Alles zu Protokollen und Schichten

TCP/IP-Schichten	OSI-Schichten	Bezeichnung	Funktionen	Dienste
4	6 + 7	Anwendungsschicht	NetBIOS, WinSock	HTTP, FTP, Telnet...
3	4	Transportschicht	TCP, UDP	diverse Kommunikations-Dienste
2	3	Internetschicht	IP, ARP	Ping
1	1 + 2	Netzwerkschicht	LAN, MAN, WAN	reine Bitübertragung

Und nun wollen wir die einzelnen Spalten der Tabelle und ihre Inhalte einmal näher beleuchten (aber nur die TCP/IP-Schichten):

In der **vierten Schicht** vereint das TCP/IP die Funktionen der OSI-Schichten 6 und 7. NetBIOS und WinSock stellen in diesem Falle zwei Programmschnittstellen zu den angegebenen Diensten wie z.B. FTP dar. Um diese Dienste nutzen zu können, sind diese Schnittstellen lebensnotwendig, besonders die Funktion WinSock. Die andere Funktion NetBIOS hat eher eine andere Aufgabe, sie übernimmt die Verbindung zur darunter liegenden nächsten Schicht Transportschicht

Die **dritte Schicht**. Hier übernimmt das Transport-Protokoll **TCP** (**T**ransmission **C**ontrol **P**rotokoll) die Aufgabe für den reinen Datentransport im Netzwerk. Das UDP-Protokoll, welches in der Tabelle erwähnt wird, ist ebenfalls ein reines Transport-Protokoll. Der Unterschied zwischen beiden liegt allein in der Kontrollfunktion. Während das **TCP** während der Datenübertragung zusätzlich **kontrolliert**, ob die Daten auch richtig am Ziel angekommen sind, verzichtet das UDP gänzlich auf diese Kontrollfunktion. Deshalb wird auch bei der Datenübertragung meistens das TCP benutzt, um Fehlern in der Übertragung vorzubeugen und diese auch dann erkennen zu können. UDP wird eher dort eingesetzt, wo größere Datenmenge übertragen werden müssen (Audiodaten oder ganze Videos im Internet)

In der **zweiten TCP/IP-Schicht** sehen wir auf einen Blick das **IP-Protokoll (I**nternet **P**rotokoll). Im Gegensatz zum TCP-Protokoll hat es die wichtige Aufgabe, allein erst einmal die Verbindung zwischen zwei Rechnern im Netzwerk aufzubauen. Das TCP-Protokoll ist also auf das IP-Protokoll angewiesen, ohne dem läuft da gar nichts. Um die Rechner exakt im Netzwerk zu 'adressieren', kommt dem IP das oben schon beschriebene **ARP-Protokoll** (**A**dress-**R**esulotion-**P**rotokoll)zur Hilfe, welches sich um die Umformung der IP-Adresse in die **MAC-Adresse** (siehe NIC - doc1) kümmert, um den Rechner im Netzwerk genau zu adressieren.
Mit dem **Ping**-Dienst / -Befehl kann man zur Prüfung im Netzwerk feststellen, ob ein Rechner erreichbar ist. Die Daten werden meist als einzelne Datenpakete nacheinander einzeln übertragen. Die Datenpakete werden mit einem IP-Header versehen.

Die **erste TCP/IP-Schicht** im LAN / MAN / WAN hat die Aufgabe, die reinen Bits zu übertragen. Alle anderen Schichten bauen somit auf dieser Schicht auf und stehen direkt mit ihr in Kontakt.

Nun sehen wir uns noch die anderen Protokolle an:

HTTP-Protokoll

Das **H**yper**T**ext **T**ransfer **P**rotokoll dürfte schon all denen über den Weg gelaufen sein, die das Internet öfters nutzen, das Forum (http://www.schulmodell.de/BS/forum/index.php) besuchen, Scripte und Seiten downloaden.

Es dient dabei zum Austausch von Hypertext-Dokumenten (z.B. den HTML-Seiten – **H**yper**t**ext **M**arkup **L**anguage) HTTP setzt auf dem Transport-Protokoll TCP auf und hat die spezielle Aufgabe, die Kommunikation zwischen dem Web-Client (z.B. deinem Web-Browser) und dem Webserver, auf dem sich die HTML-Seiten befinden, zu regeln. HTTP legt dabei das Format, den Inhalt und die Abfolge der Mitteilungen, die zwischen Client und Server ausgetauscht werden, fest. Das Protokoll ist deshalb plattformunabhängig.

Die Kommunikation zwischen Webclient und Webserver läuft immer nach dem gleichen Schema ab : Der Webclient stellt einen HTTP-Request (=Anfrage) an der Webserver. Dieser erhält diesen Request, setzt ihn entsprechend und gibt den Response (=Antwort) an den Client zurück. In der Praxis sieht das so aus :

Du tippst oben in Browser eine Internet-Adresse ein (z.B. http://www.seifert-web.de/bs.htm) und bestätigst mit <return>. Die von dir angegebene URL (die Internet-Adresse) wird von einem DNS-Server (dazu spätere Ausführungen) in die IP-Adresse umgewandelt und der Webserver mit dieser bekannten IP-Adresse erhält den Request. Er setzt ihn sofort um und schicken Daten zurück an den Client, die er ja auch davor angefordert hat. Dies kann zum Beispiel eine Webseite sein, die man sich dann auf dem Browser anschauen kann. Ein ganz simpler Ablauf !

Zum Senden einer Request-Mitteilung wird zwischen Webclient und Webserver eine TCP/IP-Verbindung aufgebaut. Die Verbindung wird aufrecht erhalten und der Server-Response wird über genau die gleiche Verbindung zurückgeschickt. Die Verbindung wird nach dem Datentransfer wieder geschlossen.

Aufbau des HTTP-Request (Vielleicht brauchen wir das auch nicht?)

Der HTTP-Request, dies kann eine normale Internet-Adresse sein, besteht grundsätzlich aus einer URL (**U**niform **R**esource **L**ocator), dazu kommt der sogenannte Method-Token, die wichtigen Header-Informationen sowie dem Entity-Body, der die zu transportierenden Daten enthält

Der Aufbau des HTTP-Requests ist in drei voneinander getrennte Segmente unterteilt : Die Status-Line enthält den Method-Token, der Header enthält die General-, Request- und Entity-Header und der Entity-Body enthält, wie schon erwähnt, die zu übertragenden Daten.

Der Method-Token hat dabei die Aufgabe, die Übertragungsmethode festzulegen. Am häufigsten gebraucht werden die Methoden **GET** und **POST**. Durch die im Request evtl. enthaltene Methode GET fordert der Webclient ein in der URL festgelegtes Dokument vom Webserver zur Übertragung an. Die Methode POST wird eher in der Webprogrammierung und dabei speziell bei CGI-Skripten (Common Gateway Interface) verwendet. Hierbei werden beim HTTP-Request Informationen im Entity-Body an den Webserver übermittelt, die dann auf dem Server verarbeitet werden können

Die Header-Informationen des HTTP-Requests befinden sich unterteilt, im oben schon erwähnten General-Header (allgemeine Informationen, wie z.B. Caching-Informationen oder das Datum von Request von Response) und im Request- bzw. Entity-Header (sonstige Informationen, wie z.B. Dateiformate, Zeichensätze, Sprachen oder auch spezielle Informationen über die Daten im Entity-Body).

Da im HTTP-Request in der Regel keine Daten zum Webserver übertragen werden, sondern ja einfach nur die Anfrage gestellt wird, um Dokumente abzuholen, ist der Entity-Body meist leer.

Aufbau der HTTP-Response

Der HTTP-Response ist ähnlich wie der HTTP-Request aufgebaut.

Er beinhaltet zusätzlich zu allen anderen Informationen des Requests die Status-Line und eben den Entity-Body, der die zu übertragenden Daten für den Webclient enthält.

Die Status-Line des HTTP-Responses beinhaltet dann den sogenannten Status-Code und die Reason-Phrase, die eine kurze Erläuterung zum Status-Code enthält. Der Code "200" hat dabei zum Beispiel die Bedeutung, dass alles ok ist und die Daten übertragen werden können. Der Fehlercode "500" gibt zum Beispiel andererseits an, daß beim Webserver ein interner Fehler vorliegt.

FTP-Protokoll

Das **F**ile**T**ransfer-**P**rotokoll dürfte auch bekannt sein, es ermöglicht einen vom Betriebssystem unabhängigen Austausch von beliebigen Daten zwischen zwei vernetzen Rechnern.

Das Protokoll FTP setzt ebenfalls, wie auch HTTP, auf dem Protokoll TCP auf. Es nutzt dabei bei beiden kommunizierenden Rechern den festgelegten Port 20 für die FTP-Datenverbindung und den Port 21 für die Steuerung der Verbindung.

Der FTP-Client kann bei der Datenübertragung zwischen mehreren Formaten wählen. Die am häufigsten Genutzten sind ASCII und die binäre Übertragung.

Die wichtigsten Protokolle für E-Mail!

SMTP-Protokoll

Das **S**imple **M**ail **T**ransfer **P**rotokoll hat im Internet die Aufgabe, die elektronische Post (Email) zwischen entfernten Rechnern auszutauschen und zu übertragen. Es setzt wie viele verwandte Protokolle auf TCP auf und benutzt dabei auch einen festgelegten Port, und zwar den Port 25.

Der SMTP-Client baut die Verbindung zum Server über den Port 25 auf mittels eines Requests, welcher vom Server mit dem Response sogleich beantwortet wird. Der SMTP-Server teilt dem Client damit mit, dass er sich dem Client als Server identifiziert hat und ob er denn auch bereit ist, Emails zu empfangen und zu verschicken. Kommt kein Response vom Server, kappt der SMTP-Client die Verbindung wieder. Warum auch warten, es kostet ja auch Geld.

Wenn der Server andererseits bereit ist, gibt der Client dem Server weiter, einerseits von wem die Post stammt und andererseits an wen sie verschickt werden soll. Falls die Empfänger-Mailadresse stimmt, gibt der Server dem Client das Signal, dass gesendet werden kann. Der Client kann jetzt die Mail senden und bekommt vom Server die Bestätigung, dass die Mail soeben verschickt wurde. Am Schluss wird die SMTP-Verbindung wieder aufgelöst.

POP-Protokoll

Das **P**ost **O**ffice **P**rotokoll stellt genau das Gegenteil zum SMTP-Protokoll dar. Es hat die Aufgabe, als Standard-Protokoll dem Client die Möglichkeit zu geben, seine Emails vom POP-Server abzuholen und zu lesen. Die aktuellste Version ist zur Zeit POP3.

Die Verbindung wird, wie beim SMTP, zum Server aufgebaut. Damit wird auch schon die Übertragung der elektronischen Post vom Server zum Client automatisch ausgelöst. Man hat aber als Client die Möglichkeit zu entscheiden, ob die Emails weiter auf dem Server bestehen bleiben sollen oder nach der erfolgreichen Übertragung gelöscht werden sollen. Die Nachrichten können auch schon vor der Übertragung, wenn sie zum Beispiel nicht wichtig sind, auf Wunsch gelöscht werden.

IMAP-Protokoll

Das **I**nternet **M**essage **A**ccess **P**rotokoll ist ein zum Protokoll POP erweitertes EMail-Protokoll. Der große Vorteil hierbei ist, dass man hiermit Nachrichten nach Bedarf übertragen kann. Es werden zunächst nur die Kopfzeilen der Emails übertragen, somit kann man entscheiden, welche Email wichtig ist oder welche man gar nicht lesen möchte. IMAP hat noch weitere Vorteile. Zum einen können hiermit auf dem IMAP-Server hierarchische Mailboxen aufgebaut werden, man kann mit einer Verbindung gleich auf mehrere Mailboxen zugreifen. Zum anderen hat man mit dem IMAP-Client die Möglichkeit, den Status der Mails zu verändern, sie z.B. von ungelesen auf gelesen zu ändern. Suchoptionen auf dem IMAP-Server erleichtern weiterhin die tägliche Arbeit mit dem Maileingang.

NNTP (Network News Transfer Protocol)

Jeder kennt ja wohl Usenet, oder hat schon ein Mal davon gehört. NNTP (Network News Transfer Protocol) ermöglicht den Zugang zu diesem Nachrichtendienst und benutzt üblicherweise den Port 119.
Es ist eigentlich so ziemlich gleich wie SMTP (Simple Mail Transfer Protocol) indem es TCP als Netzwerkprotokoll benutzt und einfache Befehle von einem Prompt akzeptiert.

Der Zweck ist in RFC 977 so beschrieben:
(Wer das drauf hat, punktet bei Anders!)
NNTP spezifiziert ein Protokoll für die Verteilung, die Recherche, die Wiedergewinnung und die Veröffentlichung von Nachrichtenartikeln durch eine zuverlässige Datenstrom-basierte Übertragung von Nachrichten innerhalb der ARPA-Internet-Gemeinde. NNETP ist so aufgebaut, dass Nachrichtenartikel in einer zentralen Datenbank gespeichert werden. Dem Abonnent wird ermöglicht, nur auf die Artikel zuzugreifen, die er lesen möchte. Indexierung, Querverweise und das Löschen von veralteten Artikeln sind ebenfalls vorgesehen.

Noch zwei Protokolle

NetBEUI (NetBUI)

NetBUI ist wohl das einfachste Protokolle von allen, welches sich aber als sehr nützlich erweist und vor allem leicht zu handeln ist.
Bereits bei Windows Version 95 war es standardmäßig integriert und sorgte für den zuverlässigen Datenfluss. Diese Tradition ging weiter über die Versionen 98 bis hin zu Wiindows NT.
Dieses Protokoll bietet sich vor allem bei kleinen Netzwerken an, so z.B. bei kleinen Peer-to-Peer-Netzen mit zwei oder mehreren Windows-Rechnern. (Sollte man bei Linux-Fetischisten nicht erwähnen!)

IPX / SPX

IPX / SPX, das eigens von der Firma Novell entwickelte Netzwerkprotokoll, sorgt ebenfalls für einen großen Datendurchsatz im Netzwerk. Auch dieses Protokoll steht bei Windows allgemein zur Verfügung, gerade im Multplayer-Bereich ist es noch sehr populär.