-Protokoll ist als internationale Norm EIA 709.1 definiert. Die Grundlage bildet das OSI-Referenzmodell. Das LonWorks-Protokoll besitzt die Schichten **„physical Layer“, „Link Layer“** und **„Application/Presentation Layer“,** wie andere Feldbussystem auch. Es besitzt jedoch darüber hinaus gehende Eigenschaften. Der Neuron-Chip besitzt drei CPUs: die MAC-CPU, die Network-CPU und die Applications-CPU. Das Zugriffverfahren ist CSMA. Die Network-CPU bearbeit verschiedene Aufträge des Transports. Die Aufgaben sind verschiedenen Schichten des OSI-Modells zugeordnet. Bild 6.4.1[4] zeigt die Übertragung des PDUs im LonWorks-Protokoll.

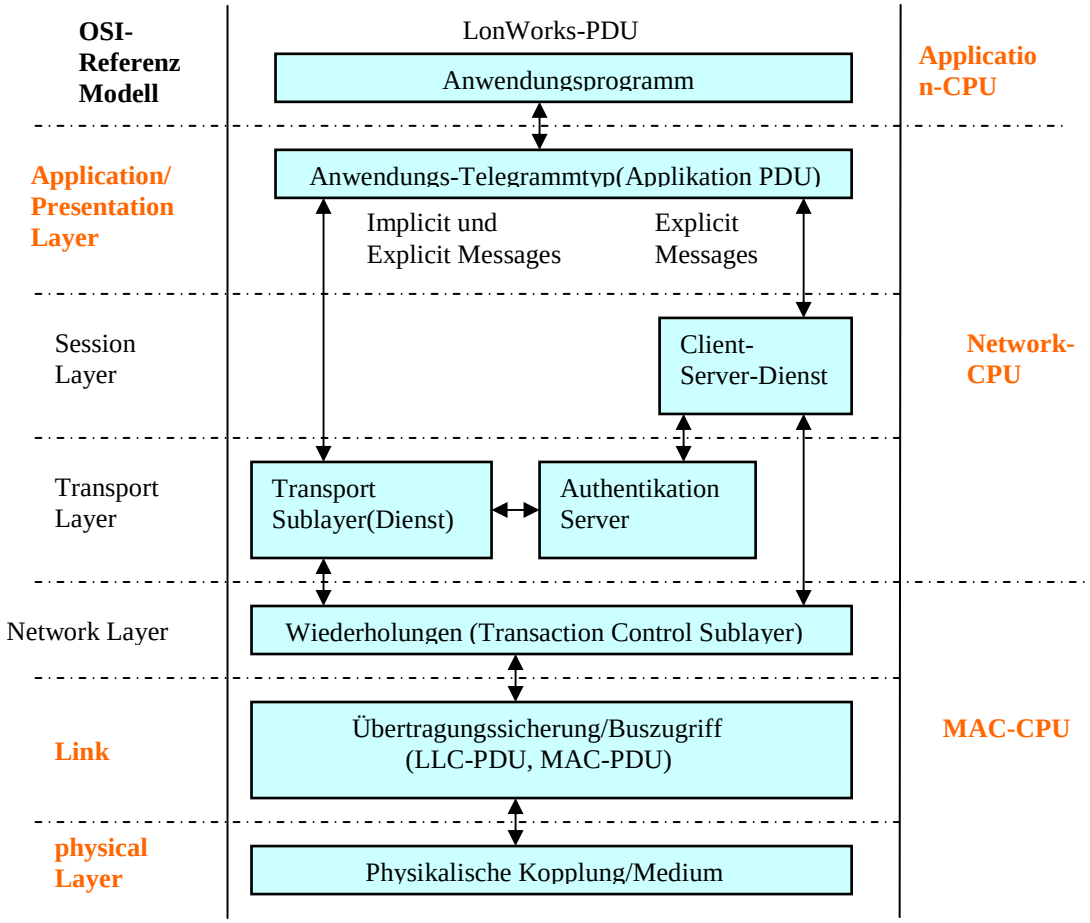


Bild 6.4.1 LonWorks-PDU und Funktionalitäten der CPUs inner Nueron-Chip gegen OSI-Modell

6.5 Interoperabilität von LON-Geräten

Das LON-Network ist ein **verteiltes Automationssystem**, d.h. die Datenverarbeitung, Steuerungsaufgaben, Datenspeicherung und Hardwareressourcen werden **dezentralisiert** implementiert. Das erfordert ein funktionales Zusammenwirken der verteilten Komponenten während des Informationsaustausches. Eine Automationsaufgabe kann man in viele Teilaufgaben (Teilfunktion) untergliedern. Die Teilfunktionen auf den verschiedenen LON-Geräten müssen durch ein funktionales Zusammenwirken in einem System eindeutig festgelegt und korrekt zugeordnet werden. Mit Hilfe von SNVT können LON-Geräte verschiedener Hersteller in einem Netzwerk eingesetzt werden, obwohl die von den konkurrierenden Firmen produzierten Geräte unterschiedlich konstruiert sind und ausgestattet wurden. Genannt wird dies Interoperabilität von LON-Geräten.
Bild 6.5.1[4] zeigt ein Beispiel des Zusammenwirkens von Teilfunktionen von LON-Geräten zur Steuerung einer Raumsituation. Die Informationen (die logische Verbindung) werden zwischen den Teilfunktionen f1 bis f8 über das Medium ausgetauscht. Diese Teilfunktionen sind eindeutig definiert, programmiert und in verschiedenen LON-Geräten implementiert. Das Gerät B ist mit Sensoren verbunden. Es übernimmt die Teilfunktion f2 für die Messung der Temperatur des Raums. Das Gerät C dient der Teilfunktion f3 für die Definition des Sollwertes, der Raumtemperatur. Die Geräte E, F, G sind mit den Aktoren verbunden. Die Steuerungsprogramme f6, f7, f8 der Aktoren befinden sich voneinander getrennt in den Geräten E, F, G. Es gibt auch die Möglichkeit, dass sich die Steuerungsprogramme f6, f7, f8 nur in einem LON-Gerät befinden. Im Gerät A liegt die Teilfunktion f1. F1 ist eine einfache Tasterfunktion, beispielsweise als ein Zwei- oder Vierfach-Taster. Durch die, auf SNVT basierenden Parameter, kann jeder Taster eine der Teilfunktionen f6, f7, f8 beeinflussen. Damit sind die logischen Schnittstellen der Teilfunktionen zum Netzwerk festgelegt. Die Teilfunktion f4 im Geräte D bearbeitet die Informationen der Teilfunktionen f1, f2, f3 und bestimmt, welcher Aktor zu aktivieren ist, der an die LON-Geräte E, F, G angekoppelt ist. Natürlich könnten Teilfunktionen, wie z.B. f5, auch im Gerät D einbettet sein, um weniger Geräte verwenden zu müssen und damit den Aufwand zu verringern.
Für die Interoperabilität von LON-Geräten gibt es **Profile** für Teilfunktionen der Automation, die weltweit gültig festgelegt wurden, ohne herstellerspezifische Besonderheiten auszuschließen. Die Funktionsprofile der Teilfunktionen im LonWorks-Geräten werden LonMark-Objekte genannt und als **Standard Functional Profile Types (SFPTs)** bezeichnet. SFPTs werden entsprechend dem Anwendungsbereich (z.B. HLK/Heizung Lüftung Klima, Ein-/Ausgabe, Beleuchtung, usw.) nummeriert. Die Interoperabilität von LON-Geräten basiert auf der Nutzung von LonMark-kompatiblen, herstellerdefinierten, gelisteten und international festgeschriebenen Profilen für Teilfunktionen der Automation (Funktional Blocks oder Funktionsblöcke entsprechend den Interoperabilitäts-Richtlinien der **LonMark Interoperability Association**).

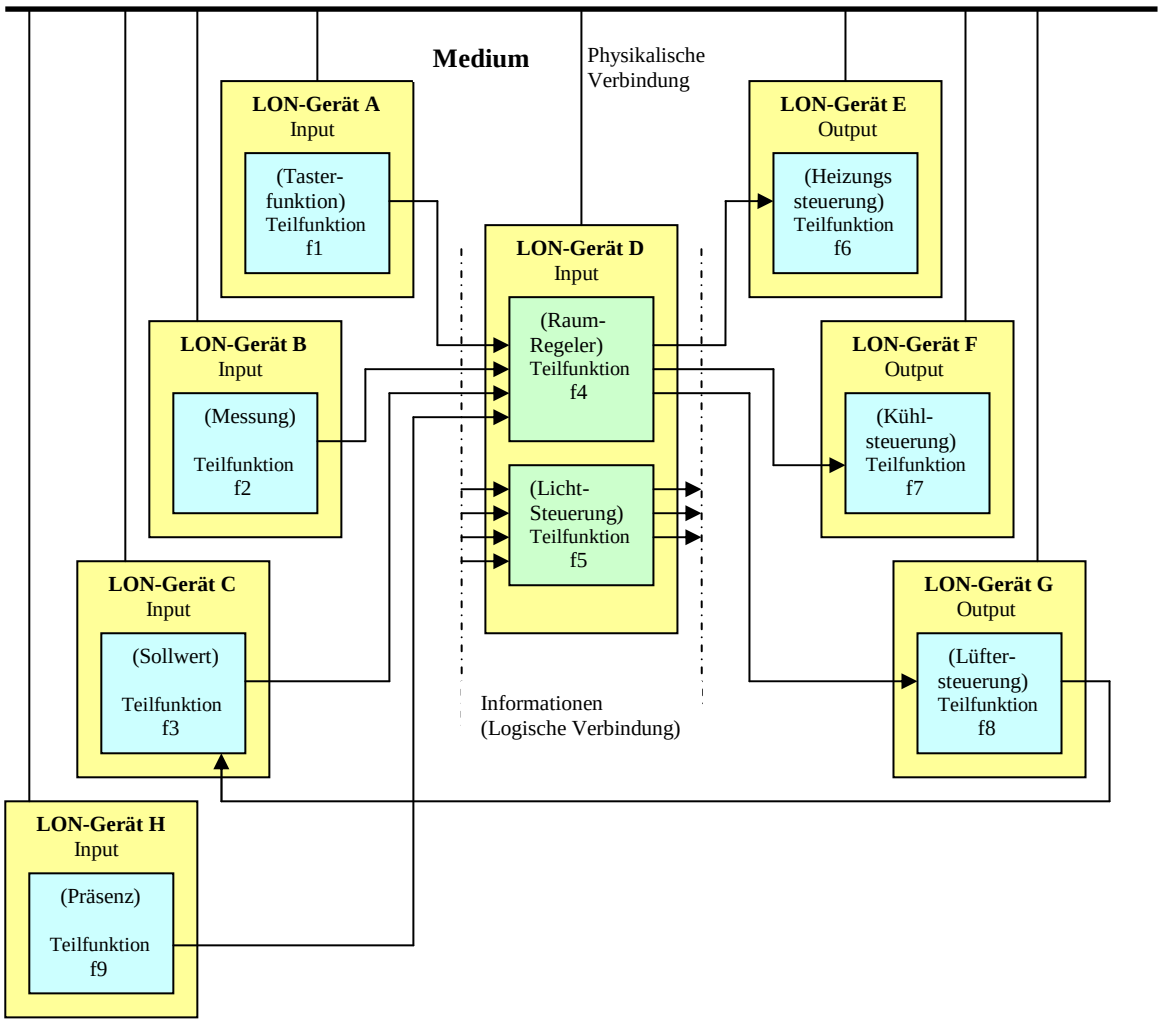


Bild 6.5.1 Zusammenwirkung der Teilfunktionen im LON-Networksystem

6.6 LonWorks-Netzwerke in der Gebäudeautomation

Für die Datenübertragung in Netzwerken müssen LON-Geräte das LonWorks-Protokoll EIA 709.1 unterstützen. Es verlangt genormte Kommunikationstypen (SNVTs, SCPTs) sowie Standard-Funktionsprofil-Typen (SFPTs). Die Werkzeuge, welche für die Inbetriebnahme und das Management der LON-Geräte von den Herstellern mitgegeben werden, sind interoperabl und unterstützen offene Plattformen.

6.6.1 Inbebtriebnahme von Netzwerken mit LON-Geräten

Für den Aufbau eines LonWorks-Netzwerkes ist Planung erforderlich und sinnvoll. Natürlich außerdem ausgereichende Sachkenntnisse über LON-Geräte, logische Schnittstellen (mögliche logische Verbindungen über **Netzwerk-Variablen**, **Konfigurationsparameter** und deren **Voreinstellungen**) und Netzwerktopologie. Die Anwendungsprogramme in LON-Geräten und ihre logischen Schnittstellen werden von den Herstellern dargestellt (im CD, Diskette oder per Download von Internet). Völlig neuartige Aufgaben können mit den in Netzwerken integrierten LON-Geräten, mittels Erweiterungen in der Feld- und Automationsebene realisiert werden. Dies

wird von den Herstellern mit unterschiedlichen Planungshandbüchern unterstützt. Außerdem gibt es ergänzende Literatur, Informationslisten und Leistungsverzeichnisse.
Für die Inbetriebnahme ist es wichtig, LON-Geräte im Netzwerksystem bekannt zu machen. Jedes LON-Gerät hat eine physikalische Adresse. Diese Adresse ist eigentlich die **Neuron-ID** des Neuron-Chips im LON-Gerät. Im LonWorks-Netzwerk besitzt das LON-Gerät zusätzlich eine **logische Adresse**, die durch Domain, Teilnetz (Subnet) und das Gerät (Node) definiert wird. Die logische Adresse muss der physischen zugeordnet werden. Teilfunktionen, die durch die im EPROM innerhalb des LON-Geräts gespeicherten Anwendungsprogramme beschrieben werden, kann man mit Parametern im LonWorks-Netzwerk konfigurieren. Jedes Gerät kennt die Parameter der logischen Schnittstelle. Normalerweise sind diese Parameter bereits vom Herstellern voreingestellt. Jedoch dürfen diese jederzeit, was insbesondere vor der Erst-Inbetriebnahme sehr wichtig ist, an spezielle Besonderheiten angepasst werden.
Besonders hilfreich für den Aufbau von LonWorks-Netzwerken, sind **Netzwerk-Inbetriebnahme-Management-Tools**. Sie sind Bestandteil der Lieferung der Hersteller und können auf **PC**s installiert und ausgeführt werden. Die physikalische Ankopplung eines Inbetriebnahme-Management-Tools an das Netzwerk erfolgt über ein Interface mit Neuron-Chip-Funktionalität hinsichtlich des implementierten Protokolls (das LonWorks- bzw. EIA 709.1-Protokoll kann auch mit einem anderen Controller oder einer anderen Programmtechnik realisiert werden). Es stellt selbst ein LON-Gerät dar.

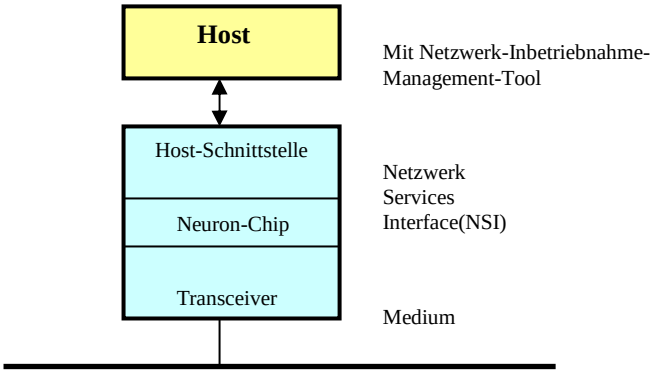


Bild 6.6.1.1 Verbindung von Host (PC) über NSI mit dem Medium

Die Host-Schnittstelle kann man auswählen. Geeignet sind z.B. **PCI**- und **PCMCIA**-, serielle (SLTA Serial LonTalk Adapter) und parallele (als Dongle) sowie die **USB-Schnittstelle** (USB Universal Serial Bus). Der Zugang zum LonWorks-Netzwerk erfolgt über entsprechende Netzwerk-Infrastrukturkomponenten (wie Tabelle 6.4.1) und ein RNI (Remote Network Interface) im Intranet.
Für die bessere Adressierung, Bindung, Konfiguration und das Management des LonWorks-Netzwerks hat „Echelon Corporation“ ein NetzWerk-Betriebssystem **„LonWorks Network Service (LNS)“** für Microsoft-Windows-basierende Systeme entwickelt. Es wurde objektorientiert entwickelt und verwendet einen **Client-Server-Mechanismus** (Bild 6.6.1.2[4]). Auf diese Weise wird u.a. das LonWorks-Netzwerk mit dem Intranet und dem Internet verbunden.

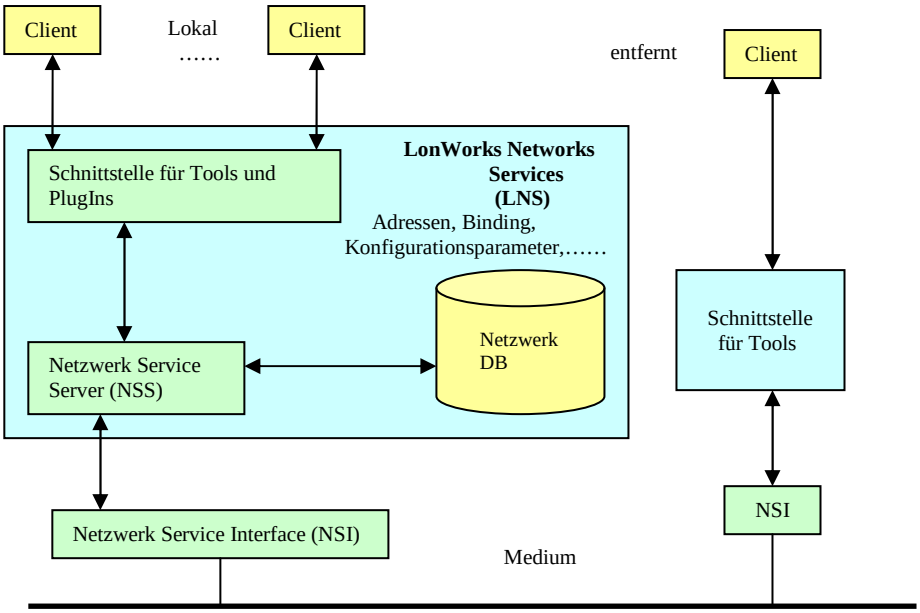


Bild 6.6.1.2 Client-Server-Mechanismus von LNS

Die LNS-Architektur besitzt zwei wichtige Komponenten: das **Netzwerk Service Interface (NSI)** und den **Netzwerk Services Server** (**NSS**). Über NSI kann der Server (PC) an das LON-Netzwerk physikalisch angekoppelt werden. Die Netzwerk-Dienste und Netzwerk-Datenbank werden vom NSS verwaltet.

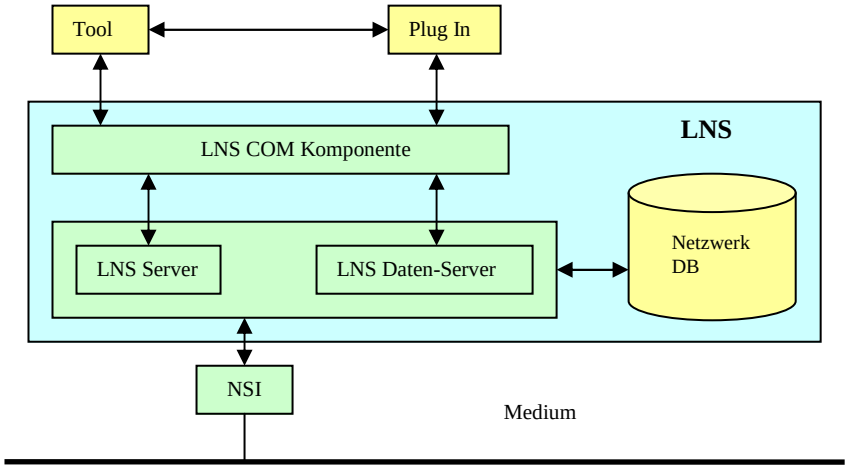


Bild 6.6.1.3 Struktur des LonWorks Netzwerk Services

Im COM (Component Objekt Model) von Microsoft gibt es eine Schichtstruktur mit LNS Server und LNS Daten-Server, die einen einheitlichen Zugang zu den Diensten und der Datenbank von LNS ermöglicht. Für Tools sind LNS-Objekte transparent und zeigen sie als Text oder Grafik an. Damit können Benutzer unkompliziert auf die Objekte zugreifen und Befehlen ausführen. Mit Geräte-PlugIns ist es möglich, herstellerspezifische Zusatzprogramme anzubinden.

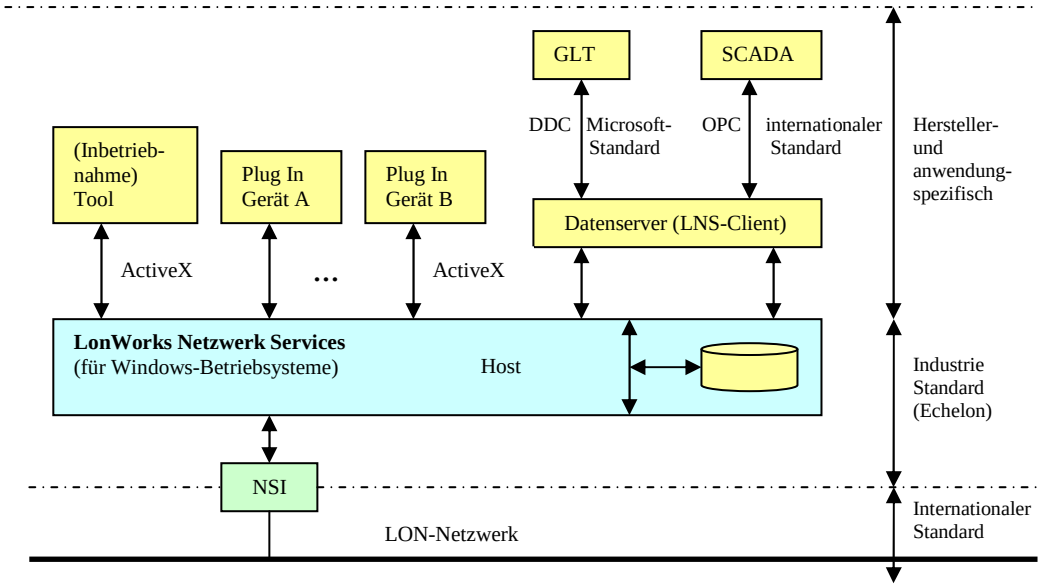


Bild 6.6.1.4 LNS-Netzwerk-Betriebsystem für Management und Leiter (GLT/SCADA) auf Host

Bild 6.6.1.4[4] zeigt die Verbindung des Management- mit dem Leitsystem. Für den Windows-basierten Host wird der Microsoftstandard OLE (Objekt Linking and Embedding) und AktiveX für den Datenaustausch genutzt. Die AktiveX-Technologie vereinfacht die Anbindung der herstellerspezifischen Tools und Geräte-PlugIns (z.B. für die Parametrierung von Teilfunktionen). Die Funktion von PlugIns ist die Anpassung der parametrierbaren Eigenschaften der Teilfunktionen über Konfigurationsparameter. LNS-Objekte werden vom LNS-Objekt-Server als OLE-Objekte dargestellt. Sie können mit Hilfe verschiedener Sprachen (Visual Basic, Visual C++ u.a.) entwickelt werden. Der Start entwickelter OLE-Objekte erfolgt lokal über Icons auf der Arbeitsfläche. Seit 2000 bietet LNS 3.0 zusätzlich die Möglichkeit, über Intranet oder Internet auf Daten zuzugreifen, die auf entfernten Datenbanken in Netzwerken gespeichert sind. Auf DDE (Dynamic Data Exchange)- und OPC (OLE for Process Contol)-Servern werden Netzwerk- und Prozess-Daten der GLT (Gebäudeleittechnik) sowie der SCADA (Supervisory Control and Data Aquisition) verwaltet und visualisiert. OPC stellt einen Standard für eine offene Datenschnittstelle in den Leitsystemen der Industrieautomation dar.

Tabelle 6.6.1.1 Dimension von LonWorks-Netzwerken

Dimension	**Erklärung**
Medium und Kabeltyp	Verschiedene Medien sind möglich (TP mit/ohne Energieversorgung und verschiedene Übertragungsraten, Infrarot, Strarkstromleitung, Funkkanal, Ultraschall, Koaxialkabel)
Entfernungen und Anzahl der LON-Geräte pro Segment	Flexible Lösungen mit Repeater und Router (abhängig von Leitungstyp, Transceiver, Übertragungsrate)
Übertragungsrate	Verschiedene kanaltypabhängige Übertragungsraten, einschließlich eines Ehternet-Kanals mit 10 Mb/s oder 100 Mb/s als Backbone
Netzwer-Strukturierung(Segmentierung)	Verfügbarkeit verschiedener Infrastrukturkomponenten: Repeater, Bridge, Router, Gateway
Adressierung	Logische Netzwerkteilnehmer, Teilnetze, Gruppen von Teilnehmern

In Tabelle 6.6.1.1[4] sind Dimensionen aufgelistet, welche im direkten Zusammenhang mit dem physikalischen und logischen Aufbau von Lonworks-Netzwerken stehen.
In Bild 6.6.1.5[4] wird schematisch gezeigt, dass ein **LON-Router** im Intranet bzw. Internet auch ein LON-Kanal (Tunnel) sein kann, durch welchen LON-interne Nachrichten ausgetauscht werden. Dies auf der Basis des Internetprotokolls TCP/IP. Die verfügbare Länge des Datenfeldes im TCP-Rahmen (64 bis zu 1500 Byte) ist für LonWorks ausreichend. Über den Weg der

Tunnelung der Daten ist es außerdem möglich, Komponenten im Intranet/Internet zu managen. Der Router verhält sich von IP-Seite wie ein IP-Host.

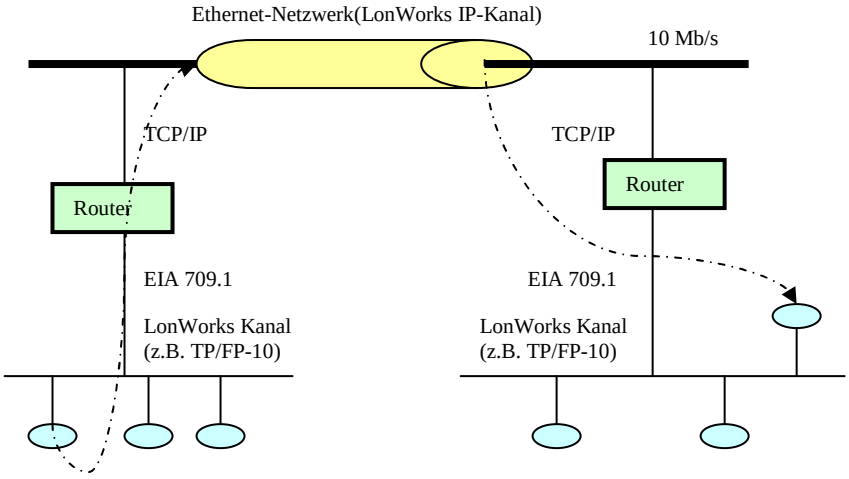


Bild 6.6.1.5 Tunnel des EIA 709.1 über TCP/IP

In der Gebäudeautomation kann man den TCP/IP-Kanal als schnellen Backbone nutzen. An die Router werden EIA 709.1-Netzwerke angekoppelt. Die Router sind auch mit integriertem Webserver (z.B. i.LON 1000) verfügbar.
Wenn in bestehende Anlagen ein LON-Netzwerk integriert werden soll oder an ein LonWorks-Netzwerk ein weiteres Bussystem angekoppelt werden muss, übernehmen Gateways die Funktion der Übersetzung zwischen beiden Systemen. Für Lösung des ersten Falls wird eine entsprechende **Schnittstellen-Karte** für das LonWorks-Netzwerk in einer DDC/SPS eingesetzt. Für die zweite Situation wird ein **Gateway** verwendet. Beispielsweise werden in der GLT Informationen für Beleuchtungen im LonWorks-Netzwerk über LON-DALI-Gateways versendet.
Das Netzwerk-Management für die Inbetriebnahme und Wartung des Netzwerkes und auch die zentrale Gebäudeleittechnik (**GLT,** die schon in Kapital 3.2.1 „GLT“ erklärt wird) bzw. das technische Gebäudemanagement, gehören zu den Facility Managements. Sie können die LNS-Datenbasis gemeinsam nutzen, da das Netzwerk-Betriebssystem LNS den gleichzeitigen Zugriff mehrerer Hosts auf die LNS-Datenbank erlaubt. Damit ist eine Anbindung der GLT (als DDE- oder OPC-Client) über die auf das LNS aufsetzenden DDE- bzw. OPC-Server möglich. Das Netzwerk-Management verteilter Kommunikationen (z.B. mit Hilfe mobiler Clients) wird damit ermöglicht.
Weil LNS die Anbindung an TCP/IP-basierte Netzwerke unterstützt, können globalere IT-Netzwerke, wie Intranet und Internet zur Anbindung örtlich entfernter EIA 709.1-Netzwerke für die Gebäudeautomation genutzt werden (siehe Bild 6.6.1.6[4]).
Auf die Gebäudeautomation spezialisierte Clients nutzen offene Datenschnittstellen wie OPC- und Web-Server oder Mechanismen des Datenaustausches, die von Windows-basierten Betriebssystemen unterstützt werden (DDC/OLE bzw. ActiveX). Da Web-Server bereits auf EIA 709.1-IP-Routern implementiert sind, ist auch der direkte Zugang zum LonWorks-Netzwerk mit einem Standard-Browser möglich. Das Bild 6.6.1.7[4] zeigt die verteilte Managementebene mit dem LNS- und dem zentralen Datenserver sowie verschiedenen lokalen und abgesetzten Clients und die Anbindung des LonWorks-Netzwerkes über einen EIA 709.1-**IP-Router (i.LON 1000)**. Varianten sind i.LON 10/100, welche entfernte Netzwerkinterfaces (RNI Remote Network Interface) darstellen.
Die Installation des LNS ist nicht zwingend notwendig. Über die **LON-Schnittstellen-Karte** eines Hosts können Scanner (Observatorium) des Informationsaustauschs im LonWorks-Netzwerk und die Bereitstellung der Daten für einen DDE-, OPC- oder Web-Server, gebäudeleittechnischer Aufgaben realisiert werden. Damit ist auch ein prinzipieller (mittlerweile klassischer) Lösungsansatz aufgezeigt, um auf der Managementebene die gewerkeübergreifende

Integration verschiedener Bussysteme in gebäudetechnische Anlagen zu sichern. Über geeignete Bus-Interfaces werden die Daten einem OPC-, DDE- oder Web-Server zur Verfügung gestellt.

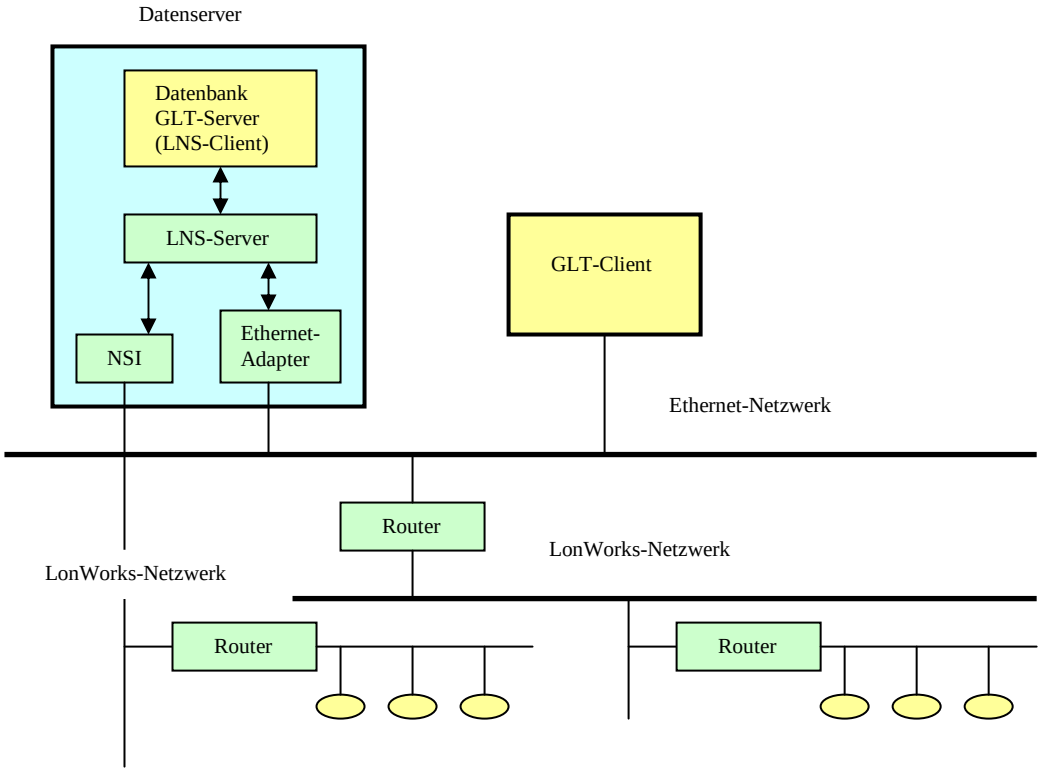


Bild 6.6.1.6 mittel- und unmittelbare Anbindung des LonWorks-Netzwerkes und Ethernet-Netzwerk

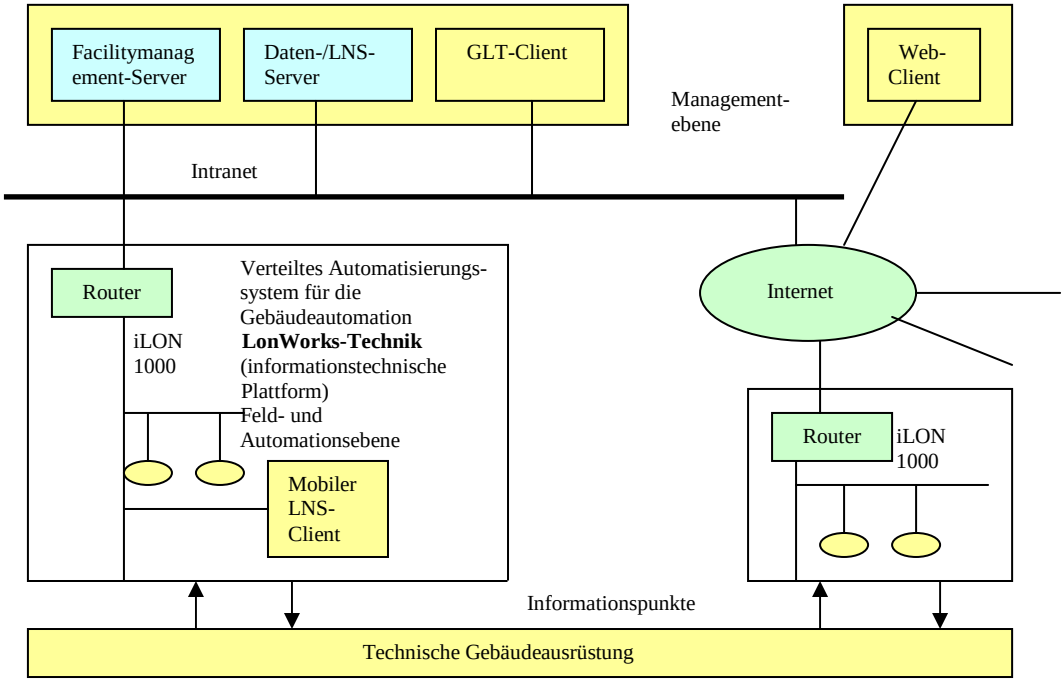


Bild 6.6.1.7 gemeinsame LonWorks-Technik mit der Managementebene

6.6.2 einige mögliche Verbindungsweisen zwischen LON und LAN

Von Kapital 3.2.2 „OPC“, Kapital 3.2.3 „OSGi“, Kapital 6.6.1 „Inbetriebnahme von Netzwerken mit LON-Geräten“ und anderen Informationsquellen kann man in Praxen viele Möglichkeiten haben, LON-Bus und LAN-Bus miteinander zu verbinden.
Hier schlage ich vor, die Verbindungsweise in vier Klassen einzuordnen:

i. Eine Klasse ist basiert auf Router iLON 10/100/1000.
ii. Eine Klasse ist basiert auf NSI.
iii. Die andere Klasse ist basiert auf die LON-IP Gateways
iv. Eine Klasse ist basiert auf Java-Technik mit OSGi

Natürlich gibt es auch andere Möglichkeiten, die nicht in den vier Klassen eingeordnet werden

6.6.2.1 mit Router iLON 10/100/1000

Hier ist ein Bespiel, dass LON-Bus über einem LON-Router an Ethernet verbunden wird (Bild 6.6.2.1.2). Dieser Router „iLon 1000“ (Bild 6.6.2.1.1) ist von „iApplianceWeb“ hergestellt, der zwei Interfacen (TP/FT-10 oder TP/XF-1250) mit LonWorks zu kommunizieren und einen „10Mbps 10BaseT RJ45“ Port mit Ethernet zu kommunizieren hat. Das Detail des Produkts kann man in „http://www.iappliancewеb.com/story/OEG20010703S0016.htm“ anfragen.

Bild 6.6.2.1.1 Router iLON 1000

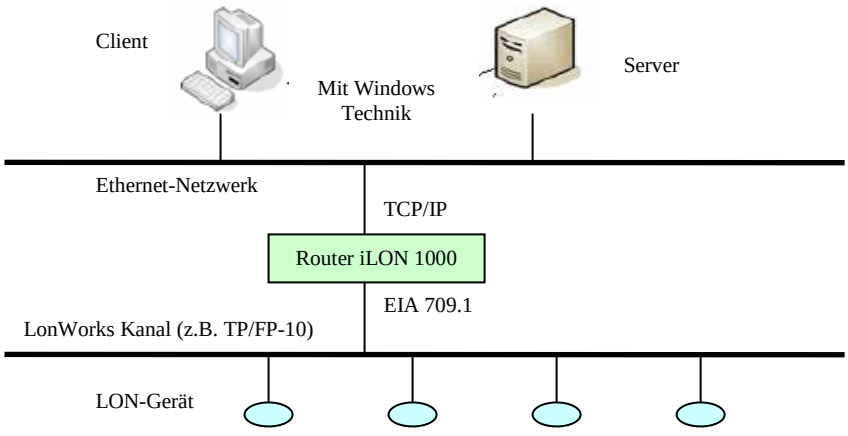


Bild 6.6.2.1.2 Verbindung zwischen LON und LAN mit Router iLON 1000

6.6.2.2 mit NSI

NSI (Netzwerk Services Interface) ist ein LON-Gerät, dessen Form ähnlich wie eine Netzkarte ist. NSI hat eine PCI- und PCMCIA-Schnittstelle, serielle (SLTA Serial LonTalk Adapter) und parallele (als Dongle) Schnittstellen sowie USB-Schnittstelle.
Mit der PCI-Schnittstelle kann NSI unmittelbar im PC eingebettet werden. Mit SLTA kann NSI mit LonWorks Netzwek verbindet werden. Wie Bild 6.6.2.2.1 ist die Weise der Verbindung zwischen LON und LAN.

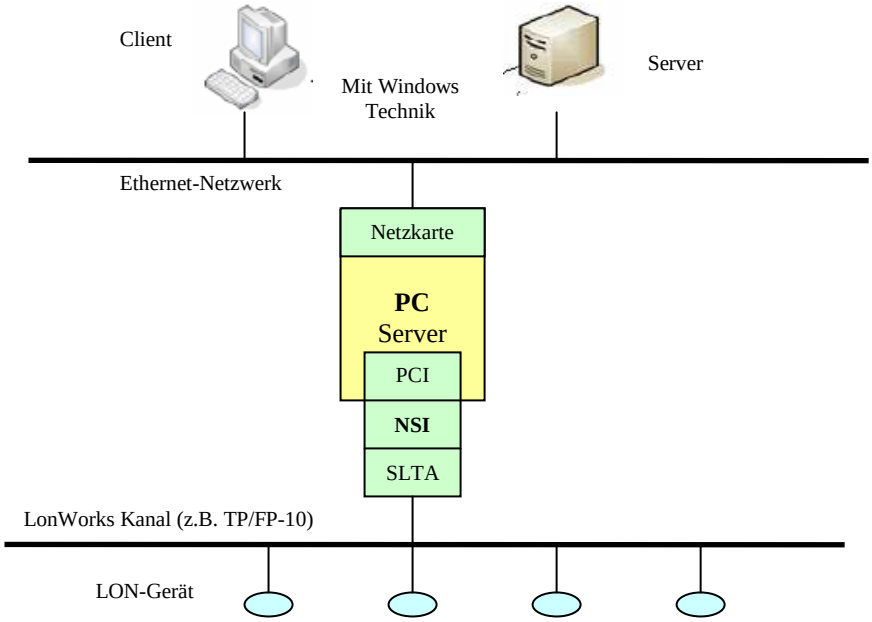


Bild 6.6.2.2.1 Verbindung zwischen LON und LAN mit NSI

6.6.2.3 mit LON to IP Gateway

Im Markt werden auch einige LON to IP Gateways von verschiedenen Herstellern dargestellt, z.B. DLA IP.
DLA IP Adapter ist mächtige LON Adapter und Brücke über Ethernet, die von „DTI“ hergestellt wird. Es hat ein Interface „FTT10“ (78Kbps) für die Verbindung zur LonWorks-Netzwerk und einen „serielle RS232“ Port für Ethernet (10/100Mbps).
Das Detail des Produkts kann man in der Webseite anfragen:
„http://www.dti-be.com/index.html?lon/en/title.htm?dlaip.htm“

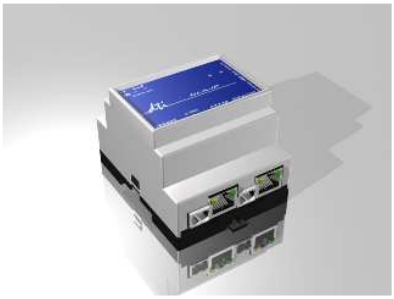


Bild 6.6.2.3.1 DLA IP Adapter

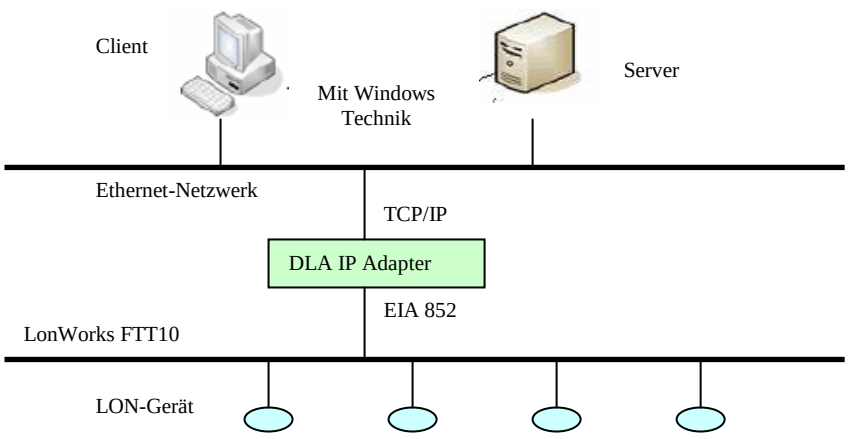


Bild 6.6.2.3.2 Verbindung zwischen LON und LAN mit LON/IP Gateway

6.6.2.4 mit OSGi Gateway

OSGi-Framework ist basiert auf Java-Technik und kann für die eingebetteten Geräte die alle intelligenten Endgeräte vernetzen. Deshalb kann man einen eingebetteten PC „SOM-4470“, das „OSGi Service Gateway“ und „JAVA Embedded Server“ enthält, zwischen LAN und LON einsetzen. Das Produkt wird von „Advantech Co. Ltd“ hergestellt.
Das Detail des Produkts kann man in der Webseite anfragen:
„http://www.advantech.com.tw/products/Model_Detail.asp?model_id=1-KOV3G&PD=EC“

Bild 6.6.2.4.1 Advantech SOM-4470 (Embedded PC)

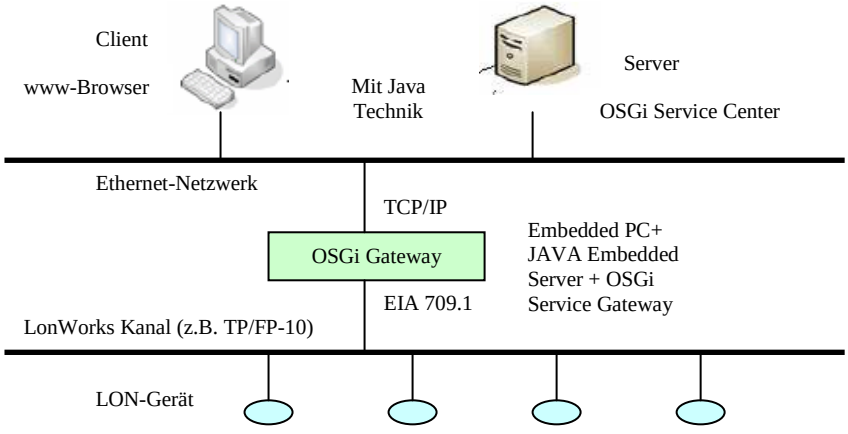


Bild 6.6.2.4.2 Verbindung zwischen LON und LAN mit OGSi Gateway

6.7 Beispiel-Szenario von LON-Bus

Zusammen mit Bild 6.7.1 und Bild 6.7.3 wird ein Beispiel von Gebäudeautomation mit LonWorks-Technik gezeigt.
Im Beispiel wie Bild 6.7.1 werden die Sensoren, Aktoren und Controller von LON-Technik installiert. Der Aktor von Jalousie ist intelligent, selbst die Fenster zu schließen oder öffnen. Und der Sensor von Helligkeit und der Aktor von Leuchtband sind auch intelligent. Wenn die Helligkeitswerte, die von den Sensor übermittelt werden, die Sollwerte überschreiten (zu dunkel oder zu hell.), will der Aktor vom Leuchtband die Leuchten ein- oder ausschalten. Die busunfähigen Sensoren und Aktoren, z.B. Kühlenaktor, Heizungsaktor… sind durch einen zentralen Raumcontroller verwaltet. Die Bustopologie ist Linie. TP/FT-10-Kanale werden als die Übertragungsmedien mit 78kBit/s Datenübertragungsrate genutzt. Für das ganze Gebäude wie Bild 7.3 wird TP/XF-1250 MIT 1,25MBit/s als Backbone von LON-Bus genutzt.

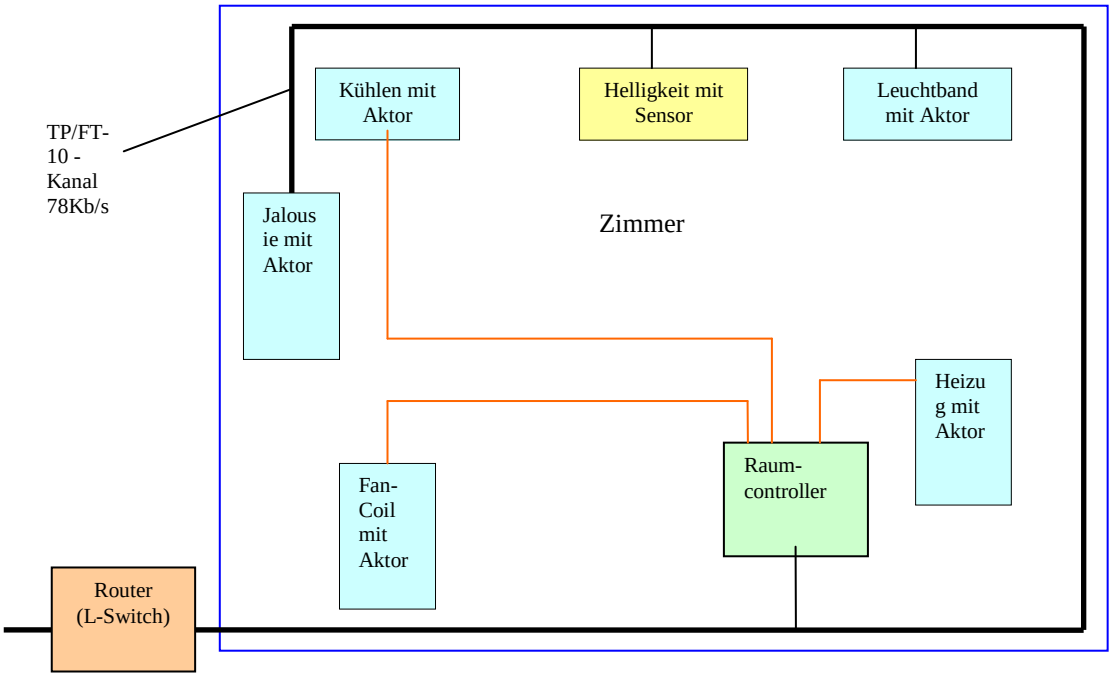


Bild 6.7.1 eine Raumautomation

Für die Anbindung des Teilnetzsegments jedes Zimmers an den Backbone wird ein Multiport-Router (z.B. 5 Port L-Switch wie Bild 6.7.2) eingesetzt.
„5 Port L-Switch“ wird von „LOYTEC electronics GmbH.“ Hergestellt. Das hat vier Porte von „FT-10“ (78Kbps) und einen Port von „TP-1250“ (1,25Mbps). Das Detail des Produkts kann man in „http://www.loytec.com/deutsch/products/lswitch_spec.htm“ anfragen.

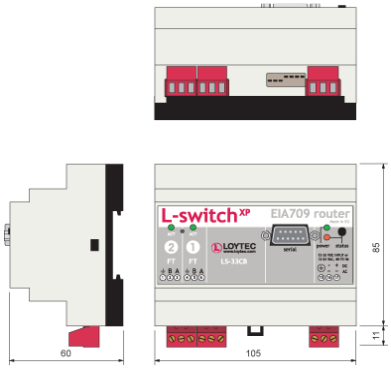


Bild 6.7.2 5 Port L-Switch

Für entfernte Überwachung der Situationen innerhalb des Gebäudes kann man mit iLON 1000 das LonWorks – Netzwerksystem an Ethernet mit Hochgeschwindigkeit von 10 bis zu 100MBit/s anschließen.

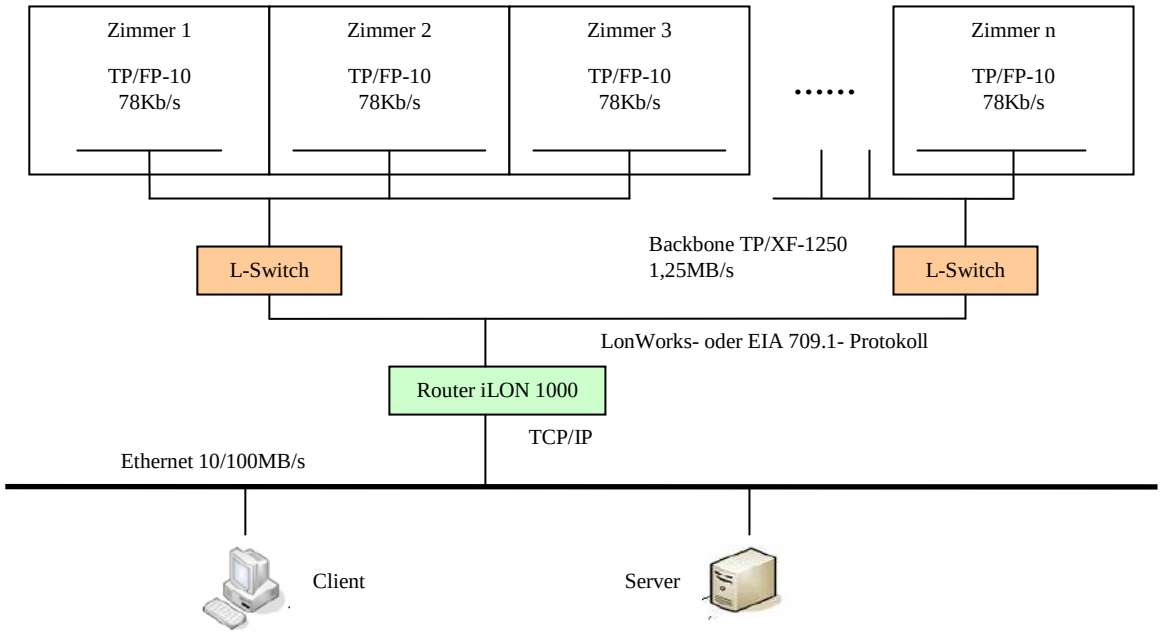


Bild 6.7.3 Verbindung zwischen den LON-Segmenten jedes Zimmers und entferntem Ethernet

Weil das LON-Bus-System ein System vom dezentralen Multi-Master System ist, hat jeder Knoten des Systems mit CSMA – Zugriffverfahren das gleiche Recht, auf die Daten zuzugreifen. Eigentlich für Gebäudeautomatisierung entwickelt, hat das LON-Bus-System auch geringe Nachteile im Bereich von Echtzeitigkeit. Dafür kann man z.B. schnellere Kabel auswählen, wenigere Knoten auf den Segment anschließen oder eine Linietopologie nutzen, um die Nachteile zu überschreiten.

7. Projekt CANDY

CANDY (Computer-Aided Network Design utility) ist ein Konzept einer Netzwerk-Design-Umgebung, die auf der effizienten XML- Sprache basiert. Mit der Hilfe von CANDY kann man schnell, einfache und günstig Netzwerke entwerfen.
Die Struktur von CANDY ist in Bild 7.1[21] gezeigt:

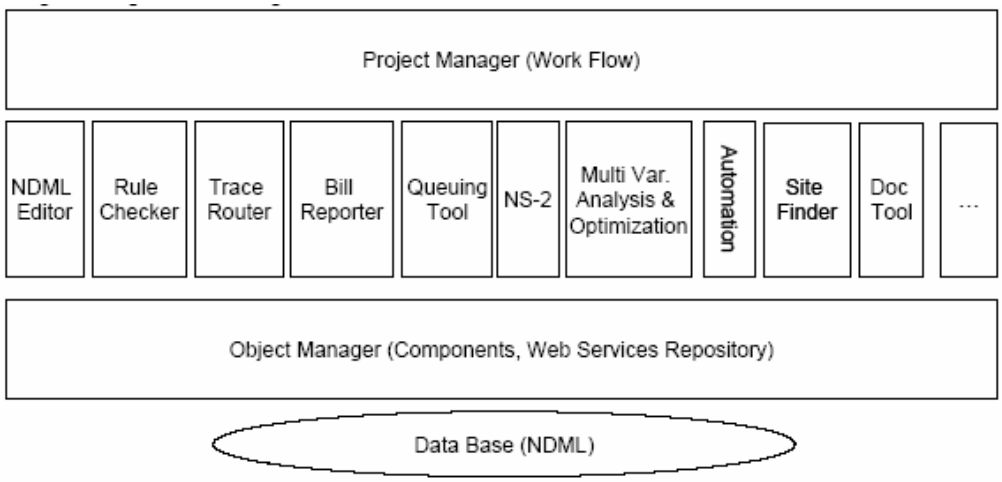


Bild 7.1 die Architektur von erweiterbarem CANDY

Der Kernentwurfsmodul von CANDY ist **NDM (Network Design Model)**. Dieser Modul nutzt die XML-basierte Sprache NDML. Diese ist auf der Basis einer Topologiebeschreibung des Netzwerks mit allen konkreten Knoten und ihren Parametern und auch allen Daten, z.B. für Strukturierung, Kostenrechnung, Performanzanalyse, aufgebaut. NDM besitzt eine komfortable grafische Oberfläche zum Editieren und Anzeigen von Netzstrukturen. NDM wurde mit JAVA-Technologien implementiert.
Für CANDY wurde eine spezielle auf XML basierte, problemorientierte Projektierungssprache **NDML (Network Design Markup Language)** entwickelt. Durch diese Sprache werden die standardisierten und organisierten Netzwerkbereiche, strukturierte Verkablungssysteme, Datenübertragungssimulationen, Netzwerkkopplungsgeräte, Netzwerkknoten, Interfaces von In-/Exporte, Protokolle für RPC usw. beschrieben. Die Beschreibungen enthalten abstrakte Ebenen, die sogenannten Viewpoints (Gesichtpunkte). Viewpoints sind z.B. Basic, Load, Topology, Costs (Tabelle 7.1[22])

Tabelle 7.1 In NDML unterstützte Viewpoints

Viewpoint	Eigenschaften	Abgeleitet von
Basic	Projekt Beschreibung, Datum...	/
Load	Netzwerkknotens, Bandbreite...	Basic
Topology	Technologie, Nic ...	Load
Geometry	Räume, Medium Länge...	Topology
Simulation	Protokolle, Agents, Applikationen...	Topology
Queuing	Verzögerung, load...	Topology
Cost	Gerät Kosten, Medium Kosten...	Topology, Geometry

Im Bild 7.2[22] werden die innere Beziehungen zwischen den Viewpoints gezeigt.

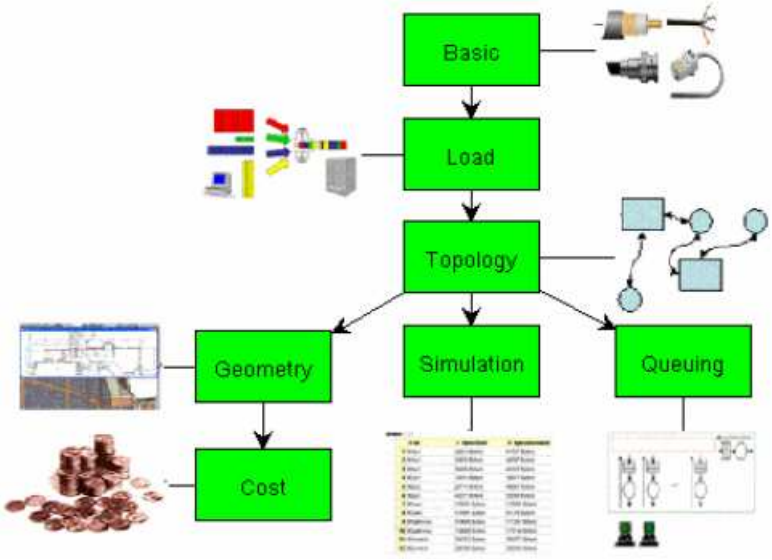


Bild 7.2 die Unterordnung von den abstrakten Ebenen(Viewpoints)

7.1 CANDY 2.0

Im Dezember 2003 ist das Candy Projekt begonnen worden. Die aktuelle Version zu Zeit ist CANDY 2.0.
Wie CANDY1.0 basiert Candy 2.0 auch auf einer Client-Server Architektur und besteht aus ***CANDY Client*** und ***CANDY Manager***.
Die gesamte Architektur sieht folgendermaßen (Bild 7.1.1[22]):

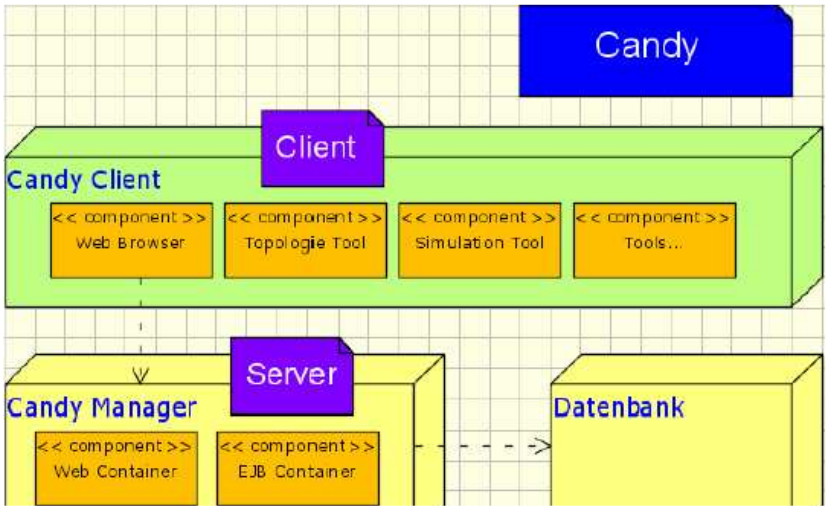


Bild 7.1.1 die Architektur von CANDY2.0

Auf der Client-Seite liegt der Candy Editor, der über Java Webstart installiert und gestartet werden kann. Der Candy Editor verfügt über verschiedene Tools, Topologie Tool, Bill Reporter Tool, Simulation Tool, eingebetteten Webbrowser usw. Mit Hilfe dieses Browsers soll der Client auf Candy Manager auf der Server Seite zugreifen können. Candy kann in jedem Betriebssystem laufen, weil Java unabhängig vom Betriebssystem ist.

Architektur Vergleich zwischen CANDY1.0 und CANDY2.0

Ungleich CANDY1.0 ist die Anwendung entsprechend der klassischen 3-Tier-Architektur mit einer klaren Trennung der Schichten angelegt, Geschäftslogik wird ausschließlich in **EJB Beans** realisiert. Im Candy 1.0 wird die Geschäftslogik nur direkt in **Servlets** eingebettet.
Die Bearbeitungen der Topologie, Geometrie usw. erfolgen im CANDY1.0 im Server. Die Client stellen Abfragen an den Server und erhalten Antworten. Im CANDY2.0 findet die Arbeite ausschließlich direkt auf dem Client statt.

Technologie Vergleich zwischen Candy 1.0 und Candy 2.0

Anstatt **JGraph** in CANDY1.0 wird Framework **JHotDraw** im Client bei der Suche aufgrund der leicht erweiterbaren und klaren Architektur in CANDY2.0 für graphische Verarbeitung gewählt.
Mittlerweile existieren manche Open Source Projekte, die JHotDraw einsetzen und wiederverwendbare Komponenten anbieten. JARP20 ist z.B ein auf JHotDraw und PNML(Petri Net Markup Language, XML basiert) basierende Petri analyzing tool.
Und **Candy Manager** im CANDY2.0 ist eine auf J2EE basierende Anwendung für die Verwaltung der Projekte und Benutzers.

7.2 NDML3.0

Für die Netzwerkbeschreibungen ist die aktuelle Version NDML3.0.
Bevor die NDML3.0 erkläret wird, gibt es ein Rückblick auf NDML1.0 und NDML2.0.
NDML1.0[23] war die erste Beschreibung einer problemorientierten, XML-basierten Sprache für die Rechnernetzprojektierung im Rahmen des CANDY-Projektes.
Damals war es nur ein Projekt, das als ein XML-Dokument (.Projektdatei.) realisiert wurde. Im Projekt kann man die Viewpoints Basic, Load und Topology finden.
Dann wird **NDML2.0**[23] entwickelt, die in zwei Teile **Core NDML** und die **Additional NDML** spaltet wird.
Core NDML soll die Viewpoints Basic, Load und Topology umfassen, die betreffenden Informationen eines Projektes werden zusammen in einem XML- Dokument gespeichert. **Additional NDML** umfasst alle anderen Viewpoints (z.B. Simulation, Geometrie, Kosten, Dokument, usw.), und die Daten von jedem Viewpoint in einem eigenen XML- Dokument(z.B. SNDML, GNDML, CNDML, DNDML…) gespeichert werden, in welchem sich auch ein Verweis auf die zugehörige Projektbeziehungsweise Core NDML- Datei befindet. Die darin definierten Objekte werden über ihre jeweiligen ID's ausgewiesen.
Wie Bild 7.2.1[23] zeigt die Zusammenhänge zwischen Core NDML und Additional NDML.

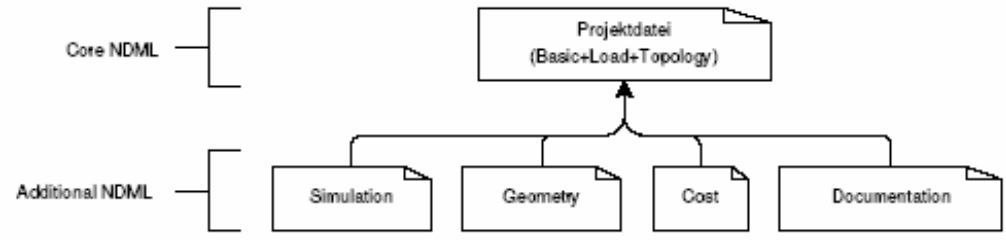


Bild 7.2.1 der Strukturmodul von NDML2.0

Ähnlich wie NDML2.0 wird jeder Viewpoint in **NDML3.0**[23] durch eigene XML- Dokumente beschrieben und durch die Verweise hängen die Viewpoints zusammen.

Als Erweiterung von NDML2.0 wird Core.NDML-Datei in drei Dokumenten für die Viewpoints Basic, Load und Topology verteilt. Damit gibt es in NDML3.0 für jeden Viewpoint genau definierte Dokumente. Diese kleineren, sauberer getrennten Dokumente werden schneller verarbeitet. Außerdem können die Dokumente parallel bearbeitet werden, um die Arbeitsabläufe zu optimieren.
Wie Bild 7.2.2[23] ist das Verhältnis von den Dokumenten bei NDML3.0.

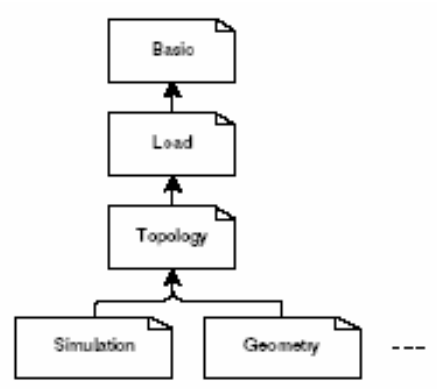


Bild 7.2.2 der Strukturmodul von NDML3.0

Im **NDML3.0** hat jede XML-Dokument eine Referenz an ein XML-Schema.
Bei **Viewpoint Basic** gibt es drei Dokumente: eines ist das eigentliche Dokument für die Projektbeschreibung, die anderen zwei zusätzlichen sind für Kunden und Dateien.
Für **Viewpoints Load** gibt es nur ein Load-Dokument, indem die Netzwerkgeräte definiert und zu Mengen zusammengefasst werden. Und in dem Dokument werden die Verbindungen zwischen den Geräten auch beschrieben.
Zum **Viewpoint Topologie** gehören sieben kleine Dokumente:

i. Technologien,
ii. Händler,
iii. Stecker-Produkte,
iv. Medien-Produkte,
v. Interface-Produkte,
vi. Geräte-Produkt,
vii. Netzwerk-Produkt

Darunter ist die Beziehung[23] zwischen den Objekten.

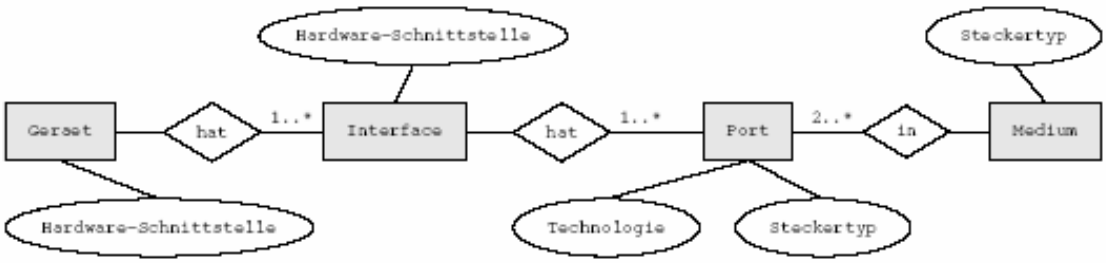


Bild 7.2.3 Objektbeziehungen beim Topology-Viewpoint (NDML3.0)

Die anderen Viewpoints wurden auch mit spezifischen Dokumenten gegen NDML1.0 erweitet. Die sind im Beleg von Robert Hänel erklärt.

7.3 Gebäudeautomationsnetz in CANDY

Candy ist ein Netzwerk-Design-Werkzeug. Die Vorschläge von mir sind, dass die Gebäudeautomationsnetzwerke z.B. LON-Bus auch in CANDY entworfen und simuliert werden können.
Die Netze können die Tools in CANDY von Gebäudeautomationsdesign weiter benutzen. Wie z.B.:

i. NDML Editor
ii. Rule checker
iii. Queuing tool
iv. Bill reporter
v. Documentation

Die Gebäudeautomatisierungsnetzwerke haben ihre besonderen Eigenschaften. Deshalb muss CANDY gegen ältere Viewpoints um einige neuere Informationen erweitet werden. Z.B.:

i. **Realtime Requirement:** Diese Anforderungen spielen die wichtigsten Rollen in den Knoten-zu-Knoten Kommunikationen und der entfernten Überwachung. Die maximale Zeitgrenze müssen unbedingt nicht überschritten werden. Die Eigenschaft ist im **Viewpoint Simulation** einzuführen.
ii. **The traffic behavior of automation nodes:** Diese Paketübertragungshandiung zwischen den automatischen Knoten sind im **Viewpoint Queuing** besser anzeigt.
iii. **Lifetime of the components:** Jedes Gebäudeautomationsgerät hat seine eigene Lebenszeit. Die Zeit ist sehr wichtig. Wenn die Zeit überschritten wird, gibt es die Möglichkeit, dass das Automationssystem nicht funktioniert oder sogar gefährlich ist. Das Element muss im **Viewpoint Topology** beschrieben werden.
iv. **The range of environment:** Jedes Sensor hat die Grenzwerte, die in den lokalen Stationen (z.B. SPS, DDC, LON-Gerät…) oder entfernten PCs eingesetzt werden. Die Daten aus Umgebungen werden über den Sensoren im Automationssystem eingegeben. Die Daten werden in lokale Stationen durch die Programme bearbeitet. Wenn diese Daten die Grenzwerte überschreiten, werden Befehle an die entsprechenden Aktoren gesendet.
v. **The based services of the components:** Gegenüber Rechnernetzkomponenten hat jedes Gebäudeautomationsgerät seine eigene Funktion. Das muss auch im **Viewpoint Topology** erklärt werden.
vi. **The price of Software:** Manche Software für GLT, Facility-Managementsystem oder Interface zwischen Anwendungssoftware und Verwaltungsoftware für Automationsinfrastruktur muss man extra bezahlen.

7.4 Szenariobeschreibung mit NDML

Darunter wie Bild 7.4.1 gibt es ein Gebäudeautomationsszenario von zwei Zimmern mit LON-Bussystem.

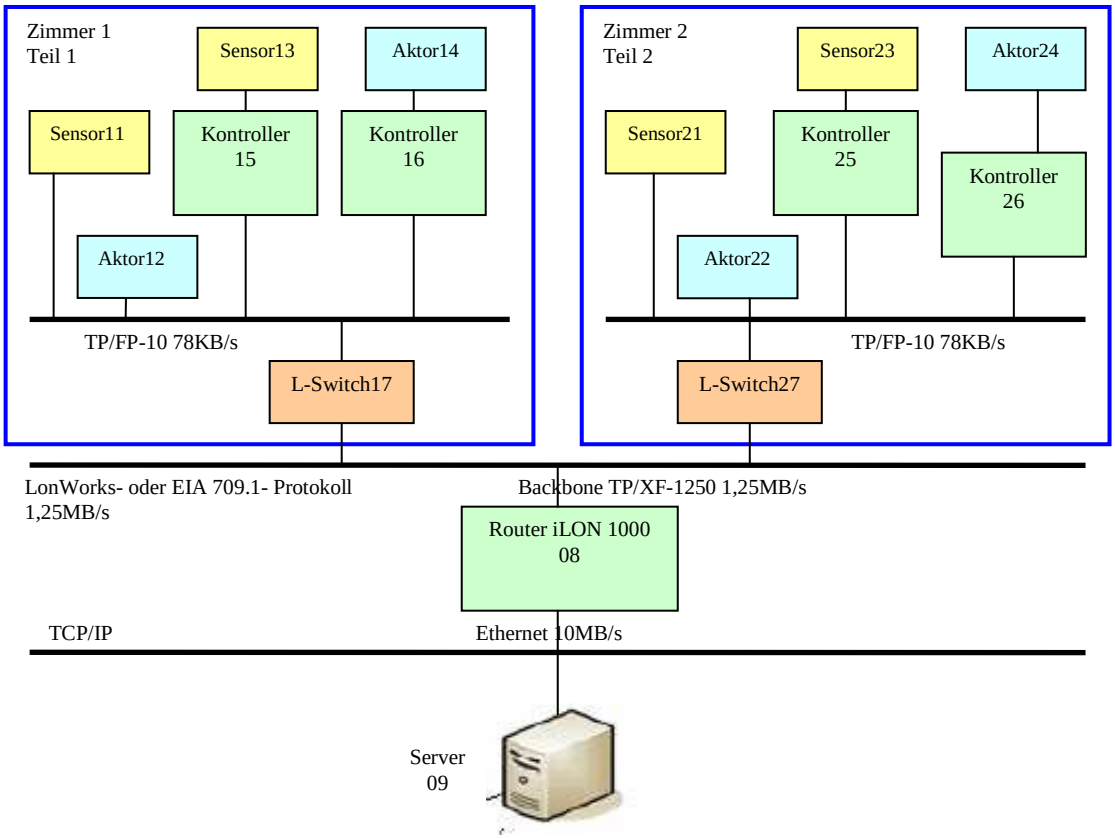


Bild 7.4.1 ein Szenario von Gebäudeautomationssystem mit zwei Zimmern

Eigentlich können die Hersteller ausreichend vielfältige Produkte für LON Automatisierungssysteme anbieten. Hier gibt es nur das einfachste Beispiel.
Im Beispiel werden zwei Zimmer dargestellt: Zimmer1, Zimmer2. Jedes Zimmer hat seines eigene kleine LON-Bussystem: Teil 1, Teil 2. Die zwei Teil LON-Bussysteme sind über die Kabel TP/FP-10 mit der Datenübertragungsrate 78KB/s getrennt durch die zwei Multiport-Routers, die L-Switch genannt werden, an dem Backbone-Kabel TP/XF-1250 mit der Datenübertragungsrate 1,25MB/s angekoppelt.
Das gesamte LON-Bussystem ist durch ein LON-Gerät, das Router iLON 1000 genannt wird, an Ethernet mit der Datenübertragungsrate 10MB/s verbunden. Durch iLON 1000 werden die zwei unterschiedliche Protokolle EIA 709.1- Protokoll und TCP/IP ineinander umgewandelt. An das Ethernet wird ein PC angekoppelt. Für LON-Bus ist der PC ein Client, der die Daten in LON-Geräte abfragt. Im Ethernet ist PC ein Server, mit dem die anderen PCs in Ethernet die Daten aus LON-Bus austauschen.
Sensor11 in Zimmer1 und Sensor21 in Zimmer2 sind die intelligenten Sensoren von Leuchten. Aktor12 in Zimmer1 und Aktor22 in Zimmer2 sind auch intelligent und kontrollieren den Leuchtband von jedem Zimmer. Diese Sensoren und Aktoren werden unmittelbar mit L-Switch verbindet.
Sensor13 in Zimmer1 und Sensor23 in Zimmer2 sind die unintelligenten Sensoren, die für die Temperatur des Zimmers und nicht direkt an L-Switch angekoppelt sind. Aktor14 in Zimmer1 und Aktor24 in Zimmer2 sind auch unintelligent und steuern die Klimaanlage jedes Zimmer, die auch an L-Switch angekoppelt werden dürfen. Dann werden die LON-Geräte mit Kontrollern an die Teilbusse angeschlossen. Die unintelligenten Sensoren und Aktoren sind durch Kontoller15，16 und Kontroller25，26 mit L-Switchs verbunden. Normalerweise wird jeder Kontroller nur Sensoren oder Aktoren getrennt verbinden. Das bedeutet, ein Kontroller darf Sensoren und Aktoren unmischbar angekoppeln.
Das ganz gesamte Bussystem darf man mit NDML3.0 in CANDY2.0 beschrieben.

Diese Bussystembeschreibung ist sehr ähnlich wie die Rechnernetzbeschreibung im der Diplomarbeit von Herrn Robert Hänel. Hier mit **NDML3.0** werden nur Viewpoint Basic, Viewpoint Load, Viewpoint Topology und Viewpoint Simulation beschrieben. Jede Viewpoint hat seine eigene XML- Dokumente und XML- Schema.

7.4.1 Viewpoint Load

Das Dokument projekt.xml ist ein Kerndokument im Viewpoint Load. Im diesem Dokument liegen die gründlichen Eigenschaften für die Bussystem-Projekt, wie die Zeitfrist des Projekts, die Kost des ganzen Projekts. Früher in NDML1.0/2.0 waren die Informationen nur die Attribute von Element <projekt>, jetzt sind sie auch eigene Elemente, damit können sie flexibler entsprechend den Anforderungen der Praxis verändert werden.

projekt.xml :

```
<project schemaVersion="3.0" projectid="projekt"…>
 <name>BasicProjekt</name>
 <description>eine LON-Busnetzwerkssystem von einem Geb鋳de mit zwei Zimmer</description>
 <customer>customer.xml</customer>
 <projectStart>2006-03-01</projectStart>
 <projectEnd>2006-09-31</projectEnd>
 <maxInstallationCosts currency="euro">10000</maxInstallationCosts>
 <maxRunningCosts currency="euro" per="month">250</maxRunningCosts>
 <files>
  <file>file1.xml</file>
  <file>file2.xml</file>
 </files>
</project>
```

Im Dokument Projekt gibt es die Elemente <customer> und <file>. Die Elemente haben die Referenzen zu den externen Dokumente customer.xml, file1.xml und file2.xml.
Customer.xml enthält die Informationen über den Kunden. Diese sind isoliert von projekt.xml und existieren für jeden Kunden.

customer.xml:

```
<customer schemaVersion="3.0" customerid="customer"…>
 <name>Muster GmbH</name>
 <contactPersons>
  <contactPerson>
   <firstName>Xiaojun</firstName>
   <lastName>Wei</lastName>
   <email>basara@XXX.com</email>
   <phone>123456789</phone>
   <notes>Mitteiler</notes>
  </contactPerson>
 </contactPersons>
</customer>
```

Neben projekt.xml geben die Dokumente file1.xml und file2.xml die zusätzlichen Informationen von dem Gebäudegrundriss des Projekts und den Notizen vom Gespräch mit dem Auftraggeber.

File1.xml erklärt den Gebäudegrundriss:

```
<file schemaVersion="3.0" fileid="file1"…>
 <ref>http://www.XXX.com/private/grundriss.jpg</ref>
```

```
 <type>Gebäudeautomation LON</type>
 <description>Skizze der Räume</description>
</file>
```

File2.xml gibt Notizen vom Gespräch mit dem Auftraggeber:

```
<file schemaVersion="3.0" fileid="file2"…>
 <ref>http://projects.planning.com/XXX.doc</ref>
 <type>text</type>
 <description>
  Notizen vom Gespräche mit dem Auftraggeber
 </description>
</file>
```

7.4.2 Viewpoint Load

In load.xml wird eine Lastbeschreibung des geplanten Netzwerks, dessen Struktur wie Bild 7.4.1 ist, grob erklärte.
Zuerst ist Element <devices>. Es besteht aus die Kinderelemente <device>. Die Elemente beschreiben alle belastenden Geräte des Netzwerkssystems.
Jedes Element <device> hat ein Attribut id, das eine eigene Identität vom Element <device> dokumentweit einmalig definiert. Das Kindelement <name> ist der Gerätename in der Praxis. Kindelement <description> erklärt grob die Funktion und Lokation des Gerätes. z.B. Die erste <device> ist Sensor11. Der ist ein intelligenter Sensor für die Leuchten im Zimmer1.
Besonderheiten sind, Router iLON1000 funktioniert als ein Gateway zwischen den Protokollen LonWork/EIA 709.1 und TCP/IP. Dieses Gerät liegt im Zugang des ganzen Gebäudes. <device id=“09“> ist ein PC. Der PC ist ein Client, der die Daten im LON-Netzwerksystem abfragt. Dieser PC muß nicht im Gebäude, sondern kann sehr entfernt (vielleicht in anderer Stadt) stehen. Der PC ist z.B. durch Ethernet mit LON-Netzwerksystem verbunden. Im Ethernet dient der PC auch als Server. Die anderen PCs im Ethernet können die Daten über LON-Netzwerksystem aus diesem Server bekommen.

```
<load schemaVersion="3.0" loadid="load"…>
<project>projekt.xml</project>
<devices>
  <device id="11">
   <name>Sensor 11</name>
   <description>intelligente Leuchtsensor in Zimmer1</description>
  </device>
  <device id="12">
   <name>Aktor 12</name>
   <description>intelligente Leuchtbandaktor in Zimmer1</description>
  </device>
    <device id="13">
   <name>Sensor 13</name>
   <description>unintelligente Temperatursensor in Zimmer1</description>
  </device>
  <device id="14">
   <name>Aktor 14</name>
   <description>unintelligente Klimaanlageaktor in Zimmer1</description>
  </device>
  <device id="15">
   <name>Kontroller 15</name>
   <description>Sensorkontroller in Zimmer1</description>
  </device>
  <device id="16">
   <name>Kontroller 16</name>
   <description>Aktorkontroller in Zimmer1</description>
```

```
 </device>
 <device id="17">
  <name>L-Switch 17</name>
  <description>Multiport-Router in Zimmer1</description>
 </device>
  … … … …
 <device id="08">
  <name>iLON1000</name>
  <description>Verbindung zwischen LON und Ethernet</description>
 </device>
 <device id="09">
  <name>PC</name>
  <description>ein Server</description>
 </device>
</devices>
```

Unter Element <buildautomation> werden alle <device> in 4 Segmenten verteilt. Die ersten 2 Teile sind die zwei Teil-LON-Netze in Zimmer1 und Zimmer2. Der dritte Teil ist <device> iLON1000, welches eine Verbindung zwischen LON und LAN ist. Der vierte Teil ist <device> Server, der an Ethernet angekoppelt ist. mit id=“5“ beschreibt das ganze Netzwerksystem.

```
<buildautomation>
 <segment id="1">
  <name>LON-Bus Segment 1</name>
  <contains>
   <device inDocument="this">Sensor 11</device>
   <device inDocument="this">Aktor 12</device>
   <device inDocument="this">Sensor 13</device>
   <device inDocument="this">Aktor 14</device>
   <device inDocument="this">Kontroller 15</device>
   <device inDocument="this">Kontroller 16</device>
   <device inDocument="this">L-Switch 17</device>
  </contains>
  <internalTraffic rate="KBit" per="second">78</internalTraffic>
 </segment>
 <segment id="2">
  <name>LON-Bus Segment 2</name>
  <contains>
   <device inDocument="this">Sensor 21</device>
   <device inDocument="this">Aktor 22</device>
   <device inDocument="this">Sensor 23</device>
   <device inDocument="this">Aktor 24</device>
   <device inDocument="this">Kontroller 25</device>
   <device inDocument="this">Kontroller 26</device>
   <device inDocument="this">L-Switch 27</device>
  </contains>
  <internalTraffic rate="KBit" per="second">78</internalTraffic>
 </segment>
 <segment id="3">
  <name>Router</name>
  <contains>
   <device inDocument="this">iLON1000</device>
  </contains>
 </segment>
 <segment id="4">
  <name>Server</name>
  <contains>
  <device inDocument="this">PC</device>
  </contains>
```

```
 </segment>
 <segment id="5">
  <name>Gebäude Bus</name>
  <contains>
   <segmentteil inDocument="this">LON-Bus Segment 1</segmentteil>
   <segmentteil inDocument="this">LON-Bus Segment 2</segmentteil>
   <segmentteil inDocument="this">Router</segmentteil>
   <segmentteil inDocument="this">Server</segmentteil>
  </contains>
 </segment>
</buildautomation>
```

Unter Element <connections> werden die Verbindungen von den 1 bis 4 beschrieben. In diesen Beschreibungen werden die Datenübertragungsrate und Protokolltyp erklärt. Z.B. bei Kindelement <connect> mit seinem Attribut id="1" ist eine Verbindung von 1 und 2; das bedeutet: eine Ankoppelung von LON-Netzwerksystem in Zimmer1 an einem LON-Busbackbone. Die Verbindung erfolgt mit Ethernet-Kabel. Die Datenübertragungsrate ist 1,25 MBit/s und die Protokoll ist LonWork/EIA 709.1.

```
<connections>
 <connection id="1">
  <description>Verbindung zwischen Netz von Zimmer1 und Backbone</description>
  <connects>
   <segmentteil inDocument="this">LON-Bus Segment 1</segmentteil>
   <segmentteil inDocument="this">Router</segmentteil>
  </connects>
  <type>wired</type>
  <traffic rate="MBit" per="second">1.25</traffic>
  <protokoll>LonWork-/EIA 709.1</protokoll>
 </connection>
 <connection id="2">
  <description>Verbindung zwischen Netz von Zimmer2 und Backbone</description>
  <connects>
   <segmentteil inDocument="this">LON-Bus Segment 2</segmentteil>
   <segmentteil inDocument="this">Router</segmentteil>
  </connects>
  <type>wired</type>
  <traffic rate="MBit" per="second">1.25</traffic>
  <protokoll>LonWork-/EIA 709.1</protokoll>
 </connection>
 <connection id="3">
  <description>Verbindung zwischen Netz von Backbone und Ethernet</description>
  <connects>
   <segmentteil inDocument="this">Router</segmentteil>
   <segmentteil inDocument="this">Server</segmentteil>
  </connects>
 <type>wired</type>
 <traffic rate="MBit" per="second">10</traffic>
 <protokoll>TCP/IP</protokoll>
 </connection>
</connections>
</load>
```

7.4.3 Viewpoint Topology

Viewpoint Topology ist ein Kern von CANDY. Der ist eine Erweiterung von Viewpoint Load und eine Basis von anderen Viewpoints, z.B. Viewpoint Simulation, Viewpoint Geometry, Viewpoint Cost, usw.

Zur Beschreibung der Topologie sind nicht nur ein XML Dokument, sondern **ein Satz von XML Dokumenten**, die Technology, Vendor, Plugprodukte, Mediumprodukte, Sensoren, Akoren, Controllern, Server und Netzwerkverbindungsgeräte (L-Switchs, iLON1000) getrennt beschrieben. Die Beziehungen zwischen den Dokumenten ist in Bild 7.4.3.1 gezeigt.

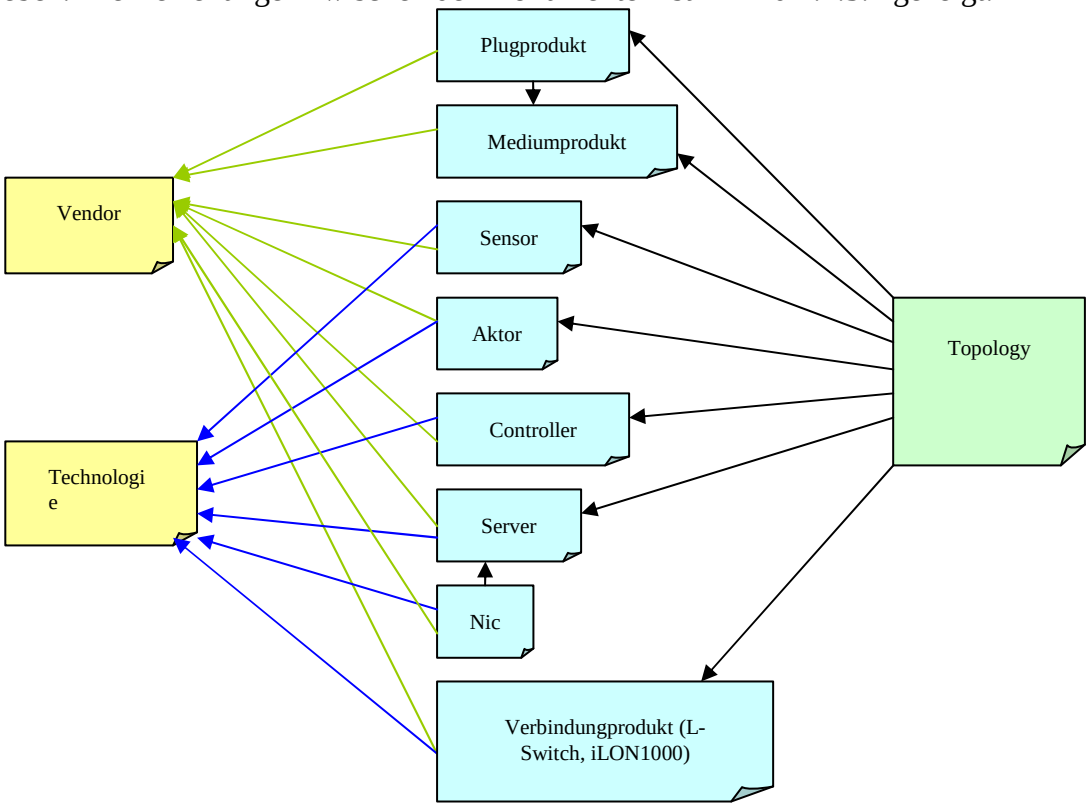


Bild 7.4.3.1 Beziehungen von den XML Dokumenten der Viewpoint Topologie

Die Vorteile von diesem getrennten Programmierungsverfahren sind, dass entsprechend der Änderung der Praxis nur wenige isolierte Dokumente verändert werden. Für die Benutzer von CANDY und Verwalter von Datenbank ist das Verfahren einfach und dafür braucht man wenig Arbeit. Aber für die Programmierer sind die vielen XML Dokumentkomponenten und Anweisungen der Beziehungen zwischen den Dokumenten schwer und kompliziert.
Hier muss man bemerkt die große **Unterschieden** der Topologiebeschreibungen zwischen dem Rechnernetz und dem Gebäudeautomationsnetz.:

i. In dem Rechnernetz sind PCs die Hauptgeräte. Sie sind Clients und Servers. Die Andere sind die Netzkopplungselemente z.B. Router, Switch, Gateway, Bridge usw. In der Diplomarbeit „Optimierung der Fachsprache NDML mit Abbildung in eine XML-basierte Datenbank“ von Robert Hänel haben die XML-Dokumente, die diese Geräte (PCs und Netzkopplungselemente) beschreiben, von nur **einem XML-Schema** definiert.
ii. Aber in dem Gebäudeautomationsnetz gibt es nicht nur PCs, die für Visualisierung, Verwaltung oder Datenbank verwendet werden, sondern auch die Automatisierungsgeräte z.B. DDC, LON-Gerät. Die Funktionen dieser Geräte sind unterschiedlich. Ich habe diese Geräte in meinem Beleg mit den XML-Dokumente beschrieben, die mit **den XML-Schemen „sensor.xsd“, „actor.xsd“, „controller.xsd“, „server.xsd“ und „verbindungdevice.xsd“** definiert werden. Damit werden die Funktionen dieser Geräte mehr detailliert erklärt. Aber für die externe Navigationen zwischen diesen XML-Schemen von den Geräte und dem Schema „topology.xsd“ ist es zu kompliziert und unflexibel. Wenn die Automationsgeräte von einer neuern Art im Gebäudeautomationsnetz einsetzt werden, ist die Topologie-Beschreibung wegen der komplizierten Navigationen und Verknüpfungen schwer zu erweitern. Deshalb hat die Topologie-Beschreibung in meinem Beleg **wenig externe**

Navigationen und Verknüpfungen. Die Funktionen von Navigationen und Verknüpfungen werden von dem Verwaltungsmechanismus (z.B. Berkeley DB XML) in der Datenbank oder anderen zusätzlichen Navigationsbeschreibungsdokumenten realisiert.

Technology1.xml:

Technology1.xml beschreibt die Kabeltechnik für LON-Bus-Verkabelung im Zimmer, die TP/FP-10 genannt ist. Hier benutzt man die TP (Twisted Pair) Kabel. Man darf STP (Shielded Twisted Pair) oder UTP (Unshielded Twisted Pair) auswählen. Die Datenübertragungsrate ist 78 KBit/s. Die max. Länge jedes Segments ist 1400 Meter. An jedem Segment werden max. 64 LON-Geräte angekoppelt.

```
<technology schemaVersion="3.0" technologyid="technology1" …>
 <name>TP/FP-10</name>
 <description>ein LON-Bus Kabel</description>
 <dataTransferRate unit="KBit" per="second">78</dataTransferRate>
 <compatibleMedia>
  <category maxLength="1400" unit="meter">stp</category>
  <category maxLength="1400" unit="meter">utp</category>
 </compatibleMedia>
 <maxStationsPerMedium>64</maxStationsPerMedium>
</technology>
```

Technology2.xml:

Technology2.xml beschreibt die Kabeltechnik für LON-Bus-Backbone, die TP/XP-1250 genannt ist. Für diese Kabeltechnik darf man auch STP oder UTP auswählen. Die Datenübertragungsrate ist 1,25 MBit/s. Die max. Länge jedes Segments ist 130 Meter. An jedem Segment werden auch max. 64 LON-Geräte angekoppelt.

```
<technology schemaVersion="3.0" technologyid="technology2" …>
 <name>TP/XF-1250</name>
 <description>ein LON-Bus Kabel</description>
 <dataTransferRate unit="MBit" per="second">1.25</dataTransferRate>
 <compatibleMedia>
  <category maxLength="130" unit="meter">stp</category>
  <category maxLength="130" unit="meter">utp</category>
 </compatibleMedia>
 <maxStationsPerMedium>64</maxStationsPerMedium>
</technology>
```

Technology3.xml:

Technology3.xml beschreibt die Kabeltechnik für Ethernet, die 10BaseT genannt ist. Hier benutzt man die TP Kabel. Man darf STP oder UTP auswählen. Bei UTP kann man aus die Alternativen UPT_CAT3, UTP_CAT4, UTP_CAT5 oder UTP_CAT6 einen günstigen Kabeltyp auswählen. Die Datenübertragungsrate ist 10 MBit/s. Die max. Länge jedes Segments ist 100 Meter.

```
<technology schemaVersion="3.0" technologyid="technology3" …>
 <name>10BaseT</name>
 <description>802.3</description>
 <dataTransferRate unit="MBit" per="second">10</dataTransferRate>
 <compatibleMedia>
  <category maxLength="100" unit="meter">stp</category>
```

```
  <category maxLength="100" unit="meter">utp_cat3</category>
  <category maxLength="100" unit="meter">utp_cat4</category>
  <category maxLength="100" unit="meter">utp_cat5</category>
  <category maxLength="100" unit="meter">utp_cat6</category>
 </compatibleMedia>
</technology>
```

Vendor1.xml:

Im Projekt gibt es zwei Händler. Ein ist vendor1.xml. Das Dokument ist für ein LON-Geräte Händler. Das andere Dokument vendor2.xml ist für ein normalen Netzwerk-Geräte Händler. Hier ist nur vendor1 Dokument als ein Beispiel:

```
<vendor schemaVersion="3.0" vendorid="vendor2"…>
 <name>Netzwerkhardware Shop</name>
 <description>ein Händler von Netzwerkhardware</description>
 <email>Network_Device@inwh-shop.de</email>
 <phone>333/98798</phone>
 <homepage>http://www.Networkhardware.de</homepage>
</vendor>
```

plugProduct1.xml:

Plugproduct Dokumente beschreiben die zwei Stecker jedes Kabels. plugProduckt1.xml beschreibt die Stecker vom Ethernetkabel. Sein Name ist RJ45. Die Farbe ist weiß. Der ist vom Fabrik „Noname“ hergestellt. Die Lebenszeit ist 10 Jahre. Der Preis ist 0.45 Euro pro Stück, 39,00 Euro per 100 Stücken. Weitere Informationen kann man auf der Webseite „http://www.LonWorks.de“ finden.

```
<plugProduct schemaVersion="3.0" plugProductid="plugProduct1" …>
 <name>RJ45-Stecker</name>
 <description>Farbe: weiss</description>
 <manufacturer>Noname</manufacturer>
 <lifetime unit="Jahr">10</lifetime>
 <moreInformation>http://www.LonWorks.de</moreInformation>
 <prices>
  <price currency="euro" quantity="1">0.45</price>
  <price currency="euro" quantity="100">39.00</price>
 </prices>
 <vendor>vendor1.xml</vendor>
 <type>RJ45</type>
</plugProduct>
```

plugProduct2.xml:

plugProduckt2.xml beschreibt die Stecker vom LON-Buskabel. Sein Name ist TP/FT10. Der ist entsprechend TP/FT10 Transceiver. Die anderen Informationen sind ähnlich wie plugProdukt1.xml.

```
<plugProduct schemaVersion="3.0" plugProductid="plugProduct2"…>
 <name>TP/FT10-Stecker</name>
 <description>für LON-Bus Transceiver</description>
 <manufacturer>Noname</manufacturer>
 <lifetime unit="Jahr">10</lifetime>
 <moreInformation>http://www.LonWorks.de</moreInformation>
 <prices>
  <price currency="euro" quantity="1">0.60</price>
  <price currency="euro" quantity="100">60.00</price>
```

```
 </prices>
 <vendor>vendor1.xml</vendor>
 <type>TP/FT10</type>
</plugProduct>
```

plugProduct3.xml:

plugProduckt2.xml beschreibt die Stecker vom LON-Busbackbone. Sein Name ist TP/FT1250. Der ist entsprechend TP/XF1250 Transceiver. Die anderen Informationen sind ähnlich wie plugProdukt1.xml.

```
<plugProduct schemaVersion="3.0" plugProductid="plugProduct3" …>
 <name>TP/XF1250-Stecker</name>
 <description>für LON-Bus Transceiver</description>
 <manufacturer>Noname</manufacturer>
 <lifetime unit="Jahr">10</lifetime>
 <moreInformation>http://www.LonWorks.de</moreInformation>
 <prices>
  <price currency="euro" quantity="1">0.65</price>
  <price currency="euro" quantity="100">65.00</price>
 </prices>
 <vendor>vendor1.xml</vendor>
 <type>TP/XF1250</type>
</plugProduct>
```

mediumProduct1.xml:

mediumProduckt1.xml beschreibt ein Ethernetkabel. Sein Name ist UTP CAT5. Der Widerstand ist 100 Ohm. Der Kabeltyp wird auch vom Fabrik „Noname“ hergestellt. Sein Steckertyp ist RJ45. Die Lebenszeit ist dauert auch 10 Jahre. Der Preis ist 4.99 Euro pro ein Meter, 7,49 Euro per 3 Meter, 9,99 Euro per 5 Meter, 12,49 Euro per 10 Meter, 14,99 Euro per 20 Meter. Weitere Informationen siehe „http://www.LonWorks.de“.

```
<mediumProduct schemaVersion="3.0" mediumProductid="mediumProduct1"…>
 <name>UTP-Kabel Cat5</name>
 <description>Impedanz: 100 Ohm</description>
 <manufacturer>Noname</manufacturer>
 <lifetime unit="Jahr">10</lifetime>
 <moreInformation>http://www.LonWorks.de</moreInformation>
 <prices>
  <price currency="euro" quantity="1" unit="meter">4.99</price>
  <price currency="euro" quantity="3" unit="meter">7.49</price>
  <price currency="euro" quantity="5" unit="meter">9.99</price>
  <price currency="euro" quantity="10" unit="meter">12.49</price>
  <price currency="euro" quantity="20" unit="meter">14.99</price>
 </prices>
 <vendor>vendor1.xml</vendor>
 <category>utp_cat5</category>
 <compatiblePlugs>
  <plugType>RJ45</plugType>
 </compatiblePlugs>
</mediumProduct>
```

MediumProduct2.xml:

MediumProduckt2.xml beschreibt ein LON-Buskabel. Der ist auch vom Fabrik „Noname“ hergestellt. Der Kabeltyp ist UPT. Sein Steckertyp ist TP/FT10. Die Lebenszeit ist

dauert auch 10 Jahre. Der Preis ist 3.99 Euro pro 3 Meter, 5,99 Euro per 5 Meter, 8,99 Euro per 10 Meter. Für mehre Informationen darf man nach „http://www.LonWorks.de“ besuchen.

```
<mediumProduct schemaVersion="3.0" mediumProductid="mediumProduct2" …>
 <name>LON-Bus Kabel</name>
 <description>mit TP/FT10-Stecker</description>
 <manufacturer>Noname</manufacturer>
 <lifetime unit="Jahr">10</lifetime>
 <moreInformation>http://www.LonWorks.de</moreInformation>
 <prices>
  <price currency="euro" quantity="3" unit="meter">3.99</price>
  <price currency="euro" quantity="5" unit="meter">5.99</price>
  <price currency="euro" quantity="10" unit="meter">8.99</price>
 </prices>
 <vendor>vendor1.xml</vendor>
 <category>utp</category>
 <compatiblePlugs>
  <plugType>TP/FT10</plugType>
 </compatiblePlugs>
</mediumProduct>
```

MediumProduct3.xml:

MediumProduckt3.xml beschreibt ein LON-Busbackbone. Der ist auch vom Fabrik „Noname“ hergestellt. Der Kabeltyp ist UPT. Sein Steckertyp ist TP/XF1250. Die Lebenszeit ist dauert auch 10 Jahre. Der Preis ist ähnlich wie MediumProdukct2.xml. Für mehre Informationen darf man nach „http://www.LonWorks.de“ besuchen.

```
<mediumProduct schemaVersion="3.0" mediumProductid="mediumProduct3"…>
 <name>LON-Backbone</name>
 <description>mit TP/XF1250-Stecker</description>
 <manufacturer>Noname</manufacturer>
 <lifetime unit="Jahr">10</lifetime>
 <moreInformation>http://www.LonWorks.de</moreInformation>
 <prices>
  <price currency="euro" quantity="3" unit="meter">3.99</price>
  <price currency="euro" quantity="5" unit="meter">5.99</price>
  <price currency="euro" quantity="10" unit="meter">8.99</price>
 </prices>
 <vendor>vendor1.xml</vendor>
 <category>utp</category>
 <compatiblePlugs>
  <plugType>TP/XF1250</plugType>
 </compatiblePlugs>
</mediumProduct>
```

Sensor11.xml:

Im Projekt gibt es vier Sensoren. Im Zimmer1 liegen Sensor11, Sensor13 und im Zimmer2 liegen Sensor21 und Sensor23. Hier werden nur Sensor11 und Sensor13 als die Beispiele erklärt. Sensor11.xml besitzt ein Attribut „deviceProductid“ mit Wert „sensor11“. Er ist ein intelligentes LON-Gerät mit der Funktion von Leuchtsensor und ist vom Fabrik „LonWorks Hersteller“ hergestellt. Die Lebenszeit ist dauert auch 10 Jahre. Der Preis ist 79 Euro. Für mehre Informationen darf man nach „http://www.LonWorks.de“ besuchen. Bei Element <location> wird Sensor11 im Zimmer1 lokalisiert.
Wie in Kapital 6.2 LON-Gerät nimmt ein LON-Sensor die Daten aus Umgebungen mit dem Frequenz von 0,3Hz bis 2,5Hz per 0,5 Tast durch I/O_0 bis I/O_10 im NEURON-Chip auf. Die Dauer der zwei diskreten Pulse ist von 0,2 ms bis 1,678 s und mit jedem Puls nimmt das Gerät 8

Bit Signale. Das heißt, dass Element <timer> mit Wert von 0,2 ms bis 1678ms sein kann, Element <frequency> mit Wert von 0,3 Hz bis 2500000 Hz sein kann und <daten> 8 Bit ist.
Element <transceiver> beschreibt einen einzelnen Transceiver, mit dem ein LON-Sensor an den LON-Bus angekoppelt wird. Die Plugtyp des Transceivers ist TP/FT10 mit Technology1. Seine Datenübertragungsrate ist 78 KBit/s. Die max. Paketlänge ist 31 Byte.
Dieser LON-Sensor skaliert die Helligkeit in 0 bis 10 Grad. Jetzt ist Element <brightness> mit Wert 5 Grad.

```
<deviceSensor schemaVersion="3.0" deviceProductid="sensor11" …>
 <name>Leuchtsensor</name>
 <description>intelligente LON-Sensorgerät</description>
 <manufacturer>LonWorks Hersteller</manufacturer>
 <lifetime unit="Jahr">10</lifetime>
 <moreInformation>http://www1.euro.dell.com/content/products/...</moreInformation>
 <price currency="euro" quantity="1">79</price>
 <vendor>vendor1.xml</vendor>
 <category>Sensor</category>
 <function>Sensor für die Helligkeit des Zimmers</function>
 <location>Zimmer1</location>
 <inputdaten>
  <timer unit="ms" per="puls">1678</timer>
  <freqeuncy unit="Hz" per="0,5Tastverhältnis">2500000</freqeuncy>
  <daten unit="Bit" per="tast">8</daten>
  <description>Daten über Sensor aus der Umgebung</description>
 </inputdaten>
 <transceiverGroup>
  <transceiver transceiverid="1" plugType="TP/FT10">
   <technology>technology1.xml</technology>
   <rate unit="KBit" per="second">78</rate>
   <maxmessagelength unit="Byte">31</maxmessagelength>
   <description>Datentausch mit LON-Bus</description>
  </transceiver>
 </transceiverGroup>
 <brightness unit="lux">5</brightness>
</deviceSensor>
```

Sensor13.xml

Wie Sensor11.xml besitzt sensor13.xml auch ein Attribut „deviceProductid“ mit Wert „sensor13“. Aber der ist kein intelligentes LON-Gerät und darf nicht unmittelbar, sondern über einem LON-Kontroller an LON-Bus verbunden werden. Das bedeutet, dass der Sensor über eigener Stichleitung an LON-Kontroller angekoppelt wird. Dieser Sensor ist ein Temperatursensor und ist vom Fabrik „LonWorks Hersteller“ hergestellt. Die Lebenszeit ist dauert auch 10 Jahre. Der Preis ist 39 Euro. Für mehre Informationen darf man nach „http://www.LonWorks.de“ besuchen. Bei Element <location> wird der Sensor auch im Zimmer1 lokalisiert.
Dieser Sensor skaliert die Temperatur des Zimmers von -10 C bis 85 C. Jetzt ist Element <temperatur> mit Wert 25 C.

```
<deviceSensor schemaVersion="3.0" deviceProductid="sensor13"…>
 <name>Temperatursensor</name>
 <description>unintelligente Sensor</description>
 <manufacturer>LonWorks Hersteller</manufacturer>
 <lifetime unit="Jahr">10</lifetime>
 <moreInformation>http://www1.euro.dell.com/content/products/...</moreInformation>
 <price currency="euro" quantity="1">39</price>
 <vendor>vendor1.xml</vendor>
 <category>Sensor</category>
```

```
 <function>Sensor für die Temperatur des Zimmers</function>
 <location>Zimmer1</location>
 <inputsplug>
  <description>Datensendung von Sensor direkt zur KontrollerI/O</description>
 </inputsplug>
 <temperatur unit="C">25</temperatur>
</deviceSensor>
```

Actor12.xml:

Im Projekt gibt es vier Aktoren. Im Zimmer1 liegen Aktor12, Aktor14 und im Zimmer2 liegen Aktor22 und Aktor24. Hier werden nur Aktor12 und Aktor14 als die Beispiele erklärt.
Aktor11.xml besitzt ein Attribut „deviceProductid“ mit Wert „aktor12“. Wie Sensor11 ist der Aktor ein intelligentes LON-Gerät mit der Funktion von Leuchtbandschaltungsaktor und ist vom Fabrik „LonWorks Hersteller“ hergestellt. Die Lebenszeit ist dauert auch 10 Jahre. Der Preis ist 79 Euro. Für mehre Informationen darf man nach „http://www1.euro.dell.com/content/.....“ besuchen. Bei Element <location> wird Aktor12 im Zimmer1 lokalisiert.
Gegen Sensor11 entwickelt LON-Aktor12 die Umgebungen mit dem Frequenz von 0,3Hz bis 2,5Hz per 0,5 Tast durch I/O_0 bis I/O_10. Die Dauer der zwei diskreten Pulse ist von 0,2 ms bis 1,678 s und mit jedem Puls gibt das Gerät 8 Bit Signale aus. Das heißt, dass Element <timer> mit Wert von 0,2 ms bis 1678ms sein kann, Element <freqeuncy> mit Wert von 0,3 Hz bis 2500000 Hz sein kann und <daten> 8 Bit ist.
Element <transceiver> beschreibt einen einzelnen Transceiver, mit dem ein LON-Aktor an LON-Bus angekoppelt wird. Die Plugtyp des Transceivers ist TP/FT10 mit Technology1. Seine Datenübertragungsrate ist 78 KBit/s. Die max. Paketlänge ist 31 Byte.
Dieser Aktor hat Element <actorstate> mit die „Boolean“ Werte. „1“ bedeutet die Einschaltung des Leuchtbands, „0“ bedeutet die Ausschaltung des Leuchtbands.

```
<deviceAktor schemaVersion="3.0" deviceProductid="actor12" …>
 <name>Leuchtbandactor</name>
 <description>intelligente LON-Sensorgerät</description>
 <manufacturer>LonWorks Hersteller</manufacturer>
 <lifetime unit="Jahr">10</lifetime>
 <moreInformation>http://www1.euro.dell.com/content/products/...</moreInformation>
 <price currency="euro" quantity="1">79</price>
 <vendor>vendor1.xml</vendor>
 <category>Aktor</category>
 <function>Aktor für die Helligkeit des Zimmers</function>
 <location>Zimmer1</location>
 <outputdaten>
  <timer unit="ms" per="plus">1678</timer>
  <freqeuncy unit="Hz" per="0,5Tastverhältnis">2500000</freqeuncy>
  <daten unit="Bit" per="tast">8</daten>
  <description>Daten für Aktivierung des Aktors</description>
 </outputdaten>
 <transceiverGroup>
  <transceiver transceiverid="1" plugType="TP/FT10">
   <technology>technology1.xml</technology>
   <rate unit="KBit" per="second">78</rate>
   <maxmessagelength unit="Byte">31</maxmessagelength>
   <description>Datentausch mit LON-Bus</description>
  </transceiver>
 </transceiverGroup>
 <actorstate>1</actorstate>
</deviceAktor>
```

Actor14.xml:

Wie Sensor13.xml ist der Aktor14 auch kein intelligentes LON-Gerät und darf nicht unmittelbar, sondern nur über einem LON-Kontroller an LON-Bus verbunden werden. Der Aktor wird über eine eigene Stichleitung an LON-Kontroller angekoppelt. Dieser Aktor ist ein Klimaanlageschaltungsaktor und ist vom Fabrik „LonWorks Hersteller“ hergestellt. Die Lebenszeit ist dauert auch 10 Jahre. Der Preis ist 39 Euro. Für mehre Informationen darf man nach „http://www1.euro.dell.com/.....“ besuchen. Bei Element <location> wird der Sensor auch im Zimmer1 lokalisiert. Der Aktor hat eine Energieversorgung mit 24 VDC (Direct Current).
Dieser Aktor hat Element <actorstate> mit die „Boolean“ Werte. „1“ bedeutet die Einschaltung der Klimaanlage, „0“ bedeutet die Ausschaltung der Klimaanlage.

```
<deviceAktor schemaVersion="3.0" deviceProductid="actor14"…>
 <name>Klimaanlageaktor</name>
 <description>unintelligente Aktor</description>
 <manufacturer>LonWorks Hersteller</manufacturer>
 <lifetime unit="Jahr">10</lifetime>
 <moreInformation>http://www1.euro.dell.com/content/products/...</moreInformation>
 <price currency="euro" quantity="1">39</price>
 <vendor>vendor1.xml</vendor>
 <category>Aktor</category>
 <function>Aktor für die Klimaanlage</function>
 <location>Zimmer1</location>
 <outputsplug>
  <description>Datensendung von Aktor direkt zur KontrollerI/O</description>
 </outputsplug>
 <voltage unit="VDC">24</voltage>
 <actorstate>1</actorstate>
</deviceAktor>
```

Controller15.xml:

Im Projekt gibt es vier Controller. Im Hier werden nur Controller15 und Controller16 im Zimmer1 als die Beispiele erklärt.
controller15.xml besitzt ein Attribut „deviceProductid“ mit Wert „controller15“. Alle Controller sind intelligente LON-Geräte. Controller15 verbindet Sensor13 mit LON-Bus und verwaltet Sensor13. Der Controller ist vom Fabrik „LonWorks Hersteller“ hergestellt. Die Lebenszeit ist dauert auch 10 Jahre. Der Preis ist 179 Euro. Für mehre Informationen darf man nach „http://www1.euro.dell.com/.....“ besuchen. Bei Element <location> wird Controller15 im Zimmer1 lokalisiert.
Eigentlich kann Controller15 mit 12 unintelligente Sensoren verbunden werden. Deshalb hat Element <inputsportGroup> 12 Kindelemente <inputsport>. Bei jedem Element <inputsport> ist das Kindelement <timer> mit Werten von 0,2 ms bis 1678ms belegt, Kindelemente <frequency> mit Werten von 0,3 Hz bis 2500000 Hz und <daten> mit 8 Bits. Hier ist der erste Inputport mit Sensor13 verbunden.
Element <transceiver> beschreibt einen einzelnen Transceiver, mit dem ein Controller an LON-Bus angekoppelt wird. Die Plugtyp des Transceivers ist TP/FT10 mit Technology1. Seine Datenübertragungsrate ist 78 KBit/s. Die max. Paketlänge ist 31 Byte.

```
<deviceController schemaVersion="3.0" deviceProductid="controller15"…>
 <name>SensorKontroller</name>
 <description>intelligente LON-Kontrollergerät</description>
 <manufacturer>LonWorks Hersteller</manufacturer>
 <lifetime unit="Jahr">10</lifetime>
 <moreInformation>http://www1.euro.dell.com/content/products/...</moreInformation>
 <price currency="euro" quantity="1">179</price>
```

```
<vendor>vendor1.xml</vendor>
<category>Kontroller</category>
<function>Kontroller von den unintelligenten Sensors</function>
<location>Zimmer1</location>
<inputsportGroup portsquantity="12">
 <timer unit="ms" per="plus">1678</timer>
 <freqeuncy unit="Hz" per="0,5Tastverhältnis">2500000</freqeuncy>
 <daten unit="Bit" per="tast">8</daten>
 <description>Daten über Sensor im Kontroller</description>
 <inputsport id="1">
  <description>direkt verbinden mit einem unintelligenten Sensor</description>
  <temperatur unit="C">25</temperatur>
 </inputsport>
   … … … …
 <inputsport id="12">
  <description>direkt verbinden mit einem unintelligenten Sensor</description>
 </inputsport>
</inputsportGroup>
<transceiverGroup>
 <transceiver transceiverid="1" plugType="TP/FT10">
  <technology>technology1.xml</technology>
  <rate unit="KBit" per="second">78</rate>
  <maxmessagelength unit="Byte">31</maxmessagelength>
  <description>Datentausch mit LON-Bus</description>
 </transceiver>
</transceiverGroup>
</deviceController>
```

Controller16.xml:

Controller16.xml besitzt ein Attribut „deviceProductid“ mit Wert „controller16“. Controller16 verbindet Aktor14 mit LON-Bus und verwaltet Aktor14. Der Controller ist auch vom Fabrik „LonWorks Hersteller“ hergestellt. Die Lebenszeit dauert auch 10 Jahre. Der Preis ist 179 Euro wie Controller15. Für mehre Informationen darf man nach „http://www1.euro.dell.com/content/.....“ besuchen. Bei Element <location> wird Aktor12 im Zimmer1 lokalisiert.
Gleich wie Controller15 kann Controller16 mit 12 unintelligente Aktoren verbunden werden. Aber gegen Controller15 hat Element <outputsportGroup> des Controller16s 1. bis 6. Kinderelemente <outputsport> mit 24VDC Energieversorgung und 7. bis 12. Kinderelemente <outputsport> mit 230VAC Energieversorgung. Bei jedem Element <outputsport> ist Kinderelement <timer> mit Wert von 0,2 ms bis 1678ms sein kann, Kinderelement <freqeuncy> mit Wert von 0,3 Hz bis 2500000 Hz sein kann und <daten> 8 Bit ist. Hier ist der erste Outputport mit Aktor14 verbunden.
Element <transceiver> beschreibt der einzelne Transceiver, mit dieses LON-Aktor an LON-Bus angekoppelt wird. Die Plugtyp des Transceivers ist TP/FT10 mit Technology1. Seine Datenübertragungsrate ist 78 KBit/s. Die max. Paketlänge ist 31 Byte.

```
<deviceController schemaVersion="3.0" deviceProductid="controller16"…>
 <name>AktorKontroller</name>
 <description>intelligente LON-Kontrollergerät</description>
 <manufacturer>LonWorks Hersteller</manufacturer>
 <lifetime unit="Jahr">10</lifetime>
 <moreInformation>http://www1.euro.dell.com/content/products/...</moreInformation>
 <price currency="euro" quantity="1">179</price>
 <vendor>vendor1.xml</vendor>
 <category>Kontroller</category>
 <function>Kontroller von den unintelligenten Aktors</function>
 <location>Zimmer1</location>
```

```
<outputsportGroup portsquantity="12">
 <timer unit="ms" per="plus">1678</timer>
 <freqeuncy unit="Hz" per="0,5Tastverhältnis">2500000</freqeuncy>
 <daten unit="Bit" per="tast">8</daten>
 <description>Daten über Sensor im Kontroller</description>
 <outputsport id="1">
  <description>direkt verbinden mit einem unintelligenten Sensor</description>
  <voltage unit="VDC">24</voltage>
  <actorstate>1</actorstate>
 </outputsport>
   … … … …
 <outputsport id="6">
  <description>direkt verbinden mit einem unintelligenten Sensor</description>
  <voltage unit="VDC">24</voltage>
 </outputsport>
 <outputsport id="7">
  <description>direkt verbinden mit einem unintelligenten Sensor</description>
  <voltage unit="VAC">230</voltage>
 </outputsport>
   … … … …
 <outputsport id="12">
  <description>direkt verbinden mit einem unintelligenten Sensor</description>
  <voltage unit="VAC">230</voltage>
 </outputsport>
</outputsportGroup>
<transceiverGroup>
 <transceiver transceiverid="1" plugType="TP/FT10">
  <technology>technology1.xml</technology>
  <rate unit="KBit" per="second">78</rate>
  <maxmessagelength unit="Byte">31</maxmessagelength>
  <description>Datentausch mit LON-Bus</description>
 </transceiver>
</transceiverGroup>
</deviceController>
```

Nic.xml:

nic.xml besitzt ein Attribut „interfaceProductid“ mit Wert „nic“. Die Netzwerkkarte wird „PILA8460BN“ genannt und vom Fabrik „Intel“ hergestellt. Die Lebenszeit ist dauert auch 10 Jahre. Der Preis ist 19 Euro. Für mehre Informationen darf man nach „http://www.Networkhardware.de“ besuchen.
Element <hardwareInterface> beschreibt die Verbindung mit PC durch PCI-Slot.
Element <ports> beschreibt den Port zu Ethernet mit RJ45 Plugtyp und Technology3.

```
<interfaceProduct schemaVersion="3.0" interfaceProductid="nic" …>
 <name>PILA8460BN</name>
 <description>Netzwerkkarte PCI , 10/100 MBit/s , ...</description>
 <manufacturer>Intel</manufacturer>
 <lifetime unit="Jahr">10</lifetime>
 <moreInformation>http://www.Networkhardware.de</moreInformation>
 <prices>
  <price quantity="1" currency="euro">19</price>
  <vendor>vendor2.xml</vendor>
 </prices>
 <hardwareInterfaces>
  <hardwareInterface>pci</hardwareInterface>
 </hardwareInterfaces>
 <portGroups>
  <ports plugType="RJ45">1</ports>
  <technology>technology3.xml</technology>
```

```
 </portGroups>
 <home>PC</home>
</interfaceProduct>
```

Server09.xml:

Server09.xml besitzt ein Attribut „deviceProductid“ mit Wert „server09“. Der PC-Name ist „PowerEdge 1800 Server“ und vom Fabrik „Dell GmbH“ hergestellt. Die Lebenszeit ist dauert auch 10 Jahre. Der Preis ist 1279 Euro. Für mehre Informationen darf man nach „http://www.Networkhardware.de“ besuchen.
Unter Element < portGroups> beschreibt Kinderelement <ports> den Port zu Ethernet mit RJ45 Plugtyp und Technology3.

```
<deviceServer schemaVersion="3.0" deviceProductid="server09" …>
 <name>PowerEdge 1800 Server</name>
 <description>Intel Xeon 2.8 GHz , ...</description>
 <manufacturer>Dell GmbH</manufacturer>
 <lifetime unit="Jahr">10</lifetime>
 <moreInformation>http://www.Networkhardware.de</moreInformation>
 <price currency="euro" quantity="1">1279</price>
 <vendor>vendor2.xml</vendor>
 <category>server</category>
 <function>entfernte Kontrollierung von LON-Bus über Ethernet</function>
 <location>Entfernung</location>
 <interfaces>
  <interface interfaceid="1">
   <description>Onboard-Netzwerkadapter</description>
   <portGroups>
    <ports plugType="RJ45">1</ports>
    <technology>technology3.xml</technology>
   </portGroups>
  </interface>
 </interfaces>
 <upgradeSlots>
  <slots hardwareInterface="pci">3</slots>
 </upgradeSlots>
</deviceServer>
```

L_Switch17.xml:

L_Switch17 und L_Switch27 sind 5-Multiports-Routeren. Sie setzen die LON-Gräte zusammen und verbinden sie mit LON-Busbackbone. L_Switch17 wird für Zimmer1 gestellt und L_Switch27 für Zimmer2. Die beiden XML-Dokumente sind von XML-Schema „verbindungdevice.xsd“ definiert. Hier wird L_Switch17 als ein Beispiel erklärt.
L_Switch17.xml besitzt ein Attribut „deviceProductid“ mit Wert „L_Switch17“. Der ist vom Fabrik „LonWorks Hersteller“ hergestellt. Die Lebenszeit ist dauert auch 10 Jahre. Der Preis ist 279 Euro. Für mehre Informationen darf man nach „http://www1.euro.dell.com/content/products/...“ besuchen. Bei Element <location> wird L_Switch17 im Zimmer1 lokalisiert.
Unter Element <transceiverGroup> gibt es vier Kinderelemente <transceiver> mit „transceiverid“ von 1 bis 4 für die LON-Geräte von Zimmer1. Die Plugtyp des Transceivers ist TP/FT10 mit Technology1. Seine Datenübertragungsrate ist 78 KBit/s. Die max. Paketlänge ist 31 Byte.
Und gibt es noch ein Kindelement <transceiver> mit „transceiverid“ von 5. Mit diesem Transceiver wird L_Switch17 über LON-Busbackbone an Router iLON1000 verbunden. Die

Plugtyp dieses Transceivers ist TP/XF1250 mit Technology2. Seine Datenübertragungsrate ist 1,25 MBit/s. Die max. Paketlänge ist 31 Byte.

```
<deviceVerbindung schemaVersion="3.0" deviceProductid="L-Switch17" …>
 <name>L-Switch</name>
 <description>Multiports LON-Routergerät</description>
 <manufacturer>LonWorks Hersteller</manufacturer>
 <lifetime unit="Jahr">10</lifetime>
 <moreInformation>http://www1.euro.dell.com/content/products/...</moreInformation>
 <price currency="euro" quantity="1">279</price>
 <vendor>vendor1.xml</vendor>
 <category>5ports-Router</category>
 <function>Switch für die LON-Geräte des Zimmers</function>
 <location>Zimmer1</location>
 <transceiverGroup>
  <transceiver transceiverid="1" plugType="TP/FT10">
   <technology>technology1.xml</technology>
   <rate unit="KBit" per="second">78</rate>
   <maxmessagelength unit="Byte">31</maxmessagelength>
   <description>Datentausch mit LON-Bus</description>
  </transceiver>
    … … … …
  <transceiver transceiverid="5" plugType="TP/XF1250">
   <technology>technology2.xml</technology>
   <rate unit="MBit" per="second">1.25</rate>
   <maxmessagelength unit="Byte">31</maxmessagelength>
   <description>Datentausch mit LON-Bus</description>
  </transceiver>
 </transceiverGroup>
</deviceVerbindung>
```

iLON1000 08.xml:

iLON1000 funktioniert als ein Gateway zwischen LON-Bussystem und Ethernet. Sie übersetzt die Protokolle LonWork/EIA709.1 und TCP/IP von einander.
iLON1000 08.xml besitzt ein Attribut „deviceProductid“ mit Wert „iLON1000 08“. Der ist vom Fabrik „LonWorks Hersteller“ hergestellt. Die Lebenszeit ist dauert auch 10 Jahre. Der Preis ist 279 Euro. Für mehre Informationen darf man nach „http://www1.euro.dell.com/content/products/...“ besuchen. Bei Element <location> wird L_Switch17 im Zimmer1 lokalisiert.
Unter Element <transceiverGroup> gibt es zwei Kindelemente <transceiver> mit „transceiverid“ von 1 bis 2. Die sind mit L_Switch17 und L_Switch27 verbunden. Die Plugtyp dieses Transceivers ist TP/XF1250 mit Technology2. Seine Datenübertragungsrate ist 1,25 MBit/s. Die max. Paketlänge ist 31 Byte. (Hier muss man bemerken, entsprechend verschiedenem Typ dieses Routers gibt es unterschiedliche Plugtypen für die zwei Transceiver, im ungleichen Anwendungsgebiete zu benutzen, z.B. TP/FT10.)
Für die Verbindung mit Ethernet hat iLON1000 noch ein Kindelement <transceiver> mit „transceiverid“ von 3. Die Plugtyp dieses Transceivers ist 10BaseT RJ45 mit Technology2. Seine Datenübertragungsrate ist 10 MBit/s.
Dieses XML-Dokument ist auch von XML-Schema „verbindungdevice.xsd“ definiert.

```
<deviceVerbindung schemaVersion="3.0" deviceProductid="iLON1000 08" …>
 <name>iLON1000</name>
 <description>LON-Routergerät als Gateway</description>
 <manufacturer>LonWorks Hersteller</manufacturer>
 <lifetime unit="Jahr">10</lifetime>
 <moreInformation>http://www1.euro.dell.com/content/products/...</moreInformation>
```

```
 <price currency="euro" quantity="1">279</price>
 <vendor>vendor1.xml</vendor>
 <category>IP-Router</category>
 <function>Gateway zwischen LON-Bus und Ethernet</function>
 <location>Etage</location>
 <transceiverGroup>
  <transceiver transceiverid="1" plugType="TP/XF1250">
   <technology>technology2.xml</technology>
   <rate unit="MBit" per="second">1.25</rate>
   <maxmessagelength unit="Byte">31</maxmessagelength>
   <description>Datentausch mit LON-Bus</description>
  </transceiver>
    … … … …
  <transceiver transceiverid="3" plugType="10BaseT RJ45">
   <technology>technology3.xml</technology>
   <rate unit="MBit" per="second">10</rate>
   <description>Datentausch mit Ethernet</description>
  </transceiver>
 </transceiverGroup>
</deviceVerbindung>
```

Topology.xml:

Topology.xml ist eine detaillierte Erweiterung auf dem Grund des load.xml Dokuments. Das Dokument beschreibt die Struktur des Netzwerksystems.
Zuerst werden die Geräte oder so genannte Knoten mit den Kinderelementen <device> unter Element <devices> erklärt. Die Elemente beschreiben alle belastenden Geräte des Netzwerkssystems.
Wie im load.xml Dokument hat jedes Element <device> ein Attribut id, das eine eigene Identität vom Element <device> dokumentweit einmalig definiert. Kinderelement <name> ist der Gerätname im Praxis. Kinderelement <description> erklärt grob die Funktion und Lokation des Gerätes. z.B. Die erste <device> ist Sensor11. Der ist ein intelligente Sensor für die Leuchten im Zimmer1.
Besonderheiten sind: Router iLON1000 funktioniert als ein Gateway zwischen den Protokollen LonWork/EIA 709.1 und TCP/IP. Und man liegt dieses Gerät im Zugang des ganzen Gebäudes. Die <device id="09"> ist ein PC. Der PC ist ein Client, der die Daten im LON-Netzwerksystem abfragt. Dieser PC kann im Gebäude, aber auch sehr entfernt (vielleicht in anderer Stadt) stehen. Der PC ist durch Ethernet mit LON-Netzwerksystem verbunden. Im Ethernet dient der PC auch als Server. Die anderen PCs im Ethernet können die Daten über LON-Netzwerksystem von diesem Server bekommen.

```
<topology schemaVersion="3.0" topologyid="topology"…>
 <project>projekt.xml</project>
 <devices>
  <device id="s11">
   <name>sensor 11</name>
   <productid>sensor11</productid>
   <description>intelligente Sensor</description>
  </device>
    … … … …
  <device id="a12">
   <name>actor 12</name>
   <productid>actor12</productid>
   <description>intelligente Aktor</description>
  </device>
    … … … …
  <device id="s13">
   <name>sensor 13</name>
```

```
 <productid>sensor13</productid>
 <description>unintelligente sensor</description>
</device>
  … … … …
<device id="a14">
 <name>actor 14</name>
 <productid>actor14</productid>
 <description>unintelligente aktor</description>
</device>
  … … … …
<device id="c15">
 <name>c15</name>
 <productid>controller15</productid>
 <description>SensorKontroller</description>
</device>
  … … … …
<device id="c16">
 <name>controller 16</name>
 <productid>controller16</productid>
 <description>AktorKontroller</description>
</device>
  … … … …
<device id="L-S17">
 <name>L-Switch 17</name>
 <productid>L-Switch17</productid>
 <description>5 ports-Router</description>
</device>
   … … … …
<device id="iL08">
 <name>iLON1000</name>
 <productid>iLON1000 08</productid>
 <description>Gateway</description>
</device>
<device id="server09">
 <name>PC</name>
 <productid>server09</productid>
 <description>Server</description>
</device>
</devices>
```

Unter Element <media> wird jedes Kabel mit Element <medium> beschrieben. Jedes Element <medium> hat ein Attribut „id“, mit dem kann jedes Kabel im Dokument topology.xml identifiziert wird. Mit Elemente <connectedTo> und <plugs> wird erklärt, an welche Geräte das Kabel angekoppelt wird und welche zwei Stecker an dem Kabel zusammengesetzt werden.

```
<media>
 <medium id="m1">
  <name>m 1</name>
  <description>direkt verbindung zwischen unintelligente sensor13 und sensorcontroller15</description>
  <connectedTo>
   <device inputsport="0" inDocument="this">sensor 13</device>
   <device inputsport="1" inDocument="this">controller 15</device>
  </connectedTo>
 </medium>
  … … … …
 <medium id="m3">
  <name>m 3</name>
  <description>direkt verbindung zwischen unintelligente aktor14 und aktorcontroller16</description>
  <connectedTo>
   <device outputsport="0" inDocument="this">actor 14</device>
```

```
  <device outputsport="1" inDocument="this">controller 16</device>
 </connectedTo>
</medium>
  … … … …
<medium id="m5">
 <name>m 5</name>
 <description>Verbindung von intelligente sensor11 mit L-Switch17</description>
 <productid>mediumProduct2</productid>
 <plugs>
  <productID>plugProduct2</productID>
  <productID>plugProduct2</productID>
 </plugs>
 <connectedTo>
  <device transceiverid="1" inDocument="this">sensor 11</device>
  <device transceiverid="1" inDocument="this">L-Switch 17</device>
 </connectedTo>
 <usedTechnology>technology1</usedTechnology>
</medium>
  … … … …
<medium id="m7">
 <name>m 7</name>
 <description>Verbindung von intelligente aktor12 mit L-Switch17</description>
 <productid>mediumProduct2</productid>
 <plugs>
  <productID>plugProduct2</productID>
  <productID>plugProduct2</productID>
 </plugs>
 <connectedTo>
  <device transceiverid="1" inDocument="this">actor 12</device>
  <device transceiverid="2" inDocument="this">L-Switch 17</device>
 </connectedTo>
 <usedTechnology>technology1</usedTechnology>
</medium>
  … … … …
<medium id="m9">
 <name>m 9</name>
 <description>Verbindung von sensorcontroller15 mit L-Switch17</description>
 <productid>mediumProduct2</productid>
 <plugs>
  <productID>plugProduct2</productID>
  <productID>plugProduct2</productID>
 </plugs>
 <connectedTo>
  <device transceiverid="1" inDocument="this">controller 15</device>
  <device transceiverid="3" inDocument="this">L-Switch 17</device>
 </connectedTo>
 <usedTechnology>technology1</usedTechnology>
</medium>
  … … … …
<medium id="m11">
 <name>m 11</name>
 <description>Verbindung von aktorcontroller16 mit L-Switch17</description>
 <productid>mediumProduct2</productid>
 <plugs>
  <productID>plugProduct2</productID>
  <productID>plugProduct2</productID>
 </plugs>
 <connectedTo>
  <device transceiverid="1" inDocument="this">controller 16</device>
  <device transceiverid="4" inDocument="this">L-Switch 17</device>
 </connectedTo>
 <usedTechnology>technology1</usedTechnology>
```

```
 </medium>
    … … … …
 <medium id="m13">
  <name>m 13</name>
  <description>Verbindung von L-Switch17 mit iLON1000 08</description>
  <productid>mediumProduct3</productid>
  <plugs>
   <productID>plugProduct3</productID>
   <productID>plugProduct3</productID>
  </plugs>
  <connectedTo>
   <device transceiverid="5" inDocument="this">L-Switch 17</device>
   <device transceiverid="1" inDocument="this">iLON1000</device>
  </connectedTo>
  <usedTechnology>technology2</usedTechnology>
 </medium>
    … … … …
 <medium id="m15">
  <name>m 15</name>
  <description>Verbindung von iLON1000 08 mit Server</description>
  <productid>mediumProduct1</productid>
  <plugs>
   <productID>plugProduct1</productID>
   <productID>plugProduct1</productID>
  </plugs>
  <connectedTo>
   <device transceiverid="3" inDocument="this">iLON1000</device>
   <device interfaceid="1" inDocument="this">PC</device>
  </connectedTo>
  <usedTechnology>technology3</usedTechnology>
 </medium>
</media>
```

Unter Element <networks> werden alle <medium> in 4 Kinderelemente <network> verteilt. Die ersten 2 Teile sind die zwei Teil-LON-Netze in Zimmer1 und Zimmer2. Der dritte Teil ist <medium> LON-Busbackbone. Der vierte Teil ist Ethernet. <network> mit id=“n5“ beschreibt das ganze Netzwerksystem.

```
<networks>
 <network id="n1">
  <name>teil lon-bus von zimmer1</name>
  <contains>
   <medium inDocument="this">m 1</medium>
   <medium inDocument="this">m 3</medium>
   <medium inDocument="this">m 5</medium>
   <medium inDocument="this">m 7</medium>
   <medium inDocument="this">m 9</medium>
   <medium inDocument="this">m 11</medium>
  </contains>
 </network>
    … … … …
 <network id="n3">
  <name>lon-bus mit backbone</name>
  <contains>
   <medium inDocument="this">m 13</medium>
   <medium inDocument="this">m 14</medium>
  </contains>
 </network>
 <network id="n4">
  <name>Ethernet</name>
```

```
  <contains>
   <medium inDocument="this">m 15</medium>
  </contains>
 </network>
 <network id="n5">
  <name>Gesamtenet</name>
  <contains>
    <teilnetwork inDocument="this">teil lon-bus von zimmer1</teilnetwork>
    <teilnetwork inDocument="this">teil lon-bus von zimmer2</teilnetwork>
    <teilnetwork inDocument="this">lon-bus mit backbone</teilnetwork>
    <teilnetwork inDocument="this">Ethernet</teilnetwork>
  </contains>
 </network>
</networks>
```

7.4.4 Viewpoint Simulation

Im Netzwerksystem gibt es neun Simulationsendknoten. Sensor11, Sensor21, Aktor12 und Aktor22 sind die Endknoten. Server ist auch ein Endknoten. Besonderheiten sind die Kontroller15, Kontroller16, Kontroller25 und Kontroller26. Sie sind LON-Geräte und sind mit unintelligenten Sensoren und Aktoren verbunden. Die unintelligente Sensoren und Aktoren werden hier nicht als Simulationsendkonten gestellt. Statt dessen werden Kontroller als die Simulationsendknoten für die unintelligente Aktoren und Sensoren dargestellt.
Diese Simulationsdokumente werden durch Element <homeDevice> von dem entsprechenden Dokument jedes Geräts (Sensor, Aktor, Controller, Server) geknüpft. Es ist ungleich als die Diplomarbeit von Robert Hänel. In seiner Arbeit sind die Simulationsdokumente von Topologiebeschreibungsdokument geknüpft.
Bei Viewpoint Simulation werden hier die XML Simulationsdokumente von Sensor11, Aktor12, Kontroller15, Kontroller16 im Zimmer1 und Server09 erklärt.

Ss11.xml:

Ss11.xml ist ein Dokument, um eine Simulation auf Sensor11 zu beschreiben. Nach dem Großen Beleg „Performancesimulation mit NS-2“, der von Pfeifer. G geschrieben wurde, wird Element <simulationTime> mit Wert 20 ms definiert und <traceCount> ist 4.
Unter Element <agents> ist Kinderelement <agent> mit „agentid="ags11Null"“ ein Empfangendknoten von Pakete auf Sensor11.
Kindelement <agent agentid="ags11UDP1"> ist ein Sendendknoten von Paketen auf Sensor11. Er sendet Pakete nach dem Empfangendknoten auf Aktor12. Der Pakettyp ist UDP. Unter Kinderelement <application> ist die Traffictyp ein inneres Traffic. Die Paketgröße ist 27 Byte und die max. Größe ist 31 Byte. Die Paketsendzeit ist 500 ms je Mal. Dazwischen ist ein Pause von 100 ms. Die Paketübertragungsrate ist 13,5 KByte/s.
Unter Kindelement <agent agentid="ags11UDP2"> ist der Zielknoten von Paketsendung den Empfangendknoten auf Server09. Der Pakettyp ist UDP. Unter Kinderelement <application> ist der Traffictyp ein externes Traffic. Die Paketgröße ist 7 Byte und die max. Größe ist 31 Byte. Die Paketsendezeit ist 500 ms . Dazwischen ist eine Pause von 500 ms. Die Paketübertragungsrate ist 3,5 KByte/s.

```
<simulation schemaVersion="3.0" simulationid="ss11" projectid="projekt"…>
 <description>Signaltausch von Sensor11</description>
 <homeDevice>sensor11</homeDevice>
 <simulationTime unit="ms">20</simulationTime>
 <traceCount>4</traceCount>
 <tracefileName>flaschenhals-trace</tracefileName>
 <agents>
```

```
 <agente agentid="ags11Null">
  <type>Null</type>
 </agente>
 <agenta agentid="ags11UDP1">
  <type>UDP</type>
  <target>aga12Null</target>
  <application applicationid="aps111">
   <description>Paketsendung nach Aktor12 für Datenübertragung</description>
   <type>traffic/pareto</type>
   <packetSize unit="Byte">27</packetSize>
          <burstTime unit="ms">500</burstTime>
          <idleTime unit="ms">100</idleTime>
          <rate unit="KByte">13.5</rate>
          <shape>1.5</shape>
          <maxPackets>31</maxPackets>
  </application>
 </agenta>
 <agenta agentid="ags11UDP2">
  <type>UDP</type>
  <target>agserver09Null</target>
  <application applicationid="aps112">
   <description>Paketsendung nach Server für Statusüberwachung</description>
   <type>traffic/exponential</type>
   <packetSize unit="Byte">7</packetSize>
          <burstTime unit="ms">500</burstTime>
          <idleTime unit="ms">500</idleTime>
          <rate unit="KByte">3.5</rate>
          <shape>1.5</shape>
          <maxPackets>31</maxPackets>
  </application>
 </agenta>
</agents>
</simulation>
```

Sa12.xml:

Sa12.xml ist ein Dokument, um eine Simulation auf Aktor12 zu beschreiben. Wie ss11.xml geschrieben, wird Element <simulationTime> mit Wert 20 ms definiert und <traceCount> ist 4.
Unter Element <agents> ist Kindelement <agent> mit „agentid="aga12Null"“ ein Empfangendknoten von Pakete auf Aktor12.
Gegenüber ss11.xml gibt es im Dokument sa12.xml nur einen Agent <agent agentid="aga12UDP1">. Unter Element <agent agentid="aga12UDP1"> ist der Zielknoten von Paketsendung den Empfangendknoten auf Server09. Der Pakettyp ist UDP. Unter Kinderelement <application> ist der Traffictyp ein externes Traffic. Die Paketgröße ist 7 Byte und die max. Größe ist 31 Byte. Die Paketsendzeit ist 500 ms. Dazwischen ist eine Pause von 500 ms. Die Paketübertragungsrate ist 3,5 KByte/s.

```
<simulation schemaVersion="3.0" simulationid="sa12" projectid="projekt"…>
 <description>Signaltausch von Aktor12</description>
 <homeDevice>aktor12</homeDevice>
 <simulationTime unit="ms">20</simulationTime>
 <traceCount>4</traceCount>
 <tracefileName>flaschenhals-trace</tracefileName>
 <agents>
  <agente agentid="aga12Null">
   <type>Null</type>
  </agente>
  <agenta agentid="aga12UDP1">
   <type>UDP</type>
```

```
  <target>agserver09Null</target>
  <application applicationid="apa121">
   <description>Paketsendung nach Server für Statusüberwachung</description>
   <type>traffic/exponential</type>
   <packetSize unit="Byte">7</packetSize>
          <burstTime unit="ms">500</burstTime>
          <idleTime unit="ms">500</idleTime>
          <rate unit="KByte">3.5</rate>
          <shape>1.5</shape>
          <maxPackets>31</maxPackets>
  </application>
 </agenta>
</agents>
</simulation>
```

Sc15.xml:

Sc15.xml ist ein Dokument, um eine Simulation auf Controller15 zu beschreiben. Wie ss11.xml geschrieben, wird Element <simulationTime> mit Wert 20 ms definiert und <traceCount> ist 4.
Unter Element <agents> ist Kindelement <agent> mit „agentid="agc15Null"“ ein Empfangendknoten von Pakete auf Controller15.
Kindelement <agent agentid="agc15UDP1"> ist ein Sendendknoten von Paketen auf Controller15. Er sendet Pakete nach dem Empfangendknoten auf Controller16. Der Pakettyp ist UDP. Unter Kindelement <application> ist der Traffictyp ein inneres Traffic. Die Paketgröße ist 27 Byte und die max. Größe ist 31 Byte. Die Paketsendezeit ist 500 ms je Mal. Dazwischen ist eine Pause von 100 ms. Die Paketübertragungsrate ist 13,5 KByte/s.
Unter Kinderelement <agent agentid="agc15UDP2"> ist der Zielknoten von Paketsendung der Empfangendknoten auf Server09. Der Pakettyp ist UDP. Unter Kindelement <application> ist die Traffictyp ein externes Traffic. Die Paketgröße ist 7 Byte und die max. Größe ist 31 Byte. Die Paketsendzeit ist 500 ms. Dazwischen ist eine Pause von 500 ms. Die Paketübertragungsrate ist 3,5 KByte/s.

```
<simulation schemaVersion="3.0" simulationid="sc15" projectid="projekt"…>
 <description>Signaltausch von Controller15</description>
 <homeDevice>controller15</homeDevice>
 <simulationTime unit="ms">20</simulationTime>
 <traceCount>4</traceCount>
 <tracefileName>flaschenhals-trace</tracefileName>
 <agents>
  <agente agentid="agc15Null">
   <type>Null</type>
  </agente>
  <agenta agentid="agc15UDP1">
   <type>UDP</type>
   <target>agc16Null</target>
   <application applicationid="apc151">
    <description>Paketsendung nach Controller16 für Datenübertragung</description>
    <type>traffic/pareto</type>
    <packetSize unit="Byte">27</packetSize>
          <burstTime unit="ms">500</burstTime>
          <idleTime unit="ms">100</idleTime>
          <rate unit="KByte">13.5</rate>
          <shape>1.5</shape>
          <maxPackets>31</maxPackets>
   </application>
  </agenta>
  <agenta agentid="agc15UDP2">
   <type>UDP</type>
```

```
  <target>agserver09Null</target>
  <application applicationid="apc152">
   <description>Paketsendung nach Server für Statusüberwachung</description>
   <type>traffic/exponential</type>
   <packetSize unit="Byte">7</packetSize>
          <burstTime unit="ms">500</burstTime>
          <idleTime unit="ms">500</idleTime>
          <rate unit="KByte">3.5</rate>
          <shape>1.5</shape>
          <maxPackets>31</maxPackets>
  </application>
 </agenta>
</agents>
</simulation>
```

Sc16.xml:

Sc16.xml ist ein Dokument, um eine Simulation auf Controller16 zu beschreiben. Wie ss11.xml geschrieben, wird Element <simulationTime> mit Wert 20 ms definiert und <traceCount> ist 4.
Unter Element <agents> ist Kinderelement <agent> mit „agentid="agc16Null"“ ein Empfangendknoten von Paketen auf Controller16.
Gegenüber sc15.xml gibt es im Dokument sc16.xml nur einen Agent <agent agentid="agc16UDP1">. Unter Element <agent agentid="agc16UDP1"> ist der Zielknoten von Paketsendung der Empfangendknoten auf Server09. Der Pakettyp ist UDP. Unter Kindelement <application> ist die Traffictyp ein externes Traffic. Die Paketgröße ist 7 Byte und die max. Größe ist 31 Byte. Die Paketsendzeit ist 500 ms je Mal. Dazwischen ist ein Pause von 500 ms. Die Paketübertragungsrate ist 3,5 KByte/s.

```
<simulation schemaVersion="3.0" simulationid="sc16" projectid="projekt"…>
 <description>Signaltausch von Controller16</description>
 <homeDevice>controller16</homeDevice>
 <simulationTime unit="ms">20</simulationTime>
 <traceCount>4</traceCount>
 <tracefileName>flaschenhals-trace</tracefileName>
 <agents>
  <agente agentid="agc16Null">
   <type>Null</type>
  </agente>
  <agenta agentid="agc16UDP1">
   <type>UDP</type>
   <target>agserver09Null</target>
   <application applicationid="apc161">
    <description>Paketsendung nach Server füt Statusüberwachung</description>
    <type>traffic/exponential</type>
    <packetSize unit="Byte">7</packetSize>
          <burstTime unit="ms">500</burstTime>
          <idleTime unit="ms">500</idleTime>
          <rate unit="KByte">3.5</rate>
          <shape>1.5</shape>
          <maxPackets>31</maxPackets>
   </application>
  </agenta>
 </agents>
</simulation>
```

Sserver09.xml

Sserver09.xml ist ein Dokument, um eine Simulation auf Server zu beschreiben. Element <simulationTime> ist mit Wert 20 ms definiert und <traceCount> ist 4. <tracefileName> ist „flaschenhals-trace“.
Unter Element <agents> ist Kinderelement <agent> mit „agentid=" agserver09Null“ ein Empfangendknoten von Pakete auf Server09.
Kindelement <agent agentid="agserver06UDP1"> ist ein Sendendknoten von Paketen auf Server09. Er sendet Pakete nach dem Empfangendknoten auf Sensor11. Der Pakettyp ist UDP. Unter Kinderelement <application> ist der Traffictyp ein inneres Traffic. Die Paketgröße ist 7 Byte und die max. Größe ist 31 Byte. Die Paketsendzeit ist 500 ms. Dazwischen ist ein Pause von 500 ms. Die Paketübertragungsrate ist 3,5 KByte/s.
Unter Kinderelement <agent agentid="agserver06UDP2"> ist der Zielknoten von Paketsendung der Empfangendknoten auf Aktor12. Der Pakettyp ist UDP. Unter Kinderelement <application> ist der Traffictyp ein externes Traffic. Die Paketgröße ist 7 Byte und die max. Größe ist 31 Byte. Die Paketsendzeit ist 500 ms. Dazwischen ist eine Pause von 500 ms. Die Paketübertragungsrate ist 3,5 KByte/s.
Kinderelement <agent agentid="agserver06UDP3"> ist ein Sendendknoten von Pakete auf Server09. Er sendet Pakete nach dem Empfangendknoten auf Controller15. Die andere Informationen sind wie unter Element <agent agentid="agserver06UDP1">.
Der Zielsimulationsendknoten von Kinderelement <agent agentid="agserver06UDP4"> ist Controller16.
Und die Zielknoten von <agent> mit den id „agserver06UDP5“, „agserver06UDP6“, „agserver06UDP7“ und „agserver06UDP8“ sind Sensor21, Aktor22, Controller25 und Controller26.

```
<simulation schemaVersion="3.0" simulationid="sserver09" projectid="projekt"…>
 <description>Signaltausch von Server09</description>
 <homeDevice>server09</homeDevice>
 <simulationTime unit="ms">20</simulationTime>
 <traceCount>4</traceCount>
 <tracefileName>flaschenhals-trace</tracefileName>
 <agents>
  <agente agentid="agserver09Null">
   <type>Null</type>
  </agente>
  <agenta agentid="agserver06UDP1">
   <type>UDP</type>
   <target>ags11Null</target>
   <application applicationid="apserver091">
    <description>Paketsendung nach Sensor11</description>
    <type>traffic/exponential</type>
    <packetSize unit="Byte">7</packetSize>
          <burstTime unit="ms">500</burstTime>
          <idleTime unit="ms">500</idleTime>
          <rate unit="KByte">3.5</rate>
          <shape>1.5</shape>
          <maxPackets>31</maxPackets>
   </application>
  </agenta>
  <agenta agentid="agserver06UDP2">
   <type>UDP</type>
   <target>aga12Null</target>
   <application applicationid="apserver092">
    <description>Paketsendung nach Aktor12</description>
    <type>traffic/exponential</type>
```

```
  <packetSize unit="Byte">7</packetSize>
        <burstTime unit="ms">500</burstTime>
        <idleTime unit="ms">500</idleTime>
        <rate unit="KByte">3.5</rate>
        <shape>1.5</shape>
        <maxPackets>31</maxPackets>
  </application>
 </agenta>
 <agenta agentid="agserver06UDP3">
  <type>UDP</type>
  <target>agc15Null</target>
  <application applicationid="apserver093">
   <description>Paketsendung nach Controller15</description>
   <type>traffic/exponential</type>
   <packetSize unit="Byte">7</packetSize>
        <burstTime unit="ms">500</burstTime>
        <idleTime unit="ms">500</idleTime>
        <rate unit="KByte">3.5</rate>
        <shape>1.5</shape>
        <maxPackets>31</maxPackets>
  </application>
 </agenta>
 <agenta agentid="agserver06UDP4">
  <type>UDP</type>
  <target>agc16Null</target>
  <application applicationid="apserver094">
   <description>Paketsendung nach Controller16</description>
   <type>traffic/exponential</type>
   <packetSize unit="Byte">7</packetSize>
        <burstTime unit="ms">500</burstTime>
        <idleTime unit="ms">500</idleTime>
        <rate unit="KByte">3.5</rate>
        <shape>1.5</shape>
        <maxPackets>31</maxPackets>
  </application>
 </agenta>
   … … … …
 </agents>
</simulation>
```

Die vorliegenden XML-Dokumente sind nur grobe Beispiele von einem LON-Bussystemszenario im Bild 7.4.1. „ein Szenario von Gebäudeautomationssystem mit zwei Zimmern“. Die detaillierten XML-Dokumente und seine XML-Schemen, die das Szenario beschreiben und durch „Exchanger XML Lite 3.2“ und „Altova XMLSpay“entwickelt werden, sind in einer CD gespeichert.

8. Zusammenfassung

In diesem Großen Beleg wird der LON-Bus als ein Schwerpunkt von der Gebäudeautomation erklärt. LON-Bustechnik ist eine sehr günstige Automationstechnik nicht nur für Gebäude mit einigen Zimmern (Hausautomation), sondern auch für große Gebäude z.B. Hotel, Krankenhaus, usw. LON-Bus darf man mit einfachen Netzverbindungsgeräten für entfernte Automatisierungsverwaltung an LAN ankoppeln. Damit kann man nicht nur neue Gebäude mit dem LON-Automationsnetzwerk designen, sondern man kann auch alte Gebäude mit LON-Automatisierung erweitern.
Wegen des CMSA-Zugriffverfahrens im LON-Bussystem gibt es einen Nachteil bei Echtzeitanforderungen, besonders in den entfernten Verwaltungen über Ethernet in vrschiedenen Städten. Dafür kann man z.B. mehrere LON-Busse entsprechend unterschiedlichen Anforderungen getrennt in ein Gebäude stellen. Das heißt, z.B. kann man den LON-Bus für die Leuchtenüberwachung und den LON-Bus für den Feueralarm und Feuerlöschung getrennt im Gebäude realisieren. Die Leuchtenüberwachung wird über Internet entfernt realisiert, und der Feueralarm und -löschung werden durch Sensoren, Aktoren und Kontrolleren lokal realisiert.
Candy ist ein Netzwerk-Design-Werkzeug. Auf Basis der Ergebnisse des Beleges kann ein LON-Bus-System auch in CANDY entworfen und simuliert werden.
Zur Zeit wird die Struktur von CANDY als Client-Server Architektur entwickelt. Für LON-Netzwerkdesign kann man auch die Tools z.B. NDML Editor, Rule checker, Queuing tool, Bill reporter, Projektmanager, usw. in CANDYBOX nutzen.
Aber das Automatisierungsnetzwerk hat viele eigene Eigenschaften ungleich zu LAN, z.B. darf die Zahl von den Knoten des LON-Netzsystems nicht 32385 überschreiten. Für unterschiedliche Automatisierungsfeldbusse sind die Eigenschaften vielfältig. Die Automatisierungsverwaltungen können auch „wireless“ realisiert werden. Deshalb muss man nicht nur Kosten, sondern auch Echtzeitanforderung und die max. Anzahl der Knoten überdenken und eine optimale Topology des Netzes wählen.
Die Gebäudeautomationstechnik kann auch für die Weißwaresteuerung z.B. Waschmaschine-, Kühlschranksteuerung… verwendet werden.
In Zukunft werden die Anwendungen von CANDY um das Gebäudeautomationsnetzdesign erweitert; dabei kann man die meisten Komponenten von CANDY weiterverwenden. Für das Gebäudeautomationsdesign muß man einige Tools und Komponente des CANDYs verändern und weiterentwickeln. Die Vorschläge der Entwicklung des CANDYs sind in Bild 8.1 angezeigt.

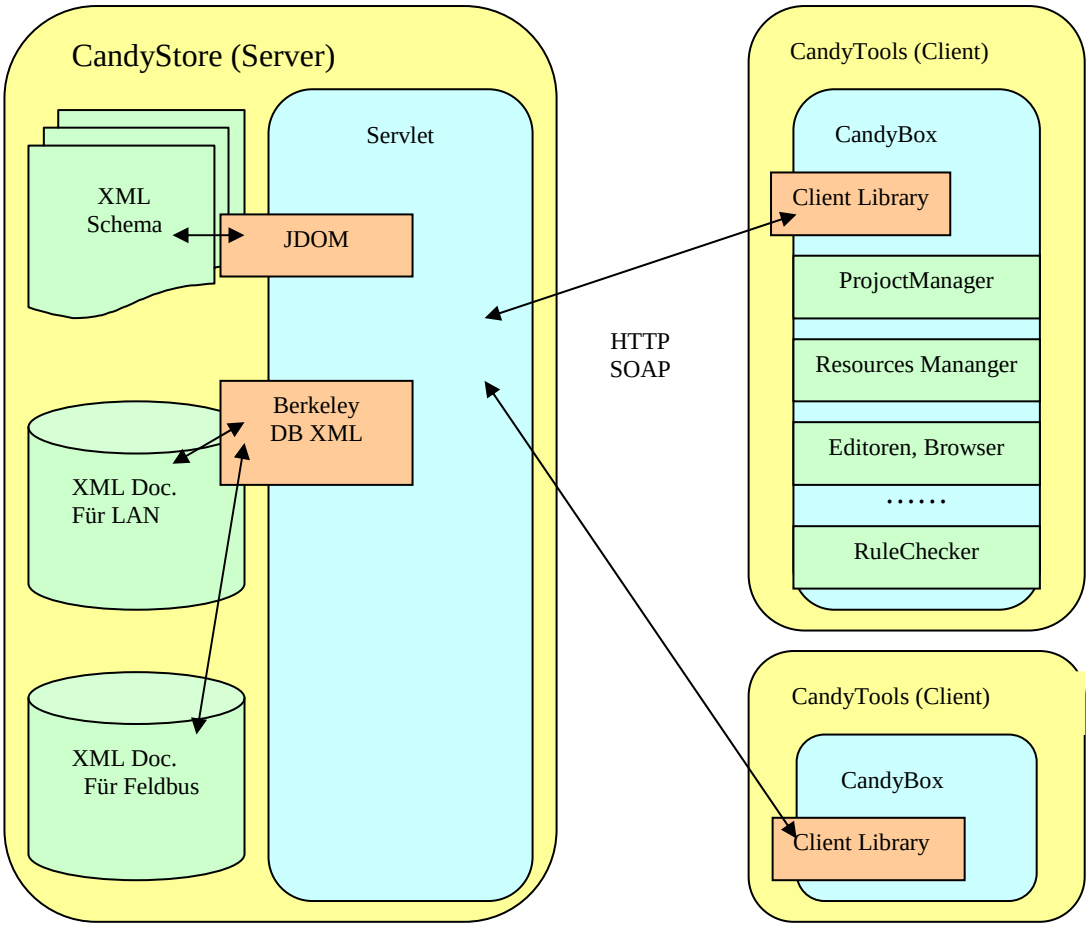


Bild 8.1 Architektur von CANDY

In dieser Client-Server-Struktur braucht man zwei Datenbanken, eine ist für die den LAN-Bus beschreibenden XML-Dokumente, die andere ist für die feldbusbeschreibenden XML-Dokumente. Oder kann man eine große Datenbank im CANDYSTORE einsetzen, in der die zwei Sorte von Beschreibungsdokumenten getrennt abgelegt werden. Diese Daten werden durch z.B. Berkeley DB XML zusammenverwaltet, weil ein gesamtes Gebäudeautomationsnetzwerk immer aus Feldbus und Rechnernetz zusammensetzt wird.
Der Schwerpunkt liegt in den Datenbankverwaltungen im CANDYSTORE. Mit NDML3.0 werden die Komponenten des Rechnernetzsystems und seine Eigenschaften in den kleinen Einheiten beschrieben. D.h. in jeder Einheit wird nur eine Komponente (z.B. Aktor, Sensor) oder eine Eigenschaft (z.B. Technology, Simulation) mit XML beschrieben. Es gibt viele kleinen XML-Dokumente und seine XML-schemen. Diese Dokumente werden entsprechend vielartiger Eigenschaften und Funktionen in den unterschiedlichen Verzeichnissen gespeichert. Dafür sind die Navigationen und Verknüpfungen schwerer und komplexer als in NDML1.0 und NDML2.0. Das muss man in Zukunft verbessern.
Bei den Clients werden einige CANDY-Tools auch verändert. Z.B.der Cost-Rechner für den Feldbus errechnet nicht nur die Gesamtkosten des Netzwerks, sondern auch die Reaktionszeit von Sensor bis Aktor entsprechend verschiedenen Topologien. Und die Funktionen von Editor und Browser werden auch für Feldbusse und seine Geräte erweitert und verstärkt.
In diesem Großen Beleg werden die XML-Dokumente von Viewpoint Basic, Load, Topology und Simulation mit XML-Schemen in einer CD gespeichert. Wegen der Weiterentwicklungen des CANDY-Tools und seiner verschiedenen Komponenten müssen die XML-Dokumente und ihre Schemen auch verbessert werden (für diese Komponenten liegen viele XML-Beschreibungsdokumente z.Zt. nicht mit XML-Schema, sondern mit DTD vor).

Literaturverzeichnis

[1] Furrer, Frank J.: Industrieautomation mit Ethernet-TCP/IP und Web-Technologie – 3. Auflage – Heidelberg, 2003

[2] Webseite der Gebäudeautomatisierung in „wikipedia“. http://de.wikipedia.org/wiki/Gebäudeautomation

[3] Phoenix Contact (Hrsg.): Grundkurs Sensor-Aktor-Feldbustechnik - 1. Auflage – Würzburg, Vogel, 1997

[4] Jürgen Beuschel: LonWorks-Technik in der Gebäudeautomation - 1. Auflage - Huss-Medien, Verl. Technik, 2003

[5] Thomas Tyczynski: SPS - Einsatz in der Gebäudetechnik -- von der Programmierung zur Komplettlösung – Berlin, 1999

[6] Webseite des GLTs (Gebäudeleittechnik) in „wikipedia“ http://de.wikipedia.org/wiki/Gebäudeleittechnik

[7] Dieter Barelmann: OPC in der Praxis - Berlin ; Offenbach, 1999

[8] Webseite „OSGi (Open Services Gateway Initiative)“ in „wikipedia“ http://de.wikipedia.org/wiki/OSGi

[9] Webseite „Feldbus“ in „wikipedia“ http://de.wikipedia.org/wiki/Feldbus

[10] Gerhard Schnell (Hrsg.). [Autoren Michael Lupik ...]: Bussysteme in der Automatisierungs- und Prozesstechnik -- Grundlagen und Systeme der industriellen Kommunikation – 5 Auflage - Braunschweig ; Wiesbaden Vieweg, 2003

[11] Birgit Scherff ; Erwin Haese ; Hagen R. Wenzek: Feldbussysteme in der Praxis--ein Leitfaden für den Anwender; Berlin ; Heidelberg [u.a.], Springer; 1999

[12] Webseite „EIB (Europäischer Installationsbus)“ in „wikipedia“ http://de.wikipedia.org/wiki/Europäischer_Installationsbus

[13] Webseite „CAN (Controller Area Network)“ in „wikipedia“ http://de.wikipedia.org/wiki/Controller_Area_Network

[14] Webseite „LON (Local Operating Network)“ in „wikipedia“ http://de.wikipedia.org/wiki/LON

[15] Horst Möbus ; Nils Gresbrand: Gebäudesystemtechnik mit LCN - 1. Auflage – Berlin, Verlag Technik; 2002

[16] Webseite „LCN (Local Control Network)“ in „wikipedia“: http://de.wikipedia.org/wiki/LCN

[17] Webseite „Profibus“ in „wikipedia“ http://de.wikipedia.org/wiki/Profibus

[18] Webseite „BACnet“ in „wikipedia“ http://de.wikipedia.org/wiki/BACnet

[19] Dietmar Dietrich ... (Hrsg.): LON-Technologie-- verteilte Systeme in der Anwendung - 2. Auflage – Heidelberg, Hüthig; 1999

[20] CANDY Projekt: CANDY: INTEGRATED ENVIRONMENT FOR NETWORK DESIGN http://www.rn.inf.tu-dresden.de/scripts_lsrn/Lehre/candy/indexgerm.htm

[21] CANDY Projekt: APPROACHES TO THE INTEGRATED NETWORK DESIGN FOR MODERN HEALTH CARE INSTITUTIONS http://www.rn.inf.tu-dresden.de/scripts_lsrn/Lehre/candy/indexgerm.htm

[22] CANDY Projekt: Zhou, Feiyue: Diplomarbeiten Entwicklung und Optimierung eines graphischen Entwurfsystems für das Rechnernetzprojektierungstool CANDY http://www.rn.inf.tu-dresden.de/scripts_lsrn/Lehre/candy/indexgerm.htm

[23] CANDY Projekt: Hänel, Robert: Diplomarbeiten

Optimierung der Fachsprache NDML mit Abbildung in eine XML-basierte Datenbank
http://www.rn.inf.tu-dresden.de/scripts_lsrn/Lehre/candy/indexgerm.htm
[24] CANDY Projekt: Pfeifer, G.: Großbelege
Performancesimulation mit NS-2
http://www.rn.inf.tu-dresden.de/scripts_lsrn/Lehre/candy/indexgerm.htm
[25] CANDY Projekt:
APPROACHES TO THE INTEGRATED NETWORK DESIGN FOR MODERN HEALTH CARE INSTITUTIONS
http://www.rn.inf.tu-dresden.de/scripts_lsrn/Lehre/candy/indexgerm.htm

Danksagung und Erklärung

Hier möchte ich besondern bei meinen Betreuern Dr. Gütter und Dr. Luntovskyy herzlich bedanken. Sie haben mit ihren Anregungen und Ideen meinen Entwurf zu verbessern und die Probleme zu lösen.
Außerdem möchte ich dem CANDY Team und insbesondere Hr. Zhou, Feiyue Hänel, Robert und Gert Pfeifer danken. Auf der Basis der früheren Arbeiten konnte ich meinen Großen Beleg realisieren.
Ich bedanke mich auch bei Hr. Volodymyr Vasyutynskyy für seine Hilfen im Fachgebiet von der Automatisierung.

Ich erkläre, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Literatur gemacht habe.

Wei, Xiaojun

Dresden, Oktober 2006

Anhang

Die XML-Dokumente und seine XML-Schemen beschreiben das LON-Bussystemszenario im Bild 7.4.1. „ein Szenario von Gebäudeautomationssystem mit zwei Zimmern“.
Ich habe diese XML-Dokumente und XML-Schemen durch „Exchanger XML Lite 3.2“ und „Altova XMLSpay“ entwickelt.
Weil es zu viele die XML-Dokumente und XML-Schemen gibt, speichere ich diese Dokumente in einer CD.