

# TCP/IP in Embedded-Systemen

## Anforderungen und Lösungen für den Einsatz von TCP/IP in Embedded-Systemen

Nachdem die Möglichkeiten des Internet für Fernwartung und Fernsteuerung schon seit längerem bekannt sind und immer mehr Hersteller von Embedded-Systemen derartige Lösungen anbieten, halten nun auch Ethernet und die Internet Protokolle Einzug in die Fabrikautomation. Mit unterschiedlichen Konzepten wollen die Open DeviceNet Vendor Association (ODVA), Interface for Distributed Automation (IDA) und die Profibus Nutzerorganisation (PNO) Prozeßkommunikation über Ethernet betreiben. Alle Ansätze haben bisher eine Gemeinsamkeit: sie basieren auf Ethernet und der Internet Protokollfamilie TCP/IP.

Während es für PC-basierende Systeme naheliegender ist, die in den PC-Betriebssystemen enthaltene TCP/IP-Protokollsoftware und -schnittstellen zu nutzen, gibt es für Embedded-Systeme verschiedene Alternativen. Grundsätzlich ist es möglich ein Echtzeitbetriebssystem mit integriertem TCP/IP-Stack einzusetzen oder – und diese Lösung ist vor allem dann interessant, wenn die Kommunikationsfähigkeit nachträglich in eine bereits vorhandene Software integriert werden soll – einen eigenständigen TCP/IP-Stack zu integrieren. Diese Lösung muß auch in Erwägung gezogen werden, wenn es wegen begrenzter Ressourcen nicht möglich oder aber auch überhaupt nicht erforderlich ist ein Echtzeitbetriebssystem einzusetzen. Hinzu kommt, daß die in Betriebssystemen integrierten TCP/IP-Protokollstacks für gewöhnlich aus dem sogenannten Berkeley-Code für UNIX abgeleitet worden sind und somit den Anforderungen, die an Embedded-Systeme hinsichtlich Ressourcenbedarf und Performance gestellt werden nur bedingt entsprechen. Soll also nicht unnötig viel Speicher und CPU-Rechenzeit des Embedded-Systems durch den TCP/IP-Stack verbraucht werden, so ergeben sich einige spezielle Anforderungen an einen für Embedded-Systeme geeigneten TCP/IP-Protokollstack.

Eine wesentliche Aufgabe eines Protokollstacks besteht in der Entgegennahme, Verarbeitung und Weiterleitung von Daten in zwei Richtungen: vom Anwendungsprogramm zum Kommunikationsmedium (Ethernet oder serielle Schnittstelle) bzw. vom Kommunikationsmedium zum Anwendungsprogramm. Hierbei durchlaufen die Daten die verschiedenen Protokollebenen innerhalb des Stacks indem sie normalerweise in Buffer abgelegt werden, wobei jede Protokollebene seine eigenen Buffer verwaltet und die Daten damit von Protokollebene zu Protokollebene kopiert werden, was einerseits einen erheblichen Aufwand an Speicherverwaltung erfordert,



**Bild 1:** Ein Fahrzeugdiagnosegerät mit 16-Bit CPU und integriertem Web-Server. Dieser ermöglicht die Darstellung und Änderung der Daten über einen Web-Browser von einem Remote-PC aus. Die Kommunikation erfolgt über ein GSM-Modem.

andererseits aber auch viel Rechenzeit in Anspruch nimmt. Eine wichtige Anforderung an einen Embedded-TCP/IP-Stack ist somit, daß es ein sogenannter „Zero-Copy-Stack“ ist. Das bedeutet, daß neue Daten nur einmal in einen Buffer kopiert werden und die einzelnen Protokollebenen direkt auf diesen Buffer arbeiten. Ein weitere wichtige Eigenschaft ist, daß Buffer nicht während der Laufzeit dynamisch allokiert werden, sondern mit sogenannten pre-allocated Buffern gearbeitet wird, die vom Stack selber verwaltet werden.

In Embedded-Systemen kommen die unterschiedlichsten CPUs, Softwarekonzepte und Betriebssysteme zum Einsatz. Daher sollte ein Embedded-TCP/IP-Stack in der Lage sein, mit allen Arten von Betriebssystemen zusammenzuarbeiten. In einem präemptiven Multitasking-Betriebssystem ist dabei ein Zugriffsschutz für bestimmte Ressourcen zu realisieren, während in einer Superloop (Endlosschleife) die Funktionsaufrufe nicht blockieren dürfen, um andere dringende Aufgaben nicht zu verzögern. In den Standard-TCP/IP-Stacks, wie sie z.B. unter Linux oder großen Echtzeit-Betriebssystemen eingesetzt werden, sind die Funktionsaufrufe der Socket-API üblicherweise blockierend, d.h. die Funktion `recv()` wartet solange, bis Daten eingetroffen sind ehe sie zum aufrufenden Programmteil zurückkehrt. Ein solches Verhalten würde die Bearbeitung der Anwendung enorm verzögern und damit eine echtzeitfähige Funktion des Gesamtsystems bei einem Einsatz des Stacks ohne echtes präemptives Multitasking-Betriebssystem verhindern.

Um eine einfache Integration in ein bestehendes System zu ermöglichen, sollte der TCP/IP-Stack nur einen Timer mit einer sehr flexibel einstellbaren Timerauflösung benötigen. Die Überwachung aller Timeouts der verschiedenen Protokollebenen sollte dabei innerhalb eines eigenen Programmoduls ge-

schehen. Damit gelingt eine effektive Integration des Stacks in ein Betriebssystem, da dieses Modul dann als zyklische Task realisiert werden kann.

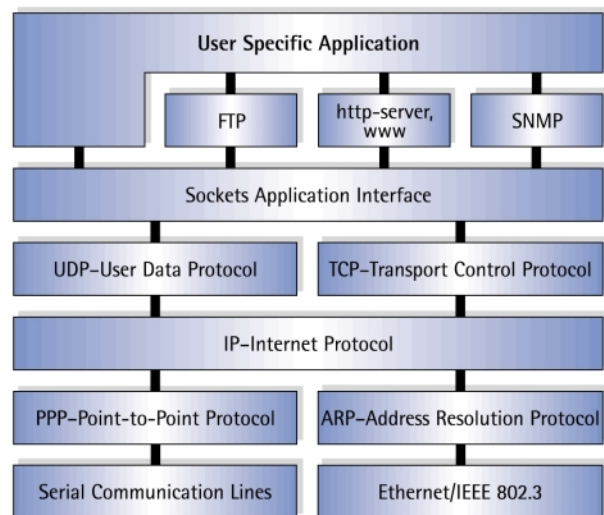
Unterschiedliche Anwendungen in unterschiedlichen Umgebungen erfordern, daß der TCP/IP-Stack mit unterschiedlichen Arten von physikalischen Schnittstellen zusammenarbeitet. Neben der seriellen Schnittstelle, die sich besonders für die Fernwartung mittels Modem und den Protokollen PPP oder SLIP anbietet, sollte auf jeden Fall noch Ethernet für den Einsatz im Intranet vorhanden sein. Werden mehr als 2 Schnittstellen gleichzeitig unterstützt, muß der TCP/IP-Stack über eine Intelligenz verfügen, die entscheidet, welche Schnittstelle für abgehende Daten benutzt wird.

Nicht alle Anwendungen erfordern die volle Funktionalität einer kompletten TCP/IP Implementierung mit all ihren ergänzenden Protokollen und Funktionen. Deshalb sollte der Stack so modular gehalten sein, daß die nicht benötigte Funktionalität beim Compilieren des Stacks einfach ausgeblendet werden kann und somit keine unnötigen Ressourcen belegt werden. Der Stack sollte von Haus aus jedoch über den vollen Umfang verfügen, denn während der Integration in die eigene Anwendung kann diese Funktionalität wertvolle Hilfe leisten. Hier wäre z.B. das Loopback-Interface zu nennen, welches bereits das Testen der Anwendung erlaubt auch wenn noch keine funktionsfähige physikalische Schnittstelle verfügbar ist.

Um die Portierung eines TCP/IP-Stacks auf ein Embedded-System in kurzer Zeit und mit wenig Aufwand zu ermöglichen, sollte der Stack über verschiedene weitere Testhilfsmittel verfügen. Bereits angesprochen sind Loopback-Interfaces, die ein Testen ohne physikalische Schnittstellen erlauben. Somit kann die erste Portierung und Anpassung des TCP/IP-Stacks bereits auf einem Evaluation-Board durchgeführt werden. Eine weitere sehr hilfreiche Debugschnittstelle ist ein in den Stack integrierter Monitor, der über eine serielle Schnittstelle mit einem einfachen Terminalprogramm verbunden wird und über den statistische Daten (z.B. aufgetretene Protokollfehler) der einzelnen Protokollebenen eingesehen bzw. Funktionen in den Protokollebenen ausgelöst werden können. Eine Analyse dieser Werte kann wertvolle Hilfe bei der Fehlersuche und Eingrenzung leisten.

Eine optimale Lösung für kleine (8/16-Bit) als auch für große (32-Bit) Embedded-Systeme stellt die TCP/IP-Protokollfamilie der Firma INTERNICHE dar. Die in Kalifornien beheimatete Firma entwickelt seit über 12 Jahren für den embedded Bereich optimierte TCP/IP Protokollstacks und gilt als einer der kompetentesten Anbieter auf diesem Gebiet. Die INTERNICHE-Protokollfamilie enthält die komplette Funktionalität, die in den RFCs spezifiziert ist, bietet aber auch die Möglichkeit, nicht benötigte Eigenschaften abzuschalten, wie z.B. das im IP-Protokoll

enthaltene, sehr aufwendige Fragmentieren von Telegrammen. Damit läßt sich das System optimal an die Bedürfnisse einer Anwendung anpassen und damit den Codebedarf erheblich reduzieren. Da die Größe des erforderlichen Datenspeichers unmittelbar von der Anzahl der gleichzeitigen Verbindungen abhängt, ist die Anzahl der Verbindungen beim INTERNICHE Stack ebenfalls konfigurierbar. Bild 2 zeigt die Strukturierung der INTERNICHE-Protokollfamilie.



**Bild 2:** Struktur eines Modulare TCP/IP-Stacks für Embedded-Systeme

Die Software ist sehr flexibel mit oder ohne Betriebssystem einsetzbar. Beim Einsatz ohne Betriebssystem müssen alle Funktionen der Protokollsoftware als nicht blockierend realisiert werden, da sie nicht auf Ereignisse warten dürfen. Ergänzt wird die Protokollfamilie durch eine Vielfalt von spezifischen Anwendungen wie Web-Server, FTP-Server, E-Mail Alerter usw. Hinzugekommen ist seit Herbst 2000 auch die Unterstützung von IP Multicasting. Damit ist eine Gruppenadressierung möglich, die besonders von der Automatisierungstechnik gefordert wird.

Das neueste Mitglied dieser Protokollfamilie ist NicheLite, welches einen vollständigen prozessorunabhängigen TCP/IP-Protokollstack mit nur 12k Bytes realisiert und damit vor allem für kleine Systeme und Mikrocontroller mit sehr wenig zur Verfügung stehenden Ressourcen geeignet ist. NicheLite ist vollständig in C geschrieben und arbeitet mit allen oben genannten Anwendungen zusammen.

Ein Beispiel für die einfache Integration des TCP/IP-Stacks in bestehende Anwendungen zeigt Bild 1. Hier wurde für ein bestehendes Fahrzeugdiagnosegerät, welches von IXXAT Automation GmbH für die ZF Friedrichshafen entwickelt wurde, nachträglich die Möglichkeit zur Ferndiagnose geschaffen. Hierzu kann ein Entwickler über ein GSM-Modem das Dia-

gnosegerät im fahrenden Fahrzeug anwählen und die Daten, die der Fahrer am LCD-Display des Diagnosegerätes sieht, auf einem PC anzeigen und ändern. Eine Forderung hierbei war, daß keine besonderen Voraussetzungen zur Anzeige der Daten auf dem PC des Entwicklers notwendig sein sollten (wie z.B. die Installation eines speziellen Softwarepakets). Daher wurde in das Diagnosegerät zusätzlich zur bereits vorhandenen Anwendung ein Web-Server mit TCP-Anbindung über das PPP-Protokoll implementiert, welcher die auf dem Display angezeigten Daten über ein Java-Applet auf einem normalen Webbrowser anzeigt, die Eingaben des Entwicklers entgegennimmt und in das Diagnosegerät einspeist. Die Entwicklung konnte mit Hilfe des INTERNICHE-TCP/IP-Stacks in relativ kurzer Zeit realisiert werden, wobei die Belastung der CPU durch den Web-Server und den TCP/IP-Stack nur geringfügig gestiegen ist.

Die Firma IXXAT Automation GmbH, ein bekanntes Systemhaus für Kommunikation in Industrie und Automobil, ist Hauptdistributor der INTERNICHE-Produkte im deutschsprachigen Raum. IXXAT beschränkt sich hierbei nicht nur auf die Distribution der Software, sondern bietet umfangreiche Beratung und Unterstützung bei der Implementierung von TCP/IP in vorhandene oder neu zu entwickelnde Produkte an. Ferner wird auch vollständige Hardware- und Softwareentwicklung für die Kunden durchgeführt oder auch die Realisierung ganzer Kommunikationssysteme übernommen. Darüber hinaus bietet IXXAT Automation Seminare zu diesem Thema an.

**Autoren:**

Uwe Weissenrieder, IXXAT Automation GmbH (ist Softwareentwickler und beschäftigt sich mit der Integration von TCP/IP in Embedded-Systemen)  
Christian Schlegel, IXXAT Automation GmbH (ist Technischer-Geschäftsführer)

**IXXAT Automation GmbH**

Leibnizstrasse 15  
88250 Weingarten  
Tel. 0751 / 56146-0  
Fax 0751 / 56146-29  
<http://www.ixxat.de>  
email: [info@ixxat.de](mailto:info@ixxat.de)