Frank Thiel

TCP/IP-Ethernet bis Web-IO

5. stark erweiterte Auflage
März 2006

Wiesemann und Theis GmbH:
Technische Grundlagen
Kompliziertes einfach

W&T
www.wut.de

W&T

www.WuT.de

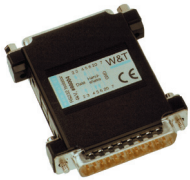
Wiesemann & Theis GmbH

Technische Grundlagen

konkret

kompakt

verständlich



www.WuT.de

TCP/IP-Ethernet bis Web-IO

Dieses Heft ist für alle gemacht, die ohne Spezialwissen über Computernetzwerke Ethernet-Endgeräte unter TCP/IP in Betrieb nehmen wollen. Es ist in fünf Teile gegliedert:

- **TCP/IP-Ethernet verstehen**
  Hier finden Sie die wichtigsten Grundlageninformationen zum Thema TCP/IP und den Basisprotokollen.

- **Individualisierte Web-Kommunikation**
  In diesem Abschnitt erfahren Sie, wie Sie die vorhandenen Netzstrukturen auch für Ihre technische Anwendung individualisiert nutzen können.

- **TCP/IP-Ethernet einrichten**
  Hier wird Schritt für Schritt die Einrichtung von TCP/IP-Ethernet auf PCs mit den gängigen Betriebssystemen aufgezeigt.

- **Kleines Netzwerk-ABC**
  Hier erläutern wir die wichtigsten Begriffe und Abkürzungen, die Ihnen beim Umgang mit Netzwerken begegnen können.

- **TCP/IP-Ethernet in der Praxis**
  Abschließend möchten wir die vielfältigen Möglichkeiten, die TCP/IP-Ethernet bietet, an einigen Beispielen aus der Praxis vorstellen.

Alle wichtigen Abläufe und Zusammenhänge werden leicht verständlich erklärt.

Keine Angst: Wir werden uns dort nicht bis ins letzte Detail verlaufen. Wir haben uns ganz bewusst auf die Dinge beschränkt, die zum Verständnis der beschriebenen Technologien wirklich wichtig sind.

Zur reinen Inbetriebnahme von TCP/IP-Netzwerkkomponenten ist es schließlich auch gar nicht nötig, jedes Protokoll bis ins letzte Bit zu kennen.

Autor:
Frank Thiel

Ursprüngliche Idee:
Rüdiger Theis und Frank Thiel

5. überarbeitete und stark erweiterte Auflage
80.001 - 100.000

Da wir Fehler machen können, darf keine unserer Aussagen ungeprüft verwendet werden. Bitte melden Sie uns alle Ihnen bekannt gewordenen Irrtümer oder Missverständlichkeiten, damit wir diese so schnell wie möglich erkennen und beseitigen können.

Inhalt

Einführung

Noch vor wenigen Jahren waren Computernetzwerke nur in Banken, Behörden und größeren Betrieben zu finden. Die verwendeten Netzwerkkomponenten waren meist kaum zu bezahlen, Installation und Administration ließen sich nur von speziell ausgebildeten Fachleuten bewältigen.

Doch spätestens seit den 90er Jahren nahmen Computer – vor allem der PC – rasanten Einzug in alle Bereiche des täglichen Lebens; ein immer höheres Datenaufkommen trug wesentlich zur Verbreitung und verstärkten Nutzung von Computernetzwerken bei.

Parallel zu dieser Entwicklung breitete sich das Internet explosionsartig aus und kann heute auch von privaten Anwendern problemlos in Anspruch genommen werden.

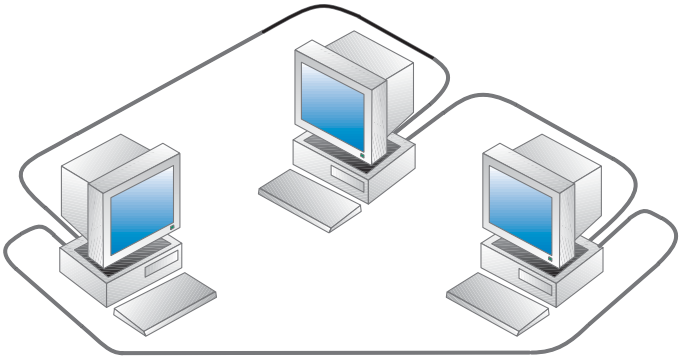
Dies alles hat dazu geführt, dass die Möglichkeit zum Zugriff auf Computernetzwerke heute fester Bestandteil moderner Betriebssysteme ist. Die wichtigste Rolle kommt dabei zwei Dingen zu: Ethernet als physikalische Grundlage und TCP/IP als Protokoll.

1. Anforderungen an ein Computernetzwerk

Jeder Computerbenutzer hat sicher schon einmal zwei Endgeräte (z.B. PC und Drucker, PC und Modem, PC und PC) miteinander verbunden. Zur Verbindung dient ein für den speziellen Anwendungsfall vorgesehenes Kabel, über das Daten zwischen beiden Geräten hin und her geschickt werden.

Man kann sich das so vorstellen: Zwei Brieffreunde senden einander Briefe, und ein Bote ist ständig damit beschäftigt, diese Briefe zu den Briefkästen der beiden zu befördern. In diesem einfachen Fall sind weder Briefumschlag noch Adresse oder Absender nötig.

Das Verfahren ist unkompliziert und funktioniert reibungslos. Es werden nur die reinen Nutzdaten verschickt. Diese Art der Verbindung nennt man auch Punkt-zu-Punkt-Verbindung. Man könnte die Punkt-zu-Punkt-Verbindung natürlich auch nutzen, um z.B. drei PCs miteinander kommunizieren zu lassen. Dazu müsste also von jedem PC je ein Kabel zu den beiden anderen PCs verlegt werden.



Für den Versand von Briefen zwischen drei Brieffreunden würden bei diesem Verfahren drei Boten gebraucht.

Schon bei vier beteiligten PCs brauchte man aber sechs Kabel, und wenn man zehn oder mehr PCs auf diese Weise „vernetzen“ wollte, wäre ein unentwirrbarer Kabelknoten die Folge. Außerdem würde jede Veränderung eines solchen Netzwerkes eine ganze Lawine von Änderungen in der Verkabelung nach sich ziehen. Die Umsetzung einer solchen Vernetzung ist also wenig praktikabel.

Ein Computernetzwerk sollte bei geringstem Material- und Verkabelungsaufwand vorhandene Ressourcen (Speicherplatz, Datenbanken, Drucker und andere beliebige Endgeräte) einer unbestimmten Zahl von angeschlossenen Nutzern zugänglich machen. Dabei muss ein Höchstmaß an Datensicherheit und Übertragungsgeschwindigkeit gegeben sein.

Aus diesen Anforderungen heraus entstanden die heute üblichen Netzwerkstandards.

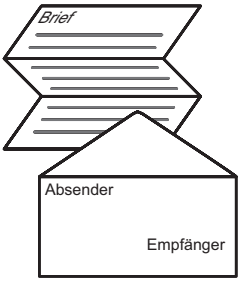
2. Grundsätzliche Funktion von Netzwerken

Grundsätzlich haben alle Netzwerktopologien eines gemeinsam:

Jeder Netzteilnehmer erhält eine eigene Adresse. Die Nutzdaten werden in einem Rahmen aus zusätzlichen Informationen (z. B. Adresse des Empfängers, Adresse des Absenders und Checksumme) „eingepackt".

Mit Hilfe der Adressinformationen können die Nutzdaten in den so entstandenen Datenpaketen über gemeinsam benutzte Leitungswege an den richtigen Empfänger übermittelt werden.

Bei einen Brief ist es nicht anders: Man steckt den Brief in einen Umschlag, auf dem Empfänger und Absender notiert sind. Der Postbote weiß dann, wem er den Brief zustellen soll; der Empfänger kann ablesen, woher er kommt und wem er bei Bedarf zu antworten hat.



Beim Datentransfer innerhalb eines Netzwerkes hat der Empfänger zusätzlich die Möglichkeit, mit Hilfe der mitversandten Checksumme die Vollständigkeit der empfangenen Nutzdaten zu überprüfen.

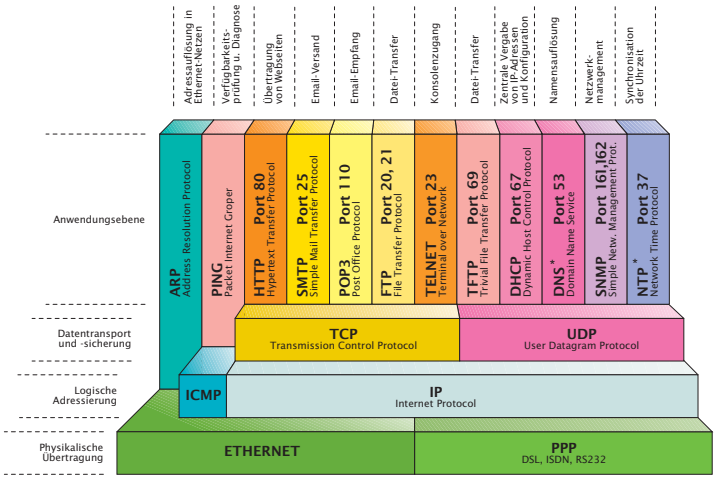
TCP/IP-Ethernet verstehen

Auf ihrem Weg von einem Netzteilnehmer zum anderen durchlaufen die Daten verschiedene Stufen. Jede dieser Stufen übernimmt dabei eine andere Funktion, auf die die nächsthöhere Stufe wiederum aufbaut.

Auf der untersten Stufe wird zunächst die Frage des Netzzugangs, d.h. der physikalischen Übertragung und der Form des Datenstroms entschieden. Für ein lokales Netz wird z.B. ein Ethernet-Standard eingesetzt.

Soll das Ethernet-Datenpaket in ein fremdes Netz versandt werden, wird es sodann von übergeordneten Protokollen, z.B. TCP/IP, adressiert und transportiert.

TCP/IP liefert das Datenpaket schließlich nicht nur beim richtigen Empfänger, sondern auch bei der richtigen Applikation ab, nämlich einem weiteren übergeordneten Protokoll, welches mit einem Anwendungsprogramm zusammenarbeitet. Sie erhalten z.B. eine E-Mail über das Protokoll POP3 und können diese mit Ihrem E-Mail-Programm abrufen.



* DNS und NTP werden in Sonderfällen auch über TCP abgewickelt

1. Physikalische Übertragung in Netzwerken

Je nach Anwendungsbereich stehen verschiedene physikalische Vernetzungstechnologien zur Verfügung. Bei lokalen Netzwerken ist Ethernet der heute am meisten verbreitete Netzwerkstandard; bereits 1996 waren ca. 86% aller bestehenden Netzwerke in dieser Technologie realisiert. Der Weg ins Internet wird dagegen mit Hilfe des öffentlichen Telefonnetzes und PPP realisiert.

1.1 Lokale Netze mit Ethernet und FastEthernet

Ethernet ist in der IEEE-Norm 802.3 standardisiert. Vereinfacht gesagt überträgt Ethernet mit Hilfe verschiedener Algorithmen Daten in Paketen über ein Medium an die Teilnehmer des Netzes, die sich jeweils durch eine eindeutige Adresse auszeichnen.

1.1.1 Ethernet-Standards

Im Laufe der Zeit haben sich verschiedene Ethernet-Varianten herausgebildet, die sich maßgeblich anhand von Übertragungsgeschwindigkeit und verwendeten Kabeltypen unterscheiden lassen. Ethernet wurde ursprünglich mit einer Übertragungsgeschwindigkeit von 10 Mbit/s betrieben; hierbei gab es drei verschiedene Grundmodelle:

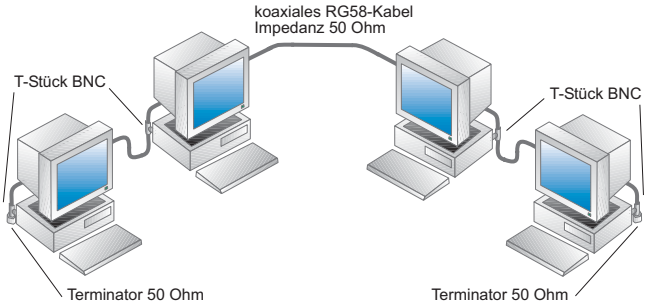
10Base5
Auch oft als „Yellow Cable“ bezeichnet; stellt den ursprünglichen Ethernetstandard dar und hat heute keine Bedeutung mehr. Verwendet wurde ein Koaxialkabel; die Reichweite betrug 500m.

10Base2
wird heute bei Neuinstallationen nicht mehr verwendet, ist aber in älteren Netzwerken noch zu finden.

10Base2 ist auch bekannt als Thin Ethernet, Cheapernet oder schlicht als BNC-Netzwerk.

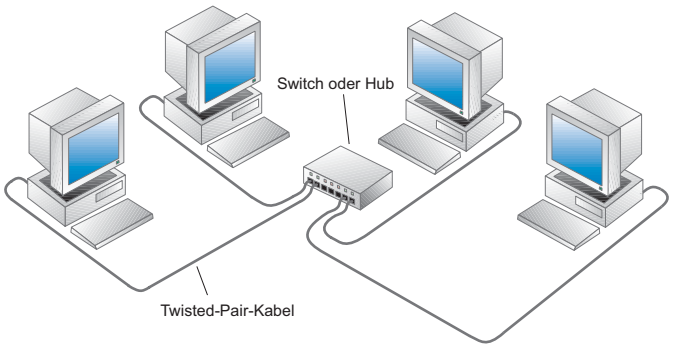
Alle Netzteilnehmer werden parallel auf ein Koaxialkabel (RG58, 50 Ohm Wellenwiderstand) aufgeschaltet. Das Kabel muss an beiden Seiten mit einem 50-Ohm-Terminator (Endwiderstand) abgeschlossen sein.

Teilen sich mehrere Geräte einen gemeinsamen Leitungsweg, spricht man auch von einer Bustopologie. Der Nachteil dieser Technik liegt in der hohen Störanfälligkeit. Wird die RG58-Verkabelung an einer beliebigen Stelle unterbrochen, ist der Netzwerkzugriff für alle angeschossenen Netzteilnehmer gestört.



10BaseT

Jeder Netzteilnehmer wird über ein eigenes Twisted-Pair-Kabel an einen sogenannten Hub (Sternverteiler) angeschlossen, der alle Datenpakete gleichermaßen an alle Netzteilnehmer weitergibt. 10BaseT arbeitet also physikalisch sternförmig, aber logisch nach dem Busprinzip (bei Datenbussen werden alle Teilnehmer wie bei 10Base2 an das selbe Kabel angeschossen) .



Mit zunehmend größeren Datenmengen wurde in den 90er Jahren Fast Ethernet mit einer Übertragungsgeschwindigkeit von 100Mbit/s eingeführt.

100BaseT
stellt den heute üblichen Standard für 100Mbit Netzwerke dar. Genau wie bei 10BaseT wird jeder Netzteilnehmer über ein eigenes Twisted-Pair-Kabel an einen Hub angeschlossen, der alle Datenpakete an alle Netzteilnehmer weitergibt. Allerdings müssen die Kabel und Komponenten wie Hubs für die höhere Übertragungsrate ausgelegt sein.

Ausführlichere Spezifikationen zu Ethernet und den verschiedenen physikalischen Topologien finden Sie auch auf den W&T Webseiten unter http://www.wut.de

HUB und Switch
Als sich 10BaseT und 100BaseT als physikalischer Standard für Ethernet-Netzwerke durchgesetzt haben, wurden zunächst nur HUBs als Sternverteiler eingesetzt. HUBs leiten, wie bereits beschrieben, den gesamten Datenverkehr des Netzwerkes an alle angeschlossenen Netzwerkteilnehmer weiter.

Inzwischen werden an Stelle von Hubs fast ausschließlich Switches eingesetzt. Switches leiten nicht mehr den gesamten Ethernet-Datenverkehr an alle angeschlossenen Netzwerkteilnehmer weiter. Stattdessen filtern Switches den Datenstrom so, dass am entsprechenden Port nur noch die Daten ausgegeben werden, die für den dort angeschlossenen Netzteilnehmer bestimmt sind.

Der Vorteil dieser Technik liegt darin, dass die Netzwerkteilnehmer nicht mehr mit Netzwerkverkehr überschwemmt werden, der ohnehin nicht für sie bestimmt ist.

1.1.2 Das Ethernet-Datenformat

Welches physikalische Grundmodell auch genutzt wird – der logische Aufbau der verwendeten Datenpakete ist bei allen Ethernet-Topologien gleich. Alle Netzteilnehmer in einem lokalen Netz erhalten alle Datenpakete einschließlich derer, die für die anderen Netzteilnehmer bestimmt sind, verarbeiten aber nur

diejenigen Pakete weiter, die tatsächlich an sie selbst adressiert sind.

Die Ethernet-Adresse

Die Ethernetadresse – auch MAC-ID oder Node-Number genannt – wird vom Hersteller in den physikalischen Ethernetadapter (Netzwerkkarte, Printserver, Com-Server, Router ...) fest „eingebrannt", steht also für jedes Endgerät fest und kann nicht geändert werden. Die Ethernet-Adresse ist ein 6-Byte-Wert, der üblicherweise in hexadezimaler Schreibweise angegeben wird. Eine Ethernetadresse sieht typischerweise so aus: 00-C0-3D-00-27-8B.

Jede Ethernet-Adresse ist weltweit einmalig!

Die ersten drei Hex-Werte bezeichnen dabei den Herstellercode, die letzten drei Hex-Werte werden vom Hersteller fortlaufend vergeben.

Das Ethernet-Datenpaket

Es gibt vier verschiedene Typen von Ethernet-Datenpaketen, die je nach Anwendung eingesetzt werden:

Datenpakettyp	*Anwendung*
Ethernet 802.2	Novell IPX/SPX
Ethernet 802.3	Novell IPX/SPX
Ethernet SNAP	APPLE TALK Phase II
Ethernet II	APPLE TALK Phase I, TCP/IP

In Verbindung mit TCP/IP werden in aller Regel Ethernet-Datenpakete vom Typ Ethernet II verwendet.

Hier der Aufbau eines Ethernet-II-Datenpakets:

Aufbau eines Ethernet-Datenpakets

⎍⎍⎍...	00C03D00278B	03A055236544	0800	Nutzdaten	Check-summe
Preamble	Destination	Source	Type	Data	FCS

Preamble Die Bitfolge mit stetigem Wechsel zwischen 0 und 1 dient zur Erkennung des Paketanfangs bzw. der Synchronisation. Das Ende der

	Preamble wird durch die Bitfolge „11“ gekennzeichnet.
Destination	Ethernet-Adresse des Empfängers
Source	Ethernet-Adresse des Absenders
Type	Gibt den übergeordneten Verwendungszweck an (z.B. IP = Internet Protocol = 0800h)
Data	Nutzdaten
FCS	Checksumme

Der Aufbau der anderen Ethernet-Pakete unterscheidet sich nur in den Feldern *Type* und *Data*, denen je nach Pakettyp eine andere Funktion zukommt. Damit verfügt ein Ethernet-Datenpaket über sämtliche erforderlichen Eigenschaften, um in lokalen Netzwerken Daten von einem Netzteilnehmer zum anderen zu verschicken.

Ethernet allein verfügt allerdings nicht über die Möglichkeit, verschiedene Netze zu adressieren. Darüber hinaus arbeitet Ethernet verbindungslos: Der Absender erhält vom Empfänger keine Bestätigung, ob ein Paket angekommen ist.

Spätestens wenn ein Ethernet-Netzwerk mit mehreren Netzen verbunden werden soll, muss also mit übergeordneten Protokollen – etwa mit TCP/IP – gearbeitet werden.

1.2 Analoge DFÜ, ISDN und DSL - Der Weg ins Internet

Eine entscheidende Einschränkung der heute üblichen Ethernet 100BaseT-Technik ist die maximale Distanz von 100m. Zwar können mit Hilfe entsprechender Komponenten wie Hubs, Switches und Routern auch größere Entfernungen erreicht werden, aber auch damit ist die Ausdehnung eines Ethernet-Netzwerkes auf das Grundstück einer Firma bzw. die Wohnung eines privaten Nutzers begrenzt.

Geht es z.B. darum, eine Verbindung zum Internet herzustellen (Datenfernübertragung, kurz DFÜ), sind oft mehrere Kilometer zu überbrücken. Internet Zugänge werden deshalb bis auf wenige Ausnahmen über das öffentliche Telefonnetz abgewickelt.

1.2.1 Physikalische Grundlagen

Drei DFÜ-Zugangsarten haben sich durchgesetzt:

- Analoges Modem
- ISDN
- DSL

Analoge Modem

Für diesen Zugang wird ein normaler, analoger Telefonanschluss (kein ISDN) benutzt.

Zwischen das Endgerät, meist ein PC, und den Telefonanschluss wird ein Modem (Modulator-Demodulator) geschaltet. Als Schnittstelle zwischen z.B. PC und Modem wird meist die serielle Schnittstelle (COM-Port / RS232) oder USB verwendet. Alternativ zu den externen Modems gibt es PC-Einsteckkarten, welche die Modem-Funktionen innerhalb des PC abwickeln.

Wird für die Übertragung das öffentliche Telefonnetz benutzt, ist es zunächst nötig, eine Einwahlverbindung zum Internet-Provider herzustellen. Auch diese Aufgabe übernimmt der Modem.

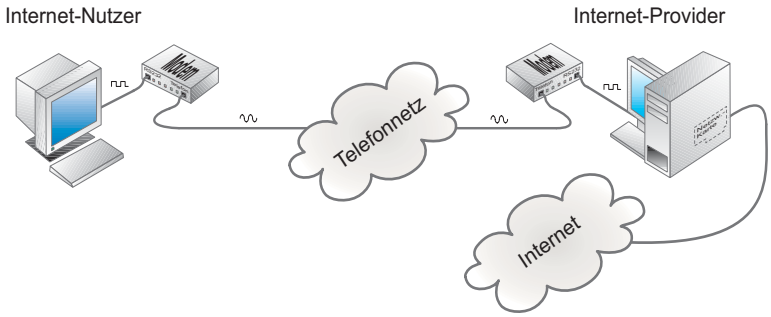
Wenn die Telefonverbindung zu Stande gekommen ist, werden die digitalen Informationen auf eine Trägerfrequenz aufmoduliert.

Wir wollen an dieser Stelle nicht näher auf die angewendeten Modulationsverfahren eingehen, sondern nur eine beispielhafte Erklärung dieser Technik geben.

Die Trägerfrequenz kann man sich vorstellen wie einen bestimmten hörbaren Ton aus dem Frequenzbereich der Sprache (40 Hz – 3.400 Hz).

Der zu übertragende Datenstrom wird in einige Bits große Blöcke zerteilt. Je nachdem welches Bitmuster vorliegt, wird der Ton in einer für dieses Bitmuster vorgegebenen Art verändert.

Am anderen Ende der Verbindungsstrecke übernimmt ein zweiter Modem die umgekehrte Aufgabe (Demodulation). Aus den empfangenen Tönen wird wieder ein Datenstrom zurückgewonnen.



Durch den eingeschränkten Frequenzbereich von analogen Telefonanschlüssen liegt die maximale Datenübertragungsrate bei 33kBit/s vom Teilnehmer in Richtung Vermittlungsstelle (Upstream). Von der Vermittlungsstelle in Richtung Teilnehmer (Downstream) sind maximal 56kBit/s möglich.

Ein großer Nachteil von DFÜ über den analogen Telefonanschluss ist neben der geringen Übertragungsrate die Tatsache, dass parallel zu DFÜ nicht telefoniert werden kann.

ISDN

Der wesentliche Unterschied zum analogen Telefonanschluss besteht darin, dass bei ISDN selbst analoge Sprachdaten

bereits am Standort des Teilnehmers in digitale vermittlungstechnische Daten umgewandelt werden.

Vom Teilnehmer zur Vermittlungsstelle werden also ausschließlich digitale Daten in Form von ISDN-Netzwerkpaketen ausgetauscht.

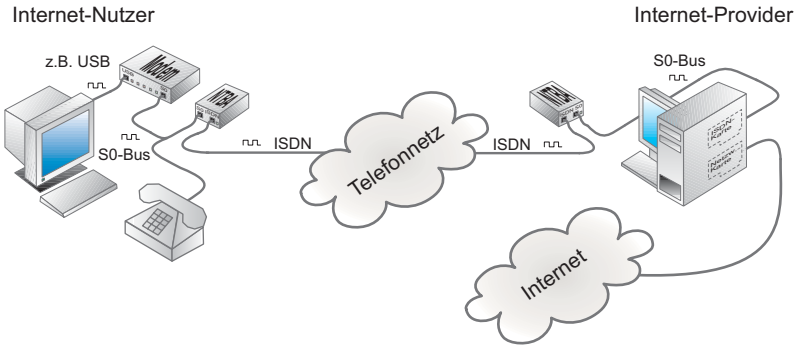
ISDN steht für Integrated Services Digital Network, was locker übersetzt so viel bedeutet wie: Integriertes digitales Netzwerk für verschiedene Dienste.

Neben der Übertragung von Sprache erlaubt ISDN von Hause aus den Austausch von digitalen Daten z.B. für Fax und DFÜ.

Eine Modulation von DFÜ-Daten ist bei ISDN im eigentlichen Sinne nicht nötig. Statt dessen werden die zu übertragenden Daten in ISDN-Pakete verpackt und versendet, wobei auch hier zunächst eine Wahlverbindung nötig ist.

Trotz dem spricht man bei externen ISDN <-> DFÜ-Daten Umsetzern gemeinhin von ISDN-Modems.

Zwischen ISDN-Modem und dem Telefonnetz bereitet der NTBA die ISDN-Daten physikalisch so auf, dass sie zur Vermittlungsstelle übertragen werden können. Die Schnittstelle zwischen den ISDN-Endgeräten und dem NTBA wird als S0-Bus bezeichnet.



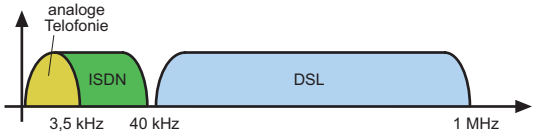
ISDN stellt dem Teilnehmer zwei Kanäle (Bereiche im ISDN-Paket) zur Verfügung, die auch für unterschiedliche Dienste, z.B. Telefonieren und DFÜ, genutzt werden können.

Pro Kanal können 64kBit/s übertragen werden. Bei paralleler Nutzung beider Kanäle (Kanalbündelung) erhöht sich die Transferrate auf 128kBit/s.

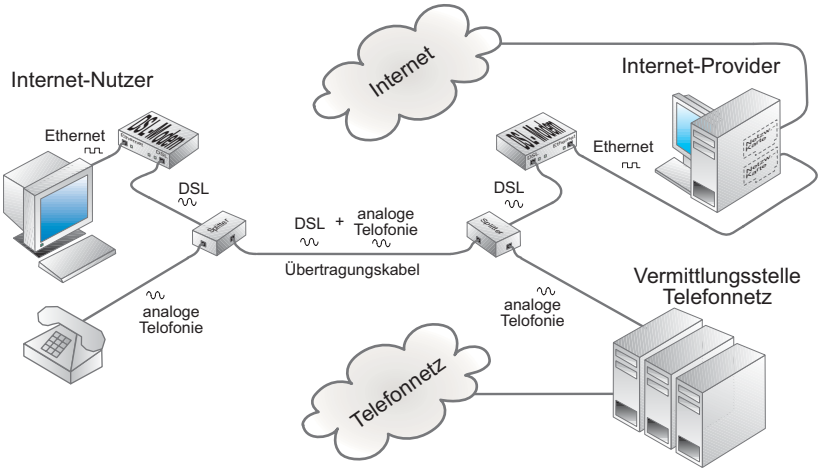
DSL

Digital Subscriber Line (deutsch: Digitale Teilnehmeranschlussleitung) bietet zur Zeit die attraktivste Möglichkeit, sich mit dem Internet zu verbinden.

Analoge Anschlüsse arbeiten auf dem Kabel mit Frequenzen bis max. 3,5 kHz. Bei ISDN liegt die Obergrenze bei ca. 40 kHz. DSL nutzt ausschließlich Frequenzen, die oberhalb 40 kHz bis ca. 1MHz angesiedelt sind. Damit kann DSL parallel zu analogen oder ISDN-Anschlüssen über dasselbe Kabel betrieben werden. Am Standort des Teilnehmeranschlusses wird über einen Splitter (eine Frequenzweiche) das DSL-Signal von den Telefonsignalen getrennt.



Die Übertragung der DSL-Daten funktioniert ähnlich wie beim analogen Modem, nur dass gleichzeitig mit mehreren, deutlich höheren Trägerfrequenzen gearbeitet wird.



DSL gibt es in verschiedenen Varianten mit Downstream Übertragungsraten zwischen 1MBit/s – 8MBit/s. Hierbei gilt: Je größer die Entfernung zur Vermittlungsstelle, desto geringer die mögliche Geschwindigkeit. Da die Upstream-Geschwindigkeit nur ca. ein Achtel der Downstream-Geschwindigkeit beträgt, spricht man auch von asynchroner Datenübertragung, kurz ADSL.

Auf Grund der hohen Übertragungsgeschwindigkeit tauschen DSL-Modems die Daten mit dem PC über USB oder direkt über Ethernet aus. Eine häufige Variante ist ein Ethernet Router mit DSL-Anschluss.

1.2.2 Übertragungsprotokolle

Die im vorigen Abschnitt beschriebenen Übertragungsverfahren analoge Modemübertragung, ISDN und DSL sind für sich betrachtet nur physikalische Standards, die aber nichts darüber aussagen, in welcher Form der Datenstrom transportiert wird.

Um Daten per DFÜ zwischen zwei Netzwerkstandorten auszutauschen, ist ein übergeordnetes Protokoll nötig, welches folgende Aufgaben übernimmt:

- Aufbau einer logischen Verbindung zwischen beiden Standorten

- Authentifizierung (Prüfen der Zugangsberechtigung)
- Aufbereitung des eingehenden Datenverkehrs für die Übertragung und Wiederherstellen des ursprünglichen Datenformates am Ende der Übertragungsstrecke
- Datensicherung
- Verschlüsselung der Übertragungsdaten
- Abbau der logischen Verbindung nach Abschluss der Datenübertragung

SLIP - Serial Line IP Protocol

Ein erster Ansatz für die Übertragung von DFÜ-Daten war SLIP. SLIP ist ein sehr einfaches Protokoll, das ausschließlich für den Transport von TCP/IP-Datenverkehr geeignet ist und nicht alle oben aufgeführten Anforderungen erfüllt.

Die kompletten IP-Datenpakete werden bei SLIP einfach um ein festgelegtes Start- und Endezeichen erweitert. Zufällig im IP-Paket vorkommende Zeichen dieses Typs ersetzt der Sender durch eine Kombination aus Ersatzzeichen.

So präpariert wird Paket für Paket auf die Leitung gegeben.

An den Start-/Ende-Zeichen eines Pakets erkennt der Empfänger wo das eigentliche IP-Paket beginnt bzw. endet. Die Ersatzzeichen werden vom Empfänger wieder gegen das Original ausgetauscht und die Start-/Endezeichen entfernt.

Durch die Beschränkung auf IP-Datenübertragung und fehlende Sicherheitsmechanismen wird SLIP heute für normale Internetzugänge nicht mehr benutzt. Dort wo räumlich abgesetzte Netzwerksegmente über Distanzen verbunden werden sollen, die mit einer normalen Ethernet-Verkabelung nicht mehr möglich sind, ist Slip aber nach wie vor eine zweckmäßige Lösung.

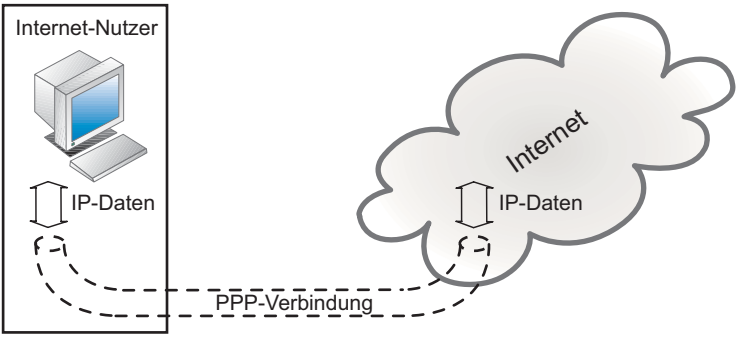
Die W&T Com-Server können z.B. als SLIP-Router konfiguriert werden und somit TCP/IP-Daten über eine RS232 oder RS422 Verkabelung übertragen.

PPP - Point-to-Point Protocol

Um allen Anforderungen für eine Datentunnelung zwischen zwei Netzwerkstandorten gerecht zu werden, wurde PPP entwickelt.

Sowohl für den Zugang ins Internet als auch zur Verbindung mit einem entfernten nicht öffentlichen Netzwerk sorgt PPP für eine gesicherte Datenübertragung.

Dazu stellt PPP sozusagen einen Tunnel durch die netzwerkfremde Umgebung her.



Der Aufbau einer PPP-Verbindung findet in mehreren Schritten statt und bedingt eine bestehende physikalische Verbindung wie z.B. DSL oder ISDN:

1. **Aushandeln der Verbindungsoptionen**
   Um festzulegen, mit welchen Optionen PPP arbeiten soll, wird das LCP-Protokoll - Link Control Protocol benutzt.
   Verhandelbar sind unter anderem:
   - Art der Authentifizierung
   - Blockgröße der Übertragungsdaten
   - Datenkompression
   - Art der zu übertragenden Daten (IP, IPX, ...)

2. **Authentifizierung**
   Hierbei werden User-ID und ein Passwort übergeben. Es gibt zwei Arten der Passwortübergabe:
   - PAP - Password Authentication Protocol
     Passwortübergabe lesbar im Klartext

- CHAP - Challange Handshake
  verschlüsselte Passwortübergabe

3. **Konfiguration übergeordneter Netzwerkprotokolle**
   Soll über PPP eine Verbindung in Netze mit übergeordneten Protokollen (z.B. Internetprotokoll) hergestellt werden, ist es erforderlich bestimmte, das entsprechende Protokoll betreffende Einstellungen vorzunehmen.
   Die nötigen Informationen werden mittels des NCP - Network Control Protocol übergeben. Im Falle eines Internetzugangs über PPP wird als NCP das Internet spezifische IPCP - Internet Protocol Control Protocol verwendet. IPCP erlaubt z.B. für die Dauer der PPP-Verbindung die Vergabe einer IP-Adresse (mehr zum Internet Protokoll im nächsten Abschnitt).

4. **Übertragung der Nutzdaten**
   Sobald alle Verbindungsoptionen festgelgt sind und der Nutzer seine Zugangsberechtigung nachgewiesen hat, beginnt der eigentliche Austausch von Nutzdaten.
   Im Fall einer Verbindung zum Internet können das Daten in Form aller IP-basierenden Protokolle sein (UDP, TCP, Telnet, FTP, HTTP...).

5. **Abbau der PPP-Verbindung**
   Auch der Verbindungsabbau wird über LCP abgewickelt.

Ähnlich wie Ethernet bettet PPP die zu transportierenden Daten in eine festgelegte Paketstruktur:

Flag	Address	Control	Protocol	Information	FCS	Flag
1 Byte	1 Byte	1 Byte	1 oder 2 Byte	n Byte	1 Byte	1 Byte

Flag (1 Byte)
Startzeichen zur Paketsynchronisation bzw. Paketerkennung

Address (1 Byte)
Eine Punkt- zu- Punkt-Verbindung erfordert keine Adressierung.

Trotzdem ist dieses Feld aus Kompatibilitätsgründen zu anderen Netzwerkprotokollen vorhanden, wird aber von PPP nicht benutzt und ist willkürlich mit dem Wert 255 gefüllt.

Control (1 Byte)
Dieses Feld war ursprünglich zur Nummerierung der Pakete vorgesehen, hat bei PPP aber immer den Wert 3, da ohne Paketnummerierung gearbeitet wird.

Protocol (1 oder 2 Byte)
Der Inhalt dieses Feldes gibt an, wie das aktuelle PPP-Paket genutzt wird: Verbindungsaufbau, Steuerinformation, Authentifizierung, Datentransport, Verbindungsabbau, ...

Information (n Byte)
An dieser Stelle wird die eigentliche Information (z.B. IP-Daten) übertragen. Bei Steuerpaketen stehen hier die Steueroptionen im LCP-Format.

Die Größe dieses Feldes ist per LCP verhandelbar, beträgt in aller Regel aber 1500 Byte. Ist die zu transportierende Information kleiner, werden Füllzeichen aufgefüllt.

FCS (2 Byte)
Checksumme zur Kontrolle der empfangenen Daten

Flag (1 Byte)
Endezeichen zur Paketsynchronisation

Durch die Möglichkeit, über eine PPP-Verbindung verschiedene unabhängige IP-Dienste und Protokolle gleichzeitig zu betreiben, können durch Einsatz geeigneter Router auch ganze Netzwerke über PPP verbunden werden.

2. Logische Adressierung und Datentransport

Weder Ethernet, noch die gängigen DFÜ-Techniken allein verfügen über die Möglichkeit, verschiedene Netze zu adressieren.

Darüber hinaus arbeitet Ethernet z.B. verbindungslos: Der Absender erhält vom Empfänger keine Bestätigung, ob ein Paket angekommen ist.

Spätestens wenn ein Ethernet-Netzwerk mit mehreren Netzen verbunden werden soll, muss also mit übergeordneten Protokollen – etwa mit TCP/IP – gearbeitet werden.

Bereits in den 60er Jahren vergab das amerikanische Militär den Auftrag, ein Protokoll zu schaffen, das unabhängig von der verwendeten Hard- und Software einen standardisierten Informationsaustausch zwischen einer beliebigen Zahl verschiedener Netzwerke möglich machen sollte. Aus dieser Vorgabe entstand im Jahr 1974 das Protokoll TCP/IP.

Obwohl TCP und IP immer in einem Wort genannt werden, handelt es sich hier um zwei aufeinander aufsetzende Protokolle. Das Internet Protocol IP übernimmt die richtige Adressierung und Zustellung der Datenpakete, während das darauf aufsetzende Transport Control Protocol TCP für den Transport und die Sicherung der Daten zuständig ist.

2.1 TCP/IP im lokalen Netz

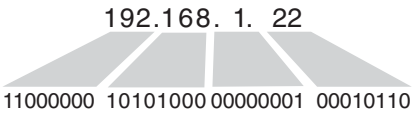
Der besseren Übersichtlichkeit halber wollen wir zunächst den Datentransport und die logische Adressierung mit TCP/IP innerhalb eines lokalen Netzes näher beleuchten.

2.1.1 IP - Internet Protocol

Für das Verständnis der Adressierung innerhalb eines lokalen Netzes reicht uns zunächst ein Blick auf die grundsätzliche Struktur des Internet Protocols IP und auf das Address Resolution Protocol ARP, welches die Zuordnung von IP-Adressen zu Ethernet-Adressen ermöglicht.

IP-Adressen

Unter IP hat jeder Netzteilnehmer eine einmalige IP-Adresse, die oft auch als „IP-Nummer“ bezeichnet wird. Diese Internet-Adresse ist ein 32-Bit-Wert, der zur besseren Lesbarkeit immer in Form von vier durch Punkte getrennte Dezimalzahlen (8-Bit-Werten) angegeben wird (Dot-Notation).

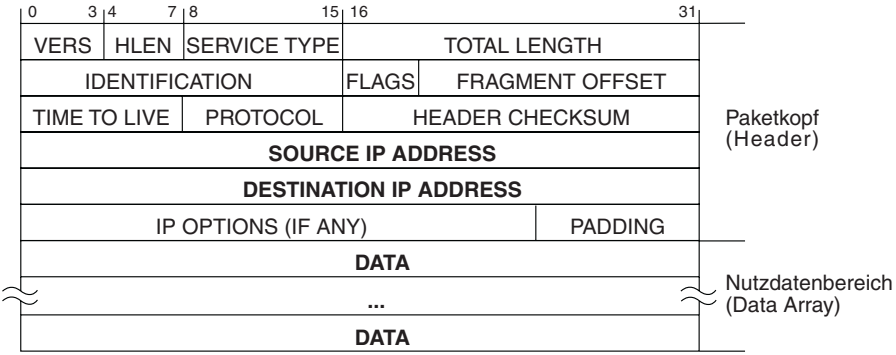


Eine IP-Adresse muss im gesamten verbundenen Netzwerk einmalig sein!

IP-Datenpakete

Auch bei der Datenübertragung mit IP werden die Nutzdaten in einen Rahmen von Adressierungsinformationen gepackt. IP-Datenpakete enthalten neben den zu transportierenden Nutzdaten eine Fülle von Adress- und Zusatzinformationen, die im sogenannten „Paketkopf“ stehen. Wir beschränken uns hier auf die Erklärung der wichtigsten Adressinformationen.

Aufbau eines IP-Datenpakets



source IP address: IP-Adresse des Absenders
destination IP address: IP-Adresse des Empfängers

ARP – Address Resolution Protocol

Der IP-Treiber übergibt neben dem IP-Datenpaket auch die physikalische Ethernet-Adresse an den Ethernetkarten-Treiber. Zur Ermittlung der Ethernet-Adresse des Empfängers bedient sich der IP-Treiber des Address Resolution Protocol ARP.

In jedem TCP/IP-fähigen Rechner gibt es eine ARP-Tabelle. Die ARP-Tabelle wird vom TCP/IP-Treiber bei Bedarf aktualisiert und enthält die Zuordnung von IP-Adressen zu Ethernet-Adressen.

InternetAddress	Physical Address	Type
172.16.232.23	00-80-48-9c-ac-03	dynamic
172.16.232.49	00-c0-3d-00-26-a1	dynamic
172.16.232.92	00-80-48-9c-a3-62	dynamic
172.16.232.98	00-c0-3d-00-1b-26	dynamic
172.16.232.105	00-c0-3d-00-18-bb	dynamic

Soll ein IP-Paket verschickt werden, sieht der IP-Treiber zunächst nach, ob die gewünschte IP-Adresse bereits in der ARP-Tabelle vorhanden ist. Ist dies der Fall, gibt der IP-Treiber die ermittelte Ethernet-Adresse zusammen mit seinem IP-Paket an den Ethernet-Kartentreiber weiter.

Kann die gewünschte IP-Adresse nicht gefunden werden, startet der IP-Treiber einen ARP-Request. Ein ARP-Request ist ein Rundruf (auch Broadcast genannt) an alle Teilnehmer im lokalen Netz.

Damit der Rundruf von allen Netzteilnehmern zur Kenntnis genommen wird, gibt der IP-Treiber als Ethernet-Adresse FF-FF-FF-FF-FF-FF an. Ein mit FF-FF-FF-FF-FF-FF adressiertes Ethernet-Paket wird grundsätzlich von allen Netzteilnehmern gelesen. Im IP-Paket wird als Destination die gewünschte IP-Adresse angegeben und im Feld Protocol des IP-Headers die Kennung für ARP ausgewiesen.

Derjenige Netzteilnehmer, der in diesem ARP-Request seine eigene IP-Adresse wiedererkennt, bestätigt das mit einem ARP-Reply. Der ARP-Reply ist ein auf Ethernet-Ebene an den ARP-Request-Absender adressiertes Datenpaket mit der ARP-Ken-

nung im Protocol-Feld. Im Datenbereich des ARP-Paketes sind außerdem die IP-Adressen von Sender und Empfänger des ARP-Reply eingetragen.

Der IP-Treiber kann nun die dem ARP-Reply entnommene Ethernet-Adresse der gewünschten IP-Adresse zuordnen und trägt sie in die ARP-Tabelle ein.

Im Normalfall bleiben die Einträge in der ARP-Tabelle nicht dauerhaft bestehen. Wird ein eingetragener Netzwerkteilnehmer über eine bestimmte Zeit (unter Windows ca. 2 Min.) nicht kontaktiert, wird der entsprechende Eintrag gelöscht. Das hält die ARP-Tabelle schlank und ermöglicht den Austausch von Hardwarekomponenten unter Beibehaltung der IP-Adresse. Man nennt diese zeitlich begrenzten Einträge auch dynamische Einträge.

Neben den dynamischen Einträgen gibt es auch statische Einträge, die der Benutzer selbst in der ARP-Tabelle ablegt. Die statischen Einträge können genutzt werden, um an neue Netzwerkkomponenten, die noch keine IP-Adresse haben, die gewünschte IP-Adresse zu übergeben.

Diese Art der Vergabe von IP-Adressen lassen auch Com-Server zu: Empfängt ein Com-Server, der noch keine eigene IP-Adresse hat, ein IP-Datenpaket, das auf Ethernet-Ebene an ihn adressiert ist, wird die IP-Adresse dieses Pakets ausgewertet und als eigene IP-Adresse übernommen.

Achtung: Nicht alle Netzwerkkomponenten besitzen diese Fähigkeit. PCs lassen sich auf diese Weise z.B. nicht konfigurieren!

2.1.2 Die Transportprotokolle TCP und UDP

Die Frage, auf welche Art und Weise Daten transportiert werden sollen, lösen Transportprotokolle, die jeweils verschiedenen Anforderungen gerecht werden.

TCP - Transport Control Protocol

Weil IP ein ungesichertes, verbindungsloses Protokoll ist, arbeitet es in der Regel mit dem aufgesetzten TCP zusammen, das die Sicherung und das Handling der Nutzdaten übernimmt. TCP stellt für die Dauer der Datenübertragung eine Verbindung zwischen zwei Netzteilnehmern her. Beim Verbindungsaufbau werden Bedingungen wie z.B. die Größe der Datenpakete festgelegt, die für die gesamte Verbindungsdauer gelten.

TCP kann man mit einer Telefonverbindung vergleichen. Teilnehmer A wählt Teilnehmer B an; Teilnehmer B akzeptiert mit dem Abheben des Hörers die Verbindung, die dann bestehen bleibt, bis einer der beiden sie beendet.

TCP arbeitet nach dem sogenannten *Client-Server-Prinzip*:

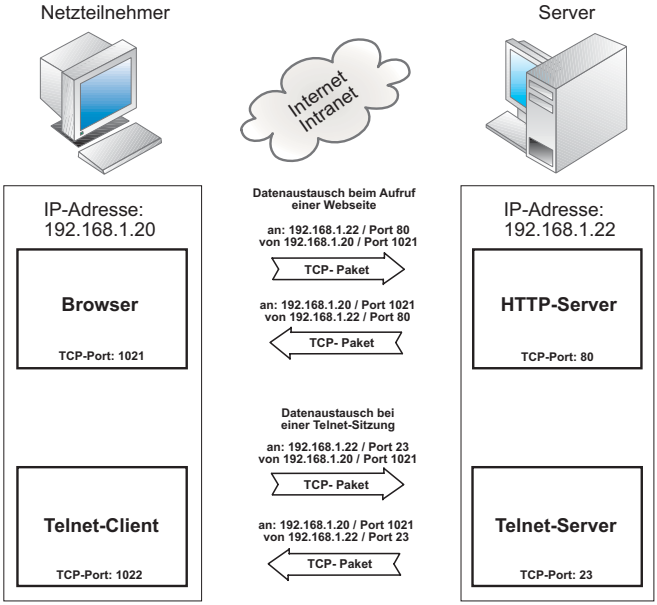
Denjenigen Netzteilnehmer, der eine Verbindung aufbaut (der also die Initiative ergreift), bezeichnet man als Client. Der Client nimmt einen vom Server angebotenen Dienst in Anspruch, wobei je nach Dienst ein Server auch mehrere Clients gleichzeitig bedienen kann.

Derjenige Netzteilnehmer, zu dem die Verbindung aufgebaut wird, wird als Server bezeichnet. Ein Server tut von sich aus nichts, sondern wartet auf einen Client, der eine Verbindung zu ihm aufbaut. Im Zusammenhang mit TCP spricht man von TCP-Client und TCP-Server.

TCP sichert die übertragenen Nutzdaten mit einer Checksumme und versieht jedes gesendete Datenpaket mit einer Sequenznummer. Der Empfänger eines TCP-Pakets prüft anhand der Checksumme den korrekten Empfang der Daten. Hat ein TCP-Server ein Paket korrekt empfangen, wird über einen vorgegebenen Algorithmus aus der Sequenznummer eine Acknowledgement-Nummer errechnet.

Die Acknowledgement-Nummer wird dem Client mit dem nächsten selbst gesendeten Paket als Quittung zurückgegeben. Der Server versieht seine gesendeten Pakete ebenfalls mit einer eigenen Sequenznummer, die wiederum vom Client mit einer Acknowledgement-Nummer quittiert wird. Dadurch ist gewährleistet, dass der Verlust von TCP-Paketen bemerkt wird, und diese im Bedarfsfall in korrekter Abfolge erneut gesendet werden können.

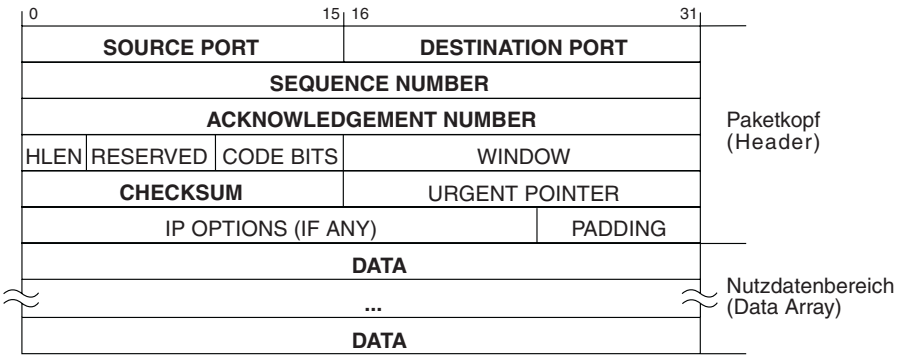
Darüber hinaus leitet TCP die Nutzdaten auf dem Zielrechner an das richtige Anwendungsprogramm weiter, indem es unterschiedliche Anwendungsprogramme – auch Dienste genannt – über unterschiedliche Portnummern anspricht. So ist Telnet z.B. über Port 23, HTTP, der Dienst über den Webseiten aufgerufen werden über Port 80 zu erreichen. Vergleicht man ein TCP-Paket mit einem Brief an eine Behörde, kann man die Portnummer mit der Raumnummer der adressierten Dienststelle vergleichen. Befindet sich z.B. das Straßenverkehrsamt in Raum 312 und man adressiert einen Brief an eben diesen Raum, dann gibt man damit zugleich auch an, dass man die Dienste des Straßenverkehrsamts in Anspruch nehmen möchte.



Damit die Antwort des Zielrechners wieder an der richtigen Stelle ankommt, hat auch die Client-Anwendung eine Portnummer. Bei PC-Anwendungen werden die Portnummern der Client-Anwendungen dynamisch und unabhängig von der Art der Anwendung vergeben.

Auch TCP verpackt die Nutzdaten in einen Rahmen von Zusatzinformationen. Solche TCP-Pakete sind wie folgt aufgebaut:

Aufbau eines TCP-Datenpakets



Source Port: Portnummer der Applikation des Absenders

Destination Port: Portnummer der Applikation des Empfängers

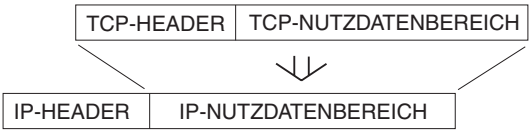
Sequence No: Offset des ersten Datenbytes relativ zum Anfang des TCP-Stroms (garantiert die Einhaltung der Reihenfolge)

Acknowl. No: im nächsten TCP-Paket erwartete Sequence No.

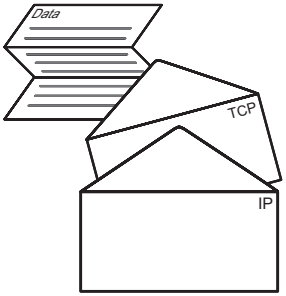
Data: Nutzdaten

Das so entstandene TCP-Paket wird in den Nutzdatenbereich eines IP-Pakets eingesetzt.

Aufbau eines TCP/IP-Datenpakets



Die Nutzdaten werden quasi in einen Briefumschlag (TCP-Paket) gesteckt, der wiederum in einen Briefumschlag (IP-Paket) gesteckt wird.



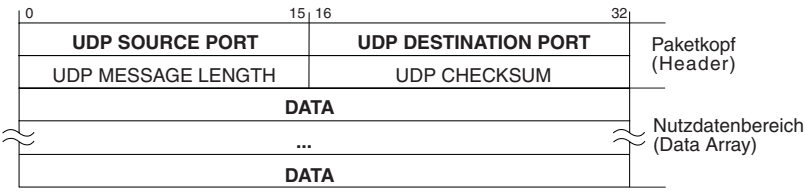
UDP – User Datagramm Protocol

UDP ist ein weiteres Transportprotokoll, das genau wie TCP auf IP aufsetzt. Im Gegensatz zu TCP arbeitet UDP verbindungslos. Jedes Datenpaket wird als Einzelsendung behandelt, und es gibt keine Rückmeldung darüber, ob ein Paket beim Empfänger angekommen ist.

Weil unter UDP aber keine Verbindungen auf- und abgebaut werden müssen und somit keine Timeout-Situationen entstehen können, kann UDP jedoch schneller als TCP sein: Wenn ein Paket verlorengeht, wird die Datenübertragung hier eben ungehindert fortgesetzt, sofern nicht ein höheres Protokoll für Wiederholungen sorgt.

Die Datensicherheit ist unter UDP also in jedem Fall durch das Anwendungsprogramm zu gewährleisten.

Aufbau eines UDP-Datenpakets



Source Port: Portnummer der sendenden Anwendung (Rücksende-Port für Empfänger).

Destination Port: Zielport, an den die Daten beim Empfänger übertragen werden sollen.

Als Faustregel kann man sagen:

- Für kontinuierliche Datenströme oder große Datenmengen sowie in Situationen, in denen ein hohes Maß an Datensicherheit gefordert ist, wird in aller Regel TCP eingesetzt.
- Bei häufig wechselnden Übertragungspartnern sowie einer Gewährleistung der Datensicherheit durch übergeordnete Protokolle macht der Einsatz von UDP Sinn.

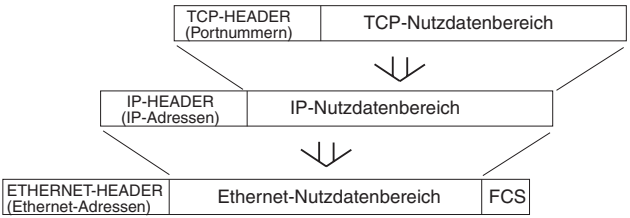
2.1.3 Der Weg eines Zeichens durch das Ethernet

Wir haben nun mit TCP/IP (bzw. UDP/IP) das Handwerkszeug kennengelernt, mit dem Daten adressiert und transportiert werden. Zusammenfassend wird im Folgenden noch einmal der Weg eines Zeichens in einem lokalen Netz aufgezeigt.

TCP/IP ist ein rein logisches Protokoll und benötigt immer eine physikalische Grundlage. Wie bereits anfänglich erwähnt, genießt Ethernet heute die größte Verbreitung bei den physikalischen Netzwerktopologien. So findet man auch in den meisten TCP/IP-Netzwerken Ethernet als physikalische Grundlage.

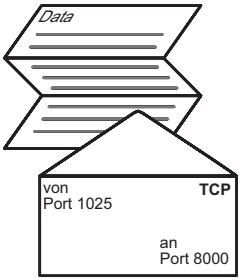
TCP/IP und Ethernet werden zusammengeführt, indem jedes TCP/IP-Paket in den Nutzdatenbereich eines Ethernet-Paketes eingebettet wird.

Aufbau eines TCP/IP-Ethernet-Datenpakets

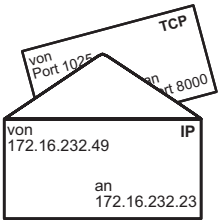


Die Nutzdaten passieren auf ihrem Weg von der Applikation auf dem PC bis ins Netzwerk mehrere Treiberebenen:

- Das Anwendungsprogramm entscheidet, an welchen anderen Netzteilnehmer die Daten gesendet werden sollen, und übergibt IP-Adresse und TCP-Port dem TCP/IP-Treiber (oft auch TCP/IP-Stack genannt).
- Der TCP/IP-Treiber koordiniert den Aufbau der TCP-Verbindung.
- Die vom Anwendungsprogramm übergebenen Nutzdaten werden vom TCP-Treiber je nach Größe in kleinere, übertragbare Blöcke geteilt.
- Jeder Datenblock wird zunächst vom TCP-Treiber in ein TCP-Paket verpackt.

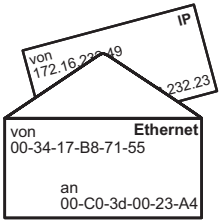


- Der TCP-Treiber übergibt das TCP-Paket und die IP-Adresse des Empfängers an den IP-Treiber.
- Der IP-Treiber verpackt das TCP-Paket in ein IP-Paket.



- Der IP-Treiber sucht in der ARP-Tabelle (Address Resolution Protocol) nach der Ethernet-Adresse des durch die IP-Adresse angegebenen Empfängers und übergibt das IP-Paket zusammen mit der ermittelten Ethernet-Adresse an den Ethernet-Kartentreiber.

- Der Ethernet-Kartentreiber verpackt das IP-Paket in ein Ethernet-Paket und gibt dieses Paket über die Netzwerkkarte auf das Netzwerk aus.



Beim Empfänger findet die Prozedur in umgekehrter Reihenfolge statt:

- Die Ethernetkarte erkennt an der Destination-Ethernet-Adresse, dass das Paket für den Netzteilnehmer bestimmt ist und gibt es an den Ethernet-Treiber weiter.

- Der Ethernet-Treiber isoliert das IP-Paket und gibt es an den IP-Treiber weiter.

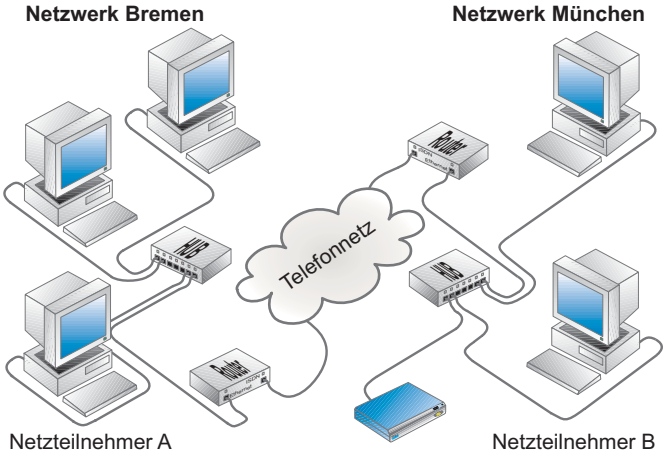
- Der IP-Treiber isoliert das TCP-Paket und gibt es an den TCP-Treiber weiter.

- Der TCP-Treiber überprüft den Inhalt des TCP-Paketes auf Richtigkeit und übergibt die Daten anhand der Portnummer an die richtige Applikation.

Auf den ersten Blick erscheint dieses vielschichtige Übertragungsverfahren ungeheuer umständlich. Aber erst die strikte Trennung von logischem Protokoll (TCP/IP) und physikalischem Protokoll (Ethernet) macht es möglich, netzübergreifend und hardwareunabhängig Daten auszutauschen.

2.2 TCP/IP bei netzübergreifender Verbindung

Das Internet-Protokoll macht es möglich, eine unbestimmte Anzahl von Einzelnetzen zu einem Gesamtnetzwerk zusammenzufügen. Es ermöglicht also den Datenaustausch zwischen zwei beliebigen Netzteilnehmern, die jeweils in beliebigen Einzelnetzen positioniert sind. Die physikalische Ausführung der Netze bzw. Übertragungswege (Ethernet, Token Ring, ISDN....) spielt hierbei keine Rolle.



Die verschiedenen Einzelnetze werden über Gateways/Router miteinander verbunden und fügen sich so zum Internet bzw. Intranet zusammen. Die Adressierung erfolgt nach wie vor über die IP-Adresse, die wir uns nun einmal genauer ansehen werden.

2.2.1 Netzklassen

Die IP-Adresse unterteilt sich in Net-ID und Host-ID, wobei die Net-ID zur Adressierung des Netzes und die Host-ID zur Adressierung des Netzteilnehmers innerhalb eines Netzes dient. An der Net-ID erkennt man, ob der Empfänger, zu dem die Verbindung aufgebaut werden soll, im gleichen Netzwerk wie der Sender zu finden ist. Stimmt dieser Teil der IP-Adresse bei Sender und Empfänger überein, befinden sich beide im selben Netzwerk; stimmt er nicht überein, ist der Empfänger in einem anderen Netzwerk zu finden.

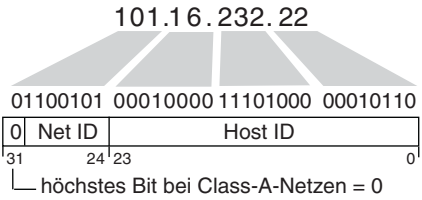
Ähnlich sind auch Telefonnummern aufgebaut. Hier unterscheidet man ebenfalls zwischen Vorwahl und Teilnehmerrufnummer.

Je nachdem, wie groß der Anteil der Net-ID an einer IP-Adresse ist, sind wenige große Netze mit jeweils vielen Teilnehmern und viele kleine Netze mit jeweils wenigen Teilnehmern denkbar. In den Anfängen des Internets hat man den IP-Adressraum anhand der Größe der möglichen Netzwerke in Klassen unterschieden.

	Adressbereich des Netzwerks	Anzahl möglicher Netze	mögl. Anzahl Hosts/Netz
Class A	1.xxx.xxx.xxx – 126.xxx.xxx.xxx	127 (2^7)	ca. 16 Millionen (2^{24})
Class B	128.0.xxx.xxx – 191.255.xxx.xxx	ca. 16.000 (2^{14})	ca. 65.000 (2^{16})
Class C	192.0.0.xxx – 223.255.255.xxx	ca. 2 Millionen (2^{21})	254 (2^8)

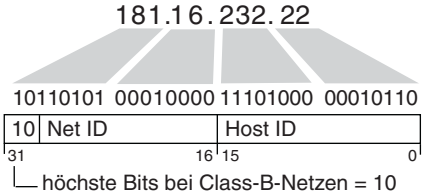
Class A

Das erste Byte der IP-Adresse dient der Adressierung des Netzes, die letzten drei Byte adressieren den Netzteilnehmer.



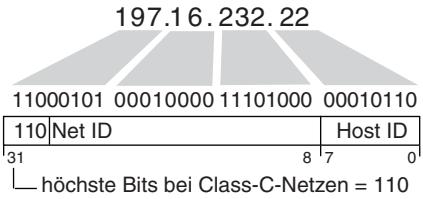
Class B

Die ersten zwei Byte der IP-Adresse dienen der Adressierung des Netzes, die letzten zwei Byte adressieren den Netzteilnehmer.



Class C

Die ersten drei Byte der IP-Adresse dienen der Adressierung des Netzes, das letzte Byte adressiert den Netzteilnehmer.

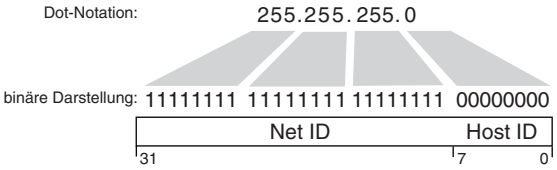


Neben den hier aufgeführten Netzen, gibt es auch noch Class-D- und Class-E-Netze, deren Adressbereiche oberhalb der Class-C-Netze liegen. Class-D-Netze und Class-E-Netze haben in der Praxis wenig Bedeutung, da sie nur zu Forschungszwecken und für Sonderaufgaben verwendet werden. Der normale Internetbenutzer kommt mit diesen Netzwerkklassen nicht in Berührung.

2.2.2 Subnet-Mask

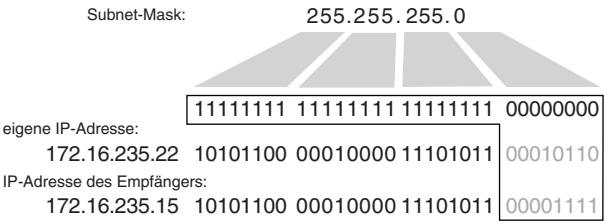
Nun ist es allerdings möglich, ein Netzwerk – egal welcher Netzwerkklasse –in weitere Unternetzwerke zu unterteilen. Zur Adressierung solcher Subnets reicht die von den einzelnen Netzwerkklassen vorgegebene Net-ID allerdings nicht aus; man muss einen Teil der Host-ID zur Adressierung der Unternetze abzweigen. Im Klartext bedeutet dies, dass die Net-ID sich vergrößert und die Host-ID entsprechend kleiner wird.

Welcher Teil der IP-Adresse als Net-ID und welcher als Host-ID ausgewertet wird, gibt die Subnet-Mask vor. Die Subnet-Mask ist genau wie die IP-Adresse ein 32-Bit-Wert, der in Dot-Notation dargestellt wird. Betrachtet man die Subnet-Mask in binärer Schreibweise, ist der Anteil der Net-ID mit Einsen, der Anteil der Host-ID mit Nullen aufgefüllt.



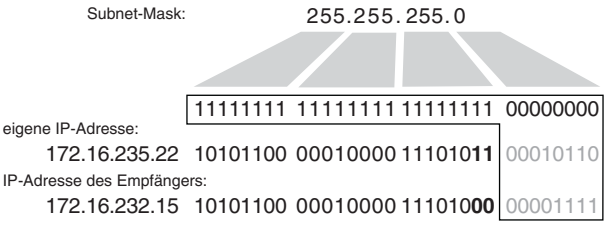
Bei jedem zu verschickenden Datenpaket vergleicht der IP-Treiber die eigene IP-Adresse mit der des Empfängers. Hierbei werden die Bits der Host-ID über den mit Nullen aufgefüllten Teil der Subnet-Mask ausgeblendet.

Sind die ausgewerteten Bits beider IP-Adressen identisch, befindet sich der gewählte Netzteilnehmer im selben Subnet.



Im oben dargestellten Beispiel kann der IP-Treiber die Ethernet-Adresse über ARP ermitteln und diese dem Netzwerkkarten-Treiber zur direkten Adressierung übergeben.

Unterscheidet sich auch nur ein einziges der ausgewerteten Bits, befindet sich der gewählte Netzteilnehmer nicht im selben Subnet.



In diesem Fall muss das IP-Paket zur weiteren Vermittlung ins Zielnetzwerk einem Gateway bzw. Router übergeben werden. Zu

diesem Zweck ermittelt der IP-Treiber über ARP die Ethernet Adresse des Routers, auch wenn im IP-Paket selbst nach wie vor die IP-Adresse des gewünschten Netzteilnehmers eingetragen ist.

2.2.3 Gateways und Router

Gateways bzw. Router sind im Prinzip nichts anderes als Computer mit zwei Netzwerkkarten. Ethernet-Datenpakete, die auf Karte A empfangen werden, werden vom Ethernet-Treiber entpackt und das enthaltene IP-Paket wird an den IP-Treiber weitergegeben. Dieser prüft, ob die Ziel-IP-Adresse zum an Karte B angeschlossenen Subnet gehört und das Paket direkt zugestellt werden kann, oder ob das IP-Paket an ein weiteres Gateway übergeben wird.

So kann ein Datenpaket auf seinem Weg von einem Netzteilnehmer zum anderen mehrere Gateways/Router passieren. Während auf IP-Ebene auf der gesamten Strecke die IP-Adresse des Empfängers eingetragen ist, wird auf Ethernet-Ebene immer nur das nächste Gateway adressiert. Erst auf dem Teilstück vom letzten Gateway/Router zum Empfänger wird in das Ethernet-Paket die Ethernet-Adresse des Empfängers eingesetzt.

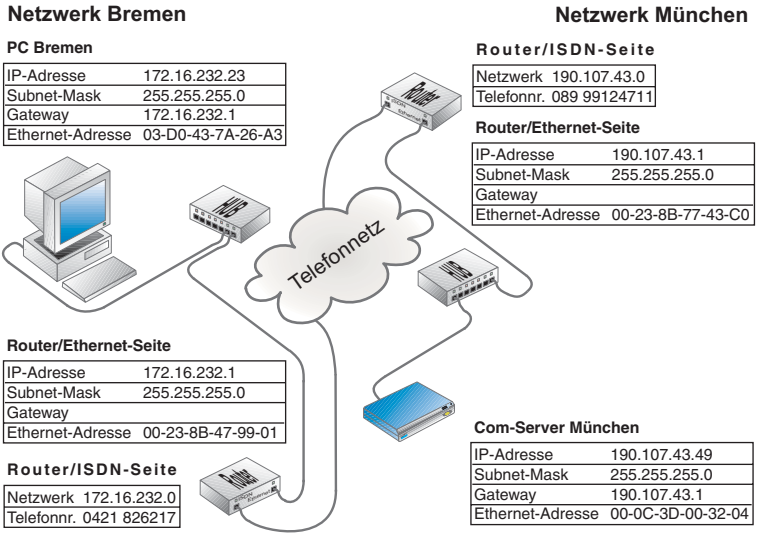
Neben Routern, die ein Ethernet-Subnet mit einem anderen Ethernet-Subnet verbinden, gibt es auch Router, die das physikalische Medium wechseln –z.B. von Ethernet auf Token Ring oder ISDN. Während auch hier die IP-Adressierung über die gesamte Strecke gleich bleibt, ist die physikalische Adressierung von einem Router zum anderen den auf den Teilstrecken geforderten physikalischen Gegebenheiten angepasst.

Zwischen zwei Ethernet-ISDN-Routern wird zum Beispiel über Telefonnummern adressiert.

2.2.4 Der Weg eines Zeichens durch mehrere Netze

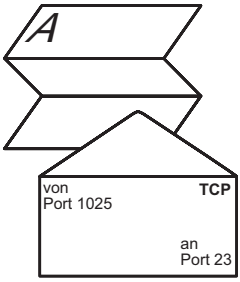
Im folgenden Abschnitt wird anhand einer bestehenden Telnetverbindung der Weg eines Zeichens über eine geroutete Netzwerkverbindung beschrieben.

Wir gehen in unserem Beispiel davon aus, dass ein Anwender in Bremen bereits eine Telnetverbindung zu einem W&T Com-Server in München aufgebaut hat; die Verbindung der Netze Bremen und München besteht in Form einer Routerverbindung über das ISDN-Netz.



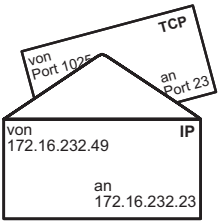
Der Anwender in Bremen gibt in der Telnet-Client-Anwendung das Zeichen „A“ ein.

- Das Telnet-Client-Programm auf dem PC übergibt dem TCP/IP-Stack das „A“ als Nutzdatum. Die IP-Adresse des Empfängers (190.107.43.49) und die Portnummer 23 für Telnet wurden dem TCP/IP-Stack bereits bei Aufbau der Verbindung übergeben.
- Der TCP-Treiber schreibt das „A“ in den Nutzdatenbereich eines TCP-Pakets und trägt als Destination-Port die 23 ein.

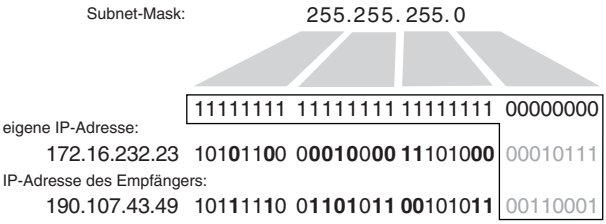


- Der TCP-Treiber übergibt das TCP-Paket und die IP-Adresse des Empfängers an den IP-Treiber.

- Der IP-Treiber verpackt das TCP-Paket in ein IP-Paket.



- Der IP-Treiber ermittelt über den Vergleich der Net-ID-Anteile von eigener IP-Adresse und IP-Adresse des Empfängers, ob das IP-Paket im eigenen Subnet zugestellt werden kann oder einem Router übergeben wird.



Hier sind die Net-ID-Anteile der beiden Adressen nicht gleich; das IP-Paket muss folglich an den eingetragenen Router übergeben werden.

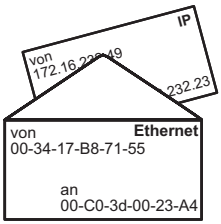
- Der IP-Treiber ermittelt über ARP die Ethernet-Adresse des Routers. Da die TCP-Verbindung bereits aufgebaut ist, wird

die IP-Adresse des Routers bereits in der ARP-Tabelle aufgelöst sein.

```
Internet Address      Physical Address      Type
→  172.16.232.1       00-23-8B-74-99-01     dynamic
   172.16.232.49      00-c0-3d-00-26-a1     dynamic
   172.16.232.92      00-80-48-9c-a3-62     dynamic
```

- Der IP-Treiber entnimmt der ARP-Tabelle die Ethernet-Adresse des Routers und übergibt sie zusammen mit dem IP-Paket dem Ethernet-Kartentreiber.

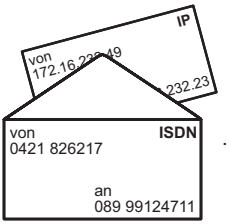
- Der Ethernet-Kartentreiber verpackt das IP-Paket in ein Ethernet-Paket und gibt dieses Paket über die Netzwerkkarte auf das Netzwerk aus.



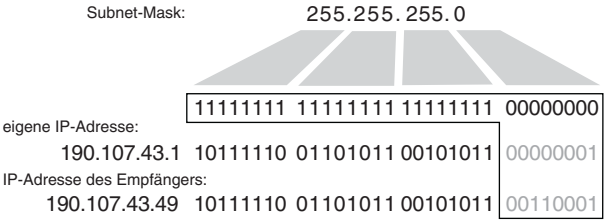
- Der Router entnimmt dem empfangenen Ethernet-Paket das IP-Paket.

- Die IP-Adresse des Empfängers wird mit einer sogenannten Routing-Tabelle verglichen. Anhand dieser Routing-Tabelle entscheidet der ISDN-Router, über welche Rufnummer das gesuchte Netzwerk zu finden ist. Da die TCP-Verbindung bereits besteht, ist vermutlich auch die ISDN-Verbindung zu diesem Zeitpunkt schon aufgebaut. Sollte dies nicht mehr der Fall sein, wählt der Router die der Routing-Tabelle entnommene Rufnummer und stellt die ISDN-Verbindung zum Gegen-Router im Zielnetzwerk wieder her.

- Auch im ISDN-Netz wird das IP-Paket in einen Rahmen von Adressinformationen eingepackt. Für uns ist nur wichtig, dass es in seinem Adressierungsbereich unverändert in das ISDN-Paket übernommen wird.

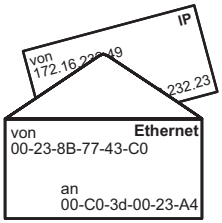


- Der Router im Zielnetz entnimmt dem empfangenen ISDN-Paket das IP-Paket. Über IP-Adressen und Subnet-Mask wird festgestellt, ob das empfangene IP-Paket im lokalen Subnet zugestellt werden kann oder einem weiteren Router übergeben werden muss.



In unserem Beispiel hat das IP-Paket das Zielnetzwerk erreicht und kann im lokalen Netz über Ethernet adressiert werden.

- Der Router, der intern ebenfalls eine ARP-Tabelle führt, ermittelt über ARP die zur IP-Adresse passende Ethernet-Adresse und verpackt das im Adressierungsbereich immer noch unveränderte IP-Paket in ein Ethernet-Paket.



- Der Com-Server erkennt an der Destination-Ethernet-Adresse, dass das Paket für ihn bestimmt ist, und entnimmt das IP-Paket.

- Der IP-Treiber des Com-Servers isoliert das TCP-Paket und gibt es an den TCP-Treiber weiter.

- Der TCP-Treiber überprüft den Inhalt des TCP-Paketes auf Richtigkeit und übergibt die Daten – in diesem Fall das „A“ – an den seriellen Treiber.

- Der serielle Treiber gibt das „A“ auf der seriellen Schnittstelle aus.

Bei einer TCP-Verbindung wird der korrekte Empfang eines Datenpaketes mit dem Rücksenden einer Acknowledgement-Nummer quittiert. Das Quittungspaket durchläuft den gesamten Übertragungsweg und alle damit verbundenen Prozeduren in Gegenrichtung. All dies spielt sich innerhalb weniger Millisekunden ab.

2.3 Exkurs: NAT - Network Address Translation

Möchte man über einen normalen Router ein Netzwerk mit 10 Endgeräten z.B. mittels PPP mit dem Internet verbinden, so würde jedes dieser Endgeräte eine eigene, einmalige IP-Adresse benötigen.

Wie bereits mehrfach angesprochen, sind öffentliche IP-Adressen, d.h. solche, die von der IANA einmalig vergeben werden und daher mit dem Internet verbunden werden können, inzwischen knapp.

Neben diesen öffentlichen IP-Adressen gibt es jedoch noch einen Adressraum für private Netze. Die Bezeichnung „privat“ steht hier für „nicht öffentlich“ und schließt auch Firmennetze mit ein. Je nach Netzwerkgröße sind für private Netze die Adressbereiche 192.168.x.x für Class C Netze und 172.16.x.x für Class B Netze vorgesehen. In diesem Adressbereich können sich Anwender bei der Einrichtung ihres privaten Netzwerks frei bedienen. Da eine Adresse jedoch in mehreren Netzwerken vorkommen kann, darf der entsprechende Netzteilnehmer nicht mit dem Internet verbunden werden.

Mit NAT (Network Address Translation) wurde nun eine Art des Routings geschaffen, die es erlaubt, eine Vielzahl von Teilnehmern in einem privaten Netzwerk zum Internet hin mit nur einer öffentlichen IP-Adresse zu repräsentieren.

Zur Erinnerung: Bei normalem TCP/IP-Datenverkehr adressiert die IP-Adresse den Netzwerkteilnehmer, die Portnummer die Anwendung im Gerät.

Beim NAT-Routing wird auch die Portnummer als zusätzliche Adressinformation für das Endgerät selbst mitgenutzt.

2.3.1 Client im privaten Netzwerk

Die Arbeitsweise von NAT-Routing soll hier anhand eines kleinen Beispiels erläutert werden.

In einem privaten Class C Netzwerk wird im Adressraum 192.168.1.x gearbeitet. Als Übergang zum Internet ist ein NAT-

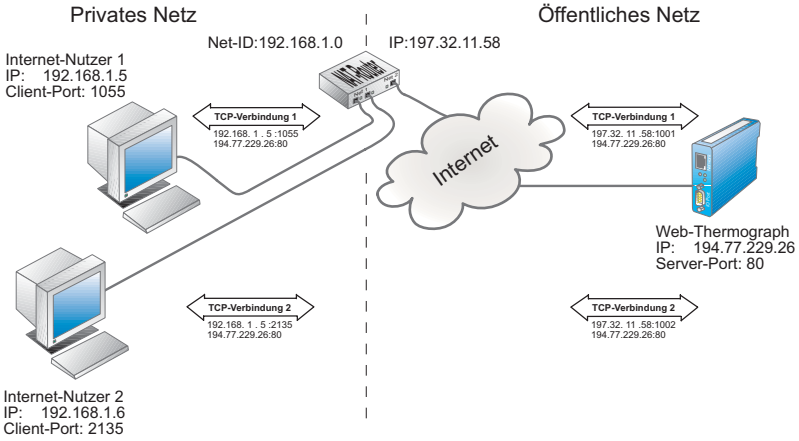
Router im Einsatz, der nach außen mit der IP-Adresse 197.32.11.58 arbeitet.

Der PC mit der netzinternen IP-Adresse 192.168.1.5 baut eine TCP-Verbindung zum W&T Web-Thermographen (IP 194.77.229.26, Port 80) im Internet auf auf und benutzt dazu den lokalen Port 1055.

Ein zweiter PC mit der IP-Adresse 192.168.1.6 baut ebenfalls eine TCP-Verbindung zum Web-Thermographen auf und benutzt dazu den lokalen Port 2135.

Um mit dem Web-Thermographen verbunden zu werden, wenden sich die PCs zunächst an den NAT-Router.

Der NAT-Router wechselt in den TCP/IP-Datenpaketen, die zum Web-Thermographen weitergesendet werden, die IP-Adresse des jeweiligen PCs gegen seine eigene aus. Auch die vom PC vorgegebene Port-Nr. wird gegen eine vom NAT Router verwaltete Port-Nr. ausgetauscht.



Die vergebenen Port-Nr. verwaltet der NAT-Router in einer Tabelle, die folgendermaßen aufgebaut ist:

nach außen	im privaten Netz	
Port-Nr.	zugehörige IP	zugehörige Port-Nr.
1001	192.168.1.5	1055
1002	192.168.1.6	2135

Der Web-Thermograph empfängt also für beide Verbindungen Datenpakete, in denen der NAT-Router als Absender eingetragen ist. Dabei wird aber für jede Verbindung jeweils eine eigene Port-Nr. verwendet.

In alle Datenpakete in Richtung der beiden PCs setzt der Web-Thermograph diese „verbogenen" Adressinformationen ein. Das bedeutet die TCP/IP-Pakete werden so aufgebaut, dass der NAT-Router der Empfänger ist.

Empfängt der NAT-Router ein solches an ihn adressiertes Datenpaket, stellt er mit Hilfe der Zuordnungstabelle fest, wer der tatsächliche Empfänger ist und ersetzt die empfangenen Adressdaten durch die ursprünglichen netzinternen Verbindungsparameter.

Die Zuordnungstabelle für ausgehende Verbindungen (Client im privaten Netz, Server außerhalb) wird dynamisch verwaltet und kann natürlich deutlich mehr als zwei Verbindungen beinhalten. So können beliebig viele Verbindungen nach außen geroutet werden.

2.3.2 Server im privaten Netzwerk

Die andere Richtung (Server im privaten Netz, Client außerhalb) kann natürlich genauso über NAT abgewickelt werden.

Auch hier wird mit Hilfe einer Zuordnungstabelle bestimmt, zu welchem Endgerät und auf welchen Port eingehende Verbindungsanforderungen bzw. Datenpakete geroutet werden sollen.

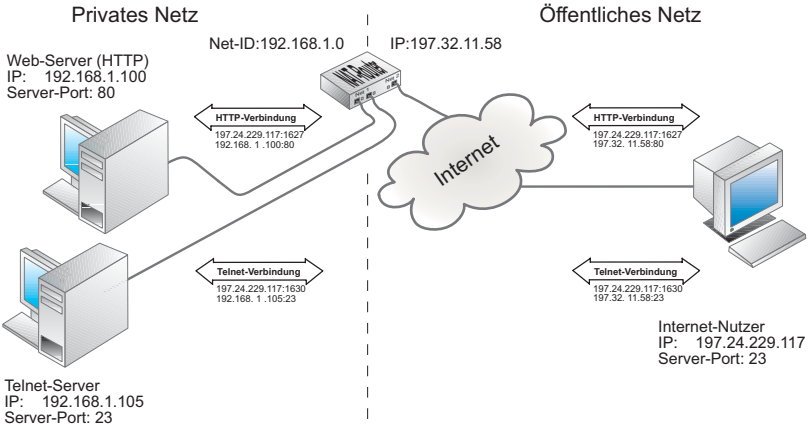
Im Gegensatz zu der Zuordnungsliste für ausgehende Verbindung ist die Serverzuordnungsliste aber statisch und muss vom Administrator angelegt und gepflegt werden.

Für jeden Serverdienst, der aus dem öffentlichen Netz zugänglich sein soll, ist ein Eintrag in der Serverliste nötig.

Sollen von außen z.B. ein Web-Server (HTTP = Port 80) und ein Telnet-Server (Telnet = Port23) erreichbar sein, könnte die Servertabelle so aussehen:

nach außen	im privaten Netz	
Server Port	zugehörige IP	zugehörige Port-Nr
80	192.168.1.100	80
23	192.168.1.105	23

Detaillierte Informationen zu den Protokollen Telnet und HTTP folgen im weiteren Verlauf dieses Buches.



Den Austausch der Verbindungsparameter nimmt der NAT-Router genauso vor, wie bei den im vorhergehenden Abschnitt gezeigten Verbindungen.

In einem privaten Netzwerk, das über einen NAT-Router mit nur einer IP-Adresse zum Internet abgebildet wird, darf natürlich jeder Server-Port nur einmal in der Servertabelle vorkommen. Das bedeutet, dass ein spezieller Serverdienst mit einer spezifischen Port-Nr. nur von einem internen Endgerät angeboten werden kann.

3. Protokolle auf Anwendungsebene

Nachdem im vorangegangenen Kapitel die grundlegenden Protokolle der TCP/IP-Datenübertragung erklärt wurden, soll im Folgenden auf die Anwendungsprotokolle eingegangen werden, die auf diese Basisprotokolle aufsetzen.

Bei den Anwendungsprotokollen unterscheidet man zwischen Hilfsprotokollen und tatsächlichen Anwendungsprotokollen.

Hilfsprotokolle werden für Management- und Diagnosezwecke genutzt und laufen oft für den Anwender unsichtbar im Hintergrund ab.

Zu den Hilfsprotokollen zählen:

- DHCP
- DNS
- DDNS
- DynDNS
- ICMP-Ping
- SNMP
- SYSLOG

Anwendungsprotokolle verrichten eine für den Anwender sofort erkennbare Aufgabe oder können direkt durch den Anwender benutzt werden, schaffen also eine Schnittstelle zum Nutzer.

Im Anschluss an die oben genannten Hilfsprotokolle gehen wir in diesem Kapitel noch auf die folgenden Anwendungsprotokolle näher ein:

- Telnet
- FTP
- TFTP
- HTTP
- SMTP
- POP3

3.1 DHCP - Dynamic Host Configuration Protocol

Zur Erinnerung: Jedes Ethernet-Endgerät hat eine weltweit einmalige Ethernet-Adresse (MAC-Adresse), die vom Hersteller vorgegeben und nicht veränderbar ist. Für den Einsatz in TCP/IP-Netzen vergibt der Netzwerkadministrator dem Endgerät zusätzlich eine zum Netzwerk passende IP-Adresse.

Wird kein DHCP benutzt, werden die IP-Adressen „klassisch" vergeben:

- Bei Geräten, die direkte User-Eingaben erlauben (z.B. PCs), kann die IP-Nummer direkt in ein entsprechendes Konfigurationsmenü eingegeben werden.
- Bei „Black-Box-Geräten" (z.B. Com-Servern) gibt es zum einen das ARP-Verfahren über das Netzwerk, zum anderen besteht die Möglichkeit, die Konfigurationsinformation über eine serielle Schnittstelle einzugeben.

Neben der IP-Adresse müssen als weitere Parameter noch Subnet-Mask und Gateway sowie ggf. ein DNS-Server (mehr dazu im nächsten Kapitel) konfiguriert werden. Bei großen Netzen mit vielen unterschiedlichen Engeräten bringt das allerdings schnell ein hohes Maß an Konfigurations- und Verwaltungsaufwand mit sich.

Mit DHCP wird dem Netzwerkadministrator ein Werkzeug angeboten, mit dem die Netzwerkeinstellungen der einzelnen Endgeräte automatisch, einheitlich und zentral konfigurierbar sind.

Für die Nutzung von DHCP wird im Netzwerk mindestens ein DHCP-Server benötigt, der die Konfigurationsdaten für einen vorgegebenen IP-Adressbereich verwaltet. DHCP-fähige Endgeräte erfragen beim Booten von diesem Server ihre IP-Adresse und die zugehörigen Parameter wie Subnet-Mask und Gateway. DHCP-Server sehen drei grundsätzliche Möglichkeiten der IP-Adresszuteilung und Konfiguration vor:

3.1.1 Vergabe der IP-Adresse aus einem Adresspool

Auf dem DHCP-Server wird ein Bereich von IP-Adressen festgelegt, aus dem einem anfragenden Netzteilnehmer eine zur Zeit nicht benutzte Adresse zugeteilt wird. Die Zuteilung ist bei diesem Verfahren in aller Regel zeitlich begrenzt, wobei die Nutzungsdauer (Lease-Time) vom Netzwerkadministrator festgelegt oder ganz deaktiviert werden kann. Darüber hinaus lassen sich wichtige Daten (Lease-Time, Subnet-Mask, Gateway, DNS-Server usw.) in einem Konfigurationsprofil hinterlegen, das für alle Endgeräte gilt, die aus dem Adresspool bedient werden.

Vorteile

Geringer Administrationsaufwand; Anwender können mit demselben Endgerät ohne Konfigurationsaufwand an verschiedenen Standorten ins Netzwerk.

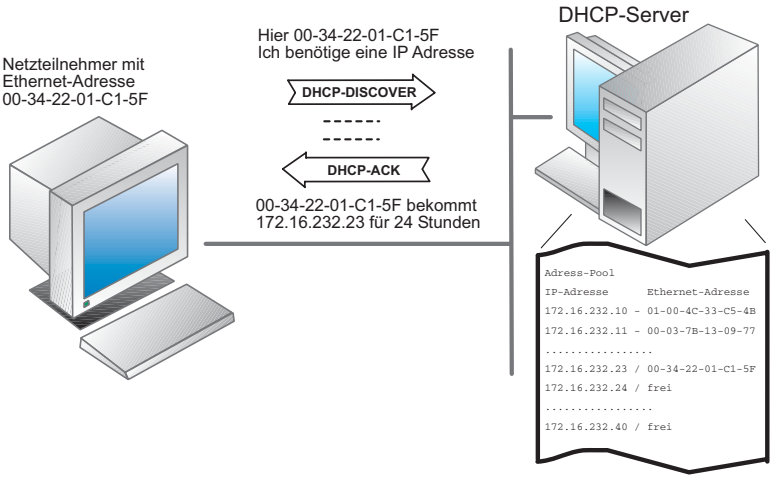
Sofern nicht alle Endgeräte gleichzeitig im Netzwerk aktiv sind, kann die Anzahl der möglichen Endgeräte größer sein als die Zahl der verfügbaren IP-Adressen.

Nachteile

Ein Netzteilnehmer kann nicht anhand seiner IP-Adresse identifiziert werden, da die eindeutige Zuordnung von IP-Adresse und Endgerät verloren geht.

Es kann vorkommen, dass ein Endgerät bei jedem Start eine andere IP-Adresse zugewiesen bekommt.

Beispiel: Typische Fälle für die Vergabe von IP-Adressen aus einem Adresspool sind Universitätsnetzwerke. Hier gibt es Netze mit einer fast unbegrenzten Zahl potentieller Anwender, von denen aber nur jeweils wenige tatsächlich im Netzwerk arbeiten. Dank DHCP haben die Studenten die Möglichkeit, ihr Notebook ohne Konfigurationsänderung von einem Labor ins andere mitzunehmen und im Netzwerk zu betreiben.



3.1.2 Vergabe einer reservierten IP-Adresse

Der Netzwerkadministrator hat die Möglichkeit, einzelne IP-Adressen für bestimmte Endgeräte zu reservieren. Auf dem DHCP-Server wird dazu der IP-Adresse die Ethernet-Adresse des Endgeräts zugeordnet; für jede reservierte IP-Adresse kann außerdem ein individuelles Konfigurationsprofil angelegt werden. Die Angabe einer Lease-Time ist in diesem Fall nicht sinnvoll (aber trotzdem möglich), da die IP-Adresse ohnehin nur vom zugeordneten Endgerät benutzt wird.

Vorteile: Trotz individueller Konfiguration lassen sich alle Netzwerkeinstellungen an zentraler Stelle erledigen und müssen nicht am Endgerät selbst vorgenommen werden.

Endgeräte können gezielt über ihre IP-Adresse angesprochen werden.

Nachteile: Da für jedes Endgerät spezifische Einstellungen angegeben werden müssen, steigt der Administrationsaufwand.

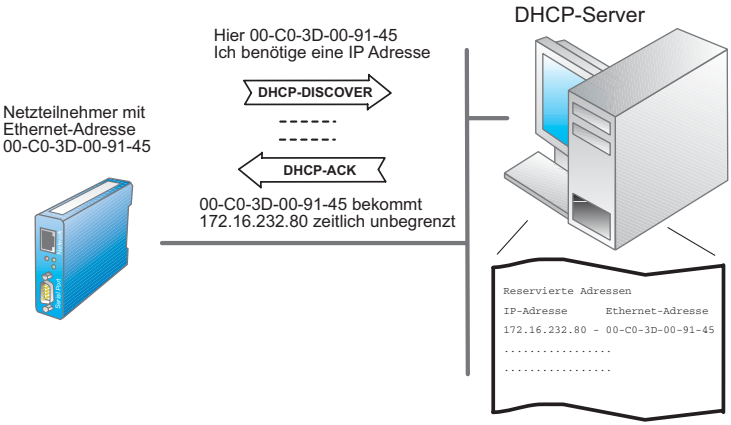
Beim Austausch von Endgeräten muss auf dem DHCP-Server im Konfigurationsprofil

mindestens die Ethernet-Adresse neu eingetragen werden.

Beispiel: Konfiguration von DHCP-fähigen Endgeräten wie Printservern oder Com-Servern, bei denen je nach Einsatzfall eine Adressierung über IP-Adresse benötigt wird. Im DHCP-Manager wird bei der reservierten IP-Adresse die Ethernet-Adresse des zugehörigen Entgerätes eingetragen; die Lease-Time sollte deaktiviert sein. Beim Com-Server können als zusätzliche Parameter Subnet-Mask und Gateway (Router) angegeben werden.

DHCP-Server unter Windows 2000 vergeben auch auf BootP-Anfragen hin IP-Adressen aus dem normalen Adress-Pool. Diese Eigenschaft lässt sich aber deaktivieren, was Ihr Netzwerkadministrator unbedingt tun sollte!

Hierzu muss ergänzend gesagt werden, dass viele Endgeräte auch das ältere BootP-Protokoll nutzen, um ihre Konfiguration zu erfragen. BootP ist ein Vorläufer von DHCP und wird von DHCP-Servern unterstützt. Allerdings kann BootP nur mit reservierten IP-Adressen arbeiten.



Bei „Black-Box-Geräten“ wie dem Com-Server sollte das BootP-Protokoll eingesetzt werden, um in jedem Fall die Übergabe einer reservierten IP-Adresse zu erzwingen. Ist beim DHCP-Server kein zur Ethernet-Adresse des Com-Server passender Eintrag vorhanden, wird die BootP-Anfrage ignoriert und der Com-Server behält die aktuell eingestellte IP-Adresse.

3.1.3 Ausschluss bestimmter IP-Adressen aus der DHCP-Konfiguration

Für Endgeräte, die weder DHCP- noch BootP-fähig sind, hat der Netzwerkadministrator die Möglichkeit, einzelne IP-Adressen oder auch ganze Adressbereiche von der Vergabe durch DHCP auszuschließen.

Die Konfiguration muss in diesem Fall entweder am Endgerät selbst vorgenommen werden oder durch den Einsatz mitgelieferter Tools erfolgen.

Nachteil: Uneinheitliche und ggf. dezentrale Konfiguration; höherer Administrationsaufwand ist erforderlich.

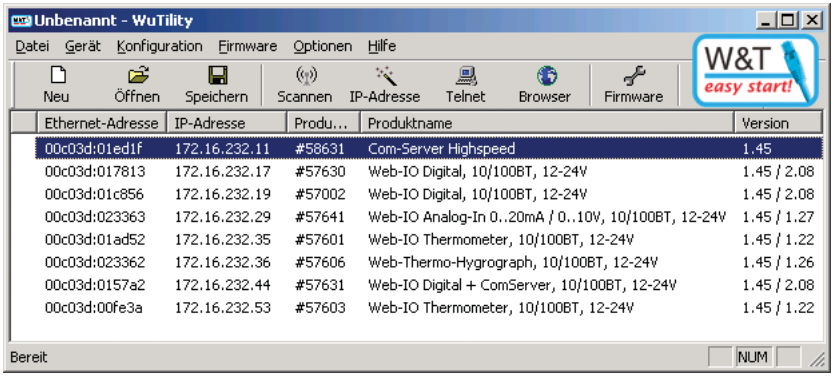
Beispiel: PCs mit älteren DOS-Versionen oder ältere Printserver sind nicht DHCP-fähig und müssen auf jeden Fall „von Hand" konfiguriert werden.

Alle drei Verfahren können in Netzwerken mit DHCP-Unterstützung nebeneinander angewandt werden.

Natürlich gibt es auch Sonderfälle, in denen es sinnvoll ist, auf DHCP zur Adressvergabe zu verzichten. In technischen Anwendungen gilt es oft neben der Vergabe der IP-Adressdaten noch weitere gerätespezifische Einstellungen vorzunehmen, die ohnehin nicht von DHCP unterstützt werden.

Hier bieten die vom Hersteller mitgelieferten Softwarewerkzeuge in vielen Fällen mehr Komfort als DHCP.

W&T bietet dem Anwender zum Beispiel mit dem Wutility Tool ein Werkzeug zur einfachen Inbetriebnahme, Inventarisierung, Wartung und Verwaltung von Com-Servern und Web-IO Geräten.



Natürlich können auch solche W&T Endgeräte, die Ihre IP-Adresse über DHCP bekommen haben, mit Wutility verwaltet werden.

3.1.4 DHCP und Router

Der Informationsaustausch zwischen Endgeräten und DHCP-Servern erfolgt auf physikalischer Ebene in Form von UDP-Broadcasts (Rundrufen ins Netz). Erstreckt sich die DHCP-Konfiguration über mehrere Subnetze, sollte bei der Auswahl geeigneter Router darauf geachtet werden, dass diese auch DHCP-Broadcasts weiterleiten.

3.2 DNS – das Domain Name System

Das Domain Name System ist das Adressbuch des Internet. Obwohl es vom Anwender nur im Hintergrund genutzt wird, ist es doch einer der wichtigsten Internetdienste.

Auf IP-Ebene werden die Millionen von Teilnehmern im Internet über IP-Adressen angesprochen. Für den Nutzer wäre der Umgang mit IP-Adressen aber schwierig: Wer kann sich schon merken, dass das Web-Thermometer von W&T unter der IP-Adresse 194.77.229.26 zu erreichen ist? Einen aussagekräftigen Namen, wie *klima.wut.de,* kann man sich dagegen viel leichter merken.

Schon in den Anfängen des Internet trug man dem Bedürfnis Rechnung, IP-Adressen symbolische Namen zuzuordnen: auf jedem lokalen Rechner wurde eine *Hosts*-Tabelle gepflegt, in der die entsprechenden Zuordnungen hinterlegt waren. Der Nachteil bestand jedoch darin, dass eben nur diejenigen Netzwerkteilnehmer erreichbar waren, deren Namen in der lokalen Liste standen. Zudem nahmen diese lokalen Listen mit dem rapiden Wachstum des Internet bald eine nicht mehr handhabbare Größe an. Man stand also vor der Notwendigkeit, ein einheitliches System zur Namensauflösung zu schaffen. Aus diesem Grund wurde 1984 der DNS-Standard verabschiedet, an dem sich bis heute kaum etwas geändert hat.

Das Prinzip ist einfach. Die Zuordnung von IP-Adressen und Domainnamen wird auf sogenannten DNS-Servern hinterlegt und dort bei Bedarf „angefragt“. Doch ehe wir hier in die Details gehen, noch einige Anmerkungen zum Aufbau von Domainnamen:

3.2.1 Domainnamen

Das DNS sieht eine einheitliche Namensvergabe vor, bei der jeder einzelne Host (Teilnehmer im Netz), Teil mindestens einer übergeordneten „Top-Level-Domain“ ist.

Als Top-Level-Domain bietet sich ein länderspezifischer Domainname an:

- *.de* für Deutschland
- *.at* für Österreich
- *.ch* für Schweiz usw.

Die Domain kann aber auch nach Inhalt bzw. Betreiber gewählt werden:

- *.com* für kommerzielle Angebote
- *.net* für Netzbetreiber
- *.edu* für Bildungseinrichtungen
- *.gov* ist der US-Regierung vorbehalten
- *.mil* ist dem US-Militär vorbehalten
- *.org* für Organisationen

Alle untergeordneten (Sub-Level-) Domainnamen können vom Betreiber selbst gewählt werden, müssen in der übergeordneten Domain aber einmalig sein. Für jede Top-Level-Domain gibt es eine selbstverwaltende Institution, bei der die Sub-Level-Domains beantragt werden müssen und die damit eine Mehrfachvergabe ausschließt. Für die *de*-Domain ist in solchen Fragen die DENIC (*De*utsches *N*etwork *I*nformation *C*enter; *http://www.denic.de*) zuständig.

Ein Beispiel: *klima.wut.de* setzt sich zusammen aus:

- *de* für Deutschland als Top-Level-Domain
- *wut* für Wiesemann und Theis als Sub-Level-Domain
- *klima* für das Web-Thermometer in der Domain *wut.de*

Der gesamte Domainname darf maximal 255 Zeichen lang sein, wobei jeder Subdomainname höchstens 63 Zeichen umfassen darf. Die einzelnen Subdomainnamen werden mit Punkten getrennt. Eine Unterscheidung zwischen Groß- und Kleinschreibung gibt es nicht. *WWW.WUT.DE* führt Sie genauso auf die Homepage von W&T wie *www.wut.de* oder *www.WuT.de*.

3.2.2 Namensauflösung im DNS

Wie bereits angesprochen, werden auf DNS-Servern (auch Nameserver genannt) Listen mit der Zuordnung von Domainnamen und IP-Adresse geführt. Gäbe es bei den heutigen Ausdehnungen des Internets nur einen einzigen DNS-Server, wäre

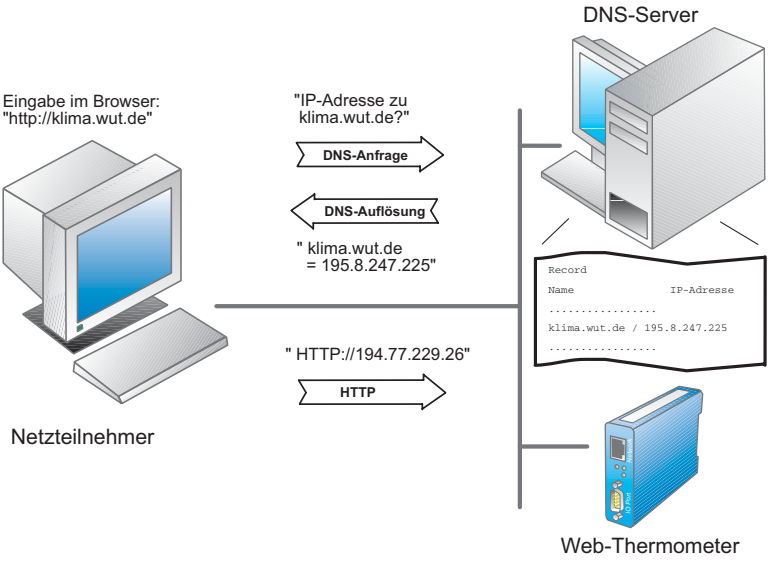
dieser vermutlich mit der immensen Zahl der DNS-Anfragen hoffnungslos überfordert. Aus diesem Grund wird das Internet in Zonen aufgeteilt, für die ein bzw. mehrere DNS-Server zuständig sind.

Netzteilnehmer, die das DNS nutzen möchten, müssen in ihrem TCP/IP-Stack die IP-Adresse eines in Ihrer Zone liegenden DNS-Servers angeben. Um auch bei Ausfall dieses Servers arbeiten zu können, verlangen die üblichen TCP/IP-Stacks sogar die Angabe eines zweiten DNS-Servers.

Welcher DNS-Server für den jeweiligen Netzteilnehmer zuständig ist, erfährt man beim Provider bzw. beim Netzwerkadministrator.

Um Domainnamen in IP-Adressen auflösen zu können, verfügen heutige TCP/IP-Stacks über ein Resolver-Programm. Gibt der Anwender anstatt einer IP-Adresse einen Domainnamen an, startet das Resolver-Programm eine Anfrage beim eingetragenen DNS-Server. Liegt dort kein Eintrag für den gesuchten Domainnamen vor, wird die Anfrage an den in der Hierarchie nächsthöheren DNS-Server weitergegeben. Dies geschieht so lange, bis die Anfrage entweder aufgelöst ist oder festgestellt wird, dass es den angefragten Domainnamen nicht gibt.

Die zum Domainnamen gehörende IP-Adresse wird von DNS-Server zu DNS-Server zurückgereicht und schließlich wieder dem Resolver–Programm übergeben. Der TCP/IP-Stack kann nun die Adressierung des Zielteilnehmers ganz normal über dessen IP-Adresse vornehmen.



Die Zuordnung von IP-Adresse und Domainnamen wird vom TCP/IP-Stack in einem Cache hinterlegt. Diese Cache-Einträge sind dynamisch: Wird der hinterlegte Netzteilnehmer für bestimmte Zeit nicht angesprochen, löscht der Stack den Eintrag wieder. Das hält den Cache schlank und macht es möglich, die zu einem Domainnamen gehörende IP-Adresse bei Bedarf auszutauschen.

3.2.3 DNS in Embedded-Systemen

Embedded-Systeme bieten in aller Regel nicht die Möglichkeit, am Gerät selbst einen Domainnamen einzugeben.

Das ist auch gar nicht nötig, denn das Endgerät muss seinen eigenen Namen gar nicht wissen. Vielmehr wird die Zuordnung von Name und IP-Adresse auch hier auf dem DNS-Server festgehalten. Soll z.B. von einem Client eine Verbindung auf ein als Server arbeitendes Embedded-System aufgebaut werden, erfragt der Client die zum Namen gehörende IP-Adresse wie gehabt beim DNS-Server.

Da Embedded-Syteme aber häufiger in „Maschine-Maschine-Verbindungen“ als in „Mensch-Maschine-Verbindungen“ arbei-

ten, kann eine direkte Adressierung über IP-Adresse hier effizienter sein, da die Zeit für die DNS-Auflösung entfällt.

Die Adressierung über Namen ist bei Embedded-Systemen nur dann sinnvoll, wenn entweder nur der Name bekannt ist (z.B. E-Mail-Adressen) oder mit einem „Umzug“ eines Servers (Name bleibt, IP-Adresse ändert sich) gerechnet werden muss (z.B. Webserver).

3.2.4 DDNS - dynamisches DNS in Verbindung mit DHCP

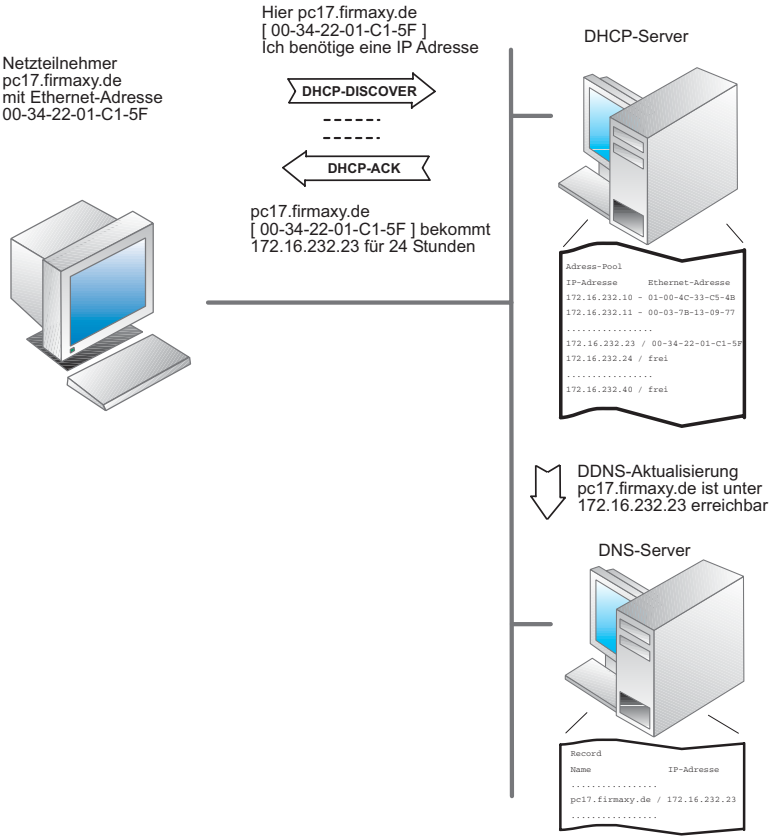
Zusammengefasst kann man sagen: DNS ist eine Art Telefonbuch für’s Netzwerk. Nun hat DNS in seiner Urform die gleichen Nachteile wie ein Telefonbuch. Ändert sich die Telefonnummer eines Teilnehmers, nachdem das Buch gedruckt wurde, kann der Teilnehmer mit Hilfe dieses nun veralteten Telefonbuches nicht mehr erreicht werden.

Die Zuordnungen in DNS-Servern werden natürlich regelmäßig aktualisiert und nicht nur einmal pro Jahr erneuert. Wird aber mit dynamischen IP-Adressen gearbeitet, die mittels DHCP vergeben werden, macht DNS nur Sinn, wenn eine ständige Korrektur der DNS-Listen betrieben wird.

Die Technik des automatischen Abgleiches zwischen DHCP-Server und DNS-Server wird als DDNS - dynamisches DNS bezeichnet.

DDNS ist kein Standard TCP/IP Dienst.

Auf welchem Weg und in welcher Form die Synchronisation zwischen DHCP-Server und DNS-Server erfolgt, hängt davon ab, unter welchem Betriebssystem die Server laufen.



Der prinzipielle DDNS-Ablauf bei Vergabe einer IP-Adresse via DHCP verläuft folgendermaßen:

1. Das Endgerät versucht vom DHCP-Server eine IP-Adresse zu beziehen. Dabei ist der Host-Name des Gerätes (hier pc17.firmaxy.de) im Endgerät fest konfiguriert.
2. Der DHCP-Server vergibt eine IP-Adresse aus seinem Adress-Pool an das Endgerät und trägt die Zuordnung zur Ethernet-Adresse in die Adressverwaltung ein.
3. Zusätzlich übergibt der DHCP-Server dem DNS-Server IP-Adresse und Host-Namen des Endgerätes.
4. Der DNS-Server aktualisiert die Namensverwaltung mit dem neuen Eintrag.

Bei dem gezeigten Ablauf spielt es keine Rolle, ob DNS-Server und DHCP-Server auf zwei getrennten Rechnern oder auf einer gemeinsamen Hardware laufen.

Da die DDNS-Kopplung vom Netzwerkadministrator eingerichtet werden muss, kommt DDNS nur in abgeschlossenen Netzen wie z.B. Firmen-Netzen zum Einsatz.

3.2.5 DynDNS

Nicht nur in lokalen Netzen, in denen die IP-Adress Vergabe der DHCP -Server abwickelt, wird mit dynamischen IP-Adressen gearbeitet.

Zur Erinnerung: In Netzen, die miteinander verbunden sind, man spricht auch von WAN (Wide Area Network), muss jedes angeschlossene Endgerät eine einmalige IP-Adresse haben.

Diese Regel gilt auch für das Internet, welches den mit Abstand größten Netzwerkverbund darstellt.

Da die Anzahl der verfügbaren IP-Adressen begenzt ist, die Anzahl der Internetnutzer aber stetig wächst, werden auch im Internet dynamische IP-Adressen vergeben.

Der größte Teil der Internet-Nutzer ist nur zeitweise über einen Zugang bei einem Provider mit dem Internet verbunden. Ähnlich wie bei DHCP teilt der Provider dem verbundenen Endgerät für die Dauer der Nutzung eine IP-Adresse zu. Diese IP-Adresse wird voraussichtlich bei jeder Internet-Nutzung eine andere sein.

Da die meisten Internet-Nutzer nur Server-Dienste (E-Mail, Abruf von Webseiten, ...) in Anspruch nehmen, also Verbindungen zu diesen Servern aufnehmen, ist das kein Problem.

Soll aber das Endgerät des Internet-Nutzers (meist ein PC) auch für andere Internet-Nutzer erreichbar sein, ist die dynamische IP-Adresse ein Problem, da die aktuell zugeteilte IP-Adresse ja nur dem Provider und dem dort angekoppelten Endgerät bekannt ist.

Um dieses Problem zu umgehen gibt es zwei Möglichkeiten:

1. Permanenter Anschluss an das Internet
Feste Internet-Zugänge mit einer festen IP-Adresse sind aber ungleich teurer als z.B. normale DSL- oder Modem-Zugänge. Diese Lösung bietet sich deshalb nur für größere Firmen an.

2. Verwendung von DynDNS
DynDNS ist ein z.Zt. kostenloser Dienst, bei dem sich der Anwender einen weltweit einmaligen Hostnamen registrieren lassen kann.

Dabei ist aber nur der Hostname individuell zu bestimmen. Als Domain-Namen bietet DynDNS einige Alternativen wie z.B. homeip.net oder dynalias.org und viele mehr an.

Ein registrierter Name könnte dann z.B. so aussehen:

meinefirma.homeip.net

Um dyndns benutzen zu können, ist es nötig, zunächst ein Konto bei www.dyndns.org anzulegen.

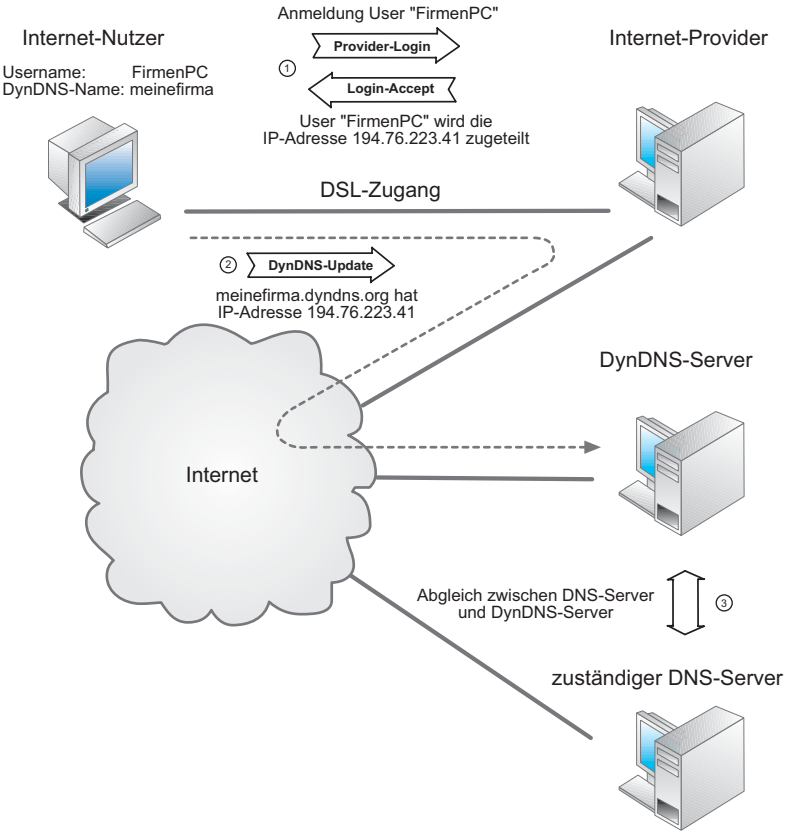
Eine detaillierte Beschreibung der Vorgehensweise ist auf den Webseiten von DynDNS unter *http://www.dyndns.org* verfügbar.

Die Abwicklung der Adressauflösung mittels DynDNS erfolgt in drei Schritten.

1.) Der Internet-User stellt z.B. via DSL eine Verbindung zu seinem Internet-Provider her und bekommt nach erfolgreichem Login eine IP-Adresse zugeteilt.
2.) Im Gegensatz zu DDNS muss der Anwender bzw. sein Endgerät dafür sorgen, dass DynDNS weiß, unter welcher IP-Adresse das Endgerät erreichbar ist. Dazu nutzt das Endgerät den DynDNS-Update-Client. Für PC gibt es entsprechende Programme, die diese Aufgabe übernehmen. Andere Endgeräte müssen spezielle Funktionen integriert haben.

3.) Erfolgt nun bei einem DNS-Server die Anfrage nach dem vom Internet-User benutzten DynDNS-Namen und der zugehörigen IP-Adresse, fragt der zuständige DNS-Server diese beim DynDNS-Server an und gleicht seine Daten ab.

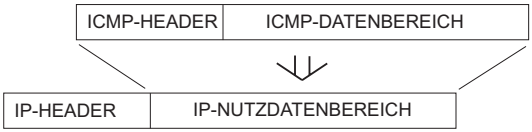
Damit ist das Endgerät unter dem gewählten Namen weltweit ansprechbar, kann also auch Server-Dienste anbieten.



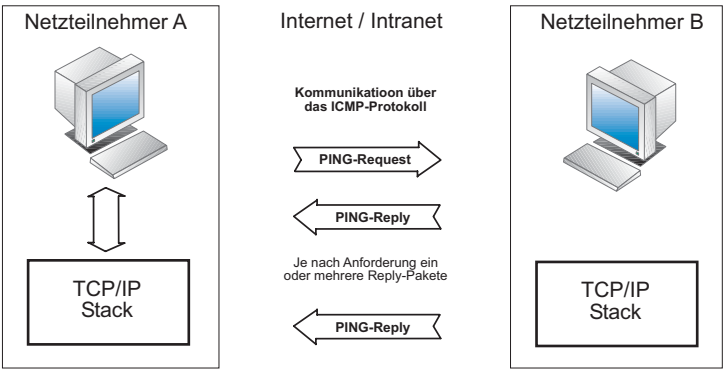
3.3 Ping – Erreichbarkeit prüfen

Die Ping-Funktion dient in TCP/IP-Netzen zu Diagnosezwecken. Mit Hilfe von Ping lässt sich überprüfen, ob ein bestimmter Teilnehmer im Netz existiert und tatsächlich ansprechbar ist.

Ping arbeitet mit dem ICMP-Protokoll, welches auf das IP-Protokoll aufsetzt.



Setzt ein Netzteilnehmer durch Eingabe des Ping-Kommandos einen ICMP-Request ab, gibt die angesprochene Station einen ICMP-Reply an den Absender zurück.



Der Aufruf des Kommandos *PING <IP-Adresse>* in der DOS-Box fordert den durch die IP-Adresse angegeben Netzteilnehmer auf, eine Rückmeldung zu geben.

Zusätzlich können noch diverse Parameter angegeben werden:

-t
Wiederholt das Ping-Kommando in Dauerschleife, bis der Anwender mit <Strg> C unterbricht.

-n count
Wiederholt das Ping-Kommando „count“ mal.

-l size
„size“ gibt an, mit wieviel Byte das ICMP-Packet aufgefüllt wird. Bei Com-Servern in Default-Einstellung sind dies maximal 512 Byte.

-w timeout
„timeout“ spezifiziert, wie lange (in Millisekunden) auf die Rückmeldung gewartet wird.

Ein Beispiel:

```
PING 172.16.232.49 -n 50
```

sendet 50 Ping-Kommandos an die Station 172.16.232.49.
Ist der Netzteilnehmer vorhanden, erscheint folgende Rückmeldung:

```
Reply from 172.16.232.49: bytes=32 time=10ms TTL=32
```

Bleibt die Rückmeldung aus, wird folgende Meldung zurückgegeben:

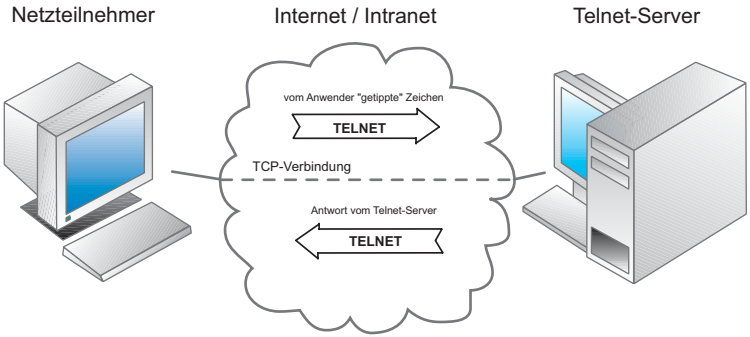
```
Request timed out.
```

Anstelle der IP-Adresse kann natürlich auch ein Host-Name eingegeben werden. Die Voraussetzung hierzu ist der Zugang zu einem DNS-Server.

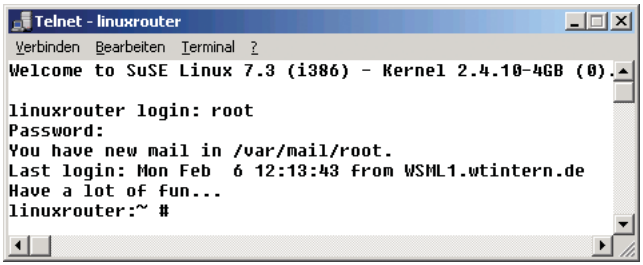
Die von Ping verwendeten ICMP-Pakete sind im Internet-Standard RFC-792 definiert.

3.4 Telnet - Terminal over Network

Einfach ausgedrückt ist Telnet ein Textfenster bzw. text-orientiertes Programm, über das ein anderer Rechner (Host) im Netzwerk vom Anwender fernbedient werden kann.



Eine Telnet-Sitzung kann man sich vorstellen wie eine DOS-Box, allerdings werden die eingetippten Befehle auf dem entfernten Rechner ausgeführt.



Dafür werden mehrere Elemente benötigt.

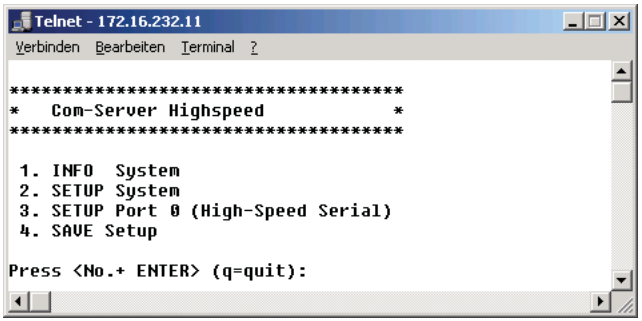
3.4.1 Der Telnet Client

Alle modernen Betriebssysteme verfügen heute über ein Telnet-Clientprogramm.

Der Telnet-Client baut eine TCP-Verbindung zu einem Telnetserver auf, nimmt Tastatureingaben vom Anwender entgegen, gibt sie an den Telnetserver weiter und stellt die vom Server gesendeten Zeichen auf dem Bildschirm dar.

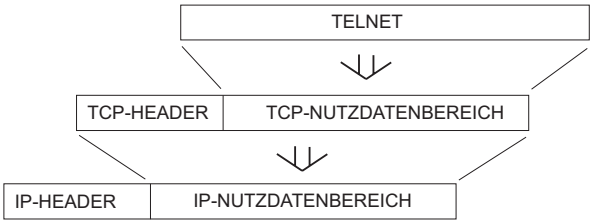
3.4.2 Der Telnet-Server

ist auf dem entfernten Rechner aktiv und gibt einem oder ggf. mehreren Nutzern die Gelegenheit sich dort „einzuloggen“. Damit ist der Telnet-Server (in Unix Systemen auch oft als Telnet-Daemon bezeichnet) das Bindeglied zwischen Netzwerkzugang via Telnet-Client und dem zu bedienenden Prozess. In seinem Ursprung wurde Telnet eingesetzt, um einen Remote-Zugang zu UNIX-Betriebssystemen zu schaffen. Es verfügen auch viele Embedded-Systeme wie Com-Server oder Printer-Server, Switches, Hubs und Router über einen Telnet-Server, der als Konfigurationszugang dient.



3.4.3 Das Telnet Protokoll

Auch Telnet setzt auf TCP als Basisprotokoll auf.



Hierbei wird, wenn vom Anwender nicht anders vorgegeben, der Port 23 genutzt. Es kann aber auch jeder beliebige andere Port angegeben werden. Wichtig ist, dass auf dem gewählten Port ein Telnet-Server aktiv ist.

Das Telnet Protokoll übernimmt im Wesentlichen drei Aufgaben:

1. Festlegung benutzter Zeichensätze und Steuercodes zur Cursorpositionierung

   Als gemeinsame Basis für Client und Server wird hierzu der NVT-Standard „Network Virtual Terminal“ eingesetzt. NVT benutzt den 7Bit ASCII Zeichensatz und legt fest, welche Zeichen dargestellt werden und welche zur Steuerung und Positionierung genutzt werden.

2. Aushandeln und Einstellen von Verbindungsoptionen

   Über die Festlegungen im NTV hinaus kann Telnet von einer Vielzahl spezieller Funktionen Gebrauch machen. Das Telnet-Protokoll gibt Client und Server die Möglichkeit Verbindungsoptionen auszuhandeln. Zum Beispiel: ob der Server alle vom Client empfangenen Zeichen als Echo zurückgeben soll.
   Hierzu werden Steuerzeichen benutzt, bei denen das 8.Bit gesetzt ist, also Zeichen oberhalb 127 und damit außerhalb des NTV-Zeichensatzes.

3. Transport der Zeichen, die zwischen Client und Server ausgetauscht werden

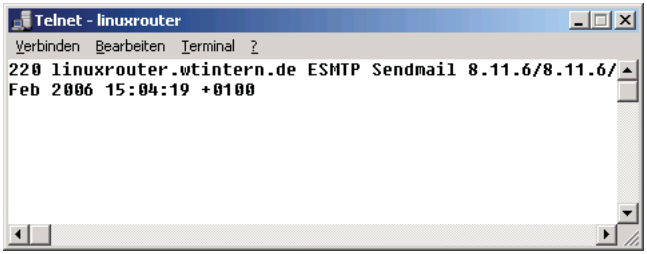
   Alle vom Anwender eingegebenen oder vom Server gesendeten Zeichen des NTV Zeichensatzes werden 1:1 in den Nutzdatenbereich eines TCP-Paketes gepackt und übers Netzwerk transportiert.

Die Einfachheit des Telnetprotokolls sowie die Transparenz bei der Zeichenübertragung, haben Telnet auch zu einem beliebten Diagnosetool gemacht. So lassen sich Verbindungen zu http, SMTP oder POP3 Servern herstellen.

Es lässt sich zum Beispiel durch Eingabe der folgenden Zeile in einer Dosbox überprüfen, ob der SMTP-Server (Port25) arbeitet:

```
telnet <IP-Addresse eines Mail-Servers> 25
```

Ist der SMTP-Server aktiv, wird eine Begrüßungsmeldung zurückgegeben.



Durch konsequentes Eintippen des SMTP-Protokolls könnte man nun theoretisch per Telnet-Client E-Mails verschicken.

Auch andere einfache Protokolle wie http oder POP3 lassen sich via Telnet-Client per Hand nachvollziehen.

3.5 FTP - File Transfer Protocol

In einfachen Worten ausgedrückt, erlaubt FTP einem Anwender im Netzwerk den Zugriff auf das Datei-System, bzw. die Festplatte eines entfernten Rechners.

Eine der Hauptanwendungen für FTP ist heute das Aufspielen von HTML-Seiten auf WWW-Server, die zu diesem Zweck auch immer einen FTP-Zugang haben.

FTP kann aber auch genutzt werden, um über embedded FTP-Clients, wie zum Beispiel den W&T Com-Server, serielle Daten von Endgeräten in eine Datei auf dem Server zu speichern.

Ein weiteres Anwendungsfeld ist das Data-Logging (zyklisches Abspeichern von Datensätzen) via FTP. Auf diesem Weg kann z.B. ein Web-Thermograph die Werte für Temperatur und Luftfeuchte in vorgegebenen Abständen mit Zeitstempel in eine Datei auf dem FTP-Server schreiben.

3.5.1 Der FTP-Client

FTP arbeitet nach dem Client/Server Prinzip. Ein FTP-Client ist heute Bestandteil jedes Betriebssystems. Unter Windows z.B. wird durch Eingabe des FTP-Befehls in einer Dosbox der FTP-Client gestartet.

Mit dem OPEN-Kommando, gefolgt von der IP-Adresse bzw. dem Hostnamen des FTP-Servers, wird die FTP Verbindung geöffnet und der Nutzer muss seinen Login-Namen und ein Passwort eingeben. Nach erfolgreichem Login, sind je nach Zugriffsrecht unter anderem folgende Dateioperationen möglich:

	FTP Befehl
Speicher von Dateien auf dem Server	PUT
Laden von Dateien vom Server	GET
Daten an eine bestehende Datei anhängen	APPEND
Löschen von Dateien auf dem Server	DELETE
Anzeigen des Verzeichnisinhaltes	DIR

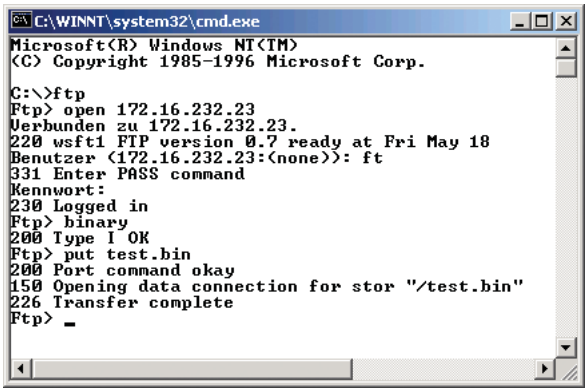
Eine Auflistung aller unterstützten Kommandos erhält man mit der Eingabe eines “?“ hinter dem FTP-Prompt. Eine kurze Be-

schreibung der einzelnen Kommandos kann mit „? Kommando“ abgerufen werden.

Eine wichtige Eigenschaft von FTP ist die unterschiedliche Handhabung von Text- und Binärdateien. Um die gewünschte Betriebsart auszuwählen, stellt FTP zwei weitere Kommandos zur Verfügung:

	FTP Befehl
für die Übertragung von Textdateien	ASCII
für die Übertragung von Binärdateien (z.B. ausführbare Programmdateien)	BINARY

Nach der Eingabe von FTP findet die Bedienung in einer Art Dialog statt, wie hier beispielhaft für das Speichern der Datei „test.bin“ auf dem Server „172.16.232.23“ gezeigt.



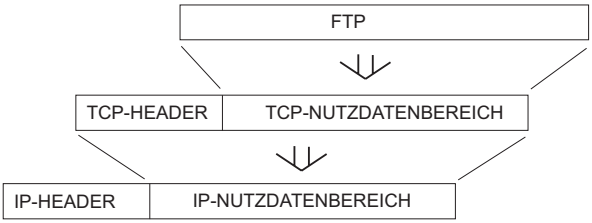
Je nach Betriebssystem können sowohl die Bedienung als auch die Kommandos des FTP-Client variieren.

In Unix-Betriebssystemen ist außerdem strikt auf Groß- und Kleinschreibung zu achten.

Eine komfortablere Handhabung von FTP lässt sich durch den Einsatz von zugekauften FTP-Client Programmen mit grafischer Benutzeroberfläche erreichen.

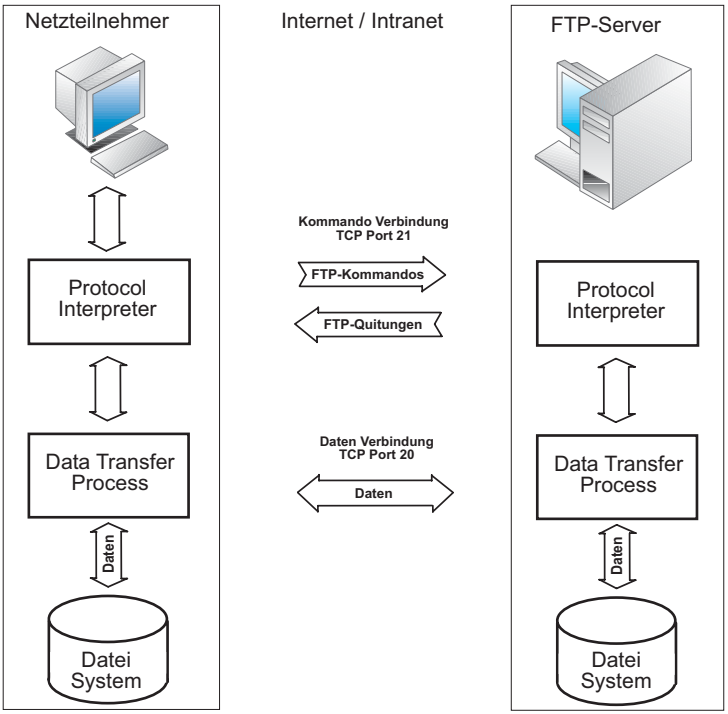
3.5.2 Das FTP-Protokoll

Als Basis-Protokoll setzt FTP auf das verbindungsorientierte und gesicherte TCP auf.



Im Gegensatz zu anderen Internetdiensten nutzt FTP aber zwei TCP-Verbindungen und damit zwei TCP-Ports:

- Port 21 als Kommando-Verbindung
- der zweite Port wird für die Übertragung der Dateien benutzt. Die verwendete Portnummer wird ausgehandelt.



Die Steuerung des Datei-Transfers zwischen Client und Server wird über einen Kommandodialog gesteuert. Diesen Part wikkeln die Protocol-Interpreter über die Kommando-Verbindung ab. Die Kommando-Verbindung bleibt für die gesamte Dauer der FTP-Sitzung bestehen.

Der eigentliche Datei-Transfer erfolgt über die Daten-Verbindung, die vom Data Transfer Prozess für jede Dateioperation neu geöffnet wird. Der Data Transfer Prozess ist dabei das Bindeglied zwischen Netzwerk und Dateisystem und wird vom Protocol-Interpreter gesteuert.

3.5.3 Der FTP-Server

Ein FTP-Server steht in der Regel nur bei Server-Betriebssystemen zur Verfügung und muss ggf. erst gestartet werden.

FTP-Server bieten zwei Zugriffsmöglichkeiten:

1. Nur eingetragene Nutzer haben Zugriff und können, je nach in einer Userliste festgehaltenem Zugriffsrecht, Dateioperationen ausführen.
2. Jeder Nutzer kann auf den Server zugreifen. Ein Login findet entweder gar nicht statt oder es wird der Username „anonymous“ angegeben. Man spricht dann von Anonymous-FTP.

3.6 TFTP - Trivial File Transfer Protocol

Neben FTP ist TFTP ein weiterer Dienst, um übers Netzwerk auf die Dateien eines entfernten Rechners zugreifen zu können.

TFTP ist allerdings sowohl vom Funktionsumfang als auch von der Größe des Programmcodes deutlich „schlanker“ als FTP.

Ein TFTP-Client ist nicht unbedingt Bestandteil des Betriebssystems und z.B. im Windows-Umfeld nur in Windows NT und Windows 2000 implementiert.

TFTP-Server kommen im Officebereich selten zum Einsatz.

Besonders geeignet ist TFTP für den Einsatz in Embedded Systemen, in denen nur ein begrenzter Speicherplatz für Betriebssystemkomponenten zur Verfügung steht. TFTP bietet hier bei minimalem Programmcode ein hohes Maß an Effizienz.

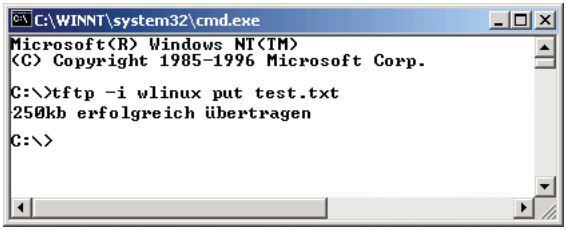
In Com-Servern, Printerservern und Miniterminals wird beispielsweise TFTP genutzt, um Konfigurations- und Firmware-Dateien zu übertragen.

TFTP stellt nur zwei Dateioperationen zur Verfügung:

	TFTP Befehl
Speicher von Dateien auf dem Server	PUT
Laden von Dateien vom Server	GET

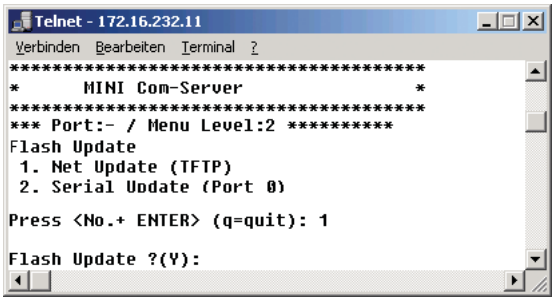
Wie auch FTP unterscheidet TFTP zwischen der Übertragung von Text- und Binär-Dateien. Sollen Binär-Dateien übertragen werden, wird dies durch den zusätzlichen Parameter „-i“ angegeben.

Hier als kurzes Beispiel: Die binäre Datei „test.txt“ wird von einem Windows NT Rechner auf den Server wlinux gespeichert.



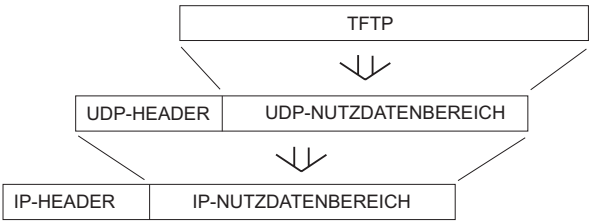
Auf eine Authentifizierung, also ein Login mit Passwortabfrage wie bei FTP, wird verzichtet.

Eine Möglichkeit, dennoch unerwünschte Zugriffe auszuschließen, möchten wir am Beispiel des Com-Servers zeigen. Um einem Com-Server via TFTP eine neue Firmware einzuspielen, muss zunächst der TFTP Zugang über Telnet bzw. den Web-Browser freigeschaltet werden.



Darüber hinaus wird geprüft, ob es sich bei den empfangenen Daten tatsächlich um Com-Server Firmware handelt.

Im Gegensatz zu FTP verwendet TFTP als Basis Protokoll UDP, wobei der Port 69 genutzt wird.

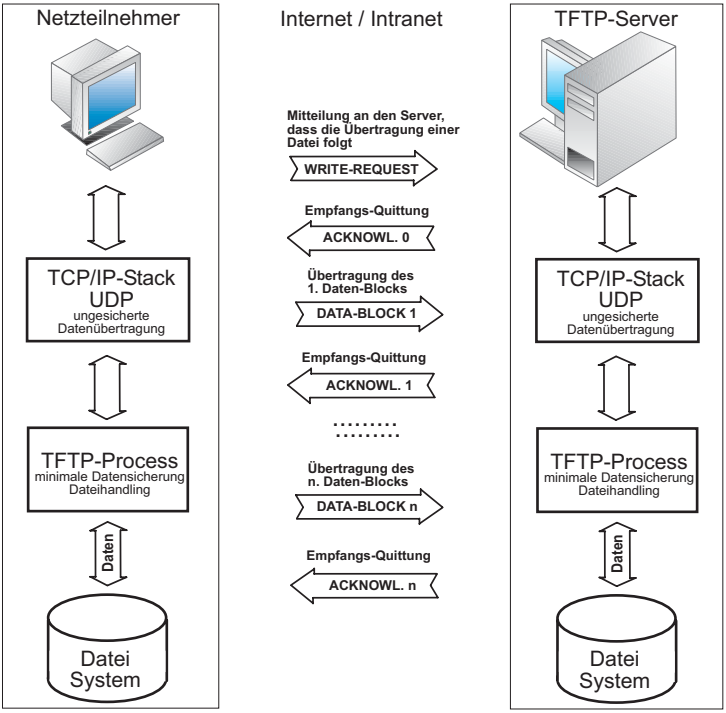


Zur Erinnerung:

UDP arbeitet verbindungslos. Man spricht bei UDP-Paketen auch von Datagrammen, wobei jedes Paket als eigenständige Datensendung behandelt wird. Auf UDP-Ebene werden empfangene Pakete nicht quittiert. Der Sender erhält keine Rückmeldung, ob ein gesendetes Paket wirklich beim Empfänger angekommen ist. UDP-Pakete bekommen keine Sequenz-Nummer. Ein Empfänger, der mehrere UDP-Pakete erhält, hat keine Möglichkeit festzustellen, ob die Pakete in der richtigen Reihenfolge empfangen wurden.

Aus diesem Grund übernimmt TFTP die Sicherung der übertragenen Daten selbst.

Die Übertragung von Dateien geschieht in Blöcken von je 512 Bytes, wobei die Blöcke mit einer laufenden Nummer versehen werden. Jeder empfangene Block wird von der Gegenseite quittiert. Erst nach Empfang der Quittung wird der nächste Block gesendet.



TFTP erkennt, ob die empfangenen Datenblöcke in Ordnung sind; eine Fehlerkorrektur gibt es aber nicht. Geht bei der Übertragung etwas schief, stimmt etwa die Paketlänge nicht oder ein komplettes Paket geht verloren, wird die Übertragung abgebrochen. In diesem Fall kann der Anwender oder eine intelligente Anwendungssoftware den Vorgang erneut starten.

3.7 SNMP – Simple Network Management Protocol

Gerade in technischen Anwendungen verbindet das Netzwerk eine Vielzahl verschiedener Endgeräte unterschiedlicher Hersteller. Jeder Hersteller hat dabei seine ganz eigene Methodik, auf welche Weise die Geräte parametriert und überwacht werden.

So stellen einige Hersteller spezielle Managementprogramme für ihre Endgeräte zur Verfügung, andere bieten dem Benutzer bzw. Administrator eine Web-Oberfläche an, über die sich die Komponenten im Browser überwachen und konfigurieren lassen.

Kleinere Netzwerke lassen sich mit diesen Mitteln bequem einrichten, überwachen und warten.

In größeren Netzwerken, mit zum Teil mehreren 100 Netzwerkteilnehmern, wäre es allerdings sehr mühsam, jedes Gerät mit anderen Mitteln zu konfigurieren und zu überwachen. Hier bietet SNMP die Grundlage für ein einheitliches und überschaubares Netzwerkmanagement.

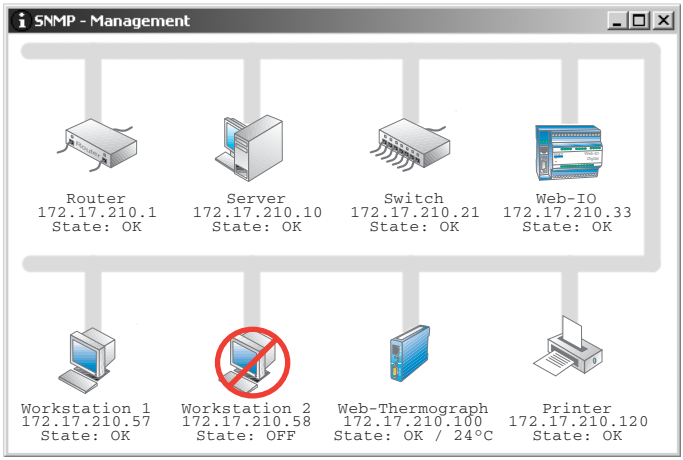
3.7.1 SNMP-Agent

Bedingung für den Einsatz von SNMP ist, dass alle beteiligten Endgeräte einen SNMP-Agenten besitzen. Der SNMP-Agent ist eine Software-Schnittstelle, die das Endgerät mit allen betriebswichtigen Parametern repräsentiert. SNMP-fähige Endgeräte werden auch als Netzwerkknoten bzw. Nodes bezeichnet. Nodes können Workstation-PC, Server, Switches, Router, Web-IO, also eigentlich alles sein, was über eine eigene IP-Adresse im Netzwerk ansprechbar ist.

3.7.2 SNMP-Manager

Neben den Nodes gibt es in SNMP-Systemen mindestens einen SNMP-Manager. Der SNMP-Manager ist eine Software-Anwendung, die auf einer Workstation oder einem Server arbeitet.

Während SNMP-Manager früher kommandozeilengesteuerte Anwendungen waren, in denen die Nodes in Listen verwaltet wurden, stellen moderne SNMP-Systeme dem Administrator mächtige Visualisierungsfunktionen zur Verfügung. Die gesamte Netzwerkinfrastruktur kann in Form von Plänen dargestellt und somit sehr übersichtlich verwaltet werden.



Zu den Aufgaben eines SNMP-Manager zählen: Konfiguration, Verwalten von Zugriffsrechten, Überwachen, Fehlermanagement und Netzwerksicherheit.

3.7.3 SNMP-MIB

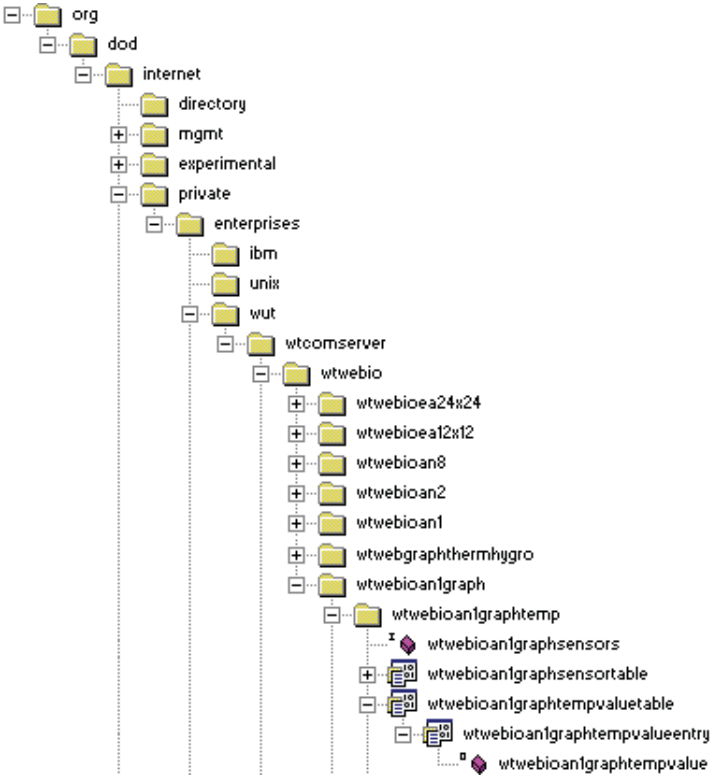
Die Abkürzung MIB steht für Management Information Base. Zu jedem Netzwerkknoten gehört eine spezifische MIB, d.h. eine Struktur aus Variablen, in denen die Eigenschaften und Zustände des Netzteilnehmers beschrieben sind.

Im Normalfall muss der Anwender sich nicht im Detail mit dem Aufbau der MIB beschäftigen. Moderne Managementsysteme verfügen über einen MIB-Compiler, der die MIB-Daten ins System integriert und dem Nutzer in einer gut handhabbaren Form zur Verfügung stellt.

Um das Verständnis für die Abläufe bei SNMP zu erhöhen, möchten wir hier dennoch einen groben Überblick über den Aufbau vermitteln.

Die MIB besteht aus zwei grundsätzlichen Teilen: der Standard MIB, in der System-Variablen verwaltet werden, die für alle Knoten benötigt werden, und der Private-MIB, in der die gerätespezifischen Variablen untergebracht sind und auf die wir hier näher eingehen.

Die Datenstruktur der MIB hat einen baumartigen Aufbau, ähnlich der Verzeichnisstruktur auf einer Festplatte. Die einzelnen Variablen sind in Gruppen, Untergruppen usw. gegliedert, so wie einzelne Dateien auf einem Datenträger in Ordnern und Unterordnern gespeichert werden.



Die Abbildung zeigt in der Darstellungsweise eines Verzeichnisbaumes, an welcher Stelle zum Beispiel bei einem

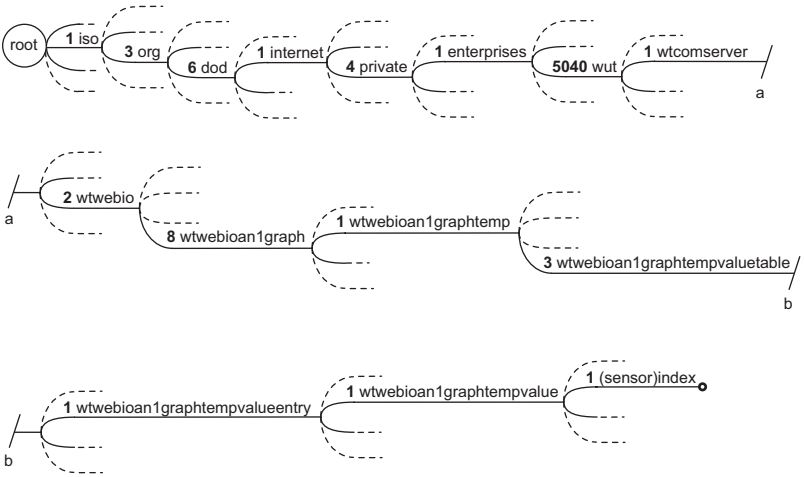
Web-Thermographen die gemessene Temperatur per SNMP abgerufen werden kann.

Bei den MIB-Variablen spricht man auch von Objekten. Zu jedem Objekt einer MIB gehört die MIB-OID. OID steht für Object Identification Number. Die OID ist eine durch Punkte getrennte Kette von Zahlen, wobei jede Zahl für einen Abzweig im MIB-Baum angibt, wohin verzweigt wird.

Die OID für die Sensortemperatur des Wiesemann & Theis Web-Thermographen sieht z.B. so aus:

1.3.6.1.4.1.5040.1.2.8.1.3.1.1.1

Da solche Datenketten für den Anwender nicht überschaubar sind, kann man die OID auch als MIB-Diagramm darstellen:



Die von den Herstellern der verschiedenen Netzwerkknoten mitgelieferten MIB-Dateien beschreiben die OID-Struktur im ASN1-Format (Abstract Syntax Notification).

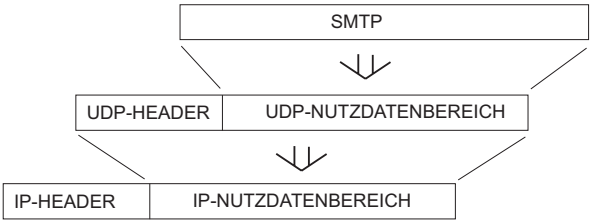
ASN1 Dateien sind zwar lesbar, eine Entschlüsselung durch den Anwender ist aber kompliziert und nicht vorgesehen.

Wie bereits angesprochen, verfügen SNMP-Managementsysteme über einen ASN.1 MIB-Compiler. Dieser Compiler wer-

tet das ASN.1 Format aus und vermittelt dem Manager, welche Variablen eines Netzwerkknotens an welcher Stelle zu finden sind.

3.7.4 SNMP-Kommunikation

Die Kommunikation zwischen SNMP Managementsystem und SNMP-Netzknoten wird über das UDP-Protokoll abgewickelt.



Hierbei empfängt der Netzwerkknoten die Datensendungen vom SNMP-Managementsystem auf Port 161. Das SNMP-Managementsystem benutzt für den Empfang Port 162.

Die normale Kommunikation geht immer vom Managementsystem aus. Dieses sendet ein GET-Kommando mit der OID des gewünschten Wertes an den Netzwerkknoten. Der Netzwerkknoten sendet darauf hin ein RESPONSE-Paket zurück, welches ebenfalls die OID und zusätzlich den zugehörigen Wert enthält.

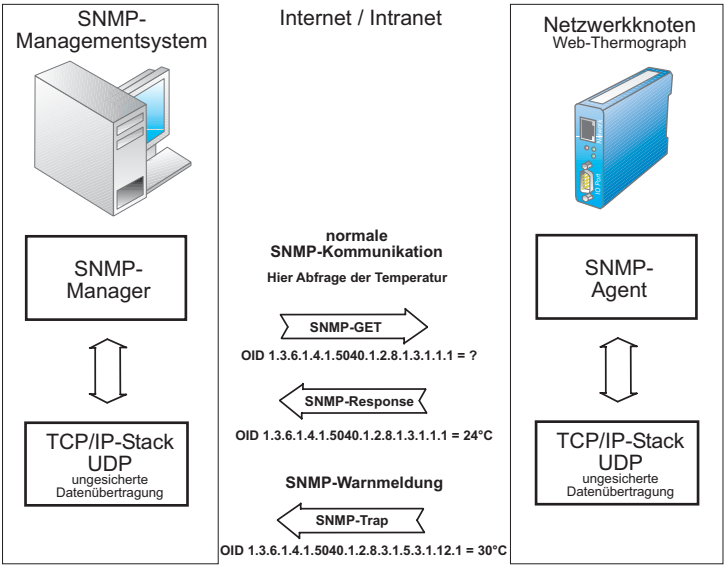
3.7.5 SNMP-Trap

Neben der normalen Kommunikation gibt SNMP den Netzwerkknoten die Möglichkeit, unaufgefordert Meldungen an den SNMP-Manager zu senden.

Diese SNMP-Traps werden als Status- oder Warnmeldungen genutzt. So kann z.B. ein Switch auf diesem Weg melden, wenn ein Port seinen Link verliert.

SNMP-Traps werden ebenfalls an Port 162 gesendet.

Beim Web-Thermographen können z.B. Alarme definiert werden, die bei Temperaturüberschreitung (z.B. im Serverraum) einen SNMP-Trap senden.



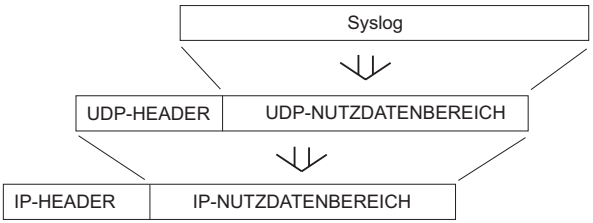
SNMP Traps haben eigene OIDs, die in einem gesonderten Teil der MIB untergebracht sind, auch wenn der gleiche Wert in einem anderen Teil der MIB ggf. noch einmal auftaucht.

Für Administratoren ausgedehnter Netze mit vielen Netzwerkteilnehmern bietet SNMP alle Voraussetzungen, die Wartung und Überwachung aller beteiligten Geräte einheitlich und übersichtlich abzuwickeln.

3.8 Syslog - Der Systemlogger

Syslog ist ähnlich dem SNMP ein Protokoll um Systemmeldungen an zentraler Stelle zu überwachen. Im Gegensatz zu SNMP ist Syslog aber eine Einbahnstraße. Das heißt mit Syslog können Netzwerkgeräte wie PCs, Router, Switches, Hubs, aber auch embedded Geräte wie Web-IO und Web-Thermographen, Systemmeldungen an einen zentralen Server senden; Datensendungen vom Server zu den Endgeräten sind jedoch nicht vorgesehen.

Auf Netzwerkebene werden Syslog-Meldungen über das UDP-Protokoll auf Port 514 übertragen.



Syslog-Meldungen können normale Statusinformationen, Warnmeldungen und Fehlermeldungen sein.

Je nach Dringlichkeit werden den Syslog-Meldungen vom Absender Prioritäten zugeordnet. Auf diese Weise kann beeinflusst werden, welche Meldungen bevorzugt bearbeitet werden.

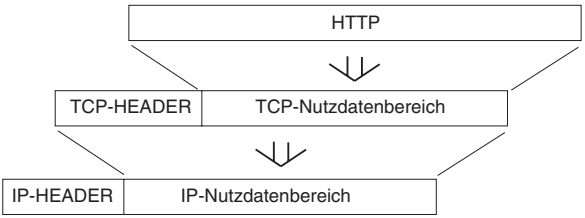
Ferner enthält jede Syslog-Meldung einen Zeitstempel mit Uhrzeit und Datum.

Der Prozess, der auf dem Server die Syslog-Meldungen entgegennimmt und weiterverarbeitet, wird als Syslog-Daemon bezeichnet.

Syslog stammt im Ursprung aus der Unix- bzw. Linux-Welt, wird heute aber auch im Windows-Umfeld eingesetzt.

3.9 HTTP – Hypertext Transfer Protocol

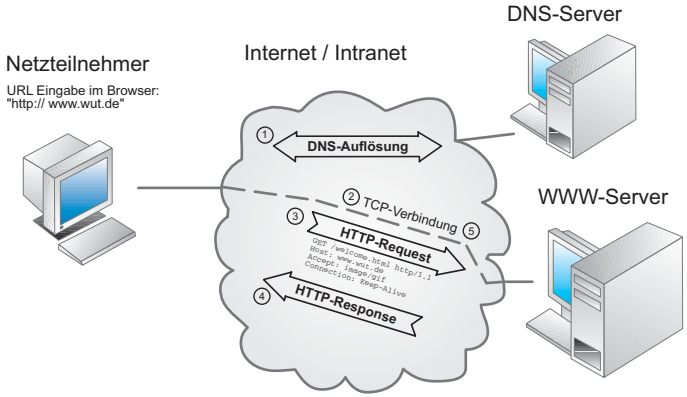
Durch die rasante Zunahme von WWW-Nutzern ist HTTP heute das mit Abstand meist genutzte Protokoll im Internet. HTTP setzt auf TCP als Basisprotokoll auf, wobei in aller Regel der TCP-Port 80 genutzt wird (abweichende Ports sind möglich, müssen aber explizit im URL angegeben werden).



Die Anforderung und Übertragung einer Webseite erfolgt in vier Schritten:

1. **Auflösen des angegebenen Host- und Domainnamens in eine IP-Adresse**
   Der TCP/IP-Stack startet eine DNS-Anfrage, um die IP-Adresse des gewünschten Servers zu ermitteln.
2. **Aufbau der TCP-Verbindung**
   Zur Erinnerung: Bei einer TCP-Verbindung gilt das Client-Server-Prinzip. Bei HTTP übernimmt der Browser die Rolle des Client und stellt die TCP-Verbindung zum angegebenen WWW-Server her.
3. **Senden der HTTP-Anforderung**
   Nach erfolgreichem Aufbau der TCP-Verbindung fordert der Browser die gewünschte Webseite beim WWW-Server an. An dieser Stelle beginnt das eigentliche HTTP-Protokoll: Der Browser sendet das *GET*-Kommando mit den erforderlichen Parametern zum WWW-Server.
4. **Senden der angeforderten Webseite**
   Der WWW-Server sendet erst eine HTTP-Bestätigung und dann die Webseite selbst.
5. **Beenden der TCP-Verbindung durch den WWW-Server**
   Eine Besonderheit bei HTTP ist, dass die TCP-Verbindung nicht wie sonst üblich durch den Client, sondern durch den Server abgebaut wird. Dafür gibt es zwei Gründe:

- Der WWW-Server signalisiert dem Browser auf einfache Art und Weise, dass die Übertragung abgeschlossen ist. Eine empfangene Webseite wird im Browser deshalb auch erst dann angezeigt, wenn die TCP-Verbindung beendet ist.
- WWW-Server müssen eine Vielzahl von TCP-Verbindungen gleichzeitig bedienen. Dabei verlangt jede offene Verbindung dem Server ein gewisses Maß an Leistung ab. Um die Verbindungszeiten so kurz wie möglich zu halten, baut der Server die Verbindung einfach ab, sobald alle angeforderten Daten übertragen wurden.



3.9.1 Die wichtigsten HTTP-Kommandos und Parameter

Wie bereits angesprochen basiert auch HTTP auf dem Client-Server-Prinzip: Der Browser als Client kann durch das Senden bestimmter Kommandos die Kommunikation steuern.

Das GET-Kommando

Das mit Abstand am häufigsten verwendete Kommando ist die **GET-Anfrage**, die jeden Aufruf einer Webseite einleitet. GET fordert den HTTP-Server auf, ein Dokument oder Element zu senden und ist damit das wichtigste Kommando.

Für den Einsatz von GET sind einige Parameter nötig; man spricht auch von einer Kommandozeile (engl. Request Line).

```
GET /pfadname/filename http-Version
```

Weitere Parameter können jeweils als neue Zeile mitgesendet werden. Diese angehängten Parameter werden auch als „Header“ bezeichnet.

Host Hostname (nur bei HTTP1.1 nötig).

Accept gibt an, welche Dateiformate der Browser verarbeiten kann
Mit *Accept: image/gif* gibt der Browser z.B. bekannt, dass er Bilder im GIF-Format anzeigen kann.

Connection über diesen Parameter *(Connection: Keep-Alive)* kann vom Browser vorgegeben werden, ob die TCP-Verbindung zum Nachladen anderer Elemente offengehalten wird.

Eine Vielzahl weiterer Parameter sind in der RFC2616 beschrieben, die unter *http://www.w3.org/Protocols/rfc2616/rfc2616.html* eingesehen werden kann.

Ein typisches GET-Kommando könnte etwa so aussehen:

```
GET /welcome.html http/1.1
Host: www.wut.de
Accept: image/gif
Connection: Keep-Alive
```

Als Antwort sendet der HTTP-Server eine Statuszeile, auf die ein Header (diesmal mit Parametern des Servers) folgt. Getrennt durch eine Leerzeile <CR LF CR LF> wird das angeforderte Element übermittelt.

```
HTTP/1.1 200 OK                                  |   Statuszeile
Date: Thu, 15 Mar 2001 11:33:41GMT               |
Server: Apache/1.3.4 (Unix) PHP/3.0.6            |
Last-Modified: Thu 15 Mar 2001 11:32:32 GMT      |
...                                              |
...                                              |   Header
Keep-Alive: timeout=15                           |
Connection: Keep-Alive                           |
Content-Type: text/html                          |
<html>                                           |
```

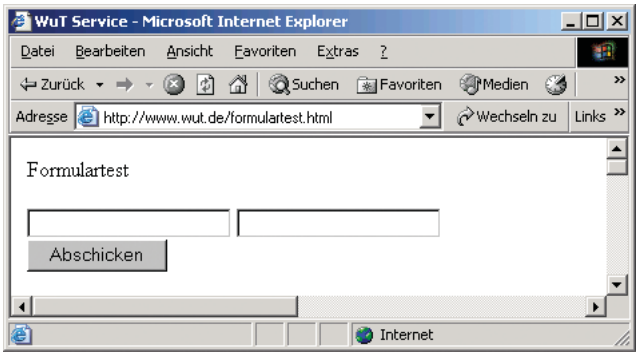
```
   ...                                    |   HTML-Seite
</html>                                   |
```

Die Statuszeile umfasst die vom Server unterstütze HTTP-Version, eine Fehlercode-Nummer und einen Kommentar. Im Header zeigt der Server unterstützte Verbindungseigenschaften und Daten an.

Das POST-Kommando

Das Gegenstück zu GET ist das POST-Kommando. POST erlaubt dem Browser, Informationen an den HTTP-Server zu übergeben.

Der klassische Einsatz für das POST-Kommando ist die Übergabe von Formulareinträgen aus einer HTML-Seite. Im Kern ist der Aufbau der POST-Anforderung identisch mit der von GET. Nach den Parametern steht eine Leerzeile <CR LF CR LF>, der die zu übergebenden Informationen folgen. Enthält eine POST-Anforderung mehrere Einzelinformationen, werden diese durch ein „&“ voneinander getrennt. Als *filename* muss in der ersten Zeile der POST-Anforderung ein auf dem Server verfügbarer Prozess angegeben werden, der die Informationen entgegennehmen und verarbeiten kann.



Für dieses Formulartest-Formular könnte die POST-Anforderung folgendermaßen aussehen; der bislang nicht besprochene Parameter *Referer* stellt hier einen Bezug zu der ursprünglich geladenen Formular-Seite her:

```
POST    /Formularauswertung.cgi HTTP/1.1
Accept: image/gif, image/jpeg
Referer: http://172.16.232.145/formulartest.html
Host: 172.16.232.145
Connection: Keep-Alive

EINGABEFELD1=test1&EINGABEFELD2=test2&submit=Abschicken
```

Tipp: Die meisten Internet Provider bieten sogenannte „CGI-Scripts“ (Programme auf dem HTTP-Server) an, die Formularangaben entgegennehmen und als E-Mail an eine beliebige Adresse weiterleiten. So kann man seinen Kunden z.B. die Gelegenheit geben, direkt von einer Webseite aus eine Bestellung oder Anfrage zu verschicken.

Das HEAD-Kommando

Als drittes Kommando sei hier der Vollständigkeit halber noch eine Variante von GET genannt. Das HEAD-Kommando arbeitet wie das GET-Kommando, doch der HTTP-Server gibt nur die Statuszeile und den Header, nicht aber das angeforderte Element selbst zurück.

Es wird fast ausschließlich zu Testzwecken und von Suchmaschinen genutzt, die über die resultierende Meldung (Fehlercode) die Existenz einer Seite überprüfen können.

3.9.2 HTTP-Versionen

HTTP wurde seit der Einführung des WWW mehrfach weiterentwickelt und kommt heute in drei Versionen vor:

HTTP 0.9	in 1989 erstmalig vorgestellt und seitdem genutzt, aber nie spezifiziert
HTTP 1.0	erst 1996 wurde HTTP in der Version 1.0 durch die RFC 1945 spezifiziert, die weitestgehend mit HTTP0.9 identisch ist
HTTP 1.1	wurde 1997 (RFC 2068) eingeführt und ist seit 1999 (RCF 2616) in überarbeiteter Form im Einsatz.

Alle heute erhältlichen Browser unterstützen standardmäßig HTTP1.1 , können aber auch problemlos mit Servern zusammenarbeiten, die HTTP0.9 oder HTTP1.0 verwenden.

Die wohl grundlegendste Änderung in HTTP1.1 liegt darin, dass die für die Übertragung des HTML-Dokuments aufgebaute TCP Verbindung auch für das Nachladen weiterer Elemente weitergenutzt wird. HTTP1.0 bzw. 0.9 haben für jedes Element eine separate TCP-Verbindung aufgebaut.

Eine persistente Verbindung wie in 1.1 erhöht den Datendurchsatz, da die Zeiten für Verbindungsaufbau und -abbau entfallen.

Als weitere Neuerung in der Version 1.1 kann ein HTTP-Server mit nur einer IP-Adresse Anfragen an verschiedene Hostadressen verarbeiten. Trägt der Anwender im Browser als URL z.B. *http://www.wut.de* ein, fragt der PC beim DNS-Server die zugehörige IP-Adresse nach.

Der Browser öffnet die TCP-Verbindung und sendet das GET-Kommando. Um auf einem HTTP-Server die Internet-Auftritte mehrerer Anbieter verwalten zu können, wurde mit *Host* ein zusätzlicher Parameter zum GET-Kommando eingeführt, der dem Server zusammen mit einer GET-Anfrage auch den Hostnamen übermittelt (z.B *Host: http://www.wut.de*). Dank dieses zusätzlichen Parameters kann der HTTP-Server über die GET-Anfrage erkennen, welchem Host die TCP-Verbindung gilt.

3.10 E-Mail

Die Möglichkeit, elektronische Post in wenigen Sekunden von einem Ende der Welt zum anderen verschicken zu können, ist sicherlich einer der Hauptgründe für die rasante Ausbreitung des Internets.

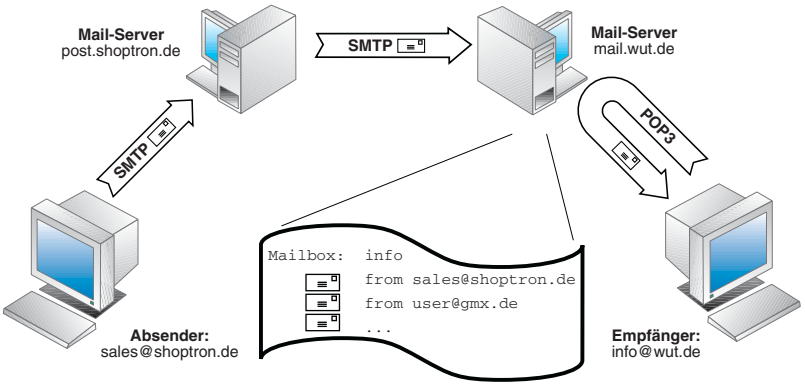
Im Gegensatz zu den meisten anderen Anwendungen im Internet ist das Versenden von E-Mail ein Dienst, bei dem keine direkte Verbindung zwischen Sender und Empfänger besteht. Das klingt zunächst verwirrend, ist aber sinnvoll, da sonst der Austausch von E-Mail nur möglich wäre, wenn Versender und Empfänger gleichzeitig im Netz aktiv sind.

Um eine zeitliche Unabhängigkeit zu gewährleisten, benötigt der E-Mail-Empfänger eine Mailbox (Postfach) auf einem Mail - Server, in der eingehende Nachrichten zunächst abgelegt werden.

Eine E-Mail-Adresse setzt sich immer aus dem Postfachnamen und der Zieldomain zusammen; als Trennzeichen steht das „@“ (engl. „at“, gesprochen „ätt“) zwischen diesen beiden Bestandteilen. Ein Beispiel: *info@wut.de* bezeichnet das Info-Postfach auf dem Mailserver von Wiesemann & Theis.

Der Weg einer E-Mail vom Versender zum Empfänger besteht aus zwei Teilabschnitten, auf denen der Transport über unterschiedliche Protokolle geregelt wird:

- vom Rechner des Absenders bis zum Postfach des Empfängers wird das SMTP-Protokoll benutzt,
- vom Postfach des Empfängers bis zum Rechner des Empfängers wird das POP3-Protokoll benutzt.



3.10.1 Aufbau einer E-Mail

Eine E-Mail setzt sich aus dem Nachrichtenkopf und der eigentlichen Nachricht zusammen. Diesen Kopf kann man mit einem Briefumschlag vergleichen, der Felder für Absender, Empfänger, Datum, Betreff und einige weitere Informationen enthält.

Hier die wichtigsten Felder im Überblick:

Die folgenden vier Felder bilden einen Minimalkopf und müssen auf jeden Fall enthalten sein.

Feld	Funktion
FROM	E-Mail-Adresse des Verfassers
TO	E-Mail-Adresse des Empfängers
DATE	Datum und Uhrzeit Hinweis: Die Uhrzeit kann willkürlich eingetragen werden und ist in aller Regel die Ortszeit des Absenders.
SUBJECT	Text der Betreffzeile
RECEIVED	Das Feld RECEIVED stellt eine Besonderheit dar, denn es wird nicht bei Erstellung der E-Mail angelegt. Jeder auf dem Weg der E-Mail liegende Mail-Router fügt ein RECEIVED-Feld ein und hinterlässt auf diese Weise einen "Durchgangsstempel" mit Datum und Uhrzeit.

Die Verwendung der im Folgenden genannten Felder ist optional.

Feld	Funktion
SENDER	E-Mail-Adresse des Absenders (in aller Regel identisch mit Eintrag unter FROM)
REPLY-TO	E-Mail-Adresse, an die der Empfänger im Bedarfsfall antworten soll. Wichtig, wenn E-Mails von einem Embedded-System wie dem W&T IO-Mailer automatisiert verschickt werden. Als Antwortadresse könnte in diesem Fall z.B die E-Mail-Adresse des Administrators eingetragen sein.
CC	E-Mail-Adresse eines weiteren Empfängers, der einen "Durchschlag" (CC = "Carbon Copy") der Nachricht erhält.
BCC	E-Mail-Adresse eines weiteren Empfängers, die für alle anderen Empfänger aber unsichtbar bleibt (BCC = "Blind Carbon Copy").
MESSAGE-ID	Eindeutige Identifikation einer E-Mail, die von der Mailsoftware willkürlich vergeben wird.
X-"MEINFELD"	Durch Voranstellen von "X-" können eigene Felder erzeugt werden.

Bei einigen Feldern ist eine RESENT-Variante möglich, die dann zum Tragen kommt, wenn es sich um eine vom ursprünglichen Empfänger weitergeleitete E-Mail handelt.

Der formale Aufbau von Nachrichtenkopf und Feldern muss den folgenden Konventionen genügen:

- Nach dem Feldnamen steht ein Doppelpunkt; danach folgt der jeweilige Parameter.
- Jedes Feld steht in einer eigenen Zeile, die mit <CR LF> (Carriage Return Line Feed; hex 0D 0A) endet.
- Nachrichtenkopf und Körper werden durch eine zusätzliche Leerzeile <CR LF> getrennt.
- Der Nachrichtenkörper selbst enthält nur den zu übermittelnden Text bzw. weitere eingefügte Dateien. Das Ende der Nachricht wird durch <CR LF . CR LF> (hex 0D 0A 2E 0D 0A) gekennzeichnet.
- Sowohl Kopf als auch Nachrichtenkörper bestehen ausschließlich aus 7-Bit-ASCII-Zeichen. Deshalb können auch alle Steuerinformationen als Klartext übertragen werden.

MIME – Multipurpose Internet Mail Extensions

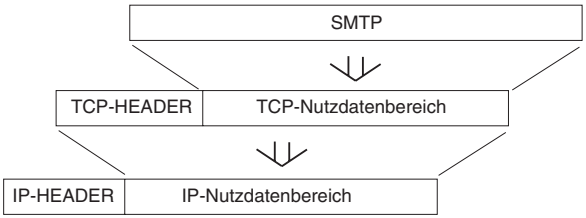
Um auch binäre Daten (8-Bit-Format) via E-Mail verschicken zu können, werden diese vor dem Einbinden in den Nachrichtenkörper nach dem „MIME-Standard“ in das 7-Bit-Format codiert und beim Empfänger wieder decodiert. Da die Verarbeitung bi-

närer Daten von heutigen E-Mail-Programmen automatisch übernommen wird, verzichten wir an dieser Stelle auf eine detaillierte Erklärung der „MIME-Codierung“.

3.10.2 SMTP – Simple Mail Transfer Protocol

SMTP regelt den Versand von E-Mails vom Mail-Client zum Mailserver (SMTP-Server). Der Mail-Client kann dabei entweder der ursprüngliche Versender oder ein auf dem Weg liegender Mail-Router sein. Mail-Router kommen zum Einsatz, wenn die E-Mail auf ihrem Weg über mehrere Domains weitergereicht wird. Häufig findet man für Mail-Router auch die Bezeichnung *MTA* (Mail-Transfer-Agent).

Für jedes Teilstück, das eine E-Mail zurücklegt, wird eine eigene TCP-Verbindung aufgebaut. SMTP setzt auf diese TCP-Verbindung auf, wobei der TCP-Port 25 genutzt wird.



SMTP stellt einige Kommandos (z.B Angabe des Absenders, Angabe des Empfängers ...) zur Verfügung. Jedes SMTP-Kommando wird einzeln vom SMTP-Server quittiert. Die eigentliche E-Mail wird komplett mit Kopf und Körper gesendet und dann erst vom SMTP-Server quittiert. Wenn keine weiteren E-Mails zum Versand anstehen, wird auch die TCP-Verbindung wieder abgebaut.

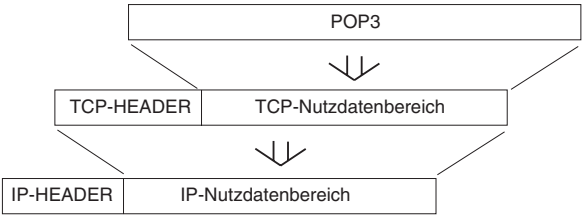
Hat die E-Mail den Ziel-Mailserver erreicht, wird sie im Postfach des Empfängers abgelegt und bleibt dort so lange liegen, bis sie vom Empfänger abgeholt wird.

3.10.3 POP3 – Post Office Protocol Version 3

Um eingegangene E-Mails aus dem Postfach auf dem Mailserver abzuholen, wird in den meisten Fällen das POP3-Protokoll benutzt. Der Empfänger wird über eingehende E-Mails nicht informiert. Er muss sein Postfach selbständig auf eingegangene E-Mails überprüfen und kann diese zu einem beliebigen Zeitpunkt abholen.

Die meisten der heute genutzten E-Mail-Programme überprüfen bei Start zunächst automatisch das Postfach des Nutzers auf eingegangene Mail. Viele E-Mail-Programme bieten darüber hinaus die Möglichkeit, ein Intervall vorzugeben, in dem das Postfach zyklisch geprüft wird. Typische Nutzer, die die meiste Zeit des Tages „offline“ sind, erhalten ihre E-Mails ohnehin nur dann, wenn sie sich beim Provider eingewählt haben. Doch bei Computern mit permanentem Internetzugang ist die zyklische Abfrage durchaus sinnvoll: Der Nutzer ist hier ständig online und erhält seine E-Mails mit nur geringer Verzögerung – quasi in Echtzeit.

Auch das POP3-Protokoll setzt auf eine TCP-Verbindung auf und ist nichts anderes als ein Klartextdialog.



POP3 nutzt die TCP-Portnummer 110. Wie bei SMTP beginnt der Dialog auch hier mit einem Login. Bei POP3 muss sich der Empfänger allerdings in zwei Schritten anmelden: mit Nutzernamen und mit Passwort. Nach erfolgreichem Login stellt POP3 einige Kommandos zur Verfügung, mit denen eingegangene Nachrichten aufgelistet, abgeholt oder gelöscht werden können.

Heute wird der Nutzer mit SMTP und POP3 nur noch in geringem Maße konfrontiert: Er muss lediglich beim Einrichten der

Mailsoftware den Namen des POP3- und SMTP-Servers angeben – das Abwickeln der Protokolle selbst wird unsichtbar im Hintergrund vom Mailprogramm übernommen.

Der Vollständigkeit halber sei noch erwähnt, dass es neben dem POP3-Protokoll noch die Protokolle POP2 und POP1 (beides Vorläufer von POP3) und IMAP4 gibt, die ebenfalls zum Abholen von E-Mails entwickelt wurden. Diese Protokolle konnten sich in der Praxis aber noch nicht durchsetzen oder wurden von POP3 verdrängt.

3.10.4 E-Mail per SMTP über gesicherte Server versenden

SMTP in seiner ursprünglichen Form sieht nicht vor, dass der Benutzer, der E-Mails versenden möchte, sich in irgendeiner Form authentifizieren, also seine Berechtigung nachweisen muss.

Das bedeutet: jeder, der Zugang zu dem Netzwerk hat, in welchem der SMTP Server platziert ist, kann von dort aus E-Mails versenden.

Im Zeitalter von Internet, Spam (unerwünschte Werbe-E-Mail) und Computerviren ist das natürlich ein nicht tragbarer Zustand.

Deshalb wurden Authentifizierungsverfahren entwickelt, die nur dem berechtigten Benutzer erlauben, E-Mails über den Server zu verschicken.

Die zwei gängigsten Verfahren möchten wir hier kurz vorstellen.

SMTP after POP3

Diese Methode ist denkbar einfach. Nur solche User, die auf dem Mail-Server ein POP3 Postfach haben, sind berechtigt über diesen Server E-Mails zu versenden.

Bevor das Senden von E-Mails zugelassen wird, muss ein Login in das POP3-Postfach erfolgen.

Der Vorteil dieser Methode ist, dass jedes normale Mailprogramm nach dem Start zunächst das POP3-Postfach nach neuem Posteingang durchsucht und über den damit verbundenen POP3-Login automatisch die Voraussetzung zum Versenden von E-Mails schafft.

Der Anwender muss also keine besondere Konfiguration an seinem Mailprogramm vornehmen.

Nur bei Embedded Endgeräten wie z.B. Web-IO oder Web-Thermographen sollte darauf geachtet werden, dass bei SMTP Authentification *SMTP after POP3* eingestellt wird.

ESMTP
Wird ESMTP benutzt, können E-Mails unabhängig vom POP3-Zugang versendet werden.

Nachdem die TCP-Verbindung zum SMTP-Server zustandegekommen ist, fragt dieser zunächst nach einem Usernamen und dem zugehörigen Passwort.

Erst wenn beides richtig übergeben wurde, können E-Mails versendet werden.

Für den Betrieb von Embedded Geräten hat diese Methode den Vorteil, dass zum Versenden von E-Mails nur eine TCP-Verbindung nötig ist.

Normale Mailprogramme müssen für den ESMTP-Betrieb speziell konfiguriert werden.

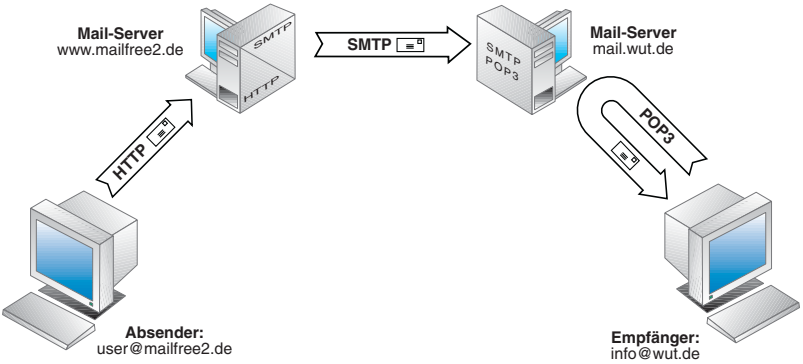
3.10.5 E-Mail über HTTP senden und empfangen

Mit der zunehmenden Nutzung von E-Mail gibt es immer mehr Freemail-Anbieter, die auf Ihrem Mailserver kostenlos Postfächer zur Verfügung stellen. Diese Dienstleistung, die jeder nutzen kann, wird in aller Regel über Werbung finanziert.

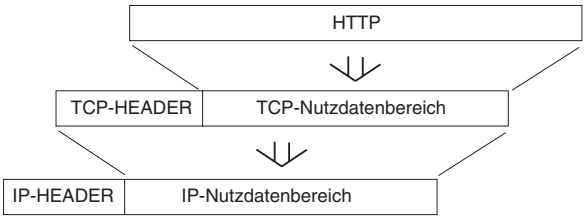
Um Raum zur Einblendung von Werbung zu schaffen, geben die meisten Freemail-Anbieter dem Nutzer die Möglichkeit, das Senden und Abrufen von E-Mails bequem über HTTP im Browser abzuwickeln, der selbstverständlich durch Werbe-

banner bereichert ist. Hierzu stehen dem Nutzer entsprechende HTML-Formulare zur Verfügung.

Um die E-Mail-Abwicklung über HTTP zu ermöglichen, muss der Freemail-Anbieter eine spezielle Mailserver-Kombination betreiben, die zur Nutzerseite als Webserver, zur anderen Seite als SMTP-Server arbeitet. Der Weg einer E-Mail sieht hier folgendermaßen aus:



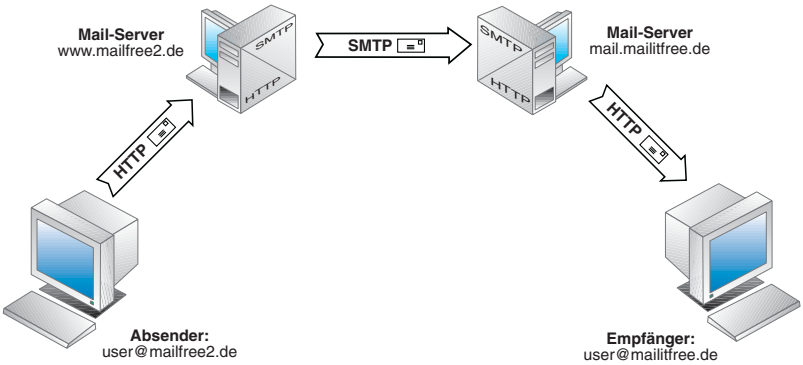
Zwischen dem Rechner des Absenders und dem Server des Freemail-Anbieters wird das HTTP-Protokoll verwendet. Wie bei anderen HTTP-Anwendungen auch, wird auch hier die TCP-Portnummer 80 genutzt.



Zwischen den Mailservern selbst ändert sich nichts. Sie kommunizieren miteinander über das SMTP-Protokoll.

Zwischen dem Ziel-Mailserver und dem Rechner des Empfängers können zwei unterschiedliche Varianten zum Einsatz kommen:

- Hat der Empfänger ein Standard-Mailkonto, werden eingegangene Mails über POP3 abgeholt.
- Nutzt auch der Empfänger die Dienste eines Freemail-Anbieters, kommt hier ebenfalls HTTP zum Einsatz.



Wer seine E-Mail lieber über SMTP und POP3 versenden möchte, sollte bei der Wahl des Freemail-Anbieters unbedingt darauf achten, dass auch Zugangsmöglichkeiten über einen SMTP- bzw. POP3-Server vorhanden sind.

3.10.6 E-Mail und DNS

Auch beim Versenden von E-Mails wird auf IP-Ebene mit IP-Adressen gearbeitet. Die Namensauflösung bei E-Mail Adressen funktioniert vom Prinzip genauso wie bei normalen Netzteilnehmern auch. Natürlich wird dabei nicht die Adresse des E-Mail-Empfängers selbst aufgelöst, sondern lediglich die des Mailservers, auf dem der Empfänger sein Postfach hat.

Zur Erinnerung: Um Namen in Adressen aufzulösen, bedient sich der TCP/IP-Stack eines Resolver-Programms, das beim DNS-Server eine entsprechende Anfrage stellt.

Nun ist der Hostname des Ziel-Mailservers aber nicht bekannt. Bekannt ist lediglich die Ziel-Domain, die ja in der E-Mail-Adresse hinter dem @-Zeichen steht. Um auch DNS-Anfragen nach Mailservern auflösen zu können, gibt es auf DNS-Servern spezielle Datensätze, in denen die zu einer Domain gehörenden Mailserver samt der zugehörigen IP-Adressen verzeichnet sind.

Das Resolver-Programm gibt also bei der Anfrage nur den Ziel-Domainnamen an und teilt zudem mit, dass es sich bei dem gesuchten Netzteilnehmer um einen Mailserver handelt. Der DNS-Server ermittelt die gesuchte IP-Adresse und gibt sie an das Resolver-Programm zurück.

Der Postfachname selbst wird für die DNS-Anfrage gar nicht benötigt. Er wird erst bei Eintreffen der Nachricht auf dem Ziel-Mailserver ausgewertet, damit diese im richtigen Postfach abgelegt werden kann.

Individualisierte Web-Kommunikation

Die grundlegende Funktionalität von TCP/IP-Ethernet und einigen weiteren klassischen Protokollen wäre nun geklärt.

In diesem Abschnitt erfahren Sie nun, wie Anwendungsprogramme sich diese Strukturen zunutze machen. Darüber hinaus stellen wir Ihnen Möglichkeiten vor, mit denen Sie durch wenige Handgriffe eine individuelle Lösung für die netzbasierte Kommunikation mit einem Endgerät oder Programm schaffen können.

Im ersten Teil stellen wir den Browser als „Mensch-Maschine-Interface“ für technische Anwendungen vor. Mit Hilfe des Browsers können Sie Sensoren und Aktoren über eine individuell gestaltete Webseite von jedem beliebigen PC aus aufrufen und visulisieren, bedienen, überwachen etc., ohne dass spezielle und meist teure Software installiert werden muss.

Anschließend beleuchten wir die Socket-Programmierung, mit der die Entwicklung eigener Anwendungen, die eine Kommunikation via TCP/IP unterstützen, nahezu uneingeschränkt möglich ist.

Schließlich gehen wir noch kurz auf OPC als eine Möglichkeit ein, die Hardware verschiedener Hersteller mit einer einheitlichen Anwendung zu verbinden.

1. Web-IO mit Browsern als Bedienoberfläche

In den ersten 20 Jahren seines Daseins war die Nutzung des Internets für normale Menschen kaum interessant. Eine für heutige Verhältnisse kleine Gruppe von Insidern musste kryptische Befehlszeilen eintippen, um Informationen auszutauschen zu können. Erst mit der Entwicklung des WWW-Standards, eines über das Internet abrufbaren Hypertext-Systems (World Wide Web), erschloss sich das Internet einem immer breiter werdenden Publikum.

Um die Möglichkeiten des WWW nutzen zu können, benötigt der Anwender einen Internetbrowser – ein Client-Programm, das über eine graphische Oberfläche die Inhalte von Webseiten anzeigt, die auf WWW-Servern hinterlegt sind.

Mit der Eingabe eines URL (Uniform Resource Locator) teilt der Anwender dem Browser mit, wie und wohin die Verbindung hergestellt werden soll. Er gibt zunächst vor, welches Protokoll genutzt wird; für den Zugriff auf Webseiten z.B. HTTP. Auf das Protokoll folgt dann die Angabe auf welchem Webserver die gewünschte Webseite liegt und wo diese dort genau zu finden ist.

```
protokoll://hostname [:tcp-port] [/pfadname][/filename][?weitere parameter]
```

http://www.wut.de/pdf/e-wwwww-12-prde-000.pdf

öffnet z.B. die Produktübersicht von W&T.

Die meisten Browser unterstützen noch weitere Protokolle, wie FTP zur Dateiübertragung (ftp://www.wut.de/download/e-58www-19-swde-000zip startet z.B. den Download der Programmbeispiele zu diesem Buch) oder Telnet. Telnet wird häufig genutzt, um Embedded Systeme via Netzwerk zu konfigurieren (Mit telnet://<IP-Adresse eines W&T Com-Servers>:1111 wird z.B. der Zugriff auf den Konfigurationsport eines W&T Com-Servers eingeleitet.)

Webseiten liegen als Hypertext vor und können neben Textinformationen auch Verweise auf Bilder, Graphiken und weitere multimediale Inhalte enthalten. Wie all diese Elemente im

Browser angezeigt werden sollen, ist ebenfalls im Hypertext hinterlegt.

Neben der statischen Anzeige von Informationen können über Webseiten jedoch auch Aktionen ausgelöst und Elemente dynamisch dargestellt werden. Das Grundgerüst einer Webseite kann mit HTML realisiert werden; für die Einbindung von interaktiven bzw. dynamischen Elementen stehen verschiedene Techniken zur Verfügung. Den Aufbau einer solchen Webseite werden wir Ihnen im Folgenden erläutern und anhand einiger Beispiele darlegen.

1.1 HTML – Hypertext Markup Language

Eines der Probleme im WWW war zunächst die Vielzahl unterschiedlicher Rechner und Betriebssysteme. Eine einheitliche Softwareschnittstelle auf Anwenderebene gab es nicht. Aus dem Bedürfnis heraus, eine auch für den Laien einfach zu bedienende Oberfläche zu schaffen, die sich auf verschiedensten Rechnern gleich darstellt, wurde HTML entwickelt.

HTML ist eine Auszeichnungssprache (Markup Language) die sich aus Schlüsselwörtern – auch Tags genannt – und den darzustellenden Inhalten zusammensetzt. Die Tags geben an, in welcher Art und Weise der nachfolgende Text darzustellen ist. So lassen sich z.B. Schriftgröße, -art und -ausrichtung vorgeben, Inhalte können in Tabellen oder in Form einer numerischen Aufzählung dargestellt werden, die Farbe von Text und Hintergrund kann festgelegt werden usw. Der Browser interpretiert diese Angaben und stellt sie dar.

Neben Text können mit Hilfe von HTML auch Grafiken angezeigt werden, und sogar multimediale Inhalte wie Musik, Sprache oder Filmsequenzen lassen sich per HTML einbinden. Das HTML-Dokument selbst transportiert dabei ausschließlich Textinhalte. Für jedes andere darzustellende Element wird via HTML angegeben, von wo es geladen werden kann, wo es auf

dem Bildschirm erscheinen soll und in welcher Größe es dargestellt werden soll.

Die wohl wichtigste Eigenschaft von HTML liegt darin, dass alle Elemente mit einem Verweis – auch *Hyperlink* oder kurz *Link* genannt – versehen werden können. Klickt der Anwender ein solches Element mit der Maus an, wird er automatisch auf eine weitere Website weitergeleitet, bekommt eine Grafik angezeigt oder startet einen Download.

Mit der Erklärung der von HTML bereitgestellten Tags lassen sich ganze Bücher füllen. Deshalb beschränken wir uns hier auf die elementaren Tags und Eigenschaften von HTML.

Für HTML-Tags gilt ein festes Schema:

- Einzelne Tags sind in spitze Klammern „eingepackt". *<HTML-Tag>*
- Das eigentliche Tag kann durch Angabe von Attributen erweitert werden. *<HTML-Tag Attribut="xy">*
- Überwiegend wird durch die paarweise Verwendung von Tags der Anfang und das Ende ihres Gültigkeitsbereichs festgelegt; die definierten Eigenschaften gelten dann für alles was zwischen den Tags steht. Das schließende Tag wiederholt das öffnende Tag mit einem vorangestellten Schrägstrich. Beispiel: <title>Willkommen</title>
- Bei HTML-Tags wird nicht zwischen Groß- und Kleinschreibung unterschieden. *<HTML>* ist gleichbedeutend mit *<html>.*

1.1.1 Grundsätzlicher Aufbau einer HTML-Datei

Jede HTML-Datei wird mit *<HTML>* eingeleitet und endet mit *</HTML>.* Man unterscheidet beim weiteren Aufbau einer Seite zwischen Kopf und Körper.

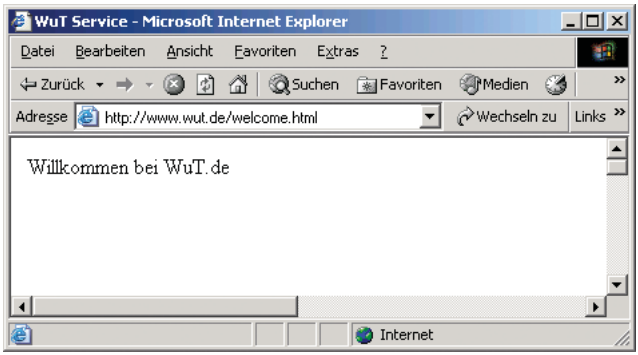
Alle Angaben im Kopf bleiben für den Betrachter unsichtbar und enthalten Eigenschaften der Seite, die nicht direkt die Darstellung betreffen. Einzige Ausnahme ist der Titel, der in der Titelleiste des Browserfensters angezeigt wird. Die Kopfinformationen stehen zwischen den Tags *<head>* und *</head>*

Auf den Kopf folgt der Seitenkörper, der mit dem *<body>*-Tag eingeleitet wird. Im Körper der HTML-Seite sind alle Angaben zu finden, die den eigentlichen Inhalt der Seite und dessen Darstellung betreffen. Das Ende des Körpers wird mit dem *</body>* Tag gekennzeichnet.

Hier ein einfaches Beispiel:

```
<html>
    <head>
        <title>WuT Service</title>
    </head>
    <body bgcolor="#FFFFFF">
        Willkommen bei WuT.de
    </body>
</html>
```

Bitte beachten Sie, dass beim *<body>*-Tag das Attribut *bgcolor="#FFFFFF"* für einen weißen Hintergrund angegeben wurde. Im Browser sieht das dann so aus:



1.1.2 Hyperlinks

Einer der großen Vorteile von HTML liegt in der Möglichkeit, einzelne Inhaltselemente mit einem Hyperlink zu versehen. Klickt der Anwender auf ein solches verlinktes Element, wird er auf eine andere Webseite weitergeleitet.

Wir erweitern unseren HTML-Code um einen Hyperlink:

```
<body bgcolor="#FFFFFF">
    Willkommen bei <a href="http://www.wut.de/index.html">WuT.de</a>
</body>
```

Bei einem Mausklick auf „WuT.de“ werden wir nun auf die Homepage von W&T gelenkt.

Das Pfadattribut des Tags *<a href="Pfadangabe">* kann die Pfadangabe entweder in absoluter oder in relativer Form enthalten.

- Absolut: es wird der komplette URL angegeben, auf den der Hyperlink verweisen soll.
- Relativ: es wird nur der Name der Datei angegeben auf die zugegriffen werden soll. Die Datei wird dann im gleichen Verzeichnis gesucht, in dem sich auch die aktuelle HTML-Datei befindet.

1.1.3 Darstellung von multimedialen Inhalten

Wie bereits angesprochen, erlaubt HTML die Darstellung von Inhalten, die nicht Bestandteil des HTML-Dokuments sind, sondern von anderer Stelle nachgeladen werden.

Bilder

Zur Einbindung von Bilddateien stellt HTML z.B. das *<img>*-Tag (*img* für Image) zur Verfügung, wobei über das Attribut *src* Namen und Quelle der Bilddatei angegeben werden.

Wir erweitern unser HTML-Dokument um eine Grafik:

```
<body bgcolor="#FFFFFF">
    <img src="http://www.wut.de/kpics/logo5.gif">
    Willkommen bei <a href="http://www.wut.de/index.html">WuT.de</a>
</body>
```

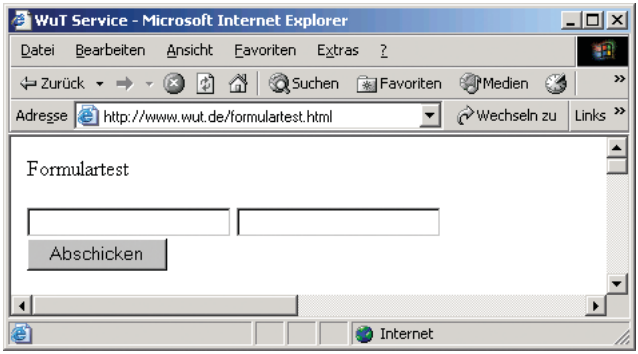
Nun wird neben dem Text ein Logo im GIF-Format dargestellt, das aus dem Verzeichnis *kpics* auf dem Webserver von W&T geladen wird. Der Pfad der Bilddatei kann wie auch beim Hyperlink absolut oder relativ angegeben werden.

Formulare

HTML ist eine reine Darstellungssprache, die eine statische Anzeige im Browser generiert. Aber wie sieht es aus, wenn der Anwender Informationen an den WWW-Server zurückgeben möchte?

Als Lösung bietet HTML die Möglichkeit, Formulare anzuzeigen, die vom Anwender ausgefüllt werden können. Die so eingegebenen Informationen können durch Anklicken eines sogenannten „Submit-Buttons“ vom Browser zum WWW-Server gesendet werden.

Hier ein kurzes Beispiel:



Im HTML-Code sieht das so aus:

```
<html>
    <head>
        <title>WuT Service</title>
    </head>
    <body bgcolor="#FFFFFF">
        Formulartest
        <form method="post" action="Formularauswertung.cgi" name="FORMULAR1">
            <input type="text" name="EINGABEFELD1">
            <input type="text" name="EINGABEFELD2">
            <input type="submit" name="submit" value="Abschicken">
        </form>
    </body>
</html>
```

Sämtliche zum Formular gehörenden Elemente sind zwischen dem einleitenden *<form>*-Tag und dem abschließenden *</form>*-Tag zu finden.

Die Attribute des Form-Tags sind:

method	gibt an, wie HTTP die Eingaben an den WWW-Server übergibt.
action	legt fest, an welchen Prozeß auf dem Server die Eingaben übergeben werden.
name	kann willkürlich vergeben werden und zeigt dem Prozeß auf dem Server, von welchem Formular die Eingaben stammen (ein Prozess kann ggf. mehrere Formulare auswerten).

Die Eingabeelemente selbst werden über das *<input>*-Tag festgelegt, wobei das Attribut *type* angibt, um welche Art von Eingabeelement es sich jeweils handelt. Mögliche Attribute sind hier:

text	Texteingabefeld
checkbox	Ankreuzkästchen
radio	Optionsbutton
submit	Button zum Abschicken oder Zurücksetzen des Formulars

Über das Attribut *name* kann dem Element ein eindeutiger Name (vergleichbar mit einem Variablennamen) gegeben werden; mit dem Attribut *value* kann ihm ein Anfangswert zugeteilt werden.

Die Angabe *<"input type="text" name="EINGABEFELD1" value="test1">* würde z.B. dazu führen, dass beim Öffnen des Formulars im Browser schon der Text *test1* im ersten Eingabefeld stehen würde.

Was mit den per Formular übergebenen Informationen geschieht – ob der Anwender eine Rückmeldung erhält und wie diese aussieht –, bestimmt allein der Prozess auf dem WWW-Server, der die Informationen entgegennimmt und auswertet.

Wie bereits angesprochen, möchten wir hier nicht bis auf das letzte Detail von HTML eingehen. Wer Webseiten erstellen möchte, sollte sich auf jeden Fall eingehender mit diesem Thema beschäftigen. Eine hervorragende Quelle für weitere Informationen zu HTML findet man unter *http://www.selfhtml.org;* sehr nützlich ist selbstverständlich auch die Website des W3-Konsortiums (*http://www.w3.org*), das in Sachen HTML als normierende Körperschaft fungiert.

1.2 Interaktive bzw. dynamische Elemente

Es gibt verschiedene Möglichkeiten, über im Browser angezeigte Webseiten Aktionen auszulösen und Elemente dynamisch darzustellen.

Dazu ist auf jeden Fall ein Programm, bzw. ein Prozess nötig, der z.B. Eingaben vom Anwender entgegennimmt und entsprechende Reaktionen auslöst. In technischen Anwendungen ist es sinnvoll, dass sich angezeigte Werte und Prozessabbilder selbstständig aktualisieren.

Unterschieden wird zwischen Programmen, die auf dem WWW-Server aktiv sind und solchen die im Browser, also auf dem lokalen Rechner ablaufen. Auch eine Kombination von beidem ist oft zu finden.

1.2.1 Serverseitige Programme

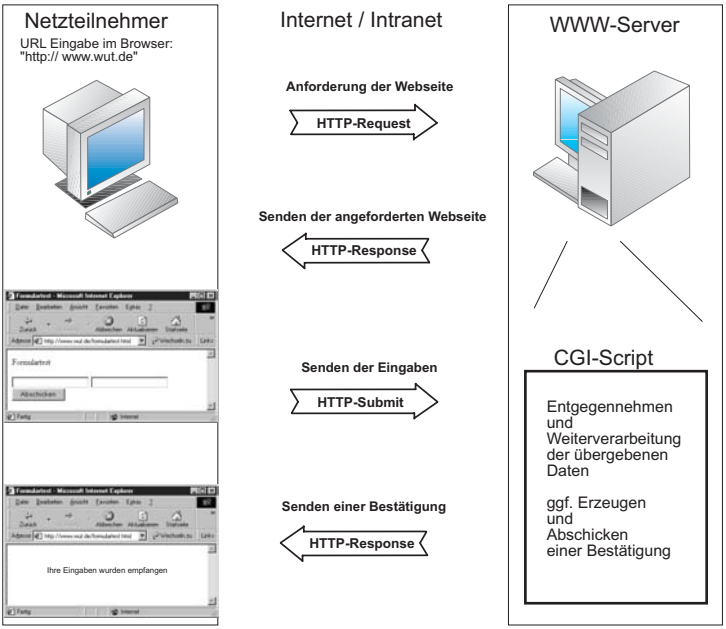
CGI - Common Gateway Interface

Der Einsatz von CGI-Scripten war lange Zeit das meistgenutzte Verfahren, im Browser interaktive Inhalte anzuzeigen bzw. Aktionen auszulösen.

Über CGI können vom Browser aus Programme auf dem WWW-Server ausgeführt werden.

Über einen Hyperlink, einen Submit-Button oder direkte Eingabe des URL wird das entsprechende Programm aufgerufen und es werden ggf. die nötigen Parameter übergeben.

Ein klassisches Beispiel sind HTML-Formulare, die vom Anwender ausgefüllt werden. Klickt der Anwender den Submit-Button (Abschicken) werden die Eingaben via http mit Hilfe des POST-Kommandos an den WWW-Server übergeben. Das angegebene CGI-Script wird gestartet und verarbeitet die Eingaben weiter.



Weitere mögliche Anwendungen sind Besucherzähler, Gästebücher, Diskussionsforen, Datenbankzugriffe oder Suchmaschinen.

CGI-Scripte können grundsätzlich in allen gängigen Programmiersprachen erstellt werden. Wichtig ist, dass der WWW-Server die gewählte Sprache unterstützt.

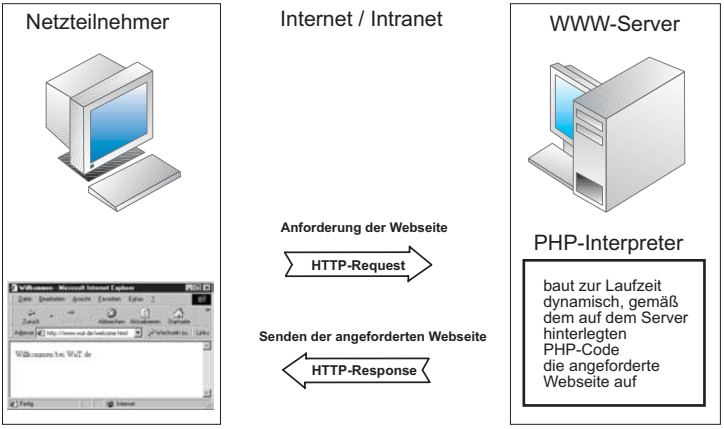
In der Praxis hat sich Perl für die Erstellung von CGI-Scripts durchgesetzt.

PHP

PHP hat CGI heute als meistgenutztes Verfahren für die Darstellung interaktiver Inhalte abgelöst und wird z.Zt. in den Versionen PHP3 und PHP4 verwendet.

Auch PHP erlaubt das Ausführen von Programmen auf einem WWW-Server. PHP ist eine Interpretersprache, deren Quelltext in eine HTML-Seite eingebunden ist, die auf dem WWW-Server liegt. Dabei können statische Inhalte der Seite im HTML-Format definiert werden, wogegen veränderbare Inhalte durch PHP-Quellcode eingebracht werden. PHP kann auch auf andere Resourcen auf dem Server, wie z.B. Datenbanken zugreifen.

Bei Anforderung der entsprechenden Seite durch den Browser wird der in der Seite integrierte PHP-Code auf dem Server vom PHP-Interpreter ausgewertet. Der PHP-Interpreter erzeugt individuell eine Seite in HTML-Code. Die so entstandene Webseite wird dann vom Server via http zum Browser gesendet. Der PHP-Quellcode bleibt für den Anwender unsichtbar.



Beim Onlineshopping könnte man zum Beispiel zu den offerierten Artikeln dynamisch Lagerstückzahlen, Lieferzeiten und Preise aus einer Warenwirtschaftsanwendung via PHP in die Webseite einbringen. Das bedingt natürlich, dass auf dem Server ein PHP-Interpreter aktiv ist.

ASP - Active Server Pages

ASP ist eine von Microsoft ins Leben gerufene Art, Webseiten dynamisch anzuzeigen. ASP-basierende Webseiten bestehen wie auch bei der PHP-Technik aus klassischen HTML-Anteilen und Scripten, die bei Aufruf der Webseite serverseitig ausgeführt werden. Ein Interpreter auf dem WWW-Server generiert, gesteuert durch diese Scripte, Webseiten in normalem HTML-Format.

Als Scriptsprachen werden zumeist VBScript (Visual Basic Syntax) oder JScript (Java Syntax) verwendet.

Ein Vorteil dieser Technik besteht darin, auf dem Server installierte DLLs und AktivX-Komponenten nutzen zu können. Dynamic Link Libraries und AktivX-Komponenten sind fertige, ausgelagerte Programmfunktionen, die dem Programmierer Arbeit abnehmen, da entsprechende, oft komplexe Funktionalitäten nicht selbst programmiert werden müssen.

Der Nachteil von ASP liegt in den Server-Betriebssystemvoraussetzungen. Im Ursprung gab es ASP-Unterstützung nur auf

Microsoft Serversystemen. Von Drittherstellern gibt es seit einiger Zeit aber auch ASP-Varianten für Linux-Server.

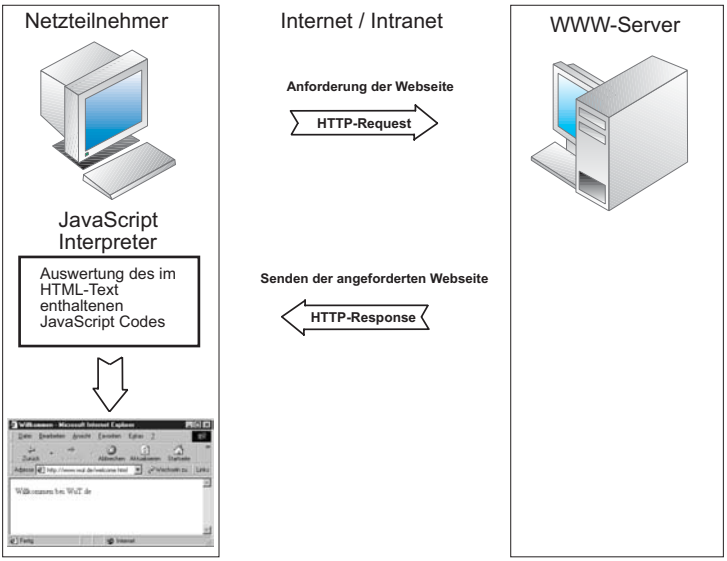
ASP-basierende Webseiten erkennt man an der Endung „.asp“ im Dateinamen.

Das klassische ASP wurde von Microsoft inzwischen durch ASP.NET abgelöst.

1.2.2 Browserseitige Programme

JavaScript

Bei JavaScript wird der Quellcode in den HTML-Text der Seite eingebunden. Der JavaScript Code wird mit dem <SCRIPT language=“JavaScript“> Tag gekennzeichnet und beim Laden einer Webseite vom Browser erkannt, interpretiert und ausgeführt.



Mit JavaScript können z.B. individuelle Anpassungen der angezeigten Inhalte einer Webseite vorgenommen werden, indem Variablen angelegt werden. In Abhängigkeit des Wertes einer

Variablen kann dann entschieden werden, wie bestimmte Dinge dargestellt werden. Auch Benutzereingaben lassen sich überprüfen, bevor sie zum WWW-Server weitergeleitet werden.

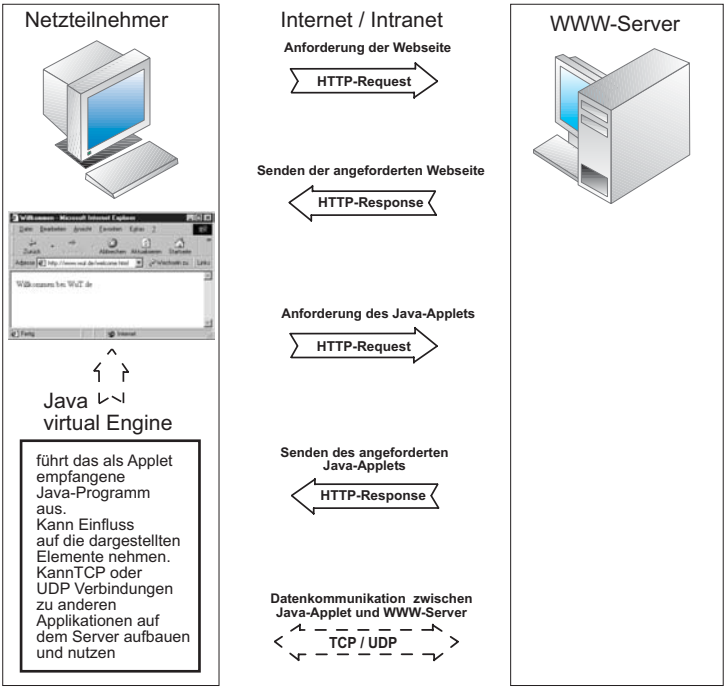
Ein Beispiel:
Der folgende Code wertet aus, ob eine Webseite über die „com" Domain oder die „de" Domain aufgerufen wurde und stellt sich entsprechend englisch oder deutsch dar.

```
<HTML>
  <HEAD>
   <TITLE>urltest</TITLE>
   <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-1">
  </HEAD>
  <BODY>
   <SCRIPT LANGUAGE="JavaScript"><!—
   if (location.hostname == "www.web-io.com") document.write("welcome at WuT");
   else document.write("willkommen bei WuT");
   //—></SCRIPT>
  </BODY>
</HTML>
```

Java Applets

Hier handelt es sich um kompilierte Programme, die in der Programmiersprache Java erstellt wurden. Java Applets werden, ähnlich wie grafische Elemente, zusätzlich zum HTML-Text geladen und im Browser ausgeführt.

Mit Java Applets können auch komplexere Aktionen, wie Netzwerkzugriffe auf TCP- und UDP-Ebene realisiert werden.



Aus Sicherheitsgründen ist allerdings nur die Kommunikation mit dem Server möglich, von dem das Applet geladen wurde. Auch auf dem lokalen Rechner des Anwenders ist der Zugriff auf Elemente und Funktionen des Browsers beschränkt. Ein Zugriff auf die Festplatte des eigenen Rechners ist zum Beispiel nicht möglich.

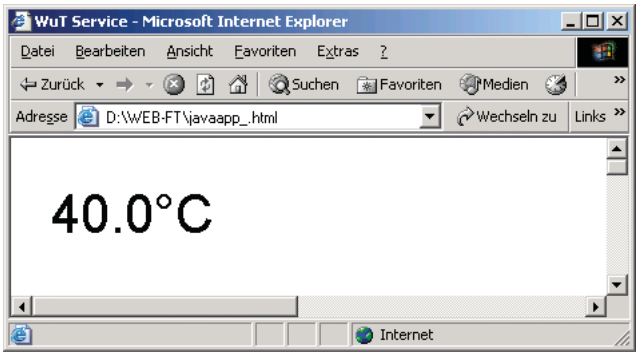
Ein Beispiel für den Einsatz von Java Applets ist das W&T Web-Thermometer. Vom Web-Thermometer wird ein Applet zur Verfügung gestellt, das, einmal im Browser gestartet, in regelmäßigen Abständen die aktuelle Temperatur abfragt und in der Webseite darstellt, von der es aufgerufen wurde.

Im HTML Code werden Applets über das Applet-Tag eingebunden, wobei mit *code=* der Name des Applets und mit *archiv=* der Name des Archivs innerhalb des Applets angegeben wird. Unter *codebase=* wird festgelegt, von welchem Host das Applet geladen wird.

Zwischen den Applet-Tags werden zusätzliche Parameter übergeben; im folgenden Quelltext z.B. welcher Sensor als Quelle für die gezeigte Temperatur ausgewählt wird.

```
<html>
  <head>
    <title>Schaltschranktemperatur</title>
  </head>
  <body bgcolor="#FFFFFF">
    <applet archive="A.jar" code="A.class" codebase = "http://172.16.232.152/" >
      <param name="sensor" value="1">
    </applet>
  </body>
</html>
```

Im Browser sieht das dann so aus:



Java und Javascript als zuverlässiges Team

Mit Java und Javascript können zuverlässig dynamische Webseiten auch für technische Anwendungen realisiert werden.

HTML stellt dabei das visuelle Grundgerüst der Webseite. HTML ist jedoch lediglich eine Beschreibungssprache, in der ausschließlich festgelegt wird, in welcher Form die Inhalte im Browser angezeigt werden.

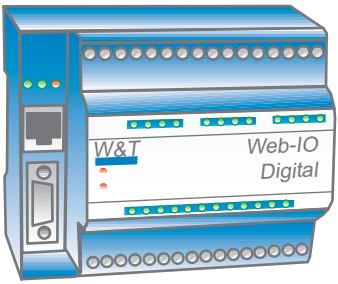
Mit JavaScript können hingegen bestimmte Inhalte der Webseite zustandsabhängig, d.h. je nach Wert der Variablen angezeigt

werden. Der Wert einer Variablen wird jedoch zum Zeitpunkt des Ladens der Webseite festgelegt.

Den kontinuierlicher Abgleich mit einer technischen Anwendung übernimmt schließlich Java in Form eines Applets. Ein Java-Applet kann nicht nur in einem vorgegebenen Format Werte anzeigen, sondern auch eine Benutzeroberfläche oder eine Software-Schnittstelle zu JavaScript bilden, die zum Datenabgleich bzw. zur Kommunikation genutzt wird. So wird die dynamische Anzeige von Werten im Browser möglich, ohne dass die zugrundeliegende Webseite neu geladen werden muss.

1.3 Beispiele dynamischer Webseiten mit Java und Javascript

Um das Zusammenspiel von Java, JavaScript und HTML möglichst praxisnah aufzuzeigen, werden die folgenden Beschreibungen am Beispiel des Web-IO Digital erläutert.



Das Web-IO 12xDigital von Wiesemann & Theis ist nicht viel größer als eine Zigarrettenschachtel und verfügt über 12 digitale Inputs mit Zählern, sowie über 12 digitale Outputs.

Das Steuern und Überwachen ist über den integrierten Web-Server möglich, der auch ein Java-Applet für technische Anwendungen im Browser bereit hält.

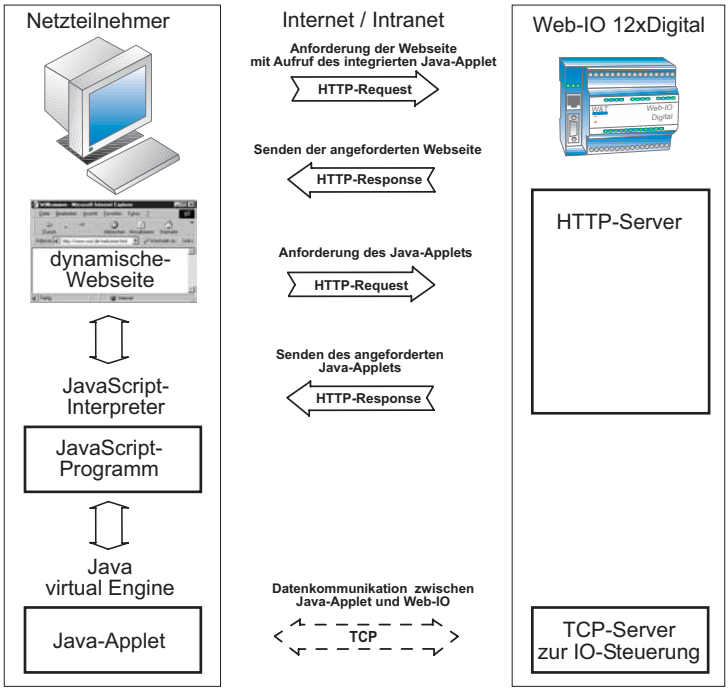
Das Java-Applet kann von einer dynamischen Webseite aus aufgerufen und gestartet werden und übernimmt dann folgende Aufgaben:

- Datenaustausch mit dem Web-IO
  Der Datenaustausch zwischen Java-Applet und Web-IO wird über den TCP-Port 80 abgewickelt, dem Port, den der Browser auch zum Abruf von Webseiten als Ziel-Port verwendet. Das Applet öffnet dazu eine separate TCP-Verbindung zum Web-IO, die bestehen bleibt, solange die Webseite im Browser angezeigt wird.

- Weitergabe der Input-, Zähler-, Output-Stati an JavaScript
  Im JavaScript Programm müssen Funktionen deklariert sein, über deren Aufruf das Java-Applet JavaScript-Variablen setzen kann. Empfängt das Java-Applet über die TCP-Verbindung eine Statusänderung, wird die entsprechende Funkti-

on aufgerufen und der neue Wert übergeben. Das JavaScript-Programm verarbeitet diesen Wert weiter und zeigt ihn in der gewünschten Form innerhalb der bereits bestehenden Webseite an. Darum ist es nicht nötig, dass die Webseite komplett neu aufgebaut wird.

- Bereitstellen einer Funktion zum Setzen der Outputs
  Das Java-Applet bietet Funktionen, die von JavaScript aus aufgerufen werden können und die JavaScript-Variablen als Parameter entgegennehmen. Das Applet generiert aus den übergebenen Werten die entsprechenden Kommandos, um die Outputs zu setzen bzw. Zähler zu löschen. Diese Kommandos werden dann vom Applet über TCP zum Web-IO gesendet.



Wie eine dynamische Webseite aussehen kann, über die das Web-IO und das daran angeschlossene technische Gerät be-

dient oder überwacht werden kann, wollen wir in drei einfachen Beispielen darstellen.

1.3.1 Statusanzeige

In diesem Beispiel soll in einem Tabellenfeld der Zustand eines Inputs am Web-IO mit „ON“ oder „OFF“ angezeigt werden. Darüber hinaus soll sich die Hintergrundfarbe des Tabellenfeldes bei ON auf grün und bei OFF auf grau ändern.

1. Schritt

Zunächst wird eine normale Webseite mit einer Tabelle mit zwei Spalten und einer Zeile erstellt.


```
<html>
<head>
  <title>Web-IO</title>
</head>

<body>
<table border="1">
  <tr>
    <td width="200">Status Input 0 </td>
    <td width="50" bgcolor="#CCCCCC">OFF</td>
  </tr>
</table>
</body>
</html>
```

2. Schritt

Anschließend wird im body-Bereich dieser Seite der Aufruf für das Java-Applet eingefügt:

```
.....
<body>
<applet name="dio0" archive="dio.jar" code="dio.class"
codebase="http://172.16.232.17" width="0" height="0" mayscript>
    <param name="device" value="0">
    <param name="inputpolling" value="on">
    <param name="pollingrate" value="1000">
</applet>
<table border="1">
........
```

Beim Aufruf des Applets werden neben Standardparametern einige Web-IO spezifische Parameter angegeben.

Zunächst die Standardparameter:

name
gibt an, unter welchen Namen das Applet unter JavaScript ansprechbar ist.

archiv und code
geben an, welches Applet geladen wird (ein Web-Server kann mehr als 1 Applet anbieten).

codebase
dieser Parameter gibt die IP-Adresse des Servers (hier des Web-IO) an und ist nur dann nötig, wenn die Webseite nicht vom gleichen Server geladen wird wie das Java-Applet

mayscript
nur wenn dem Applet-Aufruf *mayscript* hinzugefügt wurde, können JavaScript und Java-Applet Daten austauschen.

Die Web-IO spezifischen Parameter sind in dieser Form natürlich auf das spezielle Applet im Web-IO ausgerichtet. Für andere Server bzw. Anwendungen können an dieser Stelle andere Parameter zum Tragen kommen:

device
soll mit mehr als einem Web-IO gearbeitet werden, ist durch diesen Parameter eine Unterscheidung möglich.

inputpolling (outputpolling und counterpolling)
Unter Polling versteht man das zyklische Abfragen von Daten. Mit dem o.g. Parameter wird festgelegt, welche Werte (Inputs, Outputs oder Counter) vom Applet automatisch abgefragt werden sollen. Im Fall dieses Beispiels sind das nur die Inputs, so dass die Parameter outputpolling und counterpolling nicht benötigt werden.

pollingrate
legt den Abfragezyklus in ms fest (hier 1 mal pro Sekunde).

3. Schritt

Um per JavaScript bestimmte Inhalte einer Webseite auch nach dem Laden der Seite verändern zu können, muss dem zu verändernden Objekt eine ID (identification) zugeteilt werden; in unserem Fall dem Tabellenfeld, welches geändert werden soll.

```
<table border="1">
  <tr>
    <td width="200">Status Input 0 </td>
    <td width="50" bgcolor="#CCCCCC"id=”input0”>OFF</td>
  </tr>
</table>
```

4. Schritt

Nun müssen im head-Bereich der Seite die JavaScript- Funktionen eingefügt werden, die das Java-Applet bei Statusänderung aufruft.

```
function inputChanged( iDevice, iNr, iVal )
{   if iDevice==0)&(iNr==0)
    {   document.getElementById( 'input0')
        .firstChild.data = ( !iVal ) ? 'OFF' : 'ON';
        document.getElementById( 'input0' )
        .style.backgroundColor= ( !iVal ) ? '#CCCCCC' : '00FF00';
```

```
    }
}
```

Beim Aufruf der Funktion inputChanged (bitte Groß-/Kleinschreibung beachten) werden drei Werte übergeben:

iDevice
ist die Nummer, die beim Aufruf des Applets als Parameter *device* übergeben wurde. Immer dann, wenn mehr als ein Applet benutzt wird, kann anhand dieses Wertes identifiziert werden, von welchem Web-IO die Funktion aufgerufen wurde.

iNr
gibt an bei welchem Input sich der Status geändert hat.

i*Val*
übergibt den Status des entsprechenden Inputs.

true = ON
false = OFF

Über eine if-Abfrage wird sichergestellt, dass eine Aktion nur ausgeführt wird, wenn der gewünschte Input betroffen ist.

Soll mehr als ein Input ausgewertet werden, kann an dieser Stelle auch mit dem Switch/Case Kommando gearbeitet werden.

Und nun kommt die ID ins Spiel, die dem Tabellenfeld zugeteilt wurde.

Mit dem JavaScript-Kommando *document.getElementById* wird festgelegt, welches Objekt geändert wird.

In Abhängigkeit von *iVal* wird als Text ON oder OFF angezeigt. Ferner wird der Hintergrund abhängig von *iVal* grün oder grau gefärbt.

Zur besseren Übersicht hier noch einmal der komplette Quelltext:

```
<html>
<head>
<title>Web-IO</title>
<script language="JavaScript" type="text/javascript">
<!--
function inputChanged( iDevice, iNr, iVal )
{   if ((iDevice==0)&(iNr==0))
    {   document.getElementById('input0')
        .firstChild.data = ( !iVal ) ? 'OFF' : 'ON';
        document.getElementById('input0')
        .style.backgroundColor= ( !iVal ) ? '#CCCCCC' : '00FF00';
    }
}
//-->
</script>
</head>
<body>
<applet name="dio0" archive="dio.jar" code="dio.class" codebase="http://172.16.232.17"
width="0" height="0" mayscript>
    <param name="device" value="0">
    <param name="inputpolling" value="on">
    <param name="pollingrate" value="1000">
</applet>
<table border="1">
  <tr>
    <td width="200">Status Input 0 </td>
    <td width="50" bgcolor="#CCCCCC" id="input0">OFF</td>
  </tr>
</table>
</body>
</html>
```

Die hier aufgezeigte Methode lässt sich ohne großen Aufwand ebenso für die Outputs und Counter anwenden.

1.3.2 Schalten von Outputs

In diesem Beispiel soll aufgezeigt werden, wie die andere Richtung funktioniert, also wie von der Webseite aus über das Java-Applet die Outputs geschaltet werden können.

Ähnlich wie beim ersten Beispiel soll in einer Tabelle diesmal der Zustand von Output 0 angezeigt werden. In zwei zusätzlichen Spalten sollen Hyperlinks das Schalten nach ON und das Schalten nach OFF erlauben.

1. Schritt

Zunächst wird in einer Webseite eine Tabelle mit vier Spalten und einer Zeile angelegt.

In die beiden rechten Tabellenfelder wird der Text ON bzw. OFF eingetragen. Auf beide Texte wird ein Hyperlink gesetzt.

für den ON-Text: `javascript:setOutput(1);`

und für den OFF-Text: `javascript:setOutput(0);`

Das zweite Feld von links bekommt wie bei dem Input-Beispiel auch eine ID.

Hier `ID="Output0"`

Der vorläufige Quelltext der Seite sieht dann so aus:

```
<html>
<head>
<title>Web-IO</title>
</head>
<body>
```

```
<table border="1">
  <tr>
    <td width="120">Status Output 0 </td>
    <td width="50" bgcolor="#CCCCCC" id="output0">OFF</td>
    <td width="40" ><a href="javascript:setOutput(1);">ON</a></td>
    <td width="40" ><a href="javascript:setOutput(0);">OFF</a></td>
  </tr>
</table>
</body>
</html>
```

2. Schritt

Nun wird der Aufruf des Java-Applets in den body-Bereich eingefügt, wobei als spezifischer Parameter outputpolling=on angegeben wird.

```
<body>
<applet name="dio0" archive="dio.jar" code="dio.class" codebase="http://172.16.232.17"
width="0" height="0" mayscript>
    <param name="device" value="0">
    <param  name="outputpolling"  value="on">
    <param name="pollingrate" value="1000">
</applet>
<table border="1">
```

3. Schritt

Abschließend müssen noch die JavaScript-Funktionen für die Anzeige des Output-Status und das Steuern der Outputs angelegt werden.

Damit das Java-Applet den Status der Outputs an das JavaScript-Programm übergeben kann, muss die Funktion outputChanged erstellt werden.

Diese wird bei einer Änderung an den Outputs vom Java-Applet aufgerufen. Die übergebenen Variablen sind identisch mit denen der im vorherigen Beispiel vorgestellten Funktion inputsChanged, nur dass sich die Werte in diesem Fall auf den Output-Status beziehen.

Zur Erinnerung:

iDevice
ist die Nummer, die beim Aufruf des Applets als Parameter device übergeben wurde.

iNr
gibt an bei welchem Output sich der Status geändert hat.

iVal
übergibt den Status des entsprechenden Outputs.

true = ON, false = OFF
Hier die Funktion für die Anzeige des Output-Status:

```
function outputChanged( iDevice, iNr, iVal )
{   if ((iDevice==0)&(iNr==0))
    {   document.getElementById('output0')
        .firstChild.data = ( !iVal ) ? 'OFF' : 'ON';
        document.getElementById('output0')
        .style.backgroundColor= ( !iVal ) ? '#CCCCCC' : '00FF00';
    }
}
```

Auch hier wird wieder die ID (output0) des Tabellenfeldes benutzt, um das entsprechende Tabellenfeld zu verändern.

Zuletzt wird die Funktion zur Steuerung der Outputs eingefügt:

```
function setOutput(iValue )
       {   if (iValue==0)
           {   document.applets["dio0"].outputAccess( 1,0 );
           }
           else
           {   document.applets["dio0"].outputAccess( 1,1 );
           }
       }
```

Der Aufruf dieser Funktion erfolgt über die Hyperlinks in den Tabellenfeldern drei und vier.

Als Parameter *iValue* wird der Funktion der Wert 0 für Output = OFF und der Wert 1 für Output = ON übergeben.

Mit Hilfe von *iValue* wird dann über eine If-Abfrage entschieden, ob ein- oder ausgeschaltet wird.

Das Schalten selbst wird dann über Aufruf der Funktion outputAccess ausgeführt, die das Java-Applet bereitstellt.

Die Syntax des Funktionsaufrufes setzt sich folgendermaßen zusammen:

```
document.applets["Applet-Name"].outputAccess( Maske, Status )
```

document.applets[„Applet-Name“]
Dieser Teil des Aufrufes gibt an, von welchem der geladenen Applets eine Funktion aufgerufen werden soll. In unserem Fall ist zwar nur ein Applet geladen, aber grundsätzlich kann eine Webseite natürlich mit mehreren Applets zusammenarbeiten.

Als Applet-Name wird der Name eingegeben, der dem Applet beim Aufruf zugeordnet wurde. Hier also *dio0.*

outputAccess(Maske, Status)
Durch einen Punkt getrennt folgt dann der Name der Funktion.

Im Fall des Web-IO-Applets *outputAccess*. In Klammern werden dem Funktionsaufruf noch zwei Parameter übergeben.

Maske gibt an, welche Outputs geschaltet werden sollen. Mit *Status* wird der Zustand bestimmt, in den geschaltet werden soll.

Zur besseren Übersicht hier noch einmal der komplette Quelltext der Webseite:

```
<html>
<head>
<title>Web-IO</title>
<script language="JavaScript" type="text/javascript">
<!--
```

```
function outputChanged( iDevice, iNr, iVal )
{   if ((iDevice==0)&(iNr==0))
    {   document.getElementById('output0')
        .firstChild.data = ( !iVal ) ? 'OFF' : 'ON';
        document.getElementById('output0')
        .style.backgroundColor= ( !iVal ) ? '#CCCCCC' : '00FF00';
    }
}

function setOutput(iValue )
       {   if (iValue==0)
           {   document.applets["dio0"].outputAccess( 1,0 );
           }
           else
           {   document.applets["dio0"].outputAccess( 1,1 );
           }
       }
//-->
</script>
</head>
<body>
<applet name="dio0" archive="dio.jar" code="dio.class" codebase="http://172.16.232.17"
width="0" height="0" mayscript>
    <param name="device" value="0">
    <param name="outputpolling" value="off">
    <param name="pollingrate" value="1000">
</applet>
<table border="1">
  <tr>
    <td width="120">Status Output 0 </td>
    <td width="50" bgcolor="#CCCCCC" id="output0">OFF</td>
    <td width="40" ><a href="javascript:setOutput(1);">ON</a></td>
    <td width="40" ><a href="javascript:setOutput(0);">OFF</a></td>
  </tr>
</table>
</body>
</html>
```

1.3.3 Türöffner

In den beiden ersten Beispielen wurde auf rein textlicher Basis gezeigt, wie die Anzeige einer Webseite ohne komplettes Neuladen zur Laufzeit aktualisiert werden kann.

Gerade in technischen Anwendungen wird aber häufig mit grafischen Elementen gearbeitet, die das Abbild eines Prozesses visualisieren.

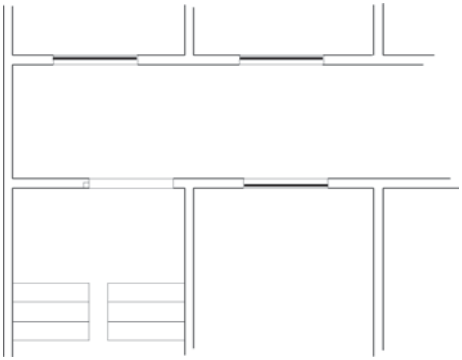
Die in diesem Beispiel gezeigte Anwendung stammt aus der Haustechnik. Mit einem Input des Web-IO Digital soll über einen Türkontakt überwacht werden, ob eine Tür offen oder geschlossen ist.

Im Browser ist dazu die Anzeige eines Grundrisses vorgesehen, in dem die Tür in ihrem aktuellen Zustand dargestellt wird.

1. Schritt

Im Vorfeld müssen die grafischen Elemente erstellt werden.

Als Hauptgrafik ist das ein Grundriss des entsprechenden Gebäudeteils:



Dieser Grundriss wird unter dem Dateinamen *sketch.gif* abgespeichert.

Es sind zusätzlich zwei weitere Grafiken notwendig, welche die Tür im geschlossenen und im offenen Zustand zeigen.

door_close.gif *door_open.gif*

Diese drei Gif-Dateien bilden die Grundlage für die Anzeige im Browser.

2. Schritt

Wie bei den vorangegangenen Beispielen muss das Java-Applet in die Webseite eingebunden werden; in diesem Fall in eine sonst völlig leere Webseite.

```
<html>
<head>
    <title>Web-IO</title>
</head>
<body>
    <applet name="dio0" archive="dio.jar" code="dio.class"
    codebase="http://172.16.232.17" width="0" height="0" mayscript>
        <param name="device" value="0">
        <param name="inputpolling" value="on">
        <param name="pollingrate" value="1000">
    </applet> </p>
</body>
</html>
```

3. Schritt

Nun muss der Grundriss an einer festen Position in die Webseite eingefügt werden. Das wird mit dem JavaScript-Befehl *document.write()* umgesetzt.

```
document.write("<img border='0' src='sketch.gif'
style ='position:absolute; top:0px; left:0px'>");
```

Durch den Parameter *style=position:absolute* wird veranlasst, dass die Grafik pixelgenau an der durch *top* und *left* spezifizierten Position angezeigt wird.

Die zwei im Folgenden gezeigten Funktionen sind sehr komplex und es würde den Rahmen dieses Buches sprengen, im Detail zu erläutern, wie genau die Funktionen arbeiten.

Wir beschränken uns an dieser Stelle darauf, zu erklären welche Wirkung mit den Funktionen erzielt wird.

```
function onoffpic(id, on_img, off_img, ypos, xpos)
    {   this.onoffpic_id = id;
        this.onoffpic_on = new Image();
        this.onoffpic_on.src = on_img;
        this.onoffpic_off = new Image();
        this.onoffpic_off.src = off_img;
        this.Set = Set;
        document.write("<img id='"+this.onoffpic_id+
        "' style=position:absolute;top:"+ypos+"px;left:"+xpos+"px src="+
        this.onoffpic_off.src+" border=0>");
    }
```

Die Funktion *onoffpic* erstellt bei Aufruf ein Grafikobjekt, bestehend aus zwei Grafiken, die wahlweise an der gleichen Position angezeigt werden können.

Bei Aufruf der Funktion werden die nötigen Parameter übergeben:

id
bestimmt, über welche ID das so erstellte Grafikobjekt nach dem Laden der Webseite angesprochen werden kann.

on_img, off_img
mit diesen Parametern wird festgelegt, welche zwei Grafiken als Quelle für die angezeigten Bilder dienen sollen.

ypos, xpos
dient der pixelgenauen Positionierung der Doppelgraphik.

Um später über JavaScript zu bestimmen, welche der beiden Grafiken angezeigt werden soll, wird für das Grafikobjekt die Methode *Set* angeboten.

Dazu muss der Quelltext die Funktion *Set* enthalten, die zum Setzen von dem durch die Funktion *onoffpic* generierten Grafikobjekt aufgerufen wird.

```
function Set(status)
{   switch(status)
    {   case false:
            document.getElementById(this.onoffpic_id).src = this.onoffpic_off.src;
        break;
        case true:
            document.getElementById(this.onoffpic_id).src = this.onoffpic_on.src;
        break;
    }
}
```

Das alles klingt zunächst sehr kompliziert, aber im Grunde reicht es aus, beim Erstellen einer neuen Objektvariablen die richtigen Parameter zu übergeben.

In unserem Fall sieht die Deklaration so aus:

```
var door = new onoffpic("10","door_open.gif","door_close.gif",264,128);
```

Die ID wurde hier willkürlich mit 10 angegeben. Als Bilder werden die Gif-Dateien der geöffneten und der geschlossenen Tür gewählt. Die Position wurde so berechnet, dass sich die Bilder der Tür in den Grundriss einpassen.

4. Schritt

Nun muss nur noch die Funktion erstellt werden, die abhängig vom Input-Zustand des Web-IO bestimmt, wie die Tür angezeigt werden soll.

```
function inputChanged( iDevice, iNr, iVal )
{   if ((iDevice==0)&(iNr==0))
    {   door.Set(iVal);
    }
}
```

Über eine If-Abfrage wird ausgewertet, ob eine Veränderung an Input 0 die Funktion aufgerufen hat.

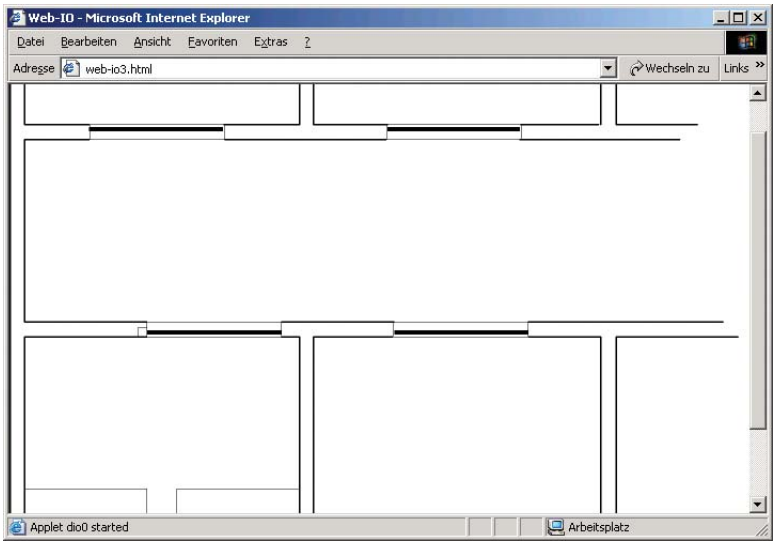
Ist das der Fall, wird die Methode Set unseres Grafikobjektes *door* aufgerufen und über *iVal* bestimmt, ob die geöffnete oder die geschlossene Tür angezeigt wird.

Zur besseren Übersicht hier noch einmal der komplette Quelltext zu diesem Beispiel:

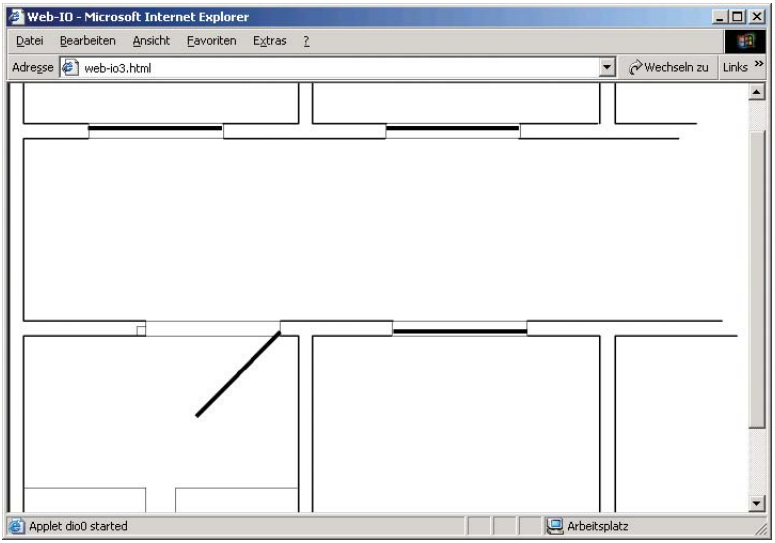
```
<html>
<head>
    <title>Web-IO</title>
    <script language="JavaScript" type="text/javascript">
    <!--
    function onoffpic(id, on_img, off_img, ypos, xpos)
    {   this.onoffpic_id = id;
        this.onoffpic_on = new Image();
        this.onoffpic_on.src = on_img;
        this.onoffpic_off = new Image();
        this.onoffpic_off.src = off_img;
        this.Set = Set;
        document.write("<img id='"+this.onoffpic_id+
        "' style=position:absolute;top:"+ypos+"px;left:"
        +xpos+"px src="+this.onoffpic_off.src+" border=0>");
    }

    function Set(status)
    {   switch(status)
        {   case false:
                document.getElementById(this.onoffpic_id).src = this.onoffpic_off.src;
            break;
            case true:
                document.getElementById(this.onoffpic_id).src = this.onoffpic_on.src;
            break;
        }
    }
    document.write("<img border='0' src='sketch.gif'
    style ='position:absolute; top:0px; left:0px'>");

    var door = new onoffpic("10","door_off.gif","door_on.gif",264,128);
```

```
    function inputChanged( iDevice, iNr, iVal )
    {   if ((iDevice==0)&(iNr==0))
        {   door.Set(iVal);
        }
    }
    //-->
    </script>
</head>
<body>
    <applet name="dio0" archive="dio.jar" code="dio.class"
        codebase="http://172.16.232.17" width="0" height="0" mayscript>
        <param name="device" value="0">
        <param name="inputpolling" value="on">
        <param name="pollingrate" value="1000">
    </applet> </p>
</body>
</html>
```

Bei geschlossener Tür ergibt sich folgende Anzeige:



Ist die Tür geöffnet, wird der angezeigte Grundriss im Bereich der Tür automatisch angepasst:



Was hier für die Visualsierung von nur einem Input umgesetzt wurde, lässt sich natürlich beliebig erweitern.

Grundrisse, Schalttafeln, Maschinenaufbauten, Prozessabbilder und vieles mehr kann mit der Kombination aus Java-Applets und JavaScript im Browser visualisiert werden.

2. Socket-Programmierung

Die in den vorherigen Kapiteln gezeigten Standard-Internet-Protokolle und Dienste bieten bereits Lösungsmöglichkeiten für diverse Anwendungen.

Oft werden aber auch speziell auf den Anwendungsfall zugeschnittene Softwarelösungen benötigt. Das kann gleichermaßen spezielle Bedien- und Eingabeoberflächen auf Anwenderebene, als auch technische Einbindung in Endgeräte und bestehende Programme betreffen.

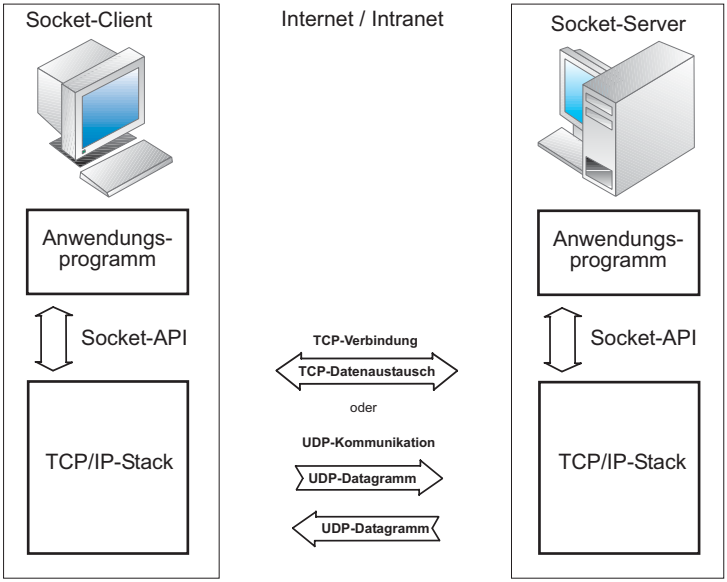
Zur Erinnerung: Den Teil eines Betriebssystems, der für die TCP/IP-Kommunikation zuständig ist, bezeichnet man als TCP/IP-Stack. Der TCP/IP-Treiber tauscht mit dem Anwendungsprogramm IP-Adressen und Ports, sowie zu übermittelnde Nutzdaten aus und stellt daraus IP-Pakete zusammen. Diese IP-Pakete werden vom TCP/IP-Stack zum physikalischen Versand an den Netzwerkkartentreiber weitergegeben.

Die eindeutige Zuordnung einer Verbindung entsteht aus dem Zusammenwirken von IP-Adresse und Port-Nummer. Man spricht bei dieser Zuordnung von einem Socket.

Wer eigene Anwendungen entwickeln möchte, die eine Kommunikation via TCP/IP unterstützen, hat mit der Socket-Programmierung nahezu uneingeschränkte Möglichkeiten.

Alle modernen Betriebssysteme verfügen heute über ein Socket-Application Interface.

Das Socket-API ist eine definierte Software-Schnittstelle, die je nach Programmiersprache und Betriebssystem über DLL-Dateien oder Controls Zugriff auf den TCP/IP-Stack erlaubt.



Eine besonders einfache Plattform für das Erstellen eigener Anwendungen bieten die Hochsprachen Visual Basic und Delphi, die hier mit kurzen Programmbeispielen vorgestellt werden.

Selbstverständlich bieten auch Programmiersprachen wie C++ und Java hervorragende Voraussetzungen für die Socket-Programmierung. Beispiele und Erklärungen hierzu finden Sie unter http://www.wut.de.

2.1 TCP-Client, TCP-Server oder UDP-Peer?

Unabhängig von der gewählten Entwicklungsumgebung, sollte anhand der Aufgabenstellung und der beteiligten Komponenten zunächst entschieden werden, welchen Teil der Kommunikation das zu erstellende Programm einnehmen soll.

2.1.1 TCP

Anwendungen, bei denen größere Datenmengen ausgetauscht werden, sollten auf Basis von TCP erstellt werden. TCP hat hier den Vorteil einer festen Verbindung, bei der die Sicherung der Daten vom TCP/IP-Stack übernommen wird.

TCP-Client

Soll das eigene Programm bestimmen, wann und mit wem Verbindung aufgenommen wird, ist es sinnvoll eine Client-Anwendung zu programmieren.

Der TCP/IP-Stack benötigt vom Anwendungsprogramm die IP-Adresse bzw. den Hostnamen des Servers und die Port-Nummer, auf der der Server eine Verbindung entgegennimmt.

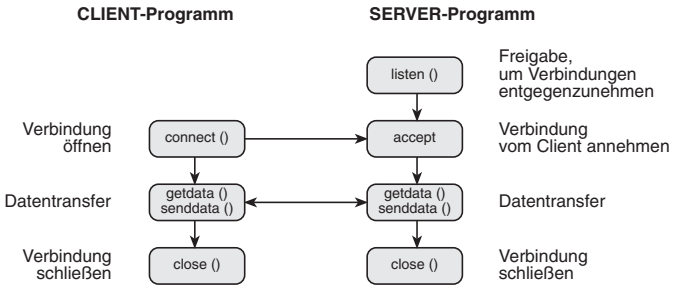
Der lokale Port, auf dem die Client-Anwendung Daten vom Server empfängt, wird in Client-Anwendungen vom TCP/IP-Stack willkürlich vergeben.

TCP-Server

Wenn die eigene Applikation dagegen einer oder mehreren anderen Anwendungen Daten und Dienste zur Verfügung stellen soll, wird ein TCP-Server programmiert.

Der TCP/IP-Stack benötigt vom Anwendungsprogramm die Information, auf welchem lokalen Port Verbindungswünsche einer Client-Anwendung entgegengenommen werden sollen.

Liegt ein Verbindungswunsch vor, informiert der Stack das Programm. Akzeptiert das Anwendungsprogramm die Verbindung, können Daten gesendet und empfangen werden.



2.1.2 UDP

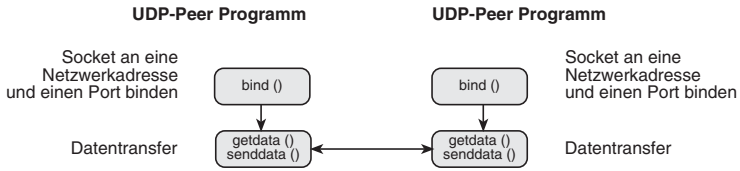
Bei Netzwerkanwendungen mit wechselnden Partnern oder nur kurzen Datensendungen ist UDP als verbindungsloses Protokoll vorzuziehen.

Auf UDP-Ebene sind beide Kommunikationspartner gleichberechtigt. Man unterscheidet nicht zwischen Client und Server. Da keine Zeit für Verbindungsauf- und -abbau verloren geht, können bei kleineren Datenmengen deutlich schnellere Zugriffszeiten erreicht werden.

Es sollte aber beachtet werden, dass die Sicherung der Daten in der eigenen Applikation realisiert werden muss.

Der TCP/IP-Stack benötigt vom Anwendungsprogramm die IP-Adresse bzw. den Hostnamen und die Port-Nr. des Kommunikationspartners sowie die eigene Port-Nummer.

Durch die Zuordnung dieser Parameter entsteht ein Socket, über das Daten gesendet und empfangen werden können.



2.2 Socket-Programmierung in Visual Basic

Die vorgestellten Beispiele wurden in Visual Basic 5 programmiert, funktionieren aber unter VB6 genauso.

Alle, die über Grundkenntnisse in VB-Programmierung verfügen, sollten den Programmbeispielen leicht folgen können.

Um in VB TCP/IP basierende Anwendungen erstellen zu können, muss zunächst das Winsock-Control in die Komponentenliste aufgenommen werden.

- Mit der rechten Maustaste in die Komponentenleiste klicken
- Menüpunkt „Komponenten“ mit linker Maustaste auswählen
- In der Liste der Steuerelemente das Häkchen für „Microsoft Winsock Control“ setzen

Die Komponentenleiste ist nun um das Winsock-Steuerelement bereichert. 

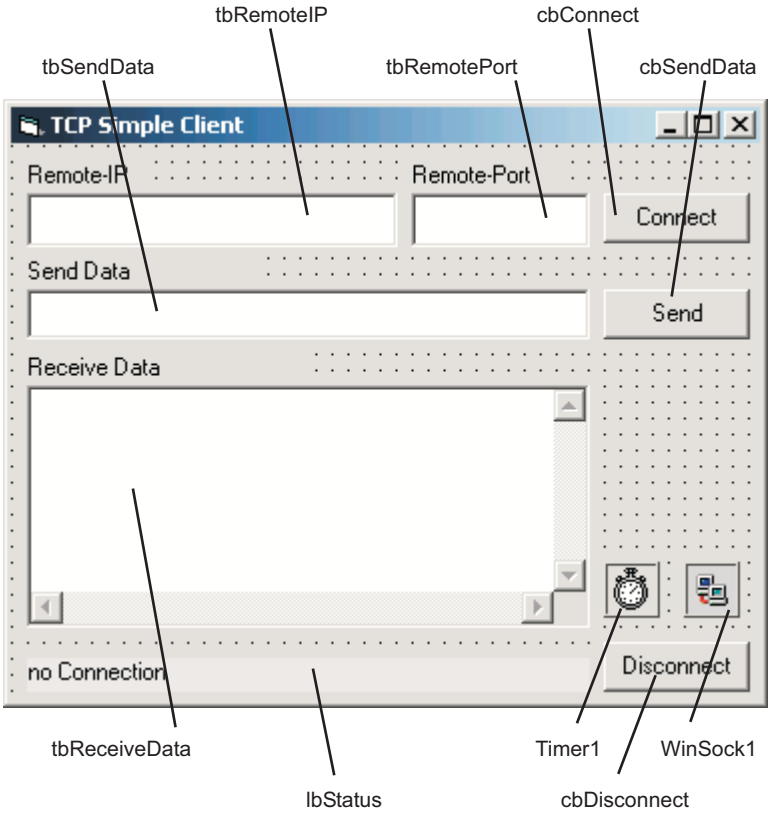
2.2.1 Ein TCP-Client in VB

Als erstes wollen wir einen TCP-Client erstellen, der folgende Aufgaben übernimmt:

(das komplette Beispiel steht unter http://www.wut.de zum Download zur Verfügung)

- Aufbau der TCP-Verbindung
- Senden und Empfangen von Textdaten
- Schließen der TCP-Verbindung
- Anzeigen des Verbindungsstatus
- Erkennen von Fehlern

Hierzu wird ein Formular mit folgenden Elementen aufgebaut:



Alle Variablen und Elementnamen sollten sich durch die gewählte Namensgebung selbst erklären.

Für Elemente vom Typ Textbox wurden mit „tb“ beginnende Namen gewählt, Command-Buttons beginnen dagegen mit „cb“.

Der gezeigte VB-Quelltext kommt deshalb auch weitestgehend ohne Kommentare aus:

Die folgende Prozedur setzt nach Klick auf den Connect-Button die vom Anwender eingegebenen Adressierungsparameter und baut, mit Hilfe der Connect-Methode des Winsock-Controls, die TCP-Verbindung auf.

```
Private Sub cbConnect_Click()
  If (tbRemotePort.Text <> "") And (tbRemoteIP.Text <> "") Then
    Winsock1.RemotePort = val(tbRemotePort.Text)
    Winsock1.RemoteHost = tbRemoteIP.Text
    Winsock1.Connect
  End If
End Sub
```

Prozedur zum Trennen der TCP-Verbindung mit Hilfe der Close-Methode.

```
Private Sub cbDisconnect_Click()
  Winsock1.Close
  cbConnect.Enabled = True
  cbConnect.SetFocus
End Sub
```

Durch Klicken auf den Send-Button wird der vom Anwender eingegebene Text über die bestehende TCP-Verbindung versandt. Hierzu wird die Senddata Methode benutzt.

```
Private Sub cbSendData_Click()
  Winsock1.SendData (tbSendData.Text)
  tbSendData.Text = ""
End Sub
```

Timerroutine überwacht den aktuellen Verbindungsstatus über die State Eigenschaft des Winsock-Controls. Ein sinnvoller Intervall für den Timer ist 500ms.

```
Private Sub Timer1_Timer()
  Select Case Winsock1.State
    Case ckClosed
      lbStatus.Caption = "no Connection"
```

```
    Case sckResolvingHost
      lbStatus.Caption = "waiting for DNS"
    Case sckHostResolved
      lbStatus.Caption = "get IP from DNS"
    Case sckConnecting
      lbStatus.Caption = "connecting"
    Case sckConnected
      lbStatus.Caption = "Connection to " + Winsock1.RemoteHost
    Case sckClosing
      lbStatus.Caption = "closing Connection"
    Case sckError
      lbStatus.Caption = "Connection Error"
      Winsock1.Close
    End Select
    If Winsock1.State <> sckConnected Then
      cbSendData.Enabled = False
      cbDisconnect.Enabled = False
      cbConnect.Enabled = True
    Else
      cbSendData.Enabled = True
      cbDisconnect.Enabled = True
      cbConnect.Enabled = False
    End If
End Sub
```

Diese Prozedur wird automatisch aufgerufen, wenn eine Verbindung von der Gegenseite beendet wird und kann z.B. genutzt werden, um über die Close-Methode auch die eigene Verbindungsverwaltung zurückzusetzen.

```
Private Sub Winsock1_Close()
  Winsock1.Close
  cbConnect.Enabled = True
  cbDisconnect.Enabled = False
  cbSendData.Enabled = False
  lbStatus.Caption = "no Connection"
  cbConnect.SetFocus
End Sub
```

Diese Prozedur wird automatisch bei erfolgreichem Verbindungsaufbau aufgerufen.

```
Private Sub Winsock1_Connect()
  cbDisconnect.Enabled = True
  cbConnect.Enabled = False
```

```
  tbReceiveData.Text = ""
  lbStatus.Caption = "connected to " + Winsock1.RemoteHost
End Sub
```

Bei Datenempfang wird automatisch diese Prozedur durchlaufen.

Eingehende Daten werden mit der Getdata-Methode entgegengenommen und im „Receive Data" Fenster angezeigt.

```
Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)

  Dim ReceiveData As String
  Winsock1.GetData ReceiveData
  tbReceiveData.Text = tbReceiveData.Text + ReceiveData
End Sub
```

Prozedur zur Behandlung von Verbindungsfehlern

```
Private Sub Winsock1_Error(ByVal Number As Integer,_
Description As String, ByVal Scode As Long,_
ByVal Source As String, ByVal HelpFile As String,_
ByVal HelpContext As Long, CancelDisplay As Boolean)
  Winsock1.Close
  dummy = MsgBox("Connection Error", vbOKOnly, "TCP simple Client")
End Sub
```

So hat man mit weniger als 2 Seiten Quelltext bereits einen TCP-Client mit Statusanzeige und Fehlerbehandlung programmiert.

Als Gegenstück wird natürlich ein Server benötigt, der den Verbindungswunsch des Clients annimmt. Das kann ein vorhandenes Gerät, wie z.B. ein W&T Com-Server, oder eine weitere VB-Server-Applikation sein.

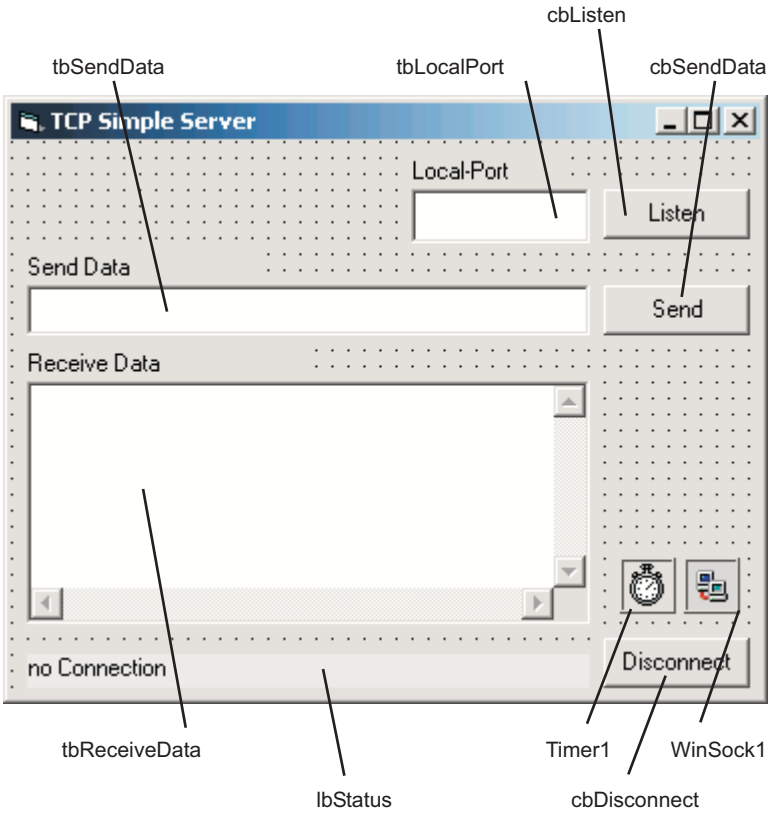
Wie ein TCP-Server aufgebaut werden kann, soll im folgenden Beispiel gezeigt werden.

2.2.2 Ein TCP-Server in VB

Die Server-Applikation übernimmt folgende Aufgaben:

- Auf dem Netz „horchen", ob es auf dem unterstützten Port einen Verbindungswunsch gibt
- Gewünschte Verbindung entgegennehmen
- Senden und Empfangen von Textdaten
- Anzeigen des Verbindungsstatus
- Anzeigen von Verbindungsfehlern
- Schließen der Verbindung von der Server-Seite

Hierzu wird ein Formular mit folgenden Elementen aufgebaut:



Alle Variablen und Elementnamen sollten sich auch hier durch die gewählte Namensgebung selbst erklären.

Die folgenden VB-Prozeduren werden für den Server benötigt:

Diese Prozedur wird für eine Server-Anwendung nicht unbedingt benötigt und steht nur am Anfang des Quelltextes, weil VB die Prozeduren alphabetisch sortiert anzeigt. Sie erlaubt es, mit der Close-Methode eine bestehende Verbindung zu schließen. Mit der Listen-Methode beginnt das Winsock-Control erneut auf etwaige Verbindungswünsche auf den gewählten Port zu horchen.

```
Private Sub cbDisconnect_Click()
  Winsock1.Close
  Winsock1.Listen
End Sub
```

Mit Aufruf der Listen-Methode beginnt das Winsock-Control auf etwaige Verbindungswünsche auf den gewählten Port zu horchen.

```
Private Sub cbListen_Click()
If tbLocalPort.Text <> "" Then
    Winsock1.LocalPort = tbLocalPort.Text
    Winsock1.Listen
    cbListen.Enabled = False
  End If
End Sub
```

Durch Klicken auf den Send-Button wird der vom Anwender eingegebene Text über die bestehende TCP-Verbindung versandt. Hierzu wird die Senddata Methode benutzt.

```
Private Sub cbSendData_Click()
  Winsock1.SendData (tbSendData.Text)
  tbSendData.Text = ""
End Sub
```

Die Timerroutine überwacht den aktuellen Verbindungsstatus über die State-Eigenschaft des Winsock-Controls. Ein sinnvolles Intervall für den Timer ist 500ms.

```
Private Sub Timer1_Timer()
  Select Case Winsock1.State
    Case ckClosed
      lbStatus.Caption = "no Connection"
    Case sckListening
      lbStatus.Caption = "listening for connection"
    Case sckConnectionPending
      lbStatus.Caption = "Connection Pending"
    Case sckConnecting
      lbStatus.Caption = "connecting"
    Case sckConnected
```

```
      lbStatus.Caption = „Connection to „ + Winsock1.RemoteHostIP
    Case sckError
      lbStatus.Caption = "Connection Error"
      Winsock1.Close
    End Select
    If Winsock1.State <> sckConnected Then
      cbSendData.Enabled = False
      cbDisconnect.Enabled = False
    Else
      cbSendData.Enabled = True
      cbDisconnect.Enabled = True
    End If
End Sub
```

Diese Prozedur wird automatisch aufgerufen, wenn eine Verbindung von der Client-Seite beendet wird. Über die Close-Methode wird die eigene Verbindungsverwaltung zurückgesetzt. Mit Aufruf der Listen-Methode beginnt das Winsock-Control erneut, auf etwaige Verbindungswünsche auf den gewählten Port zu horchen.

```
Private Sub Winsock1_Close()
  Winsock1.Close
  Winsock1.Listen
End Sub
```

Erkennt das Winsock-Steuerelement den Verbindungswunsch eines Client, wird automatisch diese Prozedur ausgeführt. Mit der Accept-Methode wird die Verbindung entgegengenommen.

```
Private Sub Winsock1_ConnectionRequest(ByVal requestID As Long)
  If Winsock1.State <> sckclose Then Winsock1.Close
  Winsock1.Accept requestID
End Sub
```

Bei Datenempfang wird automatisch diese Prozedur durchlaufen.
Eingehende Daten werden mit der Getdata-Methode entgegengenommen und im „Receive Data" Fenster angezeigt.

```
Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
  Dim ReceiveData As String
  Winsock1.GetData ReceiveData
  tbReceiveData.Text = tbReceiveData.Text + ReceiveData
End Sub
```

Prozedur zur Behandlung von Verbindungsfehlern

```
Private Sub Winsock1_Error(ByVal Number As Integer, Description As String, ByVal Scode As
Long, ByVal Source As String, ByVal HelpFile As String, ByVal HelpContext As Long,
CancelDisplay As Boolean)
  Winsock1.Close
  Winsock1.LocalPort = 0
  dummy = MsgBox("Connection Error", vbOKOnly, "TCP simple Server")
End Sub
```

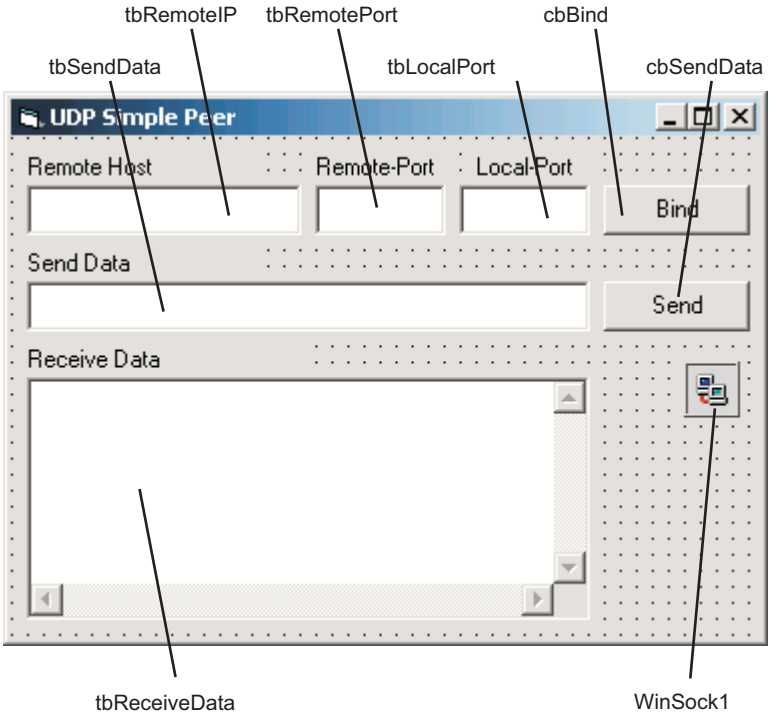
Auf den so programmierten Server kann man mit dem vorgestellten Client-Programm zugreifen. Aber auch beliebige andere Client-Anwendungen, z.B. der W&T Com-Server im Client-Mode, können bei Wahl des entsprechenden Ports Verbindung mit dem Server aufnehmen.

2.2.3 Ein einfacher UDP-Peer in VB

Die UDP-Applikation übernimmt folgende Aufgaben:

- IP-Adresse und Ports zu einem Socket binden
- Senden und Empfangen von Textdaten

Hierzu wird ein Formular mit folgenden Elementen aufgebaut:



Alle Variablen und Elementnamen sollten sich auch hier durch die gewählte Namensgebung selbst erklären.

Der folgende Quelltext kommt deshalb auch weitestgehend ohne Kommentare aus:

Mit der Bind-Methode werden Ip-Adresse und Ports in einem Socket gebunden

```
Private Sub cbBind_Click()
  Winsock1.Protocol = sckUDPProtocol
  Winsock1.RemotePort = val(tbRemotePort.Text)
```

```
  Winsock1.RemoteHost = tbRemoteIP.Text
  Winsock1.Bind val(tbLocalPort.Text)
  cbBind.Enabled = False
  cbSendData.Enabled = True
End Sub
```

Durch Klicken auf den Send-Button wird der vom Anwender eingegebene Text als UDP-Datagramm versand. Hierzu wird die Senddata Methode benutzt. Bedingung hierfür ist, dass über die Bind-Methode ein Socket gebunden wurde.

```
Private Sub cbSendData_Click()
  Winsock1.SendData (tbSendData.Text)
  tbSendData.Text = ""
End Sub
```

Bei Datenempfang wird automatisch diese Prozedur durchlaufen.
Eingehende Daten werden mit der Getdata-Methode entgegengenommen und im „Receive Data" Fenster angezeigt.

```
Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
  Dim ReceiveData As String
  Winsock1.GetData ReceiveData
  tbReceiveData.Text = tbReceiveData.Text + ReceiveData
End Sub
```

Um mit dem UDP-Peer Datenkommunikation zu betreiben, kann der gleiche Peer auf einem zweiten PC gestartet werden. Ebenso ist es aber auch möglich, über den UDP-Peer mit einem W&T Com-Server zu kommunizieren, der als UDP-Client konfiguriert ist.

Der hier gezeigte UDP-Peer verzichtet auf jede Form von Datensicherheit. Das bedeutet: Werden Daten an eine nicht existente IP-Adresse geschickt oder das adressierte Endgerät ist nicht betriebsbereit, laufen die Daten einfach ins Leere, ohne dass der Anwender etwas merkt.

2.3 Socket-Programmierung in Delphi

Die hier vorgestellten Beispiele wurden in der Delphi 5 Standardversion erstellt.

Alle, die über Grundkenntnisse in Delphi-Programmierung verfügen, sollten den Programmbeispielen leicht folgen können.

Delphi 5 stellt im Register „Internet" Standard-Steuerelemente zur Socket-Programmierung zur Verfügung.

Im Gegensatz zu Visual Basic, wo nur ein Steuerelement durch unterschiedliche Parametrierung dem gewünschten Einsatz angepasst werden kann, bietet Delphi zwei spezifische Steuerelemente an:

Steuerelement für TCP-Client (ClientSocket) 

Steuerelement für TCP-Server (ServerSocket) 

Ein Steuerelement für UDP-Anwendungen ist in der Standardversion von Delphi 5 leider nicht vorhanden.

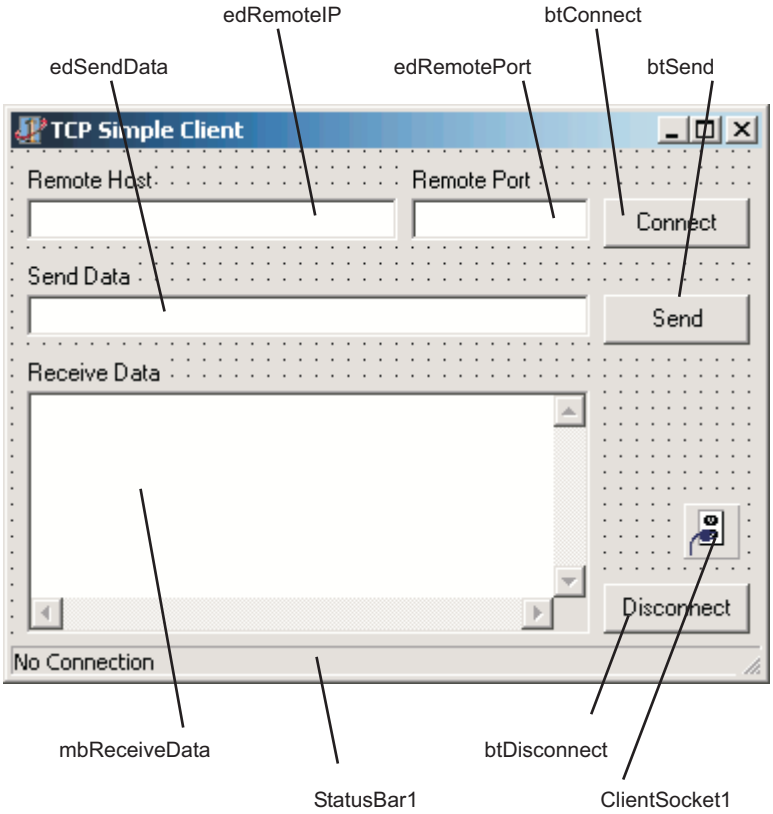
2.3.1 Ein TCP-Client in Delphi

Als erstes wollen wir einen TCP-Client erstellen, der folgende Aufgaben übernimmt:

(das komplette Beispiel steht unter http://www.wut.de zum Download zur Verfügung)

- Aufbau der TCP-Verbindung
- Senden und Empfangen von Textdaten
- Schließen der TCP-Verbindung
- Anzeigen des Verbindungsstatus
- Erkennen von Fehlern

Hierzu wird ein Formular mit folgenden Elementen aufgebaut:



Alle Variablen und Elementnamen sollten sich durch die gewählte Namensgebung selbst erklären.

Für Elemente vom Typ Edit wurden mit „ed" beginnende Namen gewählt, Buttons beginnen dagegen mit „bt" und Memoboxen mit „mb"

Der gezeigte Quelltext kommt deshalb auch weitestgehend ohne Kommentare aus:

Der erste Teil des Quelltextes wird von Delphi beim Entwerfen des Formulares selbst erstellt, und dient der Deklaration aller beteiligten Elemente.

```
unit TCP_Client;

interface
```

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ScktComp, StdCtrls, ComCtrls;

type
  TTCPClient = class(TForm)
    edRemoteIP: TEdit;
    btConnect: TButton;
    edRemotePort: TEdit;
    edSendData: TEdit;
    btSend: TButton;
    mbReceiveData: TMemo;
    btDisconnect: TButton;
    StatusBar1: TStatusBar;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    ClientSocket1: TClientSocket;
    procedure btConnectClick(Sender: TObject);
    procedure btSendClick(Sender: TObject);
    procedure OnConnect(Sender: TObject; Socket: TCustomWinSocket);
    procedure btDisconnectClick(Sender: TObject);
    procedure OnDisconnect(Sender: TObject; Socket: TCustomWinSocket);
    procedure OnRead(Sender: TObject; Socket: TCustomWinSocket);
    procedure OnError(Sender: TObject; Socket: TCustomWinSocket;
      ErrorEvent: TErrorEvent; var ErrorCode: Integer);
  private
    { Private-Deklarationen }
  public
    { Public-Deklarationen }
  end;

var
  TCPClient: TTCPClient;

implementation
{$R *.DFM}
```

Hier beginnt das eigentliche Programm.
Die folgende Prozedur setzt nach Klick auf den Connect-Button, die vom Anwender eingegebenen

Adressierungsparameter und baut durch Aktivieren des Winsock-Controls die TCP-Verbindung auf.

```
procedure TTCPClient.btConnectClick(Sender: TObject);
begin
   ClientSocket1.Host := edRemoteIP.Text;
   ClientSocket1.Port := strtoint(edRemotePort.Text);
   ClientSocket1.Active := True;
end;
```

Diese Prozedur wird automatisch bei erfolgreichem Verbindungsaufbau aufgerufen.

```
procedure TTCPClient.OnConnect(Sender: TObject; Socket: TCustomWinSocket);
begin
  btSend.Enabled := True;
  btDisconnect.Enabled := True;
  btConnect.Enabled := False;
  mbReceiveData.Clear;
  Statusbar1.SimpleText := 'Connected to ' + ClientSocket1.Host;
end;
```

Diese Prozedur wird automatisch bei Verbindungsabbau aufgerufen.

```
procedure TTCPClient.OnDisconnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  btSend.Enabled := False;
  btDisconnect.Enabled := False;
  btConnect.Enabled := True;
  Statusbar1.SimpleText := 'No Connection';
end;
```

Prozedur zur automatischen Fehlerbehandlung

```
procedure TTCPClient.OnError(Sender: TObject; Socket: TCustomWinSocket;
  ErrorEvent: TErrorEvent; var ErrorCode: Integer);
begin
  ShowMessage ('Connection Error');
  ClientSocket1.Active := False;
  btSend.Enabled := False;
  btDisconnect.Enabled := False;
  btConnect.Enabled := True;
  Statusbar1.SimpleText := 'No Connection';
end;
```

Durch klicken auf den Send-Button wird der vom Anwender eingegebene Text über die bestehende TCP-Verbindung versand.

```
procedure TTCPClient.btSendClick(Sender: TObject);
begin
  ClientSocket1.Socket.SendText (edSendData.Text);
  edSendData.Text := '';
end;
```

Bei Datenempfang wird automatisch diese Prozedur durchlaufen.
Eingehende Daten werden entgegengenommen und im „Receive Data" Fenster angezeigt.

```
procedure TTCPClient.OnRead(Sender: TObject; Socket: TCustomWinSocket);
begin
  mbReceiveData.Text := mbReceiveData.Text + ClientSocket1.Socket.ReceiveText;
end;
```

Prozedur zum Trennen der TCP-Verbindung durch Deaktivierung des ClientSocket Steuerelements.

```
procedure TTCPClient.btDisconnectClick(Sender: TObject);
begin
  ClientSocket1.Active := False;
end;

end.
```

So hat man auch in Delphi mit weniger als 2 Seiten Quelltext bereits einen TCP-Client mit Statusanzeige und Fehlerbehandlung programmiert.

Als Gegenstück wird natürlich ein Server benötigt, der den Verbindungswunsch des Clients annimmt. Das kann ein vorhandenes Gerät wie, z.B. ein W&T Com-Server, oder eine weitere Delphi-Server Applikation sein.

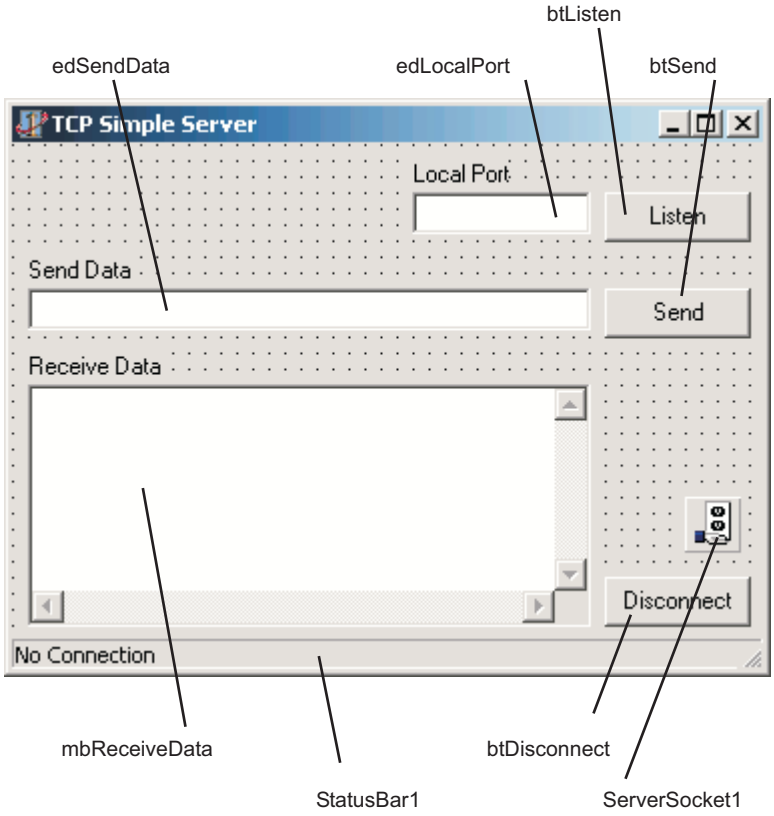
Wie ein TCP-Server in Delphi aufgebaut werden kann, soll im folgenden Beispiel gezeigt werden.

2.3.2 Ein TCP-Server in Delphi

Die Server-Applikation übernimmt folgende Aufgaben:

- Auf dem Netz „Horchen", ob es auf dem unterstützten Port einen Verbindungswunsch gibt
- Gewünschte Verbindung entgegennehmen
- Senden und Empfangen von Textdaten
- Anzeigen des Verbindungsstatus
- Anzeigen von Verbindungsfehlern
- Schließen der Verbindung von der Server-Seite (gehört nicht zu den typischen Aufgaben eines Servers, ist mit dem Programm aber auch möglich)

Hierzu wird ein Formular mit folgenden Elementen aufgebaut:



Alle Variablen und Elementnamen sollten sich auch hier durch die gewählte Namensgebung selbst erklären.

Die folgenden Delphi-Prozeduren werden für den Server benötigt:

Wie schon bei der Client-Anwendung, wird der erste Teil des Quelltextes von Delphi beim Entwerfen des Formulares selbst erstellt und dient der Deklaration aller beteiligten Elemente.

```
unit TCP_Server;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ScktComp, StdCtrls, ComCtrls;


type
  TTCPServer = class(TForm)
    btListen: TButton;
    edLocalPort: TEdit;
    edSendData: TEdit;
    btSend: TButton;
    mbReceiveData: TMemo;
    btDisconnect: TButton;
    StatusBar1: TStatusBar;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    ServerSocket1: TServerSocket;
    procedure btListenClick(Sender: TObject);
    procedure btSendClick(Sender: TObject);
    procedure btDisconnectClick(Sender: TObject);
    procedure OnListen(Sender: TObject; Socket: TCustomWinSocket);
    procedure OnAccept(Sender: TObject; Socket: TCustomWinSocket);
    procedure OnClientRead(Sender: TObject; Socket: TCustomWinSocket);
    procedure OnClientDisconnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure OnClientError(Sender: TObject; Socket: TCustomWinSocket;
      ErrorEvent: TErrorEvent; var ErrorCode: Integer);
  private
```

```
    { Private-Deklarationen }
  public
    { Public-Deklarationen }
  end;

var
  TCPServer: TTCPServer;

implementation

{$R *.DFM}
```

Hier beginnt das eigentliche Programm:
Durch Klick auf den Listen Button wird das ServerSocket Steuerelement geöffnet und beginnt auf etwaige Verbindungswünsche auf den gewählten Port zu horchen.

```
procedure TTCPServer.btListenClick(Sender: TObject);
begin
  If edLocalPort.Text <> '' Then
  begin
    ServerSocket1.Port := strtoint(edLocalPort.Text);
    ServerSocket1.Open;
  end
  Else ShowMessage ('No local port!');
end;
```

Diese Prozedur wird automatisch aufgerufen, wenn das ServerSocket Steuerelement geöffnet wurde und auf Verbindungswünsche wartet.

```
procedure TTCPServer.OnListen(Sender: TObject; Socket: TCustomWinSocket);
begin
  Statusbar1.SimpleText := 'Listening';
  btSend.Enabled := False;
  btDisconnect.Enabled := False;
  btListen.Enabled := False;
end;
```

Das Entgegennehmen von Verbindungen erledigt das ServerSocket Steuerelement automatisch im Hintergrund. Wird eine Verbindung entgegengenommen, durchläuft das Programm automatisch die folgende Prozedur.

```
procedure TTCPServer.OnAccept(Sender: TObject; Socket: TCustomWinSocket);
begin
  Statusbar1.SimpleText := 'Connected to ' + Socket.RemoteAddress;
```

```
  btSend.Enabled := True;
  btDisconnect.Enabled := True;
end;
```

Diese Prozedur wird automatisch bei Verbindungsabbau aufgerufen.

```
procedure TTCPServer.OnClientDisconnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  Statusbar1.SimpleText := 'Listening';
  ServerSocket1.Open;
  btSend.Enabled := False;
  btDisconnect.Enabled := False;
end;
```

Prozedur zur automatischen Fehlerbehandlung

```
procedure TTCPServer.OnClientError(Sender: TObject;
  Socket: TCustomWinSocket; ErrorEvent: TErrorEvent;
  var ErrorCode: Integer);
begin
  ShowMessage ('Connection Error');
  ErrorCode := 0;
  ServerSocket1.Close;
  btSend.Enabled := False;
  btDisconnect.Enabled := False;
  btListen.Enabled := True;
  Statusbar1.SimpleText := 'No Connection';
end;
```

Durch Klicken auf das Send-Button wird der vom Anwender eingegebene Text über die bestehende TCP-Verbindung versand.

```
procedure TTCPServer.btSendClick(Sender: TObject);
begin
  ServerSocket1.Socket.Connections[0].SendText(edSendData.Text);
  edSendData.Text := '';
end;
```

Bei Datenempfang wird automatisch diese Prozedur durchlaufen.
Eingehende Daten werden entgegengenommen und im „Receive Data" Fenster angezeigt.

```
procedure TTCPServer.OnClientRead(Sender: TObject;
  Socket: TCustomWinSocket);
```

```
begin
  mbReceiveData.Text := mbReceiveData.Text + Socket.ReceiveText;
end;
```

Prozedur zum Trennen der TCP-Verbindung durch Schließen des ServerSocket Steuerelements.

```
procedure TTCPServer.btDisconnectClick(Sender: TObject);
begin
  ServerSocket1.Close;
  btListen.Enabled := True;
  btSend.Enabled := False;
  btDisconnect.Enabled := False;
  Statusbar1.SimpleText := 'No Connection';
end;

end.
```

Auf den so programmierten Server kann man mit dem vorgestellten Client-Programm zugreifen. Aber auch beliebige andere Client-Anwendungen, z.B. der W&T Com-Server im Client-Mode, können bei Wahl des entsprechenden Ports Verbindung mit dem Server aufnehmen.

Wer mit Delphi UDP-Anwendungen programmieren möchte, hat die Möglichkeit, Steuerelemente von Drittanbietern einzusetzen.

Ein Beispiel hierfür ist das Internet Steuerelement des Belgiers Francois Piette, das unter http://www.overbyte.be kostenfrei zum Download bereitliegt.

Die gezeigten Beispiele sind als Anregung gedacht und sollen zum Ausprobieren und Spielen mit der Datenübertragung via TCP/IP einladen. Die Quelltexte können leicht durch Modifikation an eventuelle Anwendungswünsche angepasst werden.

2.4 Socket-Programmierung in Visual Basic .NET

Mit .Net (gesprochen Dot Net) hat die Philosophie der objektorientierten Programmierung auch in Visual Basic vollständig Einzug gehalten.

Auch wenn die Erstellung eines Programmes grundsätzlich immer geplant und strukturiert erfolgen sollte, haben die früheren Versionen von VB vor allem dem nicht routinierten Programmierer eine Plattform geboten, einfach drauflos zu programmieren.

Visual Basic.Net verlangt dem Programmierer ein deutlich höheres Maß an Disziplin ab.

Im Rahmen dieses Buches können wir natürlich nicht die Philosophie erklären, die hinter der .Net Programmierung steckt.

Allen die über Grundkenntnisse in VB-Programmierung verfügen, soll im Speziellen aufgezeigt werden, wie TCP/IP-Netzwerkkommunikation in eigene Programme integriert werden kann.

Um in VB.NET TCP/IP basierende Anwendungen erstellen zu können, reicht es leider nicht mehr aus, dem eigenen Projekt das Winsock-Control hinzuzufügen. Grundsätzlich wäre das zwar möglich, die Programmierumgebung von VB.Net stellt dieses Steuerelement aber nicht mehr zur Verfügung.

Die Socket-Klasse

Für die Abwicklung der TCP/IP-Kommunikation wird in den drei folgenden Beispielen mit einer Instanz (einer Art eigenen Kopie) der Klasse *System.Net.Sockets.Socket* gearbeitet.

Um mit der Socket-Klasse arbeiten zu können, muss in den Programmkopf ein entsprechender *Imports*-Aufruf für die übergeordnete Klasse *System.Net.Sockets* angegeben werden:

```
Imports System.Net.Sockets
```

Das Anlegen einer Socket Instanz erfolgt dann über *Dim*:

Beispiel: `Dim TCP_Client as Socket`

IPEndPoint
In Verbindung mit der Socket-Klasse werden IP-Adresse und Portnummer nicht mehr als einzelne Parameter verwendet. Statt dessen wird beides zu dem neuen Parameter IPEndPoint zusammengefasst.

Beispiel: `Dim ServerEP as new IPEnd Point(<IP-Adresse>,<Port>)`

Behandlung von Ereignissen
Ein weiterer Unterschied zum Winsock-Steuerelement liegt in der Behandlung von Ereignissen (Datenempfang, Verbindungsabbuch usw.).

Das Winsock-Steuerelement hat für alle Ereignisse Prozeduren angeboten, in die der Quelltext eingetragen werden konnte, der ausgeführt wird, wenn das entsprechende Ereignis eintritt.

In VB.Net muss der Programmierer die Behandlung von Ereignissen als Callback-Prozeduren selber in das Programm einfügen.

Für alle Socketfunktionen die ein Ereignis nach sich ziehen gibt es *Begin*-Methoden, in denen festgelegt wird welche Callback-Prozedur aufgerufen wird, wenn das Ereignis auftritt.

Beispiel: `TCP_Client.BeginReceive(.........)`

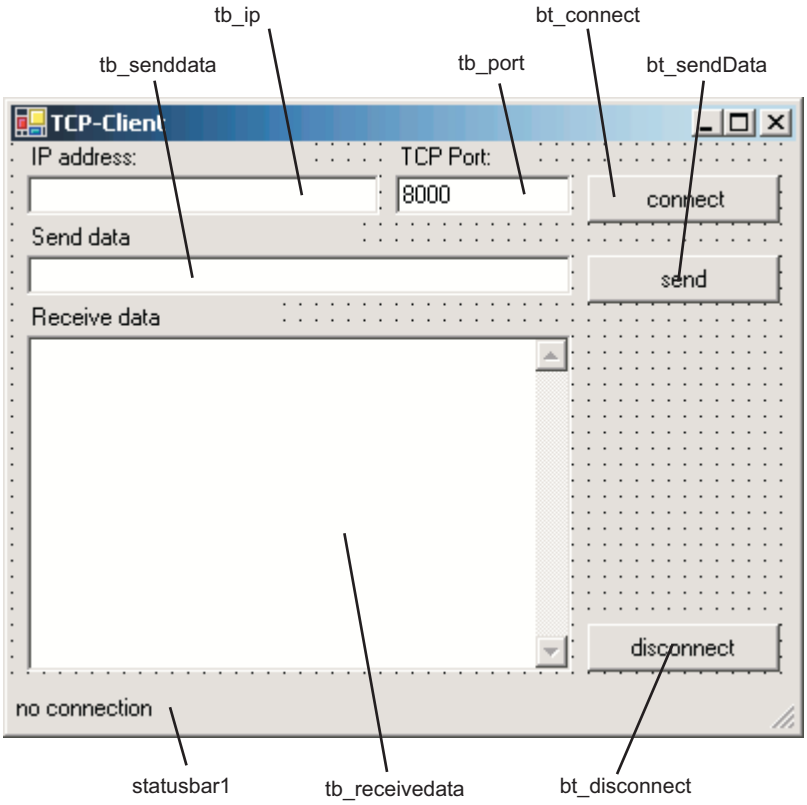
2.4.1 Ein TCP-Client in VB.Net

Als erstes wollen wir einen TCP-Client erstellen, der folgende Aufgaben übernimmt:

(das komplette Beispiel, steht unter http://www.wut.de zum Download zur Verfügung)

- Aufbau der TCP-Verbindung
- Senden und Empfangen von Textdaten
- Schließen der TCP-Verbindung
- Anzeigen des Verbindungsstatus
- Erkennen von Fehlern

Hierzu wird ein Formular mit folgenden Elementen aufgebaut:



Alle Variablen und Elementnamen sollten sich durch die gewählte Namensgebung selbst erklären.

Für Elemente vom Typ Textbox wurden mit „tb“ beginnende Namen gewählt, Command-Buttons beginnen dagegen mit „bt“.

Der gezeigte VB.Net-Quelltext kommt deshalb auch mit nur wenigen Kommentaren aus.

Hier der Quelltext, in dem der von VB.Net automatisch generierte Quellcode nicht abgedruckt ist:

Über das Import-Statement werden alle Namespaces der verwendeten Klassen angegeben. Im Quelltext der

eigenen Klasse muss dann z.B. wenn eine neue Instanz der Klasse System.Nets.Sockets.Socket benutzt werden soll nur noch Dim TCP_client As Socket eingegeben werden.

```
Imports System.Threading
Imports System.Net
Imports System.Net.Sockets
Imports System.IO
Imports System.Windows.Forms

Public Class Form1
    Inherits System.Windows.Forms.Form
```

Alle global benutzten Variablen, Objekte und Klasseninstanzen müssen zunächst dimensioniert weden.

```
Dim TCP_client As Socket
Dim receivebuffer(511) As Byte
```

.....

Hier befindet sich im Original der von VB.Net automatisch generierte Quelltext

.....

Prozedur zum Öffnen der TCP-Verbindung

```
Private Sub bt_connect_Click(ByVal sender As System.Object, _
                             ByVal e As System.EventArgs) Handles bt_connect.Click

  If tb_ip.Text <> "" And tb_port.Text <> "" Then
    Dim webioep As New IPEndPoint(IPAddress.Parse(tb_ip.Text), Val(tb_port.Text))
    TCP_client = New Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp)
    bt_connect.Enabled = False
    Try
      TCP_client.BeginConnect(webioep, _
                          New AsyncCallback(AddressOf callback_connect), TCP_client)
    Catch ex As Exception
      closeconnections()
      StatusBar1.Text = "ERROR no Connection"
    End Try
  End If
End Sub
```

Callback-Prozedur, die aufgerufen wird, wenn die TCP-Verbindung zustande kommt

```
Private Sub callback_connect(ByVal ar As IAsyncResult)
```

```
  bt_connect.Enabled = False
  bt_senddata.Enabled = True
  bt_disconnect.Enabled = True
  StatusBar1.Text = „connected“
  Try
    TCP_client.EndConnect(ar)
    TCP_client.BeginReceive(receivebuffer, 0, 512, SocketFlags.None, _
                        New AsyncCallback(AddressOf callback_readdata), TCP_client)
  Catch ex As Exception
    closeconnections()
    StatusBar1.Text = “error on connecting“
  End Try
End Sub

Callback-Prozedur, die aufgerufen wird, wenn Daten vom Netzwerk empfangen werden
Private Sub callback_readdata(ByVal ar As IAsyncResult)
  If TCP_client.Connected Then
    Dim bytesread As Integer
  Try
    bytesread = TCP_client.EndReceive(ar)
  Catch ex As Exception
  End Try
  If bytesread = 0 Then     'wenn die Callback-Prozedur aufgerufen wird, ohne dass Daten
                            'empfangen wurden, hat die Gegenseite die Verbindung geschlossen
    closeconnections()
  Else
    Dim receivestring As String
    receivestring = System.Text.Encoding.ASCII. _
                                    GetString(receivebuffer, 0, receivebuffer.Length)
    receivebuffer.Clear(receivebuffer, 0, 512)
    tb_receivedata.AppendText(receivestring)
    Try
      'Datenempfangs Callback-Prozedur starten
      TCP_client.BeginReceive(receivebuffer, 0, 512, SocketFlags.None, _
                            New AsyncCallback(AddressOf callback_readdata), TCP_client)
    Catch ex As Exception
      closeconnections()
    End Try
    End If
  End If
End Sub
```

Prozedur zum Senden von Daten

```
Private Sub bt_senddata_Click(ByVal sender As System.Object, _
                              ByVal e As System.EventArgs) Handles bt_senddata.Click
  Dim senddata As Byte() = System.Text.Encoding.ASCII.GetBytes(tb_senddata.Text)
  Try
    TCP_client.Send(senddata)
  Catch ex As Exception
    closeconnections()
  End Try
End Sub
```

Prozeduraufruf bei Klick auf Disconnect-Button

```
Private Sub bt_disconnect_Click(ByVal sender As System.Object, _
                            ByVal e As System.EventArgs) Handles bt_disconnect.Click
  closeconnections()
End Sub
```

Prozedur, die alle offenen Socket-Recourcen beendet und die Buttons auf Startzustand setzt

```
Private Sub closeconnections()
  Dim ar As IAsyncResult
  Try
    TCP_client.EndReceive(ar)
  Catch ex As Exception
  End Try
  Try
    TCP_client.Shutdown(SocketShutdown.Both)
  Catch ex As Exception
  End Try
  Try
    TCP_client.Close()
  Catch ex As Exception
  End Try
  bt_connect.Enabled = True
  bt_senddata.Enabled = False
  bt_disconnect.Enabled = False
  StatusBar1.Text = "no connection"
End Sub
```

Die folgende Prozedur sorgt dafür, dass bei Programmende alle Socket-Recourcen beendet werden.

```
Private Sub Form1_Closing(ByVal sender As Object, _
            ByVal e As System.ComponentModel.CancelEventArgs) Handles MyBase.Closing
```

```
    closeconnections()
    Application.DoEvents()    'Warten bis alle laufenden Vorgänge abgeschlossen sind
End Sub

End Class
```

Das Beispiel zeigt, dass auch mit VB.Net mit einer überschaubaren Menge Quelltext ein TCP-Client aufgebaut werden kann.

Als Gegenstück wird natürlich ein Server benötigt, der den Verbindungswunsch des Clients annimmt. Das kann ein vorhandenes Gerät, wie z.B. ein W&T Com-Server oder eine weitere VB.Net-Server-Applikation sein.

Wie ein TCP-Server aufgebaut werden kann, soll im folgenden Beispiel gezeigt werden.

2.4.2 Ein TCP-Server in VB.Net

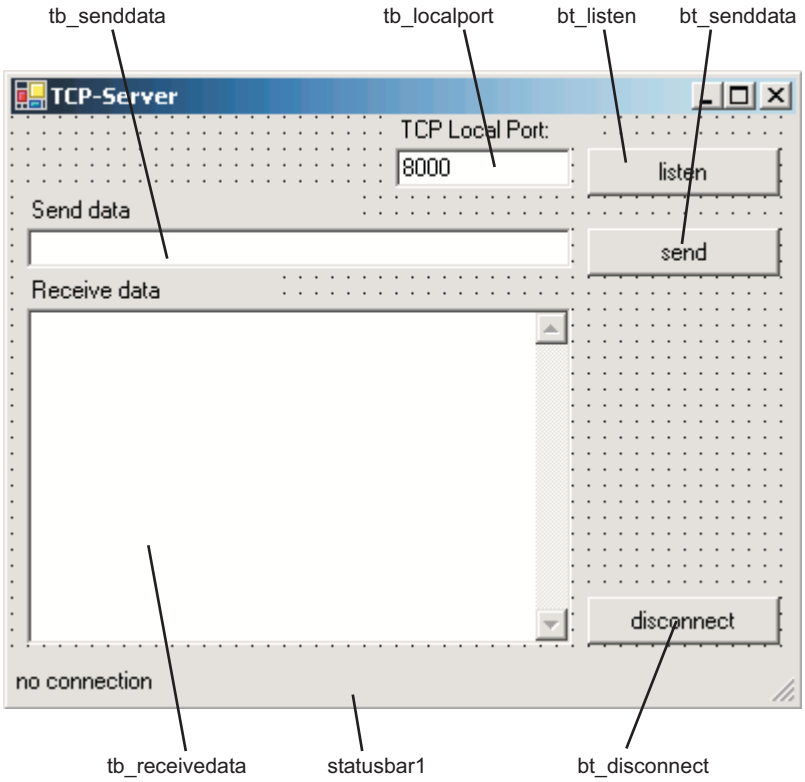
Die Server-Applikation übernimmt folgende Aufgaben:

- Auf dem Netz „Horchen", ob es auf dem unterstützten Port einen Verbindungswunsch gibt
- Gewünschte Verbindung auf einem neuen Socket entgegennehmen
- Senden und Empfangen von Daten
- Anzeigen des Verbindungsstatus
- Anzeigen von Verbindungsfehlern
- Schließen der Verbindung von der Server-Seite

Auch für den TCP-Server wird mit der Socket-Klasse von VB.Net gearbeitet. Hierbei werden zwei Instanzen der Socketklasse benutzt:

Listen_Server	*Socket zum Horchen auf dem Netz*
Connection_Server	*Socket für die Datenverbindung*

Es wird ein Formular mit folgenden Elementen aufgebaut:



Alle Variablen und Elementnamen sollten sich auch in diesem Beispiel durch die gewählte Namensgebung selbst erklären.

Die folgenden VB.Net-Prozeduren werden für den Server benötigt:

Über das Import-Statement werden alle Namespaces der verwendeten Klassen angegeben. Im Quelltext der eigenen Klasse muss dann z.B. wenn eine neue Instanz der Klasse System.Nets.Sockets.Socket benutzt werden soll nur noch Dim TCP_client As Socket eingegeben werden.

```
Imports System.Threading
Imports System.Net
Imports System.Net.Sockets
Imports System.IO
Imports System.Windows.Forms

Public Class Form1
```

```
    Inherits System.Windows.Forms.Form

Alle global benutzten Variablen, Objekte und Klasseninstanzen müssen zunächst dimensioniert werden.
Dim Listen_Server As Socket
Dim Connection_Server As Socket
Dim receivebuffer(511) As Byte

.....
Hier befindet sich im Original der von VB.Net automatisch generierte Quelltext
.....

Prozedur zum Öffnen des Listen-Sockets, welches das Horchen auf dem Netzwerk nach
Verbindungswunsch übernimmt.
Private Sub bt_listen_Click(ByVal sender As System.Object, _
                   ByVal e As System.EventArgs) Handles bt_listen.Click
  If tb_localport.Text <> "" Then
    StatusBar1.Text = "Waiting for connection"
    bt_listen.Enabled = False
    bt_disconnect.Enabled = True
    Listen_Server = New Socket(AddressFamily.InterNetwork, _
                   SocketType.Stream, ProtocolType.Tcp)
Ermitteln der eigenen IP-Adresse und Generieren des lokalen IPEndPoint
    Dim localhostinfo As IPHostEntry = Dns.Resolve(Dns.GetHostName())
    Dim localadress As IPAddress = localhostinfo.AddressList(0)
    Dim localEP As New IPEndPoint(localadress, Val(tb_localport.Text))
    Try
      Listen_Server.Bind(localEP) 'IP-Adresse und Port binden
    Catch ex As Exception
      StatusBar1.Text = "ERROR on binding socket"
    End Try
    Try
      Listen_Server.Listen(1)     'Listen-Server starten
    Catch ex As Exception
      closeconnections()
      StatusBar1.Text = "ERROR on Start Listening"
    End Try
    Zuweisen der Callback-Prozedur für das Accept-Ereignis
    Listen_Server.BeginAccept(AddressOf callback_accept, Listen_Server)
  End If
End Sub
```

Callback-Prozedur, die aufgerufen wird, wenn ein Verbindungswunsch ansteht

```
Public Sub callback_accept(ByVal ar As IAsyncResult)
  StatusBar1.Text = "connected"
  bt_senddata.Enabled = True
  Try
    Connection_Server = Listen_Server.EndAccept(ar)
    Zuweisen der Callback-Prozedur für den Datenempfang
    Connection_Server.BeginReceive(receivebuffer, 0, 512, SocketFlags.None, _
           New AsyncCallback(AddressOf callback_readdata), Connection_Server)
    Listen_Server.Close()          'Listen-Server schließen
  Catch ex As Exception
    StatusBar1.Text = "no connection"
  End Try
End Sub
```

Callback-Prozedur, die aufgerufen wird, wenn Daten vom Netzwerk empfangen werden

```
Private Sub callback_readdata(ByVal ar As IAsyncResult)
  If Connection_Server.Connected Then
    Dim bytesread As Integer
    Try
      bytesread = Connection_Server.EndReceive(ar)
    Catch ex As Exception
    End Try
    If bytesread = 0 Then  'wenn die Callback-Prozedur aufgerufen wird, ohne dass Daten
                           'empfangen wurden, hat die Gegenseite die Verbindung geschlossen
      closeconnections()
    Else
      Dim receivestring As String
      receivestring = System.Text.Encoding.ASCII.GetString(receivebuffer, _
              0, receivebuffer.Length)
      receivebuffer.Clear(receivebuffer, 0, 512)
      tb_receivedata.AppendText(receivestring)
      Try
        Connection_Server.BeginReceive(receivebuffer, 0, 512, SocketFlags.None, _
               New AsyncCallback(AddressOf callback_readdata), Connection_Server)
      Catch ex As Exception
        closeconnections()
      End Try
    End If
  End If
End Sub
```

Prozedur zum Senden von Daten

```
Private Sub bt_senddata_Click(ByVal sender As System.Object, _
            ByVal e As System.EventArgs) Handles bt_senddata.Click
  Dim senddata As Byte() = System.Text.Encoding.ASCII.GetBytes(tb_senddata.Text)
  Try
    Connection_Server.Send(senddata)
  Catch ex As Exception
    closeconnections()
  End Try
End Sub
```

Prozeduraufruf bei Klick auf Disconnect-Button

```
Private Sub bt_disconnect_Click(ByVal sender As System.Object, _
              ByVal e As System.EventArgs) Handles bt_disconnect.Click
  closeconnections()
End Sub
```

Prozedur, die alle offenen Socket-Recourcen beendet und die Buttons auf Startzustand setzt

```
Private Sub closeconnections()
  Dim ar As IAsyncResult
  Try
    Connection_Server.EndReceive(ar)
  Catch ex As Exception
  End Try
  Try
    Listen_Server.Close()
  Catch ex As Exception
  End Try
  Try
    Connection_Server.Shutdown(SocketShutdown.Both)
  Catch ex As Exception
  End Try
  Try
    Connection_Server.Close()
  Catch ex As Exception
  End Try
  bt_listen.Enabled = True
  bt_senddata.Enabled = False
  bt_disconnect.Enabled = False
  StatusBar1.Text = "no connection"
End Sub
```

Die folgende Prozedur sorgt dafür, dass bei Programmende alle Socket-Recourcen beendet werden.

```
Private Sub Form1_Closing(ByVal sender As Object, _
                ByVal e As System.ComponentModel.CancelEventArgs) Handles MyBase.Closing
  closeconnections()
  Application.DoEvents()
End Sub

End Class
```

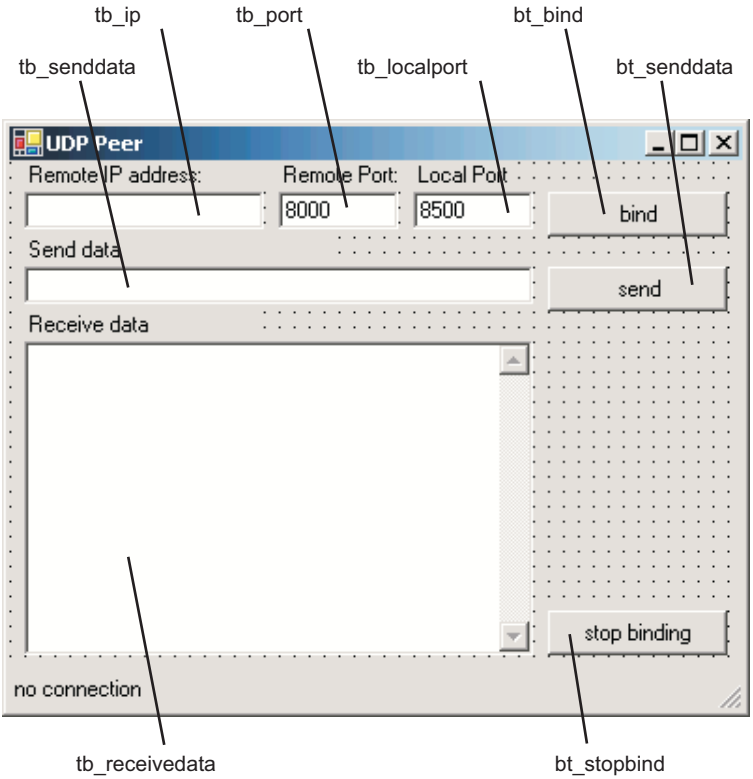
Auf den so programmierten Server kann man mit dem vorgestellten Client-Programm zugreifen. Aber auch beliebige andere Client-Anwendungen, z.B. der W&T Com-Server im Client-Mode, können bei Wahl des entsprechenden Ports Verbindung mit dem Server aufnehmen.

2.4.3 Ein einfacher UDP-Peer in VB.Net

Die UDP-Applikation übernimmt folgende Aufgaben:

- IP-Adresse und Ports zu einem Socket binden
- Senden und Empfangen von Textdaten

Hierzu wird ein Formular mit folgenden Elementen aufgebaut:



Alle Variablen und Elementnamen sollten sich auch hier durch die gewählte Namensgebung selbst erklären.

Der folgende Quelltext, in dem der von VB.Net automatisch generierte Quellcode nicht abgedruckt ist, kommt deshalb auch weitestgehend ohne Kommentare aus:

Über das Import-Statement werden alle Namespaces der verwendeten Klassen angegeben. Im Quelltext der

eigenen Klasse muss dann z.B. wenn eine neue Instanz der Klasse System.Nets.Sockets.Socket benutzt werden soll nur noch Dim TCP_client As Socket eingegeben werden.

```
Imports System.Threading
Imports System.Net
Imports System.Net.Sockets
Imports System.IO
Imports System.Windows.Forms

Public Class Form1
    Inherits System.Windows.Forms.Form
```

Alle global benutzten Variablen, Objekte und Klasseninstanzen müssen zunächst dimensioniert weden.

```
Dim UDP_peer As Socket
Dim receivebuffer(511) As Byte
```

.....

Hier befindet sich im Original der von VB.Net automatisch generierte Quelltext

.....

Prozedur zum Binden der UDP Ports

```
Private Sub bt_bind_Click(ByVal sender As System.Object, _
                          ByVal e As System.EventArgs) Handles bt_bind.Click
Dim socketerror As Boolean
  If tb_ip.Text <> „“ And tb_port.Text <> „“ And tb_localport.Text <> „“ Then
    Dim iphostinfo As IPHostEntry = Dns.Resolve(Dns.GetHostName())
    Dim pc_ip As IPAddress = iphostinfo.AddressList(0)
    Dim listenEP As New IPEndPoint(pc_ip, Val(tb_localport.Text))
    UDP_peer = New Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp)
    Try
      socketerror = False
      UDP_peer.Bind(listenEP)
    Catch ex As Exception
      socketerror = True
      MessageBox.Show(ex.Message, „Socket Error“, MessageBoxButtons.OK, _
                                                    MessageBoxIcon.Error)
    End Try
    If socketerror Then
      resetbinding()
    Else
      bt_bind.Enabled = False
      bt_senddata.Enabled = True
```

```
      bt_stopbind.Enabled = True
      UDP_peer.BeginReceive(receivebuffer, 0, 512, SocketFlags.None, _
              New AsyncCallback(AddressOf callback_readdata), UDP_peer)
      StatusBar1.Text = „binded: Local Port = „ + tb_localport.Text + „, _
              Remote Port = „ + tb_port.Text + „, Remote IP = „ + tb_ip.Text
    End If
  End If
End Sub
```

Callback-Prozedur, die aufgerufen wird, wenn Daten vom Netzwerk empfangen werden

```
Private Sub callback_readdata(ByVal ar As IAsyncResult)
  Try
    UDP_peer.EndReceive(ar)
  Catch ex As Exception
  End Try
  Dim receivestring As String
  receivestring = System.Text.Encoding.ASCII.GetString(receivebuffer, 0, _
                                                       receivebuffer.Length)
  receivebuffer.Clear(receivebuffer, 0, 512)
  tb_receivedata.AppendText(receivestring)
  Try
    UDP_peer.BeginReceive(receivebuffer, 0, 512, SocketFlags.None, _
                          New AsyncCallback(AddressOf callback_readdata), UDP_peer)
  Catch ex As Exception
    resetbinding()
  End Try
End Sub

'# sub for sending data
Private Sub bt_senddata_Click(ByVal sender As System.Object, _
                              ByVal e As System.EventArgs) Handles bt_senddata.Click
  Dim senddata As Byte() = System.Text.Encoding.ASCII.GetBytes(tb_senddata.Text)
  Dim sendtoEP = New IPEndPoint(IPAddress.Parse(tb_ip.Text), Val(tb_port.Text))
  Try
    UDP_peer.SendTo(senddata, 0, senddata.Length, SocketFlags.None, sendtoEP)
  Catch ex As Exception
    resetbinding()
  End Try
End Sub
```

Prozedur die aufgerufen wird, wenn der stop binding *Button angeklickt wird*

```
Private Sub bt_stopbind_Click(ByVal sender As System.Object, _
                              ByVal e As System.EventArgs) Handles bt_stopbind.Click
  resetbinding()
End Sub

Prozedur um die Bindung der UDP Ports zu lösen und die Recourcen wieder frei zu geben
Private Sub resetbinding()
  Dim ar As IAsyncResult
  Try
    UDP_peer.EndReceive(ar)
  Catch ex As Exception
  End Try
  Try
    UDP_peer.Shutdown(SocketShutdown.Both)
  Catch ex As Exception
  End Try
  Try
    UDP_peer.Close()
  Catch ex As Exception
  End Try
  bt_bind.Enabled = True
  bt_senddata.Enabled = False
  bt_stopbind.Enabled = False
  StatusBar1.Text = „no binding“
End Sub

Die folgende Prozedur sorgt dafür, dass bei Programmende alle Socket-Recourcen beendet werden.
Private Sub Form1_Closing(ByVal sender As Object,_
                          ByVal e As System.ComponentModel.CancelEventArgs) _
                                                  Handles MyBase.Closing
  resetbinding()
  Application.DoEvents()    'Warten bis alle laufenden Vorgänge abgeschlossen sind
End Sub

End Class
```

Um mit dem UDP-Peer Datenkommunikation zu betreiben, kann der gleiche Peer auf einem zweiten PC gestartet werden. Ebenso ist es aber auch möglich, über den UDP-Peer mit einem W&T Com-Server zu kommunizieren, der als UDP-Client konfiguriert ist.

Der hier gezeigte UDP-Peer verzichtet auf jede Form von Datensicherheit. Das bedeutet, werden Daten an eine nicht existente IP-Adresse geschickt oder das adressierte Endgerät ist nicht betriebsbereit, laufen die Daten einfach ins Leere, ohne dass der Anwender etwas merkt.

3. OPC – Der Prozessdaten Dolmetscher

In der Automatisierungstechnik werden meist Hardware-Komponenten verschiedenster Hersteller zu einer Anlage zusammen gesetzt. Hierbei verfolgt jeder Hersteller einen eigenen Weg, Prozessdaten an die Softwareebene weiterzugeben. Das betrifft sowohl den physikalischen Kommunikationsweg als auch das Datenformat.

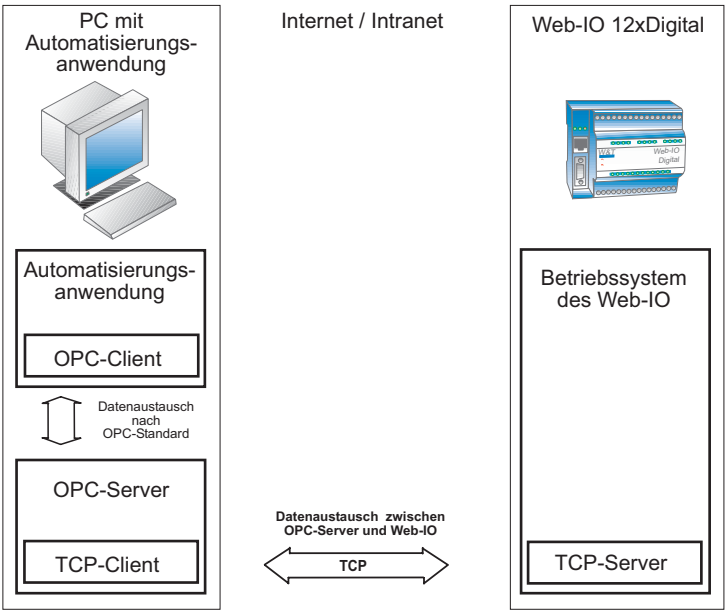
Um diesem Prozessdaten-Babylon zu entgehen, wurde der OPC-Standard eingeführt .

OPC steht für OLE for Process Control, wobei OLE die Abkürzung für Object Linking and Embedding ist. Die Grundidee von OLE ist die geregelte Einbettung von Dokumenten anderer Anwendungen in die eigene Anwendung. Zum Beispiel das Einfügen eines EXCEL Dokuments in eine Word Datei.

OPC gestützte Anwendungen kommunizieren nicht auf direktem Weg mit den angesprochenen Endgeräten. Statt dessen wird für das entsprechende Endgerät ein OPC-Server installiert. Der OPC-Server ist ein Softwareprozess, der im Hintergrund die herstellerspezifische Kommunikation mit dem Endgerät abwickelt. Die so gewonnenen Prozessdaten werden nach dem OPC-Standard aufgearbeitet und der Anwendung in einer standardisierten Form übergeben.

Der Teil der Anwendung, der mit dem OPC-Server kommuniziert, wird als OPC-Client bezeichnet.

Das folgende Beispiel zeigt den Zugriff auf ein Wiesemann & Theis Web-IO 12xDigital mittels OPC-Server:



3.1 Der Zugriff über OPC

Bei der OPC-Schnittstelle unterscheidet man zwischen zwei Hauptaufgaben:

Data Access: kurz DA beschreibt den Austausch von Daten über OPC.

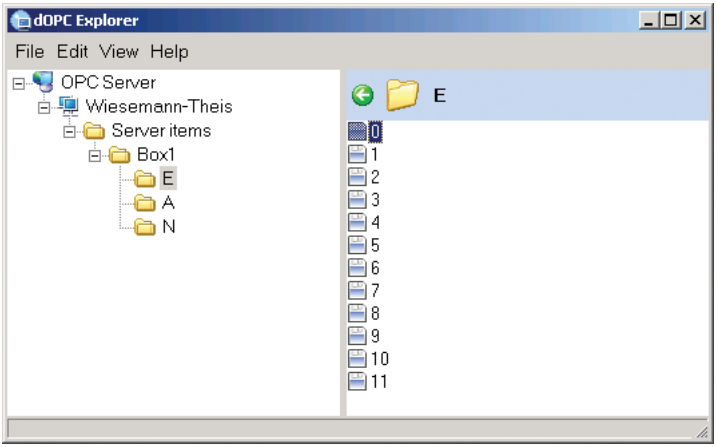
Alarm & Events: kurz AE dient zur Alarm- und Ereignisbehandlung.

Es gibt noch weitere Aufgaben, die in der Praxis aber eine untergeordnete Rolle spielen und auf die hier deswegen nicht weiter eingegangen wird.

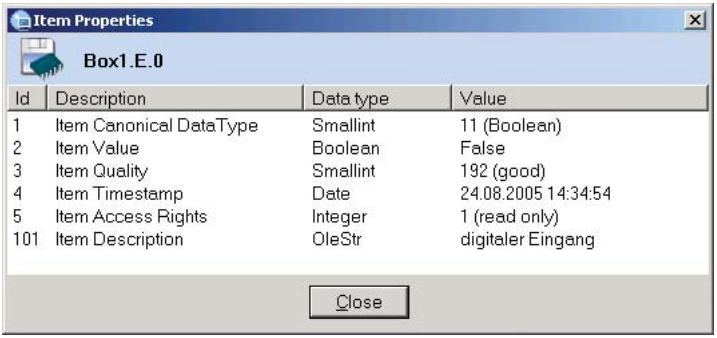
Prozessdaten werden vom OPC-Server in Form von Items bereitgestellt. Alle Items haben eine Item-ID, eine innerhalb des OPC-Servers einmalige also eindeutige Adresse. Jedes Item hat eine unbestimmte Anzahl von Properties bzw. Item-Eigenschaften, wie z.B. Wert, Qualität, Zeitstempel .....

Die Items werden vom OPC-Server meist in Gruppen zusammengefasst. Daraus ergibt sich dann eine Art Hierarchie (OPC-Server > OPC-Group > OPC Item).

Um dem OPC-Client einen einfachen Zugang zu allen verfügbaren Items zu ermöglichen, erlauben viele OPC-Server dem OPC-Client das OPC-Browsing.



Der OPC-Client kann darüber alle Items in einer Art Verzeichnisbaumstruktur abfragen. Hier als Beispiel die Struktur der Items eines W&T Web-IO 12xDigital.



3.2 Kommunikation zwischen OPC-Client und OPC-Server

Der OPC-Client kann aus allen vom OPC-Server angebotenen Items eine Teilmenge (oder auch alle) auswählen und seinerseits zu einer oder mehreren Gruppen zusammenfassen. Diese Gruppen müssen nicht identisch mit den vom OPC-Server gebildeten Gruppen sein. Die ausgewählten Items werden dann gruppenweise vom OPC-Client abonniert. Das bedeutet, der OPC-Client muss nicht ständig den Zustand der Items abfragen, sondern wird vom OPC-Server automatisch informiert, wenn sich eine der Eigenschaften eines Items ändert. Auf diese Weise entlastet der OPC-Server den OPC-Client, und somit die Anwendung.

3.3 Wann macht es Sinn mit OPC zu arbeiten?

Immer dann, wenn eine flexible Anwendung entstehen soll, die ohne großen Aufwand mit der Hardware verschiedenster Hersteller Daten austauschen muss, ist OPC die ideale Lösung.

In Applikationen der Prozessleittechnik und der Prozess- und Messdatenvisualisierung ist OPC vor allem für den Anwender eine feine Sache.

Bei allen Vorteilen der OPC-Technik soll hier aber nicht verschwiegen werden, dass die Programmierung einer universellen OPC-Client Anwendung eine komplexe Aufgabe ist, die ein hohes Maß an Programmierkompetenz voraussetzt.

Wenn es also darum geht, eine spezielle Anwendung für ein spezielles Endgerät eines Herstellers zu erstellen, sollte man abwägen, ob es nicht einfacher ist den direkten, vom Hersteller vorgesehenen Kommunikationsweg zu gehen.

TCP/IP -Ethernet einrichten

Alle aktuellen Betriebssysteme bieten heute die Möglichkeit, TCP/IP als lokales Netzwerkprotokoll zu nutzen.

Die Bedingung hierfür ist, dass der PC über eine Ethernet-Netzwerkkarte verfügt.

Wie das TCP/IP Protokoll auf den gängigen Microsoft Windows Systemen eingerichtet und konfiguriert wird, soll auf den folgenden Seiten Schritt für Schritt beschrieben werden.

Wenn Ihr PC bereits in ein Ethernet-Netzwerk eingebunden ist, sollten Sie zunächst in Erfahrung bringen, ob in diesem Netzwerk bereits Anwendungen unter TCP/IP betrieben werden. Fragen Sie in diesem Fall Ihren Netzwerkadministrator, ob für Ihren PC schon eine IP-Adresse vorgesehen ist oder welche IP-Adressen Sie für Ihren PC verwenden können. Ferner müssen Sie wissen, welche Subnet-Mask, welches Gateway und welcher DNS-Server ggf. für das Netz gültig sind.

Bitte notieren Sie sich die verwendeten Werte:

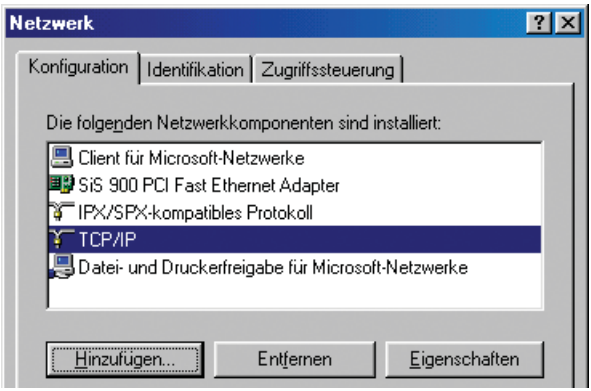
IP-Adresse	_____.____.____.____
Subnet-Mask	_____.____.____.____
Gateway	_____.____.____.____
DNS-Server	_____.____.____.____

TCP/IP unter Windows 9x installieren und konfigurieren

1. Klicken Sie auf *Start* und öffnen unter *Einstellungen* die *Systemsteuerung*.

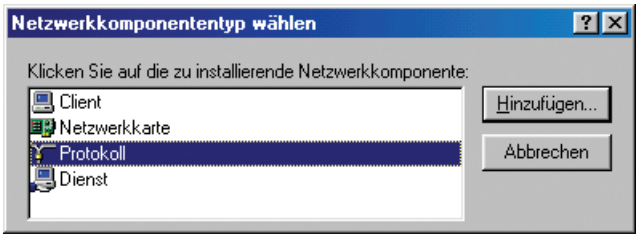
2. Doppelklicken Sie auf das Netzwerk-Symbol .

3. Kontrollieren Sie, ob im Konfigurationsfenster *TCP/IP* aufgelistet ist.

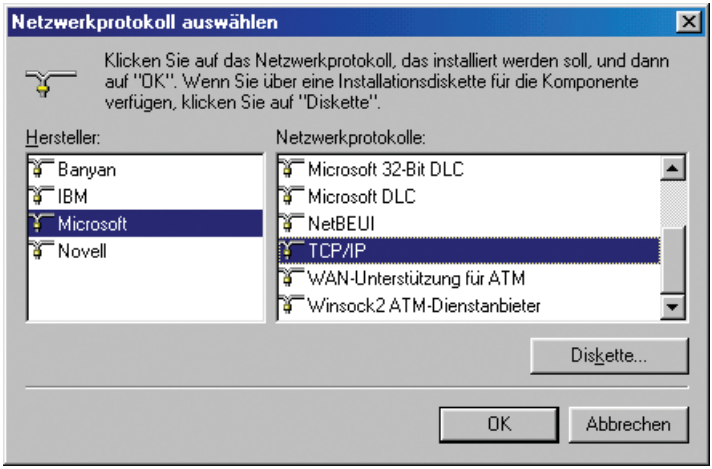


Wenn der Eintrag *TCP/IP* vorhanden ist, fahren Sie mit Punkt 5 fort.

4. Bei fehlendem Eintrag *TCP/IP* klicken Sie auf *Hinzufügen* und wählen im folgenden Fenster *Protokoll*.

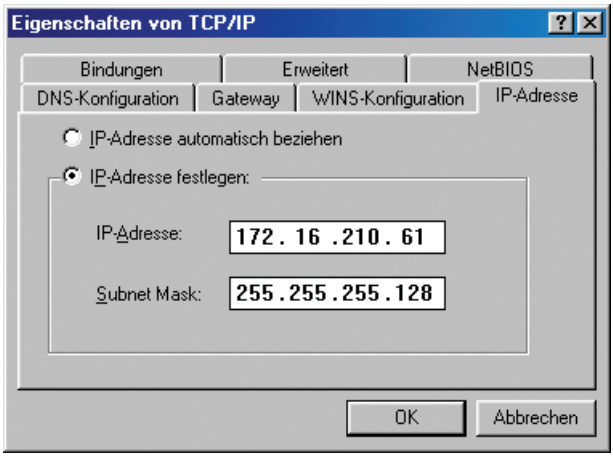


Klicken sie auf *Hinzufügen* und wählen Sie im folgenden Fenster als Hersteller *Microsoft* und als Netzwerkprotokoll *TCP/IP*.



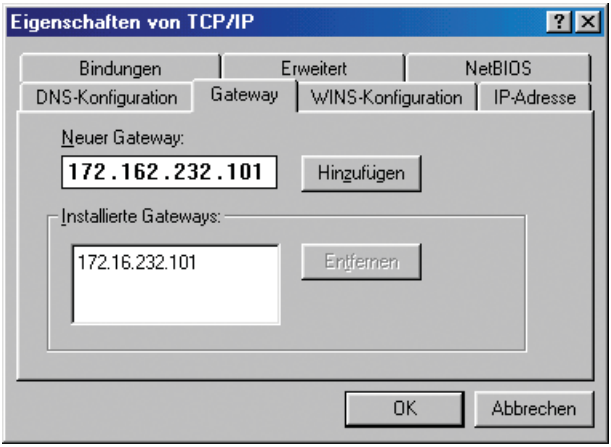
Bestätigen Sie mit *OK*.
Zur Installation des Protokolls benötigen Sie die Installations-CD Ihrer Windows-Version.

5. Markieren Sie *TCP/IP* und wählen Sie *Eigenschaften*. Fragen Sie Ihren Netzwerkadministrator, ob die IP-Adresse über DHCP automatisch bezogen wird.



Wenn nicht, tragen Sie IP-Adresse und Subnet-Mask ein.

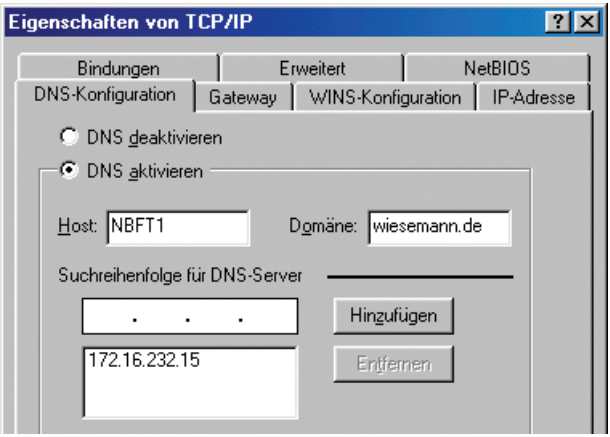
Wechseln Sie nun zum Register Gateway



Tragen Sie die IP-Adresse des Gateways im Feld *neuer Gateway* ein und klicken auf *Hinzufügen*. Nur wenn die eingetragene Gateway-Adresse im unteren Fenster erscheint, bleibt sie nach Bestätigung mit *OK* erhalten.

Arbeitet Ihr Netzwerk mit DNS-Unterstützung, sollte im Register DNS-Konfiguration auch die IP-Adresse des DNS-Servers eingetragen werden. Nur wenn die eingetragene DNS-Adresse im unteren Fenster erscheint, bleibt sie nach Bestätigung mit *OK* erhalten.

Ferner sollten Sie dort den Hostnamen des PCs und die Domain, in der er verwaltet wird, eintragen.



Bestätigen Sie mit *OK*.

Damit ist die Installation von TCP/IP abgeschlossen, und Sie werden jetzt aufgefordert, Ihren PC neu zu starten.

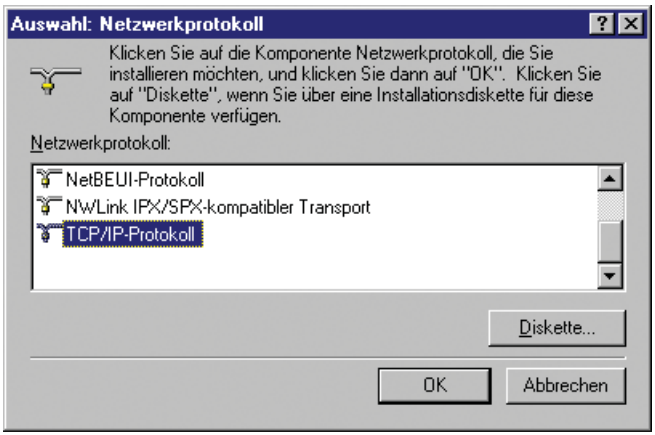
TCP/IP unter Windows NT installieren u. konfigurieren

1. Klicken Sie auf *Start* und öffnen unter *Einstellungen* die *Systemsteuerung*.

2. Doppelklicken Sie auf das Icon .

3. Wenn im Register *Protokolle TCP/IP-Protokoll* aufgelistet ist, können Sie an Punkt 5 fortfahren.

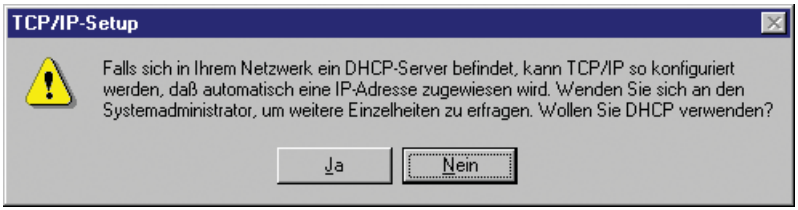
4. Bei fehlendem Eintrag *TCP/IP-Protokoll* klicken Sie auf *Hinzufügen* und wählen im folgenden Fenster *TCP/IP*.



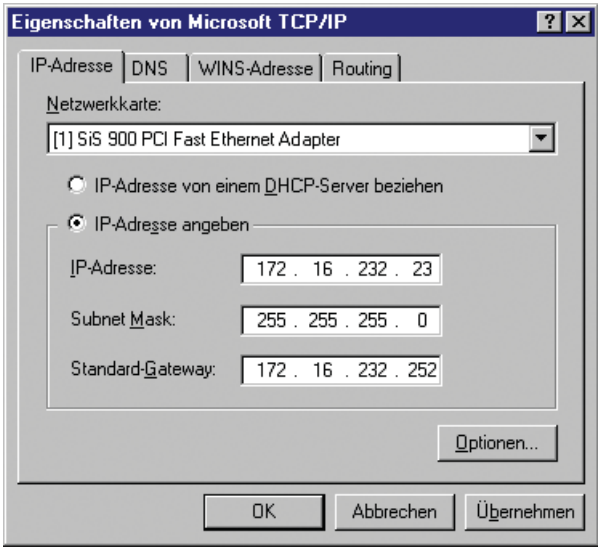
Sie benötigen nun die Windows-NT-Installations-CD. Bestätigen Sie mit *OK*.

5. Bei neu hinzugefügter TCP/IP-Unterstützung klicken Sie *OK*, um die Eigenschaften zu konfigurieren. War TCP/IP bereits auf Ihrem PC installiert, markieren Sie den Eintrag *TCP/IP-Protokoll* und klicken Sie anschließend auf *Eigenschaften.*

   Haben Sie die TCP/IP-Unterstützung neu installiert, erscheint nun folgende Meldung:

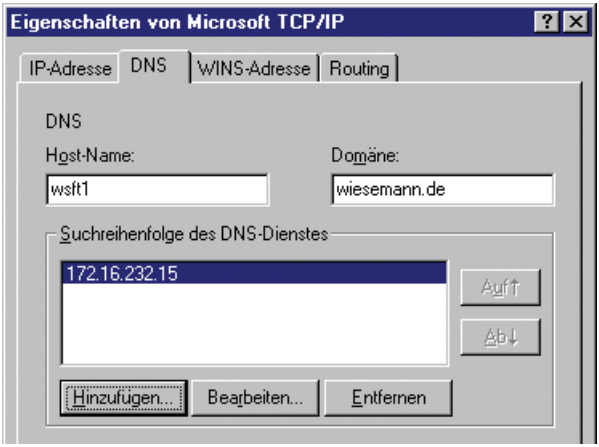


   Fragen Sie Ihren Netzwerkadministrator, ob der DHCP-Dienst unterstützt wird. Ist das nicht der Fall, klicken Sie hier auf *Nein.*



Tragen Sie im folgenden Fenster IP-Adresse, Subnet-Mask und Gateway ein.

Arbeitet Ihr Netzwerk mit DNS-Unterstützung, sollte im Register *DNS-Konfiguration* auch die IP-Adresse des DNS-Servers eingetragen werden.



Ferner sollten Sie dort den Hostnamen des PCc und die Domain, in der er verwaltet wird, eintragen.

Bestätigen Sie mit *OK.*

Damit ist die Installation von TCP/IP abgeschlossen, und Sie werden jetzt aufgefordert, Ihren PC neu zu starten.

TCP/IP unter Win 2000 installieren und konfigurieren

1. Klicken Sie auf *Start* und öffnen unter *Einstellungen* die *Systemsteuerung.*

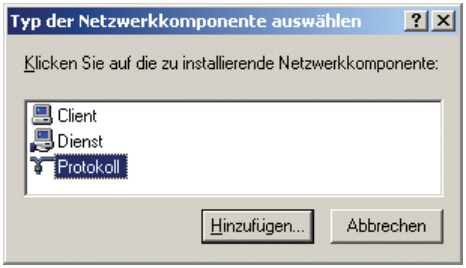
2. Doppelklicken Sie auf das Icon:

   und im nächsten Fenster auf:

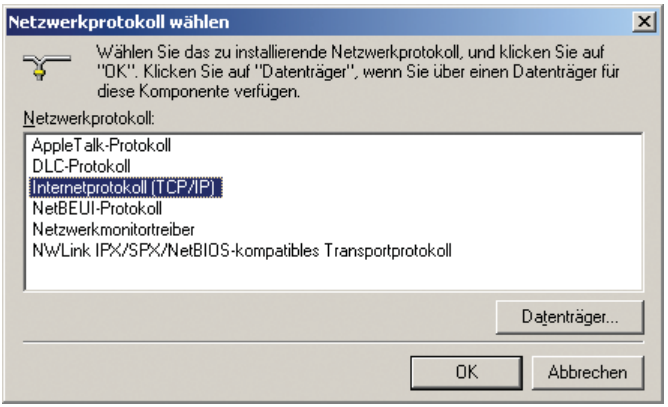
3. Kontrollieren Sie, ob *Internetprotokoll (TCP/IP)* aufgelistet ist.

   Wenn der Eintrag *Internetprotokoll (TCP/IP)* vorhanden ist, fahren Sie mit Punkt 5 fort.

4. Bei fehlendem Eintrag *Internetprotokoll (TCP/IP)* klicken Sie auf *Installieren* und wählen im folgenden Fenster *Protokoll* und *Hinzufügen*.



Wählen Sie *Internetprotokoll (TCP/IP).*



Sie benötigen nun die Windows 2000 Installations-CD. Nach Bestätigung mit *OK* ist die Liste der Netzwerkprotokolle um den Eintrag *TCP/IP-Protokoll* erweitert.

5. Es erscheint nun wieder das Fenster *Eigenschaften von LAN-Verbindung*. Markieren Sie den Eintrag *Internetprotokoll (TCP/IP)* und klicken Sie anschließend auf *Eigenschaften.*

   Wenn Ihr PC bereits in ein Netzwerk eingebunden ist, sollten Sie sich bei Ihrem Netzwerkadministrator erkundigen, ob der DHCP-Dienst unterstützt wird.

   Ist das der Fall wählen Sie *IP-Adresse automatisch beziehen.*

Sonst tragen Sie im folgenden Fenster IP-Adresse, Subnet-Mask und Gateway ein. Arbeitet Ihr Netzwerk mit DNS-Unterstützung, sollte auch die IP-Adresse des DNS-Servers eingetragen werden. Bestätigen Sie mit *OK*.

Damit ist die Installation von TCP/IP abgeschlossen, und Sie werden jetzt aufgefordert, Ihren PC neu zu starten.

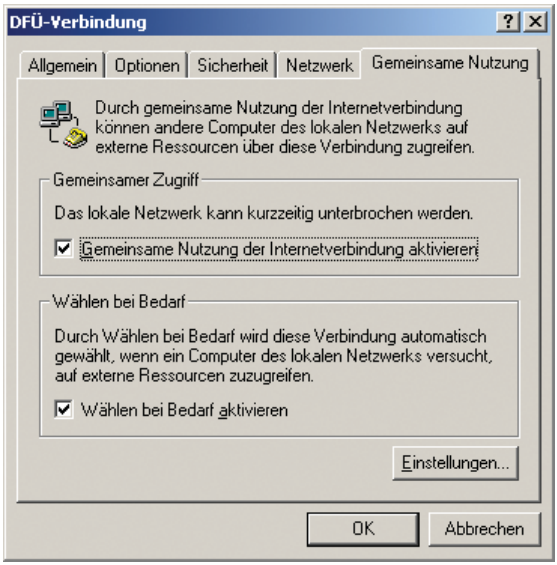
TCP/IP-Ethernet bei gleichzeitigem DFÜ-Internetzugang

Hat der PC neben dem Zugang ins lokale Netz einen Internet-Zugang über DFÜ, bietet Windows 2000 eine Besonderheit:

Der Internetzugang dieses PCs kann von anderen am Netz angeschlossenen Geräten mitgenutzt werden.

Um diesen Dienst freizuschalten, doppelklicken Sie in der Systemsteuerung das Icon für die angelegte DFÜ-Verbindung.

Klicken Sie auf *Eigenschaften* und wählen Sie im folgenden Fenster das Register *Gemeinsame Nutzung*



Aktivieren Sie die *Gemeinsame Nutzung der Internetverbindung*.

Der PC arbeitet nun quasi als Router ins Internet.

Durch Aktivierung der gemeinsamen Nutzung wird die IP-Adresse des PCs vom System fest auf 192.168.0.1 geändert.

Darüber hinaus tritt dieser PC plötzlich als DHCP- bzw. BootP-Server auf.

Zur Erinnerung: Ein DHCP Server teilt Netzwerkteilnehmern auf Anforderung automatisch eine IP-Adresse zu. Nähere Informationen finden Sie im Kapitel DHCP.

PCs, bei denen die *Gemeinsame Nutzung* aktiviert ist, vergeben also willkürlich IP-Adressen aus dem Adressraum 192.168.0 und das sogar auf BootP Anfragen hin.

Bei BootP ist im Normalfall nur die Vergabe reservierter Adressen vorgesehen!

Hier ist also Vorsicht geboten! Durch dieses Verhalten ist es möglich, dass andere Netzwerkteilnehmer in Folge der Adressänderung nicht mehr erreichbar sind.

Abhilfe

1. Bei kleineren Netzwerken, in denen auf einem Windows 2000 PC die *Gemeinsame Nutzung* freigeschaltet ist:

   - Deaktivieren Sie bei allen Netzteilnehmern das automatische Beziehen einer IP-Adresse via DHCP, bzw. BootP.
   - Vergeben Sie den Netzteilnehmern festeingestellte IP-Adressen im Adressraum 192.168.0.

   Durch diese Maßnahmen bleiben die Netzteilnehmer auch über ihre IP-Adresse erreichbar. Das ist vor allem bei Embedded-Systemen wie z.B. dem Com-Server wichtig.
   Trotzdem lassen sich so die Vorteile der gemeinsamen Nutzung in Anspruch nehmen.

2. Bei größeren Netzen sollte auf die gemeinsame Nutzung verzichtet werden. Statt dessen empfehlen wir den Einsatz eines Routers.

TCP/IP unter Win XP installieren und konfigurieren

1. Klicken Sie auf *Start* und öffnen unter *Einstellungen* die *Systemsteuerung.*

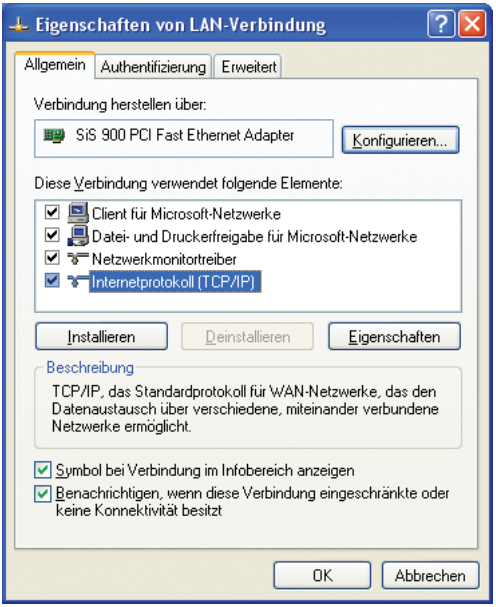
2. Doppelklicken Sie auf das Icon:

und im nächsten Fenster auf:

Anschließend klicken Sie mit

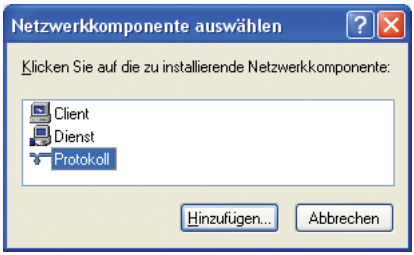
der rechten Maustaste auf:

3. Kontrollieren Sie, ob *Internetprotokoll (TCP/IP)* aufgelistet ist.



Wenn der Eintrag *Internetprotokoll (TCP/IP)* vorhanden ist, fahren Sie mit Punkt 5 fort.

4. Bei fehlendem Eintrag *Internetprotokoll (TCP/IP)* klicken Sie auf *Installieren* und wählen im folgenden Fenster *Protokoll* und *Hinzufügen*.



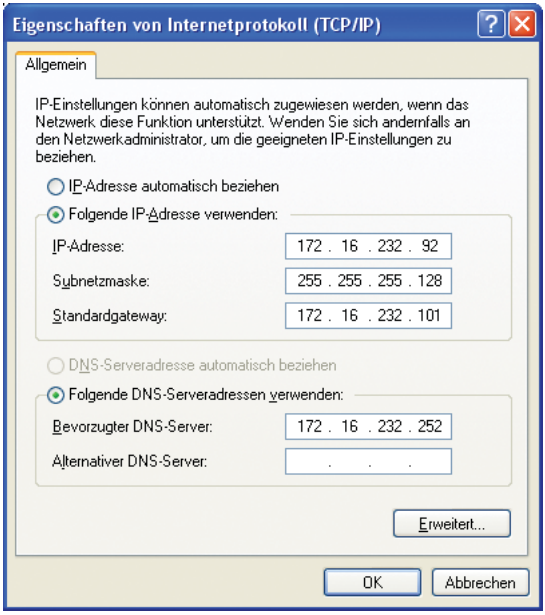
Wählen Sie *Microsoft TCP/IP.*

Sie benötigen nun die Windows XP Installations-CD. Nach Bestätigung mit *OK* ist die Liste der Netzwerkprotokolle um den Eintrag *Microsoft TCP/IP* erweitert.

5. Es erscheint nun wieder das Fenster *Eigenschaften von LAN-Verbindung*. Markieren Sie den Eintrag *Internetprotokoll (TCP/IP)* und klicken Sie anschließend auf *Eigenschaften.*

Wenn Ihr PC bereits in ein Netzwerk eingebunden ist, sollten Sie sich bei Ihrem Netzwerkadministrator erkundigen, ob der DHCP-Dienst unterstützt wird.

Ist das der Fall wählen Sie *IP-Adresse automatisch beziehen*.



Sonst tragen Sie im folgenden Fenster IP-Adresse, Subnet-Mask und Gateway ein. Arbeitet Ihr Netzwerk mit DNS-Unterstützung, sollte auch die IP-Adresse des DNS-Servers eingetragen werden. Bestätigen Sie mit *OK.*

Damit ist die Installation von TCP/IP abgeschlossen, und Sie werden jetzt aufgefordert, Ihren PC neu zu starten.

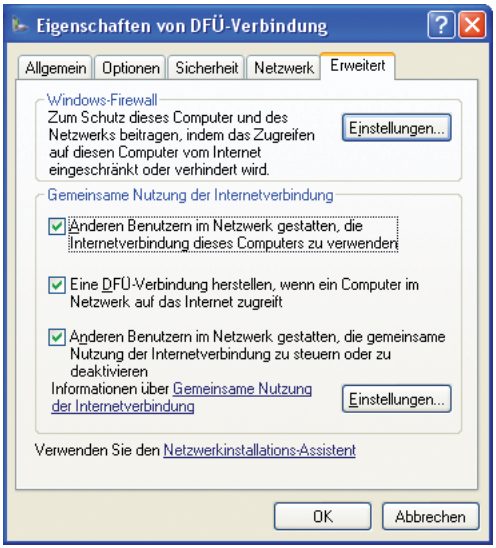
TCP/IP-Ethernet bei gleichzeitigem DFÜ-Internetzugang

Hat der PC neben dem Zugang ins lokale Netz einen Internet-Zugang über DFÜ, bietet Windows XP eine Besonderheit:

Der Internetzugang dieses PCs kann von anderen am Netzangeschlossenen Geräten mitgenutzt werden.

Um diesen Dienst freizuschalten, klicken Sie in der Systemsteuerung im Bereich Netzwerkverbindungen mit der rechten Maustaste auf das Icon für die angelegte DFÜ-Verbindung.

Klicken Sie auf *Eigenschaften* und wählen Sie im folgenden Fenster das Register *Erweitert.*



Aktivieren Sie die *Gemeinsame Nutzung der Internetverbindung*.

Der PC arbeitet nun quasi als Router ins Internet.

Durch Aktivierung der gemeinsamen Nutzung wird die IP-Adresse des PCs vom System fest auf 192.168.0.1 geändert.

Darüber hinaus tritt dieser PC plötzlich als DHCP- bzw. BootP-Server auf.

Zur Erinnerung: Ein DHCP Server teilt Netzwerkteilnehmern auf Anforderung automatisch eine IP-Adresse zu. Nähere Informationen finden Sie im Kapitel DHCP.

PCs, bei denen die *Gemeinsame Nutzung* aktiviert ist, vergeben also willkürlich IP-Adressen aus dem Adressraum 192.168.0 und das sogar auf BootP Anfragen hin.

Bei BootP ist im Normalfall nur die Vergabe reservierter Adressen vorgesehen!

Hier ist also Vorsicht geboten! Durch dieses Verhalten ist es möglich, dass andere Netzwerkteilnehmer in Folge der Adressänderung nicht mehr erreichbar sind.

Abhilfe

1. Bei kleineren Netzwerken, in denen auf einem Windows 2000 PC die *Gemeinsame Nutzung* freigeschaltet ist:

   - Deaktivieren Sie bei allen Netzteilnehmern das automatische Beziehen einer IP-Adresse via DHCP, bzw. BootP.
   - Vergeben Sie den Netzteilnehmern festeingestellte IP-Adressen im Adressraum 192.168.0.

   Durch diese Maßnahmen bleiben die Netzteilnehmer auch über ihre IP-Adresse erreichbar. Das ist vor allem bei Embedded-Systemen wie z.B. dem Com-Server wichtig.
   Trotzdem lassen sich so die Vorteile der gemeinsamen Nutzung in Anspruch nehmen.

2. Bei größeren Netzen sollte auf die gemeinsame Nutzung verzichtet werden. Statt dessen empfehlen wir den Einsatz eines Routers.

Kleines Netzwerk-ABC

10Base2 – *10Mbit/s BASEband 200 (185)m/Segment*
Ethernet-Topologie auf koaxialer Basis mit einer Übertragungsrate von 10Mbit/s.
Weitere geläufige Bezeichnungen für 10Base2 sind auch Cheapernet oder Thin-Ethernet. Es wird Koax-Kabel mit 50 Ohm Impedanz in einer dünnen und flexiblen Ausführung verwendet, um die einzelnen Stationen busförmig miteinander zu verbinden. Anfang und Ende eines Segments müssen mit Abschlusswiderständen von 50 Ohm abgeschlossen werden.
Die Transceiver sind auf den Netzwerkkarten integriert, so dass der Bus direkt bis an jeden Arbeitsplatz geführt werden muss, wo er über BNC-T-Stücke an den Rechner angeschlossen wird. Die Dämpfung des Kabels, sowie die teilweise hohe Anzahl von Steckverbindern beschränken ein 10Base2 Segment auf max. 185m mit max. 30 Anbindungen. Zwischen zwei Stationen dürfen nicht mehr als vier Repeater liegen.
Die Schwachstelle der physikalischen Bus-Topologien von Ethernet liegt in der Tatsache, dass eine Unterbrechung des Kabels – z.B. durch Abziehen eines Steckverbinders – den Stillstand des gesamten Netzsegmentes zur Folge hat.

10Base5 – *10Mbit/s BASEband 500m/Segment*
10Base5 ist die ursprüngliche Ethernet-Spezifikation. Die Verkabelung beruht hier auf einem koaxialen Buskabel mit 50 Ohm Impedanz und einer max. zulässigen Länge von 500m (Yellow Cable). Bedingt durch die koaxiale Zwei-Leiter-Technik (Seele und Schirm) lassen sowohl 10Base5 als auch 10Base2 lediglich einen Halbduplex-Betrieb zu. Die Netzwerkteilnehmer werden über externe Transceiver angeschlossen, die über Vampir-Krallen die Signale direkt vom Buskabel abgreifen, ohne dieses durch Steckverbinder o. ä. zu unterbrechen. Getrennt nach Sende-, Empfangs- und Kollisions-Information werden die Daten vom Transceiver auf einem 15-poligen D-SUB-Steckverbinder zur Verfügung gestellt. Der Anschluss des Endgerätes erfolgt über ein 8adriges TP-Kabel von max. 50m Länge. Zwischen zwei beliebigen Stationen dürfen nicht mehr als vier Repeater liegen.

Diese Regel betrifft allerdings nur „hintereinander“ liegende Repeater – bei der Realisierung baumartiger Netzwerkstrukturen kann also durchaus eine Vielzahl von Repeatern eingesetzt werden.
Durch die Verwendung von relativ hochwertigem Kabel ohne jegliche Unterbrechungen durch Steckverbinder ergeben sich die Vorteile der großen Segmentlänge und der hohen Anzahl möglicher Anbindungen pro Segment (max. 100).
Die Dicke und Unflexibilität des Yellow Cable sowie die durch externe Transceiver zusätzlich entstehenden Kosten sind die Hauptnachteile von 10Base5 und haben wohl entscheidend zur Einführung von 10Base2 beigetragen.

10BaseT – *10Mbit/s BASEband Twisted Pair*
Mit der Definition von 10BaseT wird die physikalische Topologie von der logischen getrennt. Die Verkabelung ist, ausgehend von einem Hub als zentraler aktiver Komponente, sternförmig ausgeführt. Es wird ein mindestens zweipaariges Kabel der Kategorie 3 mit 100 Ohm Impedanz verwendet, in dem die Daten getrennt nach Sende- und Empfangsrichtung übertragen werden. Als Steckverbinder werden 8-polige RJ45-Typen eingesetzt, in denen die Paare auf den Pins 1/2 und 3/6 aufgelegt sind. Die max. Länge eines Segments (= Verbindung vom Hub zum Endgerät) ist auf 100m begrenzt. Ihren Ursprung hat die 10BaseT-Topologie in den USA, weil sie ermöglichte, die dort üblichen Telefonverdrahtungen auch für den Netzwerkbetrieb zu nutzen. Für Deutschland entfiel dieser Vorteil, da hier für die Telefonie Stern-4er-Kabel verlegt wurden, die den Anforderungen der Kategorie 3 nicht entsprachen.
Kabelunterbrechungen oder abgezogene Stecker, die bei allen physikalischen Busstrukturen einen Stillstand des gesamten Segmentes bedeuten, beschränken sich bei 10BaseT lediglich auf einen Arbeitsplatz.

100BaseT4 – *100Mbit/s BASEband Twisted 4 Pairs*
100BaseT4 spezifiziert eine Ethernet-Übertragung mit 100Mbit/s. Wie bei 10BaseT handelt es sich um eine physikalische Sternstruktur mit einem Hub als Zentrum. Es wird ebenfalls ein Kabel der Kategorie 3 mit 100 Ohm Impedanz, RJ45 Steckverbindern und einer max. Länge von 100m eingesetzt. Die zehnfache Übertragungsgeschwindigkeit von 100Mbit/s bei gleichzeitiger

Einhaltung der Kategorie-3-Bandbreite von 25MHz wird u.a. auch durch die Verwendung aller vier Aderpaare erzielt. Für jede Datenrichtung werden bei 100BaseT4 immer 3 Paare gleichzeitig verwendet.

100BaseTX – *100Mbit/s BASEband Twisted 2 Pairs*
100BaseTX spezifiziert die 100Mbit/s-Übertragung auf 2 Aderpaaren über eine mit Komponenten der Kategorie 5 realisierte Verkabelung. Kabel, RJ45-Wanddosen, Patchpanel usw. müssen gemäß dieser Kategorie für eine Übertragungsfrequenz von mindestens 100MHz ausgelegt sein.

Abschlusswiderstand
Bei koaxialen Netzwerktopologien wie 10Base5 oder 10Base2 muss jeder Netzwerkstrang am Anfang und am Ende mit einem Abschlusswiderstand (Terminator) abgeschlossen werden. Der Wert des Abschlusswiderstandes muß der Kabelimpedanz entsprechen; bei 10Base5 oder 10Base2 sind dies 50 Ohm.

Administrator
Systemverwalter, der im lokalen Netzwerk uneingeschränkte Zugriffsrechte hat und für die Verwaltung und Betreuung des Netzwerks zuständig ist. Der Administrator vergibt unter anderem die IP-Adressen in seinem Netzwerk und muss die Einmaligkeit jeder IP-Adresse gewährleisten.

ARP – *Address Resolution Protocol*
Über ARP wird die zu einer IP-Adresse gehörende Ethernet-Adresse eines Netzwerkteilnehmers ermittelt. Die ermittelten Zuordnungen werden auf jedem einzelnen Rechner in der ARP-Tabelle verwaltet. In Windows-Betriebssystemen kann man auf die ARP-Tabelle mit Hilfe des ARP-Befehls Einfluss nehmen.
Eigenschaften und Parameter des ARP Kommandos in der DOS-Box:

- ARP -A listet die Einträge der ARP-Tabelle auf
- ARP -S <IP-Adresse> <Ethernet-Adresse> fügt der ARP-Tabelle einen statischen Eintrag hinzu
- ARP -D <IP-Adresse> löscht einen Eintrag aus der ARP-Tabelle

ARP ist im Internet-Standard RFC-826 definiert; *vgl. a.* 30ff.

AUI – *Attachment Unit Interface*
Schnittstelle zur Anbindung eines externen Ethernet-Transceivers.
Getrennt nach Sende-, Empfangs- und Kollisions-Information werden die Daten vom Transceiver auf einem 15-poligen D-SUB-Steckverbinder zur Verfügung gestellt. Der Anschluss des Endgerätes erfolgt über ein 8-adriges TP-Kabel von max. 50m Länge.
Während die AUI-Schnittstelle in der Vergangenheit hauptsächlich zur Ankopplung von Endgeräten an 10Base5-Transceiver (Yellow-Cable) genutzt wurde, verwendet man sie heute eher zur Anbindung an LWL-Transceiver (Glasfaser) o.ä.

BNC – *Bayonet Neill Concelmann*
Bei der BNC-Steckverbindung handelt es sich um einen Bajonettverschluss zum Verbinden zweier Koaxialkabel. BNC-Steckverbindungen werden in 10Base2-Netzwerken zur mechanischen Verbindung der RG-58-Kabel (Cheapernet) verwendet.

BootP – *Boot Protocol*
Dieses ältere Protokoll zum Booten von PCs ohne Festplatte über das Netzwerk ist der Vorläufer von DHCP. Auch moderne DHCP-Server unterstützen immer noch BootP-Anfragen. Heute wird BootP in erster Linie eingesetzt, um Embedded-Systemen eine IP-Adresse zuzuteilen. Dazu muss auf dem DHCP-Server ein reservierter Eintrag hinterlegt werden, in dem der MAC-Adresse des Embedded-Systems eine feste IP-Adresse zugeordnet ist; *vgl. a.* 58ff

Bridge
Bridges verbinden Teilnetze miteinander und entscheiden anhand der Ethernet-Adresse, welche Pakete die Bridge passieren dürfen und welche nicht. Die dazu notwendigen Informationen entnimmt die Bridge Tabellen, die je nach Modell vom Administrator eingegeben werden müssen oder von der Bridge dynamisch selbst erstellt werden; *vgl. a.* **Router**

Broadcast
Als Broadcast bezeichnet man einen Rundruf an alle Netzteilehmer. Eine typische Broadcast-Anwendung ist der ARP-

Request (siehe ARP). Auch andere Protokolle – etwa **RIP** oder **DHCP** – nutzen Broadcast-Meldungen.
Broadcast-Meldungen werden nicht über Router oder Bridges weitergegeben.

Browser
Client-Programm mit grafischer Benutzeroberfläche, das dem Anwender die Möglichkeit gibt, Webseiten anzuzeigen und andere Dienste im Internet zu nutzen. *vgl. a.* 92.

Bus-System
Bei einem Bus-System teilen sich mehrere Endgeräte eine einzige Datenleitung (Busleitung). Da zu einer gegebenen Zeit jeweils nur ein Endgerät die Datenleitung benutzen darf, erfordern Bus-Systeme immer ein Protokoll zur Regelung der Zugriffsrechte. Klassische Bus-Systeme sind die Ethernet-Topologien 10Base2 und 10Base5.

Cheapernet
Andere Bezeichnung für Ethernet auf der Basis von 10Base2.

Client
Computer oder Anwendungen, die Dienste von sogenannten Servern in Anspruch nehmen. Server-Dienste können zum Beispiel die Bereitstellung einer COM- oder Drucker-Schnittstelle im Netzwerk, aber auch Telnet und FTP sein; *vgl. a.* 32.

Client-Server-Architektur
System der „verteilten Intelligenz“, bei dem der Client Verbindung zu einem Server aufbaut, um vom Server angebotene Dienste in Anspruch zu nehmen. Manche Server-Anwendungen können mehrere Clients gleichzeitig bedienen.*vgl. a.* 32.

Com-Server
Endgerät in TCP/IP-Ethernet Netzwerken, das Schnittstellen für serielle Geräte über das Netzwerk zur Verfügung stellt.

DHCP – *Dynamic Host Configuration Protocol*
Dynamische Zuteilung von IP-Adressen aus einem Adressenpool.

DHCP wird benutzt, um PCs in einem TCP/IP-Netz automatisch – also ohne manuellen Eingriff – zentral und somit einheitlich zu konfigurieren. Der Systemadministrator bestimmt, wie die IP-Adressen zu vergeben sind und legt fest, über welchen Zeitraum sie vergeben werden.
DHCP ist in den Internet-Standards RFC 2131 (03/97) und RFC 2241 (11/97) definiert; *vgl. a.* 55ff.

DDNS – *Dynamic Domain Name Service*
DNS-Dienst, der auch die Namensauflösung für solche Netzteilnehmer unterstützt, die ihre IP-Adresse dynamisch über DHCP beziehen; *vgl. a.* 65ff.

DNS – *Domain Name Service*
Netzteilnehmer werden im Internet über numerische IP-Adressen angesprochen. Doch weil man sich Namen eben besser merken kann als Nummern, wurde der DNS eingeführt.
DNS beruht auf einem hierarchisch aufgebauten System: Jede Namensadresse wird über eine Top-Level-Domain („de“, „com“, „net“ usw.) und innerhalb dieser über eine Sub-Level-Domain identifiziert. Jede Sub-Level-Domain kann (muss aber nicht) nochmals untergeordnete Domains enthalten. Die einzelnen Teile dieser Namenshierarchie sind durch Punkte voneinander getrennt.
Wird vom Anwender zur Adressierung ein Domain-Name angegeben, erfragt der TCP/IP-Stack beim nächsten DNS-Server die zugehörige IP-Adresse.
Netzwerkressourcen sollten sinnvollerweise einen Domain-Namen erhalten, der im Kontext zu der angeboten Dienstleistung oder dem Firmennamen des Anbieters steht. So lässt sich z.B. *wut.de* in die Top-Level-Domain *de* (= Deutschland) und die Sub-Level-Domain *wut* (= Wiesemann & Theis GmbH) auflösen; *vgl. a.* 61ff.

DNS-Server
DNS-Server stellen im Internet die Dienstleistung zur Verfügung, einen Domain-Namen in eine IP-Adresse aufzulösen.

DynDNS
Bei den meisten Internetzugängen bekommt das angeschlossene Endgerät zum Zeitpunkt der Einwahl eine IP-Adresse aus dem

Adresspool des Internetproviders. Da diese temporäre IP-Adresse nach außen nicht bekannt ist, sind solche Endgeräte normalerweise vom Internet aus nicht adressierbar. Über DynDNS kann einem solchen Internetteilnehmer ein Name gegeben werden. DynDNS aktualisiert die Zuordnung zwischen Namen und temporärer IP-Adresse, sobald der Teilnehmer online geht, so dass eine Erreichbarkeit über den Namen möglich wird; *vgl. a.* 67ff.

E-Mail
Elektronische Post über Internet und Intranet; *vgl. a.* 97ff.

E-Mail-Adresse
Eine E-Mail-Adresse wird benötigt, um einem Anwender elektronische Post senden zu können und setzt sich immer aus dem Mailbox-Namen des Anwenders und der Ziel-Domain, getrennt durch das @-Zeichen zusammen. Ein Beispiel: *info@wut.de* bezeichnet das Info-Postfach auf dem Mailserver von W&T; *vgl. a.* 97ff.

Embedded System
Als Embedded System bezeichnet man eine mikroprozessorgesteuerte Baugruppe, die als eingebetteter Teil eines Gerätes oder einer Maschine im Hintergrund Daten verarbeitet und ggf. Prozesse steuert.

Ethernet
Ethernet ist die zur Zeit bei lokalen Netzen am häufigsten angewandte Technologie. Es gibt drei verschiedene Ethernet Topologien – 10Base2, 10Base5 und 10BaseT –; die Übertragungsrate beträgt 10 Mbit/s; *vgl. auch* 14ff.

Ethernet-Adresse
Die unveränderbare, physikalische Adresse einer Netzwerkkomponente im Ethernet; *vgl. a.* 17.

Fast-Ethernet
Fast-Ethernet ist quasi ein Upgrade der 10BaseT-Topologie von 10MBits/s auf 100 Mbit/s. *vgl. hierzu* **100BaseT4** und **100BaseTX**

Firewall

Unter Firewall versteht man Netzwerkkomponenten, die ähnlich einem Router ein internes Netzwerk (Intranet) an ein öffentliches Netzwerk (z.B. Internet) ankoppeln. Hierbei lassen sich die Zugriffe ins jeweils andere Netz abhängig von der Zugriffsrichtung, dem benutzten Dienst sowie der Authentifizierung und Identifikation des Netzteilnehmers begrenzen oder komplett sperren.
Ein weiteres Leistungsmerkmal kann die Verschlüsselung von Daten sein, wenn z.B. das öffentliche Netz nur als Transitweg zwischen zwei räumlich getrennten Teilen eines Intranet genutzt wird.

FTP – *File Transfer Protocol*

FTP ist ein auf TCP/IP aufsetzendes Protokoll, das es ermöglicht, ganze Dateien zwischen zwei Netzwerkteilnehmern zu übertragen. *vgl. a.* 76ff.

Gateway

Gateways verbinden – wie auch **Bridges** und **Router** – verschiedene Netze miteinander. Während Bridge und Router zwar ggf. die physikalische Art des Netzes umsetzen (z.B. Ethernet/ISDN), das eigentliche Protokoll (z.B. TCP/ IP) aber unberührt lassen, bieten Gateways die Möglichkeit, einen Zugang zu protokollfremden Netzen zu schaffen (z.B. TCP/IP auf Profibus). Ein Gateway hat also unter anderem auch die Aufgabe, unterschiedliche Kommunikationsprotokolle zu übersetzen.
Achtung: bei der Netzwerkkonfiguration in Windows-Betriebssystemen wird auch die Eingabe eines Gateways gefordert. Diese Angabe bezieht sich allerdings auf einen ggf. im Netzwerk vorhandenen Router!

HTML – *Hypertext-Markup-Language*

Auszeichnungssprache, die über Schlüsselwörter vorgibt, wie die Inhalte im Browser angezeigt werden, wo Multimedia-Elemente zu finden sind und welche Elemente wie verlinkt sind; *vgl. a.* 109ff.

HTTP – *Hypertext Transfer Protocol*

Das HTTP-Protokoll setzt auf TCP auf und regelt die Anforderung und Übertragung von Webinhalten zwischen HTTP-Server

und Browser. Damit ist HTTP heute das meistgenutzte Protokoll im Internet; *vgl. a.* 91ff.

Hyperlink
Verweis auf andere Webseiten oder Inhalte innerhalb einer Webseite. Durch einfaches Anklicken des verlinkten Elements gelangt der Anwender auf die gewünschte Webseite. 111ff.

Hub
Ein Hub – oft auch als Sternkoppler bezeichnet – bietet die Möglichkeit, mehrere Netzteilnehmer sternförmig miteinander zu verbinden. Datenpakete, die auf einem Port empfangen werden, werden gleichermaßen auf allen anderen Ports ausgegeben.
Neben Hubs für 10BaseT (10Mbit/s) und 100BaseT (100Mbit/s) gibt es sogenannte Autosensing-Hubs, die automatisch erkennen, ob das angeschlossene Endgerät mit 10 oder 100Mbit/s arbeitet. Über Autosensing-Hubs können problemlos ältere 10BaseT-Geräte in neue 100BaseT-Netzwerke eingebunden werden. *vgl. a.* 16ff.

ICMP – *Internet Control Message Protocol*
Das ICMP-Protokoll dient der Übertragung von Statusinformationen und Fehlermeldungen zwischen IP-Netzknoten. ICMP bietet außerdem die Möglichkeit einer Echo-Anforderung; auf diese Weise läßt sich feststellen, ob ein Bestimmungsort erreichbar ist; *vgl. a.* 70ff.

Internet
Das Internet ist der derzeit weltweit größte Netzverbund, der den angeschlossenen Netzteilnehmern eine nahezu grenzenlose Kommunikationsinfrastruktur zur Verfügung stellt. Durch Einsatz von TCP/IP können die Netzteilnehmer plattformunabhängig im Internet angebotenen Dienste wie E-Mail, FTP, HTTP usw. in Anspruch nehmen.

Intranet
Ein abgeschlossenes Netzwerk (etwa innerhalb eines Unternehmens), in dessen Grenzen die Netzteilnehmer Internet- typische Dienste wie E-Mail, FTP, HTTP usw. in Anspruch nehmen können. In aller Regel gibt es von einem Intranet über Router bzw. Firewalls auch Übergänge in das Internet.

IP – *Internet Protocol*
Protokoll, das die Verbindung von Teilnehmern ermöglicht, die in unterschiedlichen Netzwerken positioniert sind.
vgl. a. 29ff.

IP-Adresse
Die IP-Adresse ist eine 32-Bit-Zahl, die jeden Netzteilnehmer im Internet bzw. Intranet eindeutig identifiziert. Sie besteht aus einem Netzwerkteil (Net-ID) und einem Benutzerteil (Host-ID).
vgl. a. 29ff, 41ff.

ISDN – *Integrated Services Digital Network*
ISDN ist der neue Standard in der Fernmeldetechnik und hat das analoge Fernsprechnetz in Deutschland komplett ersetzt. Bei ISDN werden Telefon und Telefax, aber auch Bildtelefonie und Datenübermittlung integriert. Über ISDN können also abhängig von den jeweiligen Endgeräten Sprache, Texte, Grafiken und andere Daten übertragen werden.
ISDN stellt über die S0 Schnittstelle eines Basisanschlusses zwei Basiskanäle (B-Kanäle) mit je 64 kbit/s sowie einen Steuerkanal (D-Kanal) mit 16 kbit/s zur Verfügung. Der digitale Teilnehmeranschluss hat zusammengefasst eine maximale Übertragungsgeschwindigkeit von 144 kbit/s (2B+D). In den beiden B-Kanälen können gleichzeitig zwei unterschiedliche Dienste mit einer Bitrate von 64 kbit/s über eine Leitung bedient werden.
vgl. a. 20ff.

ISDN-Router
ISDN-Router gestatten es, zwei lokale Netzwerke über das ISDN-Netz eines Telefonnetz-Providers miteinander zu verbinden. Dabei übernehmen ISDN-Router neben den normalen Funktionen eines Routers auch das Handling der ISDN-Verbindung.

LAN – *Local Area Network*
Lokales Netz innerhalb eines begrenzten Gebiets unter Anwendung eines schnellen Übertragungsmediums wie z.B. Ethernet.

MAC-ID
Die unveränderbare, physikalische Adresse einer Netzwerkkomponente (MAC = Media Access Control); *vgl. a.* **Ethernet-Adresse** und 17ff.

NAT – *Network Address Translation*
Durch die explosionsartige Ausweitung des Internets in den letzten Jahren sind freie IP-Adressen knapp geworden und werden nur noch sehr sparsam vergeben. NAT kommt dort zum Einsatz, wo Firmennetze ans Internet angebunden werden. Das Firmennetz ist über einen NAT-fähigen Router mit dem Internet verbunden, arbeitet intern allerdings mit einem eigenen, vom Internet unabhängigen IP-Adressraum. Von außen ist das Netz nur über eine einzige (oder einige wenige) IP-Adresse(n) ansprechbar. Anhand der Portnummer im empfangenen TCP/IP-Paket wird dieses an einen bestimmten internen Netzteilnehmer weitergeroutet. *vgl. a.* 50ff.

Ping – *Packet Internet Groper*
Ping dient in TCP/IP-Netzen zu Diagnosezwecken; mit Hilfe dieser Funktion lässt sich überprüfen, ob ein bestimmter Teilnehmer im Netz existiert und tatsächlich ansprechbar ist.
Die von Ping verwendeten ICMP-Pakete sind im Internet-Standard RFC-792 definiert. *vgl. a.* 70ff.

POP3 – *Post Office Protocol Version 3*
Um eingegangene E-Mails aus dem Postfach auf dem Mailserver abzuholen, wird in den meisten Fällen das POP3-Protokoll benutzt. Auch POP3 setzt auf TCP auf; *vgl. a.* 101ff.

PPP – *Point to Point Protocol*
PPP ist ein erweiterter Nachfolger von **SLIP** und weist u.a. eine verbesserte Fehlerkorrektur auf.
Genau wie SLIP bietet PPP die Möglichkeit, TCP/IP-Geräte, die keinen LAN-Anschluss haben, über die serielle Schnittstelle in TCP/IP-Netze einzubinden. *vgl. a.* 25ff.

Repeater
In lokalen Netzen dient ein Repeater zur Verbindung zweier Ethernet-Segmente, um das Netz über die Ausdehnung eines einzelnen Segmentes hinaus zu erweitern. Repeater geben Datenpakete von einem Netzwerksegment zum anderen weiter, indem sie zwar die elektrischen Signale normgerecht „auffrischen", den Inhalt der Datenpakete dabei aber unverändert lassen. Erkennt der Repeater auf einem der angeschlossenen Segmente einen physikalischen Fehler, wird die Verbindung zu die-

sem Segment abgetrennt („partitioniert"). Die Partitionierung wird automatisch aufgehoben, wenn der Fehler nicht mehr vorhanden ist.
Zwischen zwei Stationen dürfen nicht mehr als vier Repeater liegen. Diese Regel betrifft allerdings lediglich „hintereinander" liegende Repeater – bei der Realisierung baumartiger Netzwerkstrukturen kann also durchaus eine Vielzahl von Repeatern eingesetzt werden.

RIP – *Routing Information Protocol*
Routingprotokolle wie RIP dienen dazu, Veränderungen der Routen zwischen zwei vernetzten Systemen an die beteiligten Systeme weiterzuleiten und so eine dynamische Änderung der Routingtabellen zu ermöglichen. RIP ist im Internet-Standard RFC-1058 definiert.

Router
Router verbinden zwei unterschiedliche Netze, wobei im Gegensatz zu Bridges nicht anhand der Ethernet-Adresse, sondern in Abhängigkeit von der IP-Adresse entschieden wird, welche Datenpakete weiterzuleiten sind.
vgl. a. **Bridge** und 44.

SLIP – *Serial Line Internet Protocol*
SLIP bietet eine einfache Möglichkeit zur Übertragung von TCP/IP-Datenpaketen über serielle Punkt-zu-Punkt-Verbindungen. Damit können Endgeräte, die nicht über einen LAN-Anschluss verfügen, auch über die serielle Schnittstelle ins Netzwerk eingebunden werden.
SLIP arbeitet nach einem sehr einfachen Algorithmus ohne eigene Datensicherungsverfahren: Dem eigentlichen IP-Datenpaket wird ein Startzeichen (dezimal 192) vorangestellt und ein Endzeichen (ebenfalls dezimal 192) angehängt. Um die binäre Transparenz zu erhalten, werden im Datenpaket vorkommende Start- und Endzeichen zuvor durch andere Sequenzen ersetzt. SLIP ist in RFC 1055 beschrieben.

SLIP-Router
Ein SLIP-Router stellt die Hardware und Funktionalität zur Verfügung, um serielle Endgeräte, die über einen TCP/IP-Stack verfügen, in ein Netztwerk einzubinden.

Com-Server stellen z.B. SLIP-Routing als Betriebsart zur Verfügung.

SMTP – *Simple Mail Transfer Protocol*
SMTP regelt den Versand von E-Mails vom Mail-Client zum Mailserver (SMTP-Server) und zwischen den Mailservern und setzt auf TCP auf; *vgl. a.* 100ff.

SNMP – *Simple Network Management Protocol*
SNMP setzt auf UDP auf und ermöglicht die zentrale Administration und Überwachung von Netzwerkkomponenten.
SNMP ist in folgenden Standards spezifiziert: RFC 1052, RFC 1155, RFC 1156, RFC 1157, RFC 1213 und RFC 1441.
vgl. a. 84ff.

STP – *Shielded Twisted Pair*
Abgeschirmtes Datenkabel, bei dem jeweils 2 Kabeladern miteinander verdrillt sind; *vgl. a.* **Twisted Pair**

Subnet-Mask
32-Bit-Wert, der festlegt, welcher Teil der IP-Adresse das Netzwerk und welcher den Netzwerkteilnehmer adressiert.
vgl. a. 42ff.

Switch
Ein Switch bietet wie ein Hub die Möglichkeit, mehrere Netzteilnehmer sternförmig miteinander zu verbinden. Switches vereinigen die Funktionalität eines **Hub** mit denen einer **Bridge**: Ein Switch „lernt" die Ethernet-Adresse des an einem Port angeschlossenen Netzteilnehmers und leitet dorthin nur noch diejenigen Datenpakete weiter, die an diesen Netzteilnehmer adressiert sind. Eine Ausnahme bilden dabei Broadcast-Meldungen, die an alle Ports weitergegeben werden (hier unterscheidet sich der Switch in seiner Funktion von einer Bridge, die Broadcast-Meldungen generell nicht weitergibt).
Neben Switches für 100Base T (100Mbit/s) gibt es sogenannte Autosensing-Switches, die automatisch erkennen, ob das angeschlossene Endgerät mit 10 oder 100Mbit/s arbeitet. Über Autosensing-Switches können problemlos ältere 10BaseT-Geräte in neue 100BaseT-Netzwerke eingebunden werden.
vgl. a. 16ff.

TCP – *Transmission Control Protocol*
TCP setzt auf IP auf und sorgt nicht nur für die Verbindung der Teilnehmer während der Datenübertragung, sondern stellt auch die Korrektheit der Daten und die richtige Abfolge der Datenpakete sicher; *vgl. a.* 32ff.

TCP/IP-Stack
Teil des Betriebssystems oder ein auf das Betriebssystem aufgesetzter Treiber, der alle für die Unterstützung des IP-Protokolls benötigten Funktionen und Treiber zur Verfügung stellt.

Telnet – *Terminal over Network*
In der Vergangenheit kam Telnet vor allem für den Fernzugriff über das Netzwerk auf UNIX-Server zum Einsatz. Über eine Telnet-Anwendung (Telnet-Client) kann von einem beliebigen Rechner im Netz ein Fernzugriff auf einen anderen Rechner (Telnet-Server) erfolgen. Heute wird Telnet auch zur Konfiguration von Netzwerkkomponenten wie z.B. Com-Servern benutzt. Telnet wird unter TCP/IP normalerweise über Portnummer 23 angesprochen; für spezielle Anwendungen können aber auch andere Portnummern verwendet werden. Telnet setzt auf TCP/IP als Übertragungs- und Sicherungsprotokoll auf. *vgl. a.* 72ff. Telnet ist im Internet-Standard RFC 854 definiert.

TFTP – *Trivial File Transfer Protocol*
Das Trivial File Transfer Protocol (TFTP) ist neben FTP ein weiteres Protokoll zur Übertragung ganzer Dateien. TFTP bietet nur ein Minimum an Kommandos, unterstützt keine aufwendigen Sicherheitsmechanismen und benutzt UDP als Übertragungsprotokoll. Da UDP ein ungesichertes Protokoll ist, wurden in TFTP eigene minimale Sicherungsmechanismen implementiert. *vgl. a.* 80ff.
Das Trivial File Transfer Protocol ist in den Standards 783, 906, 1350 und 1782 bis 1785 beschrieben.

Transceiver
Das Wort Transceiver ist eine Zusammensetzung aus Transmitter (Sender) und Receiver (Empfänger). Der Transceiver realisiert den physikalischen Netzzugang einer Station an das Ethernet und ist bei den modernen Ethernet-Topologien 10Base2 und 10BaseT auf der Netzwerkkarte integriert. Nur bei 10Base5 (*vgl.*

auch **AUI**-Anschluss) ist der Transceiver als externe Komponente direkt am Netzwerkkabel angebracht.

Twisted Pair
Datenkabel, bei dem jeweils zwei Kabeladern miteinander verdrillt sind. Durch die paarige Verseilung einzelner Doppeladern wird ein deutlich reduziertes Übersprechverhalten zwischen den Doppeladern in einem Kabel erreicht. Man unterscheidet bei Twisted-Pair-Kabeln zwischen ungeschirmten UTP-Kabeln (Unshielded Twisted Pair) und geschirmten STP-Kabeln (Shielded Twisted Pair).
TP-Kabel werden vor allem in der Netzwerktechnik eingesetzt und sind nach ihren maximalen Übertragungsfrequenzen kategorisiert; in der Praxis kommen heute meist zwei Typen zum Einsatz:

- Kategorie-3-Kabel erlauben eine maximale Übertragungsfrequenz von 25MHz, ausreichend für den Einsatz in 10BaseT-, aber auch 100BaseT4-Netzen.
- Kategorie-5-Kabel erlauben eine maximale Übertragungsfrequenz von 100MHz und reichen damit für alle heutigen Netzwerktopologien aus.

UDP – *User Datagram Protocol*
UDP ist ein Protokoll, das wie TCP auf IP aufsetzt, im Gegensatz dazu aber verbindungslos arbeitet und über keine Sicherheitsmechanismen verfügt. Der Vorteil von UDP gegenüber IP ist die höhere Übertragungsgeschwindigkeit. *vgl. a.* 36.

URL – *Uniform Resource Locator*
Adress- und Protokollinformation für den Browser. Über den URL gibt der Anwender dem Browser vor, welches Protokoll genutzt wird, auf welchem Webserver die Seite liegt, und wo diese auf dem Webserver zu finden ist. *vgl. a.* 96ff, 108.

UTP – *Unshielded Twisted Pair*
Im Gegensatz zu **Twisted Pair** ein nicht abgeschirmtes Datenkabel, bei dem jeweils zwei Kabeladern miteinander verdrillt sind.

Web-Based Management
Unter Web-Based Management versteht man die Möglichkeit, ohne spezielle Software Endgeräte übers Netzwerk direkt im Browserfenster zu konfigurieren.

WWW – *World Wide Web*
WWW wird häufig mit dem Internet gleichgesetzt. Das stimmt nicht ganz: Während das Internet die physikalischen Verbindungswege beschreibt, definiert das WWW einen Standard, der dem Anwender über eine grafische Benutzeroberfläche durch einfachste Bedienung Zugang zu den gängigsten Internetdiensten verschafft. Per Mausklick lassen sich Webseiten anfordern, E-Mails verschicken und Dateien downloaden. *vgl. a.* 108ff.

Zahlensysteme

Neben dem dezimalen Zahlensystem (Zeichenvorrat: 0–9, neue Stelle bei 10) werden in der Computertechnik auch oft das binäre Zahlensystem (Zeichenvorrat 0–1, Stellensprung bei 2) und das hexadezimale Zahlensystem (Zeichenvorrat: 0–9 + A–F, neue Stelle bei 16) verwendet.
In der folgenden Tabelle finden Sie einige Beispiele für die Darstellung gebräuchlicher Werte in den drei Zahlensystemen:

Binär	Dez.	Hex.	Binär	Dez.	Hex.
0	0	0	11111	31	1F
1	1	1	100000	32	20
10	2	2	...	...	...
11	3	3	111111	63	3F
100	4	4	1000000	64	40
101	5	5	...	...	...
110	6	6			
111	7	7	...	...	...
1000	8	8	1111111	127	7F
1001	9	9	10000000	128	80
1010	10	A	11000000	192	C0
1011	11	B	11100000	224	E0
1100	12	C	11110000	240	F0
1101	13	D	11111000	248	F8
1110	14	E	11111100	252	FC
1111	15	F	11111110	254	FE
10000	16	10	11111111	255	FF

TCP/IP-Ethernet in der Praxis

Der wohl entscheidendste Grund, ein Gerät mit einer Netzwerkschnittstelle auszurüsten, war in der Vergangenheit die hohe Übertragungsgeschwindigkeit. Mit der immer größeren Ausdehnung von Firmennetzen und dem Zusammenwachsen von Intranet und Internet gewinnt die große Standortflexibilität und die Benutzung von vorhandener Infrastruktur immer mehr Gewicht bei der Entscheidung, nicht nur PCs, File-Server und Drucker mit einem Netzwerkanschluss auszurüsten.

Zum Abschluss möchten wir Ihnen die Idee vorstellen, verschiedenste Signale direkt über das Netzwerk zu erfassen, zu steuern und auszuwerten.

Wir haben zu diesem Zweck Com-Server entwickelt, mittels derer sich serielle Endgeräte ans Netzwerk anschließen lassen.

Eine weitere Gerätegruppe erlaubt es, analoge und digitale Signale über das Netzwerk zu erfassen und zu steuern. Wir haben für diese Technik den Namen Web-IO gewählt.

Com-Server - Anwendungsbeispiele aus der Praxis

Com-Server, das sind kleine Boxen, die mit einem Ethernet-Anschluss auf der einen und mit bis zu 4 seriellen Ports auf der anderen Seite ausgerüstet sind.

Der Ethernet-Anschluss arbeitet je nach Modell mit 10Mbit oder mit 10/100Mbit autosensing Technik (automatische Erkennung). Für die serielle Seite kann RS232, RS422, RS485 oder 20mA gewählt werden.

Durch die W&T *easy start* Technik wird die Integration ins Netzwerk zum Kinderspiel: Netzwerkkabel anschließen, Gerät mit Spannung versorgen, IP-Adresse mit dem komfortablen WuTility-Tool vergeben - fertig!

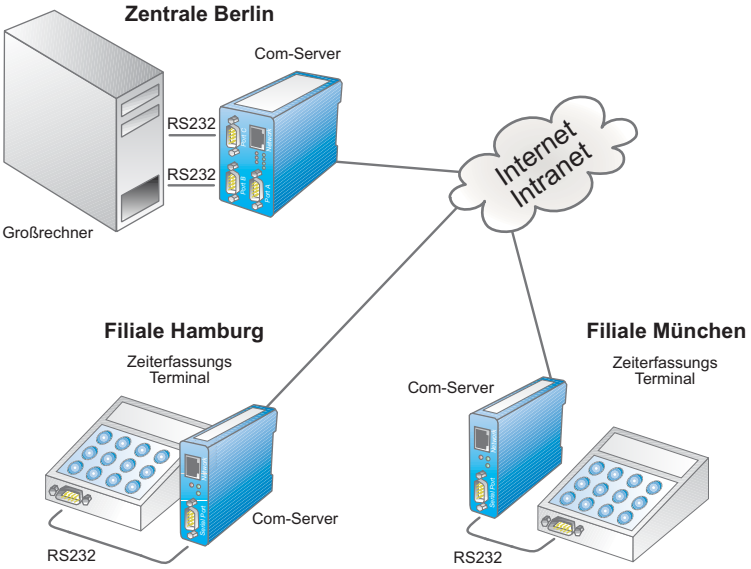
Ausführliche Datenblätter der verschiedenen Com-Server finden Sie im Anhang und auf unserer Webseite www.WuT.de.

Box-to-Box - Der Tunnel durchs Netzwerk

Zwei Com-Server werden so konfiguriert, dass Daten, die am seriellen Port des 1. Com-Servers eingehen, über das Netzwerk zum 2. Com-Server weitergeroutet werden. Der 2. Com-Server gibt die seriellen Daten dann wieder aus. Das Ganze funktioniert selbstverständlich in beide Richtungen.

Ein Beispiel:
Die Zeiterfassungsdaten bei einer Bank in Berlin werden über RS232 vom Zeiterfassungs-Terminal an einen UNIX-Großrechner übertragen. Die Daten aus den Zweigstellen Hamburg und München wurden bislang auf Diskette gezogen und mit der Post verschickt.

Heute werden die Daten über Com-Server im Box-to-Box Modus einfach über die ohnehin bestehende Intranet Verbindung mit übertragen. Der erste Com-Server „steckt" die RS232-Daten in TCP-Pakete und routet sie durchs Netz an den zweiten Com-Server. Der „packt" die RS232-Daten wieder aus und gibt sie an den Zentral-Rechner.

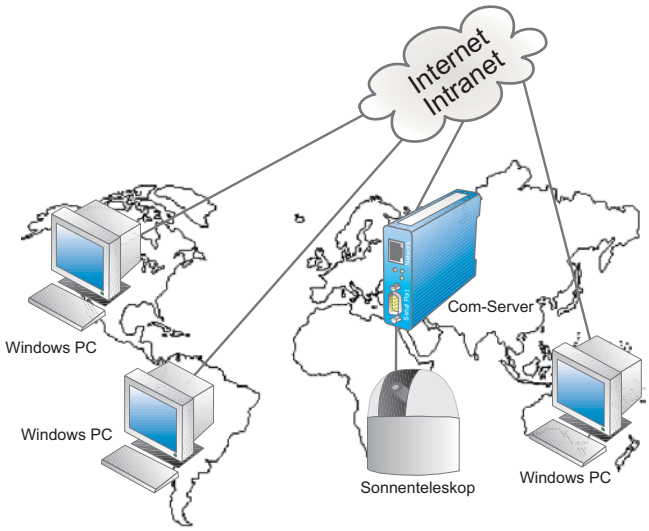


Die COM-Umlenkung - Der „ganz woanders“ COM-Port

Mit Hilfe des Com-Umlenkungs Treibers und eines oder mehrerer Com-Server lassen sich in Microsoft Windows Betriebssystemen zusätzliche COM-Ports einrichten, die an einer beliebigen Position im Netzwerk sein können.

Ein Beispiel:
Ein Sonnenteleskop in Afrika übergibt seine Bilddaten über ein TCP/IP-Netzwerk an verschiedene Universitäten auf der ganzen Welt. Die Positionierungskoordinaten des Teleskops aber lassen sich leider nur über eine RS232-Schnittstelle direkt vor Ort eingeben. Bisher mussten diese Parameter telefonisch einem Mitarbeiter durchgegeben werden, der die nötigen Einstellungen vorgenommen hat.

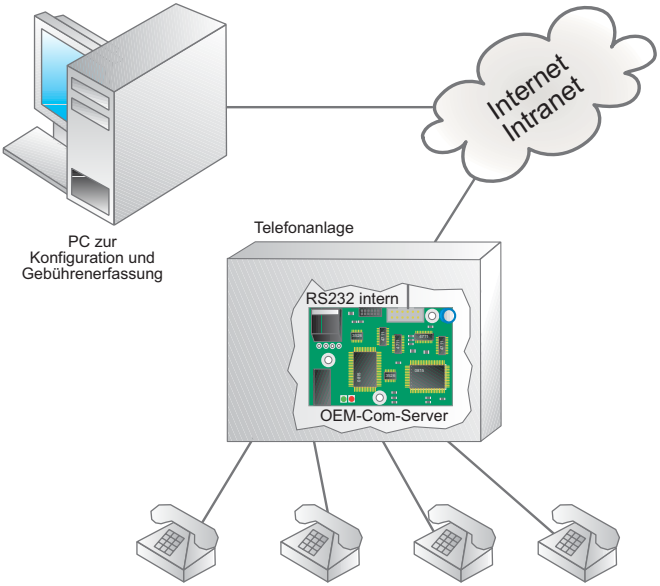
Seit der Konfigurationsport des Sonnenteleskops an einen Com-Server angeschlossen ist, können die Benutzer in Sydney, Washington und Tegucigalpa über die nun virtuelle COM3 ihres PC das Sonnenteleskop online positionieren.



TCP/IP-Sockets -
Mit dem eigenen Programm auf den seriellen Port

Über TCP oder auch UDP erlaubt der Com-Server eine direkte Kommunikation mit dem seriellen Port des Com-Servers.

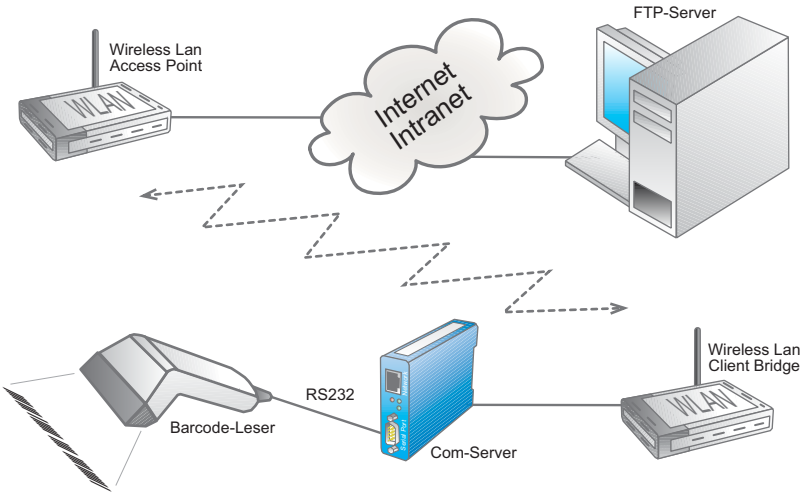
Ein Beispiel:
Ein Hersteller von Telefonnebenstellenanlagen schließt die RS232 Konfigurations- und Gebührendatenschnittstelle an die eingebaute Com-Server-OEM Platine an. Um die Gebührendaten zu erfassen und die Anlage zu konfigurieren, wurde eine kleine Software programmiert, die das bequem übers Netzwerk erledigt.



FTP - Serielle Daten direkt in eine Datei

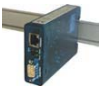
Der Com-Server unterstützt unter anderem auch FTP, als Client oder Server. Damit ist ein Dateitransfer zum oder vom seriellen Port des Com-Server problemlos möglich.

Ein Beispiel:
Im Lager einer Spedition sollen alle ein- und ausgehenden Pakete per Barcode erfasst werden. Dazu wurden auf den Gabelstaplern je ein Barcode-Leser mit einem Com-Server verbunden, der als FTP-Client konfiguriert wurde. Über eine Wireless Client Bridge kann der Com-Server via Funk-LAN den netzwerkseitig angeschalteten Access Point und somit das Firmennetzwerk erreichen. Die eingelesenen Barcodes werden nun automatisch per FTP in Dateien auf dem File-Server der Spedition geschrieben, unabhängig davon, wo der Gabelstapler auf dem Firmengelände im Einsatz ist.



Natürlich gibt es eine Vielzahl weiterer Anwendungsbeispiele für den Einsatz von Com-Servern. CNC, DNC, Betriebsdatenerfassung, Messwerterfassung, Fernwartung..... um hier nur einige zu nennen.

Com-Server - Die verschiedenen Modelle

Model	Com-Server Highspeed Industry #58631	Com-Server Highspeed Isolated #58633	Com-Server Low Cost Industry #58511	Com-Server Highspeed Kompakt #58231	Com-Server Highspeed Office #58031 #58034	OEM Com-Server OEM	Com-Server Highspeed 19" OEM OEM 19"
							
Serielle Ports							
Anzahl	**1**	**3**	1	1	1, **4**	1	1 , 4
RS232, RS422, RS422 umschaltbar	✓	✓		✓	✓	✓	✓
nur RS232			✓			TTL	
Optional 20mA	✓			✓	✓	✓	✓
Netzwerk							
10/100BaseT	✓	✓		✓	✓	✓	✓
nur 10BaseT			✓			✓	
Stromversorgung							
12V - 24V AC oder DC auf Schraubklemme	✓	✓					
5V auf Hohlklinke			✓	✓		OEM	OEM
230V integriertes Netzteil					✓		
Netzteil im Lieferumfang	✓	✓	✓	✓			
Gehäuse							
Kunststoffgehäuse für Hutschienenmontage	✓	✓	✓				
Robustes Aluminiumgehäuse				✓			
Tischgehäuse					✓		

Detaillierte technische Daten und Informationen zu diesen Produkten finden Sie auf unserer Webseite http://www.WuT.de

Web-IO - Anschlussbeispiele aus der Praxis

Web-IO, das sind kleine Baugruppen, mit denen analoge und digitale Signale per TCP/IP-Ethernet gesteuert und überwacht werden können.

Ein integrierter HTTP-Server erlaubt mittels Web-Based Control Technik einen bequemen Zugriff. Nicht nur Konfiguration, sondern auch Steuerung und Auswertung sind ohne gerätespezifische Spezialsoftware, selbst für den unbedarften Anwender, sofort von jedem Browser möglich.

Web-Based Control ermöglicht aber auch die individuelle Programmierung von Browser-Anwendungen. Technische Visualisierungen und Steueraufgaben sind so ohne Spezialsoftware auf jedem PC verfügbar.

Zusätzlich ist die Einbindung in bestehende Management- und Visualisierungssysteme ebenfalls kein Problem. SNMP und OPC, aber auch der direkte Zugang via TCP und UDP machen eine Integration ganz einfach.

SNMP-Traps und E-Mail-Versand, bei vorhandener Infrastuktur bis hin zur SMS, erlauben aber auch ganz neue Wege der Signalverarbeitung.

Das Datenbanktool Sensobase erlaubt eine kontinuierliche Archivierung von Messdaten und IO-Zuständen in beliebige ODBC-Datenbanken.

Netzwerkseitig sind alle Web-IO mit einem 10/100BaseT autosensing Anschluss ausgestattet. Die Spannungsversorgung ist in einem weiten Bereich zwischen 12 V und 24 V Gleich- oder Wechselspannung bzw. über ein 230 V Netzteil möglich.

Durch diese Flexibilität bietet Web-IO alle Eigenschaften für den Einsatz in Anlagen- und Haustechnik sowie Labor- und Büroanwendungen.

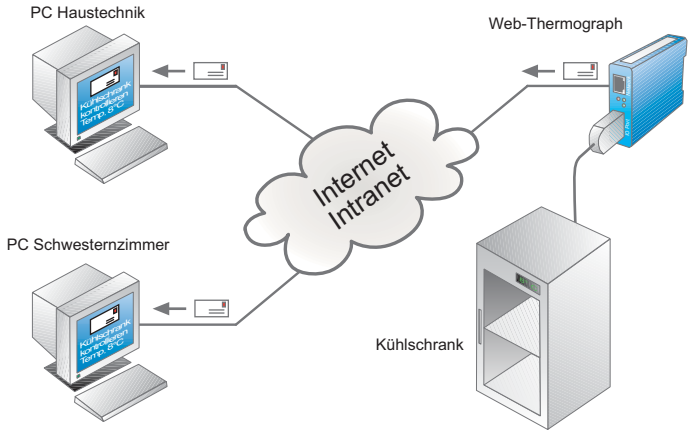
Web-Thermographen - Temperaturüberwachung im Netz

Der Web-Thermograph erlaubt den Anschluss von bis zu acht Temperatursensoren in NTC oder PT100 Technik. Die Temperaturen lassen sich jederzeit im Browser abrufen; als Tabelle, als Kennlinie oder in einer selbst erstellten Homepage. Für jeden Sensor können individuelle Grenzwerte festgelegt werden. Bei Grenzwertüber- oder -unterschreitung kann per E-Mail oder SNMP-Trap Alarm geschlagen werden. Mittels OPC lassen sich die gemessenen Werte in bestehende Visualisierungs- und Gebäudemanagementsysteme integrieren.

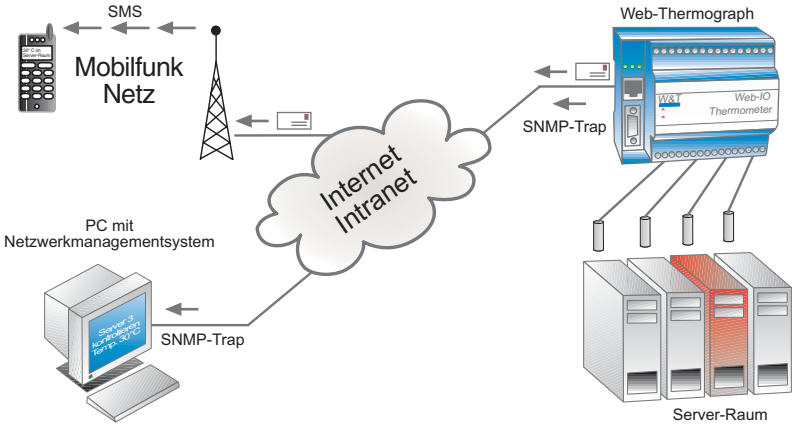
Ein Beispiel:
In einem Krankenhaus müssen spezielle Medikamente in einem Kühlschrank zwischen 3° und 8° C gelagert werden.

In der Vergangenheit wurde die Temperatur des Kühlschrankes einmal pro Stunde von der Stationsschwester kontrolliert und in eine Tabelle eingetragen.

Heute überwacht der Web-Thermograph den Kühlschrank. Steigt die Temperatur über 7,5° C bekommt die Stationsschwester eine E-Mail. Übersteigt die Temperatur sogar 8° C wird zusätzlich eine E-Mail an die Haustechnik gesendet. Darüber hinaus wird einmal pro Woche der gesamte Temperaturverlauf per Download als Excel-Tabelle gesichert.



Ein zweites Beispiel:
In einem Rechenzentrum sind bereits mehrmals diverse Festplatten den „Hitzetod“ gestorben, weil über Nacht die Klimaanlage des Serverraumes ausgefallen ist. Heute sendet der eingesetzte Web-IO Thermograph via E-Mail an den Mobilfunknetzbetreiber eine SMS direkt an den Techniker. Zusätzlich wird ein SNMP-Trap an das Netzwerkmanagementsystem gesendet. So kann zu jeder Uhrzeit rechtzeitig reagiert werden.



Web-Thermo-Hygrographen - Temperatur u. Luftfeuchte

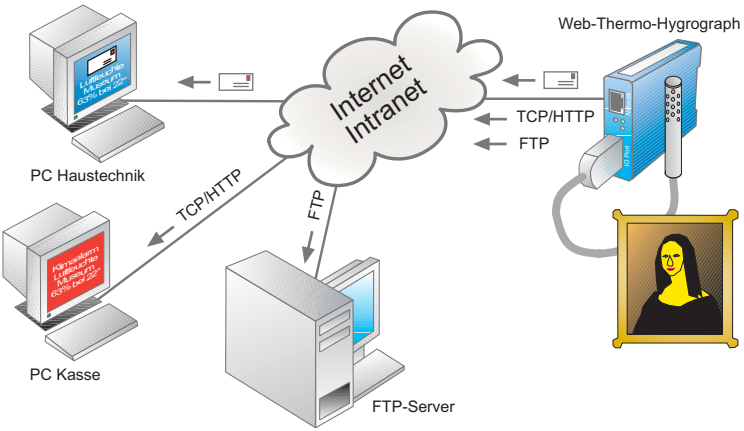
In vielen Bereichen und Anwendungen gibt es festgelegte Anforderungen an das Klima. Der Web-Thermo-Hygrograph ist mit einem kombinierten Temperatur-Feuchtesensor ausgestattet und kann neben der Temperatur auch die relative Luftfeuchte ausrechnen und überwachen. Die Klimawerte lassen sich jederzeit im Browser abrufen, auch als Kennlinie. Neben der Alarmierung per E-Mail oder SNMP-Trap bei Grenzwertüberschreitung können Temperatur und Luftfeuchte auch kontinuierlich mit Zeitstempel aufgezeichnet und mittels FTP in eine Datei geschrieben werden. Damit kann der Web-Thermo-Hygrograph beispielsweise auch zur Qualitätssicherung eingesetzt werden.

Ein Beispiel:
In einem Kunstmuseum ist bei 20-22°C eine konstante Luftfeuchte von 60% nötig, damit die ausgestellten Ölgemäde langfristig unbeschadet bleiben. Obwohl die Klimaanlage des Museums Temperatur und Luftfeuchte automatisch nachregelt, verlangt der hohe Wert der ausgestellten Exponate eine zusätzliche Kontrolle des Raumklimas.

In der Vergangenheit wurde von einem Museumsmitarbeiter als zusätzliche Kontrolle stündlich in jedem Raum ein Hygrometer abgelesen.

Diese Aufgabe übernimmt heute ein Web-Thermo-Hygrometer. Die aktuellen Werte werden permanent auf dem PC des Kassierers angezeigt. Verlässt das Klima die vorgegebenen Werte färbt sich diese Anzeige rot und es ertönt ein Warnsignal. Darüber hinaus wird eine E-Mail an die Haustechnik gesendet.

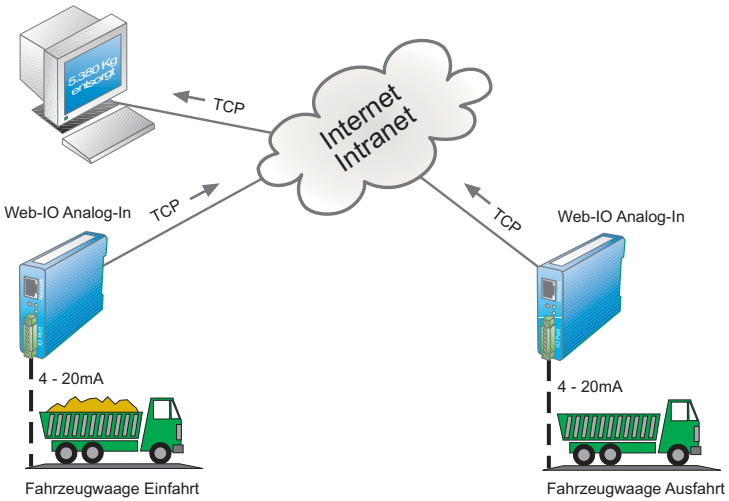
Für einen lückenlosen Nachweis schreibt das Web-Thermo-Hygrometer stündlich die Werte per FTP in eine Datei.



Web-IO Analog-In - Messdatenerfassung über's Netzwerk

Mit dem Web-IO Analog-In können Ströme von 0-20mA und Spannung von 0-10V gemessen werden. Die ermittelten Werte können im Browser angezeigt werden oder lassen sich mittels OPC in bestehende Visualisierungsprogramme übernehmen. Wie bei allen anderen Web-IO auch kann bei Grenzwertüberschreitung eine Alarmierung per E-Mail oder SNMP-Trap konfiguriert werden. Natürlich sind die Werte auch über TCP- oder UDP-Sockets abrufbar, so dass eine Integration in eigene Programme kein Problem ist.

Ein Beispiel:
Im Ein- und Ausfahrbereich einer Müllverbrennungsanlage befindet sich je eine Fahrzeugwaage mit 4-20mA Ausgang. Es sollen alle Fahrzeuge bei der Einfahrt und bei der Ausfahrt gewogen werden. Über die Gewichtsdifferenz soll eine PC-Anwendung die Entsorgungsmenge ausrechnen. Dazu werden die Waagen jeweils mit einem Web-IO Analog-In verbunden. Obwohl Ein- und Ausfahrt nicht im gleichen Bereich des Anlagengeländes liegen, werden die ermittelten Gewichte über das bestehende Firmennetzwerk an den auswertenden PC übermittelt.



Web-IO Digital

Über HTTP (Browser), TCP, UDP, SNMP oder OPC können digitale Eingänge und Ausgänge per Netzwerk gesteuert und ausgewertet werden. Zusätzlich steht ein Box-to-Box Modus zur Verfügung, mit dem über einen Eingang an Web-IO 1 ein Ausgang an Web-IO 2 gesetzt werden kann.

Die digitalen Eingänge sind in Gruppen galvanisch getrennt und lassen sich mit ±30 V ansteuern.

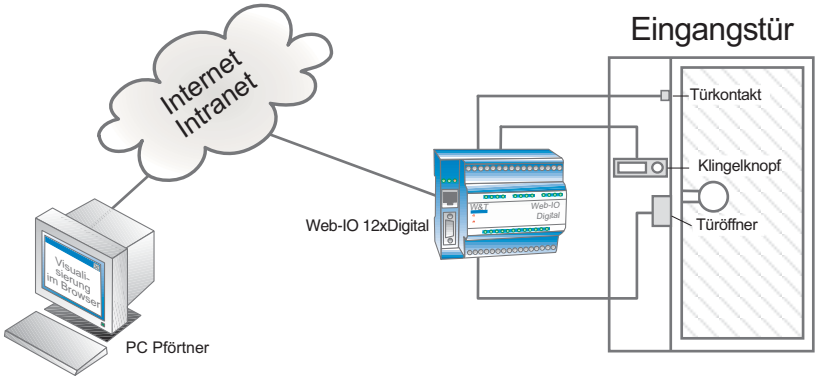
Die Ausgänge schalten über einen gemeinsamen Versorgungseingang Spannungen von 6 bis 30 V. Der maximale Ausgangsstrom pro Signal beträgt 500mA. Ein thermischer Überlastschutz sorgt dabei für Kurzschlussfestigkeit. Die Ausgänge können paarweise oder in Gruppen bis zu 4 Ausgängen parallelgeschaltet werden, um auch höhere Schaltströme zu realisieren. Freilaufdioden für Relaisanschaltung sind natürlich integriert.

Über Alarmkonfigurationen lassen sich beliebige Eingangsmuster überwachen und im Alarmfall kann per E-Mail, SNMP-Trap oder UDP-Meldung gewarnt werden.

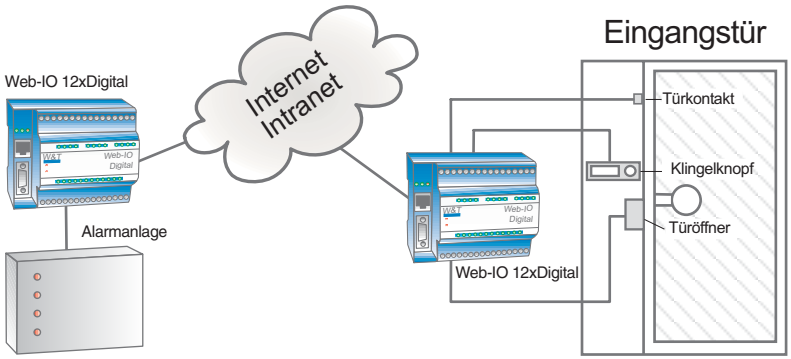
Auf den Punkt gebracht: mit Web-IO Digital kann jedes Gerät, das einen Kontakt oder ein elektrisches Signal zur Verfügung stellt, ins Netzwerk gebracht werden.

Ein Beispiel:
Der Hintereingang eines Firmengebäudes hat einen Klingelknopf, einen Türöffner und einen Kontakt, der überwacht, ob die Tür geschlossen ist.

Klingelknopf und Türkontakt sind über die Eingänge 0 und 1 an das Web-IO angeschlossen. Ausgang 0 des Web-IO steuert den Türöffner. Über TCP-Sockets sind diese Signale des Web-IO in ein Gebäudevisualisierungssystem eingebunden. So bekommt der Pförtner am Haupteingang an seinem PC ein akustisches Signal, wenn jemand den Klingelknopf drückt. Durch Mausklick kann er den Türöffner betätigen. Ob die Tür offen oder geschlossen ist, wird ebenfalls angezeigt.

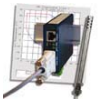


Zusätzlich ist der Türkontakt an Eingang 4 des Web-IO angeschlossen. Im Keller, neben der Alarmanlage, befindet sich ein zweites Web-IO. Ausgang 4 dieses Web-IO steuert einen Alarmeingang der Alarmanlage an. Die Web-IO wurden so konfiguriert, dass Eingang 4 des ersten Web-IO Box-to-Box mit Ausang 4 des zweiten Web-IO zusammenarbeitet. Der Zustand des Türkontaktes ist auf diese Weise über das Netzwerk 1:1 mit der Alarmanlage verbunden. Wird in der Nacht, wenn die Alarmanlage scharf geschaltet ist, die Tür geöffnet, schlägt diese sofort an.



Web-IO · verschiedene Modelle

Web-Thermographen und Web-Thermo-Hygrographen

Model	*Web-Thermograph* #57605	*Web-Thermo-Hygrograph* #57606	*Web-Graph 2x Thermometer* #57607	*Web-Graph 8x Thermometer* #57608
				
IO-Ports				
für NTC-Thermofühler	1			
für PT100 Sensoren	1	1	2	8
für PT1000 Sensoren		1	2	8
Thermo-Hygrofühler		1		
Netzwerk				
10/100BaseT autosensing	✓	✓	✓	✓
PPP über RS232 und Modem	in Vorbereitung	in Vorbereitung	in Vorbereitung	in Vorbereitung
Software-Schnittstellen				
Web-based Control	✓	✓	✓	✓
TCP-Sockets	✓	✓	✓	✓
UDP-Sockets	✓	✓	✓	✓
E-Mail	✓	✓	✓	✓
FTP	✓	✓	✓	✓
SNMP	✓	✓	✓	✓
SYSLOG	✓	✓	✓	✓
OPC	✓	✓	✓	✓
LabView	✓	✓	✓	✓
Sensobase	✓	✓	✓	✓
Stromversorgung				
12V - 24V AC oder DC auf Schraubklemme	✓	✓	✓	✓
Netzteil im Lieferumfang	✓	✓	✓	✓
Gehäuse				
Kunststoffgehäuse für Hutschienenmontage	✓	✓	✓	✓

Detaillierte technische Daten und Informationen zu diesen Produkten finden Sie auf unserer Webseite http://www.WuT.de

Web-IO Analog In

Model	Web-IO Analog In 0..20mA / 0..10V #57641	Web-IO Analog In 2x 0..20mA #57642	Web-IO Analog In 2x 0..10V #57643
			
IO-Ports			
0..20mA (4..20mA)	1	2	
0..10V	1		2
Netzwerk			
10/100BaseT autosensing	✓	✓	✓
PPP über RS232 und Modem	in Vorbereitung	in Vorbereitung	in Vorbereitung
Software-Schnittstellen			
Web-based Control	✓	✓	✓
TCP-Sockets	✓	✓	✓
UDP-Sockets	✓	✓	✓
E-Mail	✓	✓	✓
FTP	✓	✓	✓
SNMP	✓	✓	✓
SYSLOG	✓	✓	✓
OPC	✓	✓	✓
LabView	✓	✓	✓
Sensobase	✓	✓	✓
Stromversorgung			
12V - 24V AC oder DC auf Schraubklemme	✓	✓	✓
Netzteil im Lieferumfang	✓	✓	✓
Gehäuse			
Kunststoffgehäuse für Hutschienenmontage	✓	✓	✓

Detaillierte technische Daten und Informationen zu diesen Produkten finden Sie auf unserer Webseite http://www.WuT.de

Web-IO Digital

Model	Web-IO 2x Digital #57633	Web-IO 12x Digital #57630	Web-IO 12x Digital + Com-Server #57631	Web-IO 24x Digital OEM 19" #57632
				
IO-Ports				
Digitale Inputs	2	12	12	24
Digitale Outputs	2	12	12	24
Serielle Com-Server Ports				
Anzahl			1	
Netzwerk				
10/100BaseT autosensing	✓	✓	✓	✓
PPP über RS232 und Modem	in Vorbereitung	in Vorbereitung	in Vorbereitung	in Vorbereitung
Software-Schnittstellen				
Web-based Control	✓	✓	✓	✓
TCP-Sockets	✓	✓	✓	✓
UDP-Sockets	✓	✓	✓	✓
E-Mail	✓	✓	✓	✓
FTP	✓	✓	✓	✓
SNMP	✓	✓	✓	✓
SYSLOG	✓	✓	✓	✓
Box-to-Box	✓	✓	✓	✓
OPC	✓	✓	✓	✓
LabView	✓	✓	✓	✓
Logik	✓	✓	✓	✓
Stromversorgung				
12V - 24V AC oder DC auf Schraubklemme	✓	✓	✓	
24V DC auf VG96 Stecker				✓
Gehäuse				
Kunststoffgehäuse für Hutschienenmontage	✓	✓	✓	

Detaillierte technische Daten und Informationen zu diesen Produkten finden Sie auf unserer Webseite http://www.WuT.de

Probieren geht über Studieren

Web-IO Starterkit #57002

Das im Koffer eingebaute Modell einer Straßenkreuzung ist mit einer kompletten Ampelsignalanlage ausgestattet. Alle Ampellichter, Signalgeber, Anforderungsschalter und Induktionsschleifen der Ampelanlage sind mit den Ein- und Ausgängen eines W&T Web-IO 12x Digital verbunden und lassen sich über TCP/IP lesen und steuern.

Das Handbuch führt Schritt für Schritt vom Schalten einzelner Ampellichter bis zur Entwicklung einer kompletten, ereignisabhängigen Ampelsteuerung. Alle Programmbeispiele sind in Visual Basic, Delphi und C++ auf der beiliegenden CD zu finden.

Produkt-Eigenschaften:

- **Inhalt des Koffers**
  Web-IO 12x Digital
  Schulungsboard mit Ampelanlage
  Adapterplatinen für Web-IO und Flachbandkabel
  Netzteil 230V - 24V DC/1A
  2 x Modellauto mit Magnet für induktive Erkennung
  Handbuch TCP/IP-Ethernet und Web-IO
  Begleit-CD mit Anleitungen, Aufgaben und Programmierbeispielen (Delphi, VB, C++)
- **Funktionen des Schulungsboards**
  28 Ampelsignale in LED-Ausführung
  2 Beeper, 8 Taster als Anforderungsdrücker
  4 Hall-Sensoren zur Fahrzeugerkennung

EnOcean: Steuern und Überwachen per Funk

Der Begriff „EnOcean“ steht für Schalten, Bedienen, Steuern und Überwachen per Funkübertragung bei gleichzeitiger völliger Wartungsfreiheit der eingesetzten Funkkomponenten.

Funkwellen sind ein ideales Medium zur Verteilung von Steuerbefehlen an alle elektrischen Verbraucher eines Gebäudes, ohne dass hierzu Leitungen verlegt werden müssen. Durch den Verzicht auf die leitungsgebundene Übertragung lassen sich Installationen nach Bedarf erweitern, ohne dass größere Renovierungsarbeiten nötig wären. Die Geräte können dort platziert werden, wo sie benötigt werden, und nicht nur dort, wo zufällig die Leitungen liegen.

EnOcean-Funksender haben zusätzlich den Vorteil, zum Betrieb keinerlei externe Stromversorgung und keine Batterien zu benötigen, da die zum Senden benötigte Energie aus der Umwelt gewonnen wird - z.B. mit Solarzellen aus dem vorhandenen Licht oder durch die Nutzung verfügbarer kinetischer oder thermischer Energie.

Mittlerweile haben etliche Hersteller von Elektroinstallationskomponenten die EnOcean-Funktechnik direkt in ihre Geräte integriert, so dass der Anwender auf eine breite Basis an Produkten zurückgreifen kann.

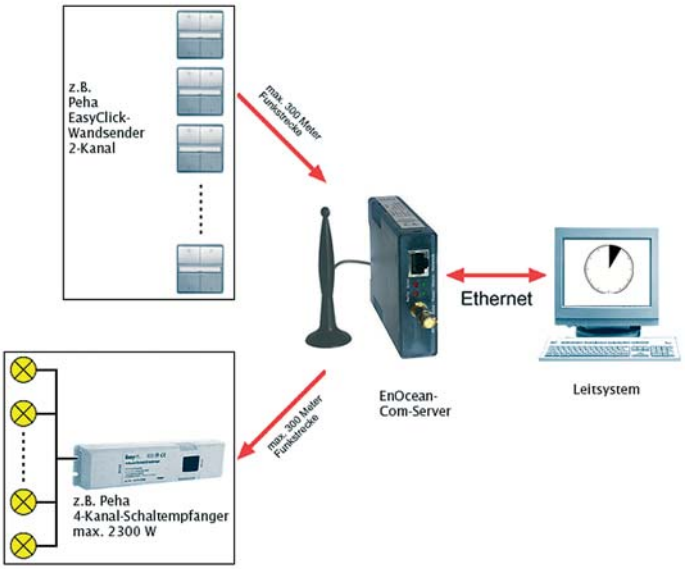
Der W&T EnOcean Com-Server kommt überall dort ins Spiel, wo über die bloße Steuerung eines Funkempfängers durch einen zugeordneten Funksender hinaus übergeordnete Steuerungs- und Überwachungsfunktionen gefordert sind.

Ein Beispiel für eine solche Aufgabenstellung finden Sie auf den folgenden Seiten.

Beispiel aus der Praxis

Einfache Steuerungsaufgaben lassen sich alleine mit EnOcean-Funksensoren und -Funkaktoren mit extrem geringem Installationsaufwand realisieren. Das System stößt jedoch bei Anwendungen an seine Grenzen, bei denen sowohl lokale als auch zentrale Steuerungs- und Überwachungsfunktionen erforderlich sind.

Die dargestellte Applikation zeigt am Beispiel einer Treppenhauslicht-Steuerung, bei der die Beleuchtung per Tastendruck eingeschaltet und zeitgesteuert wieder ausgeschaltet wird, das Zusammenspiel der Funkkomponenten mit dem EnOcean Com-Server und einem Leitsystem:



Obwohl es sich bei der dargestellten Anwendung um ein Trivial-Beispiel handelt, so ist doch erkennbar, welche Anwendungsarten prinzipiell den Zugang zu einem Leitsystem über einen EnOcean Com-Server erforderlich machen: Applikationen, bei denen lokale Ereignisse von einer zentralen Instanz

- in der Regel automatisiert - überschrieben oder überwacht werden müssen.

So kann z.B. in einem Bürogebäude die Steuerung der Beleuchtung oder die Einstellung der Rolläden von jedem Benutzer individuell nach seinen Bedürfnissen vorgenommen werden. Trotzdem kann eine zentrale Instanz dafür sorgen, dass zu einer bestimmten Zeit alle Rolläden geschlossen werden und das Licht im gesamten Gebäude gelöscht wird.

Die Software

Die Einbindung des W&T EnOcean Com-Servers in steuernde und/oder beobachtende Software-Umgebungen kann auf unterschiedlichen Wegen geschehen, die auf den folgenden Seiten kurz beschrieben werden:

Der etablierte Standard: OPC

So gut wie alle am Markt verfügbaren Leit- und Visualisierungssysteme unterstützen bereits in der Grundausstattung oder über zukaufbare Module die Standard-Softwareschnittstelle OPC. Zu den Vertretern dieser Gattung zählen z.B. die Wonderware-Produkte, NI's "LabView", das vom Ing.-Büro Bauer entwickelte "ShowIt" und viele andere mächtige Softwarepakete, die über den frei verfügbaren W&T OPC-Server eine einfache Einbindung des EnOcean Com-Servers in die jeweilige Aufgabenstellung ermöglichen.

Der Vorteil einer solchen Lösung ist schnell erklärt: Sie können sich fast ausschließlich auf die eigentliche Problemstellung konzentrieren, ohne sich um das kümmern zu müssen, was unter der Haube passiert: Mehr als die Zuordnung der Sensoren und Aktoren zu den OPC-Variablen muss der Intergrator an "Verwaltungsarbeit" nicht leisten. Der Rest der Aufgabe ist die Arbeit an der eigentlichen Funktionalität des Leit- bzw. Visualisierungssystems.

So viel Komfort hat natürlich seinen Preis: Einerseits sind die Entwicklungswerkzeuge eher hochpreisige Lösungen, bei denen für die einzelnen Projekte in der Regel noch Kosten für zusätzliche Laufzeit-Lizenzen anfallen. Andererseits ist die Einarbeitung in ein professionelles Visualisierungssystem alles andere als "mal eben" nebenbei erledigt.

Fazit: Auf OPC basierende Lösungen sind der Königsweg, wenn das Softwarepaket, das zum Einsatz kommen soll, bereits vorhanden ist und Sie im Umgang damit vertraut sind oder wenn Sie den häufigen Einsatz eines solchen Pakets für die Zukunft in Erwägung ziehen.

In diesem Fall kommen Sie auf der OPC-Schiene innerhalb kürzester Zeit zu einsatzfähigen Lösungen - wobei diese prinzipbedingt weitgehend auf die Windows-Welt beschränkt sind.

Do it yourself: Zugriff über Socket-Programmierung

Falls die OPC-Lösung nicht zum Einsatz kommen soll, ist es jedoch kein großes Problem, die Auswertung des EnOcean-Protokolls in einer beliebigen Programmiersprache selbst in die Hand zu nehmen.

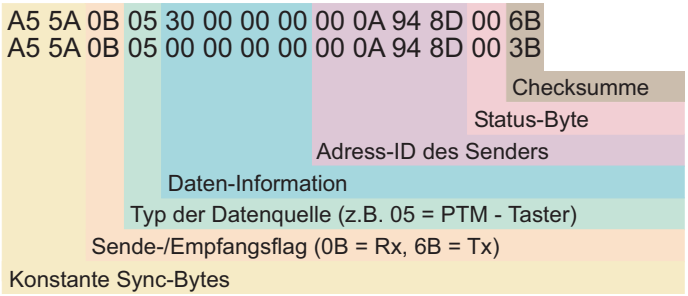
Das Protokoll wird durch den EnOcean Com-Server transparent vom Netzwerk an das integrierte Funkmodul durchgereicht, so dass Sie die Funktelegramme über die Netzwerkverbindung exakt so erhalten, wie sie von den Sensoren gesendet werden. In der gleichen Weise müssen die Telegramme so an den Com-Server übergeben werden, wie Sie sie auch direkt an das Transceivermodul senden würden.

Das EnOcean-Protokoll ist vergleichsweise einfach aufgebaut und im Handbuch zum Transceiver-Modul dokumentiert, das Sie unter www.WuT.de auf der Tools-Seite zum EnOcean Com-Server finden.

Ein Daten-Telegramm besitzt eine Länge von 14 Byte und beginnt mit zwei konstanten Sync-Bytes (0xA5, 0x5a). Die folgenden Bytes beinhalten Daten und Adress-Informationen, sowie einige wenige Statusbytes, an denen sich der Typ des Telegramms und der Datenquelle ablesen lässt. Den Abschluss bildet ein Checksummen-Byte, das aus der Summe aller Nutz-Bytes (also ohne Sync-Bytes) Modulo 256 gebildet wird.

Das folgende Diagramm zeigt ein Beispiel für empfangene Telegramme bei Betätigung und Loslassen eines Funk-Tasters:

EnOcean-Protokoll



Sende-Telegramme sind ähnlich den gezeigten Empfangs-Telegrammen aus dem obigen Beispiel aufgebaut. Es muss lediglich das Sende-/Empfangsflag auf 0x6B gesetzt und darauf geachtet werden, dass die in die Struktur geschriebene Adress-ID aus dem Adressraum des integrierten Transceiver-moduls stammt.

Noch eine Alternative: Die Windows COM-Umlenkung

All das, was zum Zugriff via Socket-Programmierung gesagt wurde, besitzt ebenfalls Gültigkeit, wenn Sie den EnOcean Com-Server über die W&T COM-Umlenkung ansprechen. Der EnOcean Com-Server präsentiert sich in diesem Fall als zusätzliche virtuelle COM-Schnittstelle im System, über die Sie das EnOcean-Protokoll in der oben beschriebenen Weise abwickeln können.

In Verbindung mit dem TCM-Monitor haben Sie quasi als Zugabe noch ein Diagnose-Hilfsmittel zur Hand, das es erlaubt, empfangene EnOcean Funktelegramme inhaltlich zu interpretieren, Telegramme mit einstellbarem Inhalt zu versenden und das Transceivermodul zu konfigurieren. Der TCM-Monitor ist ein kleines, aber effektives Software-Tool von EnOcean, das auf der seriellen Schnittstelle aufsetzt und problemlos mit der Kombination "EnOcean Com-Server + COM-Umlenkung" zusammenarbeitet.

Ein Blick in die Zukunft:
Das EnOcean Web-IO als Stand-alone - Lösung

Zur Zeit erfordert selbst eine minimale Installation bestehend aus einem EnOcean Com-Server und einigen Sensoren / Aktoren das Vorhandensein eines ständig betriebsbereiten PCs mit einer installierten Leit- bzw. Visualisierungs-Software: Der EnOcean Com-Server fungiert ausschließlich als transparentes Interface zwischen dem TCP/IP-Netzwerk und der bidirektionalen Funkschnittstelle.

Dieser Umstand ist speziell in überschaubaren Installationen eher lästig, erfordert die zusätzliche Ressource PC + Software und hat den Nachteil, permanent nicht unerheblich zur Stromrechnung beizutragen.

Dies wird sich mit dem EnOcean Web-IO ändern:

- Völlig autarke Auswertung empfangener Funktelegramme
- Steuerung von EnOcean-Aktoren in Abhängigkeit von empfangenen Ereignissen
- Konfiguration über eine Web-Oberfläche aus einem beliebigen Browser heraus

Speziell für kleine EnOcean-Anwendungen gehört mit dem EnOcean Web-IO der in Dauerbertrieb laufende, stromhungrige und mit spezieller Software ausgerüstete PC zukünftig der Vergangenheit an.

Neue Kapitel Online abrufen

Das Buch TCP/IP-Ethernet bis Web-IO wird kontinuierlich erweitert. Neue Kapitel, die bei Druck dieses Buches noch nicht vorlagen, können Sie auf unserer Webseite

http://www.WuT.de

als PDF-Datei herunterladen.

Wenn sie möchten, dass wir Sie im Bereich Web-IO auf dem Laufenden halten, werden Sie einfach Mitglied im Web-IO und Com-Server Club!

Das geht ganz einfach:

Auf unserer Webseite

http://www.wut.de

im Bereich *Service* auf den Link *Info-Clubs* klicken und das Formular ausfüllen.

Sie bekommen dann von uns alle 4 - 6 Wochen eine kurze E-Mail mit den aktuellsten Informationen.

Natürlich halten wir auch auf unserer Webseite tagesaktuell Informationen zu Web-IO, Netzwerktechnik und seriellen Schnittstellen für Sie bereit.

Index

Symbole

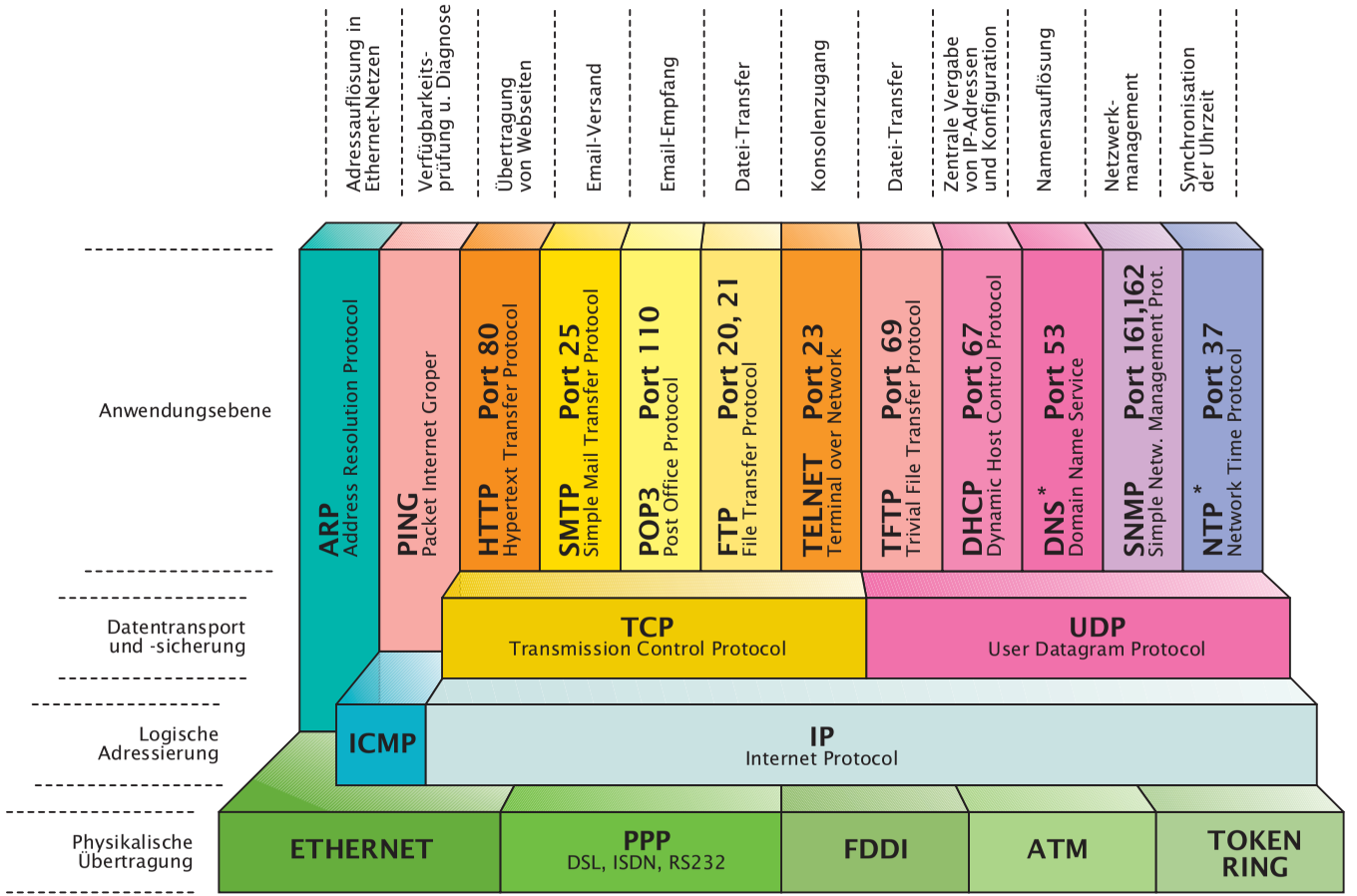
A

B

C

T

Die wichtigsten Protokolle für Intra- und Internet auf einen Blick



* DNS und NTP werden in Sonderfällen auch über TCP abgewickelt

Technische Grundlagen
Ethernet, TCP/IP bis Web-IO verstehen

Einen Toaster steckt man in die Steckdose, ohne zu wissen woher der Strom kommt. Einen Videorecorder zu programmieren, ohne etwas über Sendekanäle oder Showview zu wissen, ist bekanntlich tückisch.

Wer Geräte an Computernetze anschließen will, wird es ohne Grundkenntnisse der Netzwerktechnik nicht weit bringen; insbesondere dann, wenn es sich um größere professionelle Netze handelt. Aber auch bei der Kinderzimmervernetzung kann die Kenntnis der Grundbegriffe einigen Ärger sparen.

Dicke Bücher über Netzwerktechnik finden Sie genug. In diesem Büchlein sind die wesentlichen Grundlagen übersichtlich zusammengefasst. Alles was nur den Entwickler von Netzwerkprodukten interessieren muss, haben wir konsequent weggelassen. Dafür sind die Informationen, die auch den Anwender interessieren könnten, durchaus mit technischem Tiefgang behandelt.

W&T
www.wut.de

Wiesemann & Theis GmbH
Porschestraße 12
D-42279 Wuppertal

Mail info@wut.de
Web www.wut.de

Tel. 0202 26 80-110
Fax 0202 26 80-265

TCP/IP-Ethernet bis Web-IO
DE 1.5 03/2006