---------{ =>TCP/IP=> }---------

Überblick:

Kurze Beschreibung zum Thema

Dieser Text befasst sich mit dem Thema, wie eine Verbindung zwischen zwei Rechnern überhaupt erst entsteht, wie diese funktioniert, und welche Protokolle und Dienste dazu von Nöten sind. Erstmal sei gesagt, dass bei einer Verbindung über das Internet mit einem Protokollstapel gearbeitet wird. Der genaue Aufbau und die Unterteilung dieses Stapels in die verschiedenen Schichten wird weiter unten aufgezeigt. Wird eine Verbindung aufgebaut, so wird der Stapel Schicht um Schicht erstellt. Kommen die Daten beim Empfänger an, so werden die Schichten in umgekehrter Richtung wieder abgebaut, um die entsprechenden Prozesse in Gang zu leiten

Das ISO/OSI-Schichtenmodell im Überblick

Im folgenden Abschnitt werden jetzt die verschiedenen Schichten(Layers) des 7-Layer Protokolls, welche heute den ISO/OSI-Standart definiert, etwas genauer erklärt. Die Reihenfolge ist so dargestellt, wie sie beim aussenden von Daten aussieht. Beim Zielrechner wird der Stapel von unten nach oben gelesen. Ein Layer streift dann nach der Ausführung seiner Aufgabe den eigenen Header ab, um den Protokollen des nächsten Layers die Erledigung seiner Aufgaben zu ermöglichen

Anwendungsschicht:

Die Anwendungsschicht stellt die Schnittstelle zwischen den Anwendungen dar, mit welchen ein Benutzer Befehle über ein Netzwerk sendet bzw. empfängt

Darstellungsschicht:

In der Darstellungsschicht werden je nach System des anderen Rechners anwendungsspezifische Formatierungen durchgeführt

Sitzungsschicht:

Diese Schicht sorgt dafür, dass unterbrochene Verbindungen zwischen Anwendungen wieder hergestellt werden und z.T. auch an genau der selben Stelle fortgesetzt werden können, um Datenverlust zu verhindern

Transportschicht:

Diese Schicht sorgt für die zuverlässige Datenübertragung zwischen den zwei Rechnern und dient oft auch als Schnittstelle zwischen den übergeordneten Anwedungsschichten und den untergeordneten Netzwerkschichten

Netzwerk-/Verbindungsschicht:

In dieser Schicht wird der optimale Weg (routing) für die Übermittlung gesucht. Das Protokoll kann hier bereits unabhängig von den übergeordneten Schichten gewählt werden (z.B: IP)

Datenschicherungsschicht:

Diese Schicht hat dafür zu sorgen, dass keine Fehlübertragungen stattfinden und im Falle einer solchen, die Daten wieder hergestellt werden

Physikalische Schichte:

Diese Schicht ist für die Herstellung einer physikalischen Verbindung beim Empfangen bzw. Aussenden von Daten verantwortlich

Nachdem wir nun die Aufgaben der einzelnen Schichten kennengelernt

haben, werden wir auf die einzelnen Schichten genauer eingehen, um auch die verschiedenen Protokolle der jeweiligen Schichten kennen zu lernen. Hier auf alle Schichten Protokolle wirklich detailiert einzugehen würde den Rahmen aber deutlich sprengen. Trotzdem werden die Beschreibungen ausreichen, um ein recht gutes Wissen über diesen Protokollstapel zu erhalten

Physikalische Schicht

Diese Bitübertragungsschicht ist für die Herstellung einer physikalischen Verbindung zwischen zwei Kommunikationsendpunkten, wie etwa Programmen auf Rechner A und B verantwortlich. Dazu benötigt sie Kabel, Controller oder Modulierungsverfahren. Die physikalische Schicht wird in der Regel auch von der Treibersoftware für die netzwerkbezogene Hardware repräsentiert

Datensicherungsschicht

MAC (Media Access Control) übernimmt in Ethernet wie auch Token Ring die Zugriffssteuerung für das jeweils verwendete Medium. Es ist für die Adressierung der Netzwerkstationen zuständig und überwacht deren Zustand. Es gibt daruner drei verschiedene Typen von MAC-Adressen, auf welche wir aber nicht genauer eingehen werden

LLC (Logical Link Control) ist gegenüber MAC unabhängig vom jeweiligen Medium. Es ist also nicht ausschlaggebend, auf welchem Netzwerk es verwendet wird und bildet einen eigenen Standart. Gerade die Tatsache dieser Flexibilität macht es für TCP/IP interessant. Auch von LLC gibt es gesammthaft drei verschiedene Typen

SNAP (Sub Network Access Protocol) schafft als Erweiterung von LLC die Möglichkeit auch Protokolle, welche nicht den ISO-Standarts entsprechen in Netzwerken zu benutzen. Durch das von LLC verwendete SAP(Service Access Point) wird die Multiprotokollfähigkeit noch stärker erweitert

Netzwerkschicht

Hier kommen wir nun also zu einem der Kernpunkte des Themas. Ich werde hier nur auf IP genauer eingehen und ICMP, ARP sowie RARP nur eher am Rande beschreiben.

IP (Internet Protocol) bildet zusammen mit dem später erklärten TCP das zentrale Protokollpaar der TCP/IP Architektur. Die Transporteinheit im IP ist das Datagramm. Sein Aufbau wird in 32 Bit-Blöcken dargestellt, wobei der Header mindestens 20Bytes umfasst. Folgendes wird im Header angegeben:

Version IHL Type of Service Packet Lenght
Identification Flags Fragment Offset
Time to Live Protocol Header Checksum
Source IP-Adress
Destination IP-Adress
Options/Padding
...

Da es nicht allzu viel bringt, sich einen Header anzusehen, ohne zu wissen, wofür die einzelnen Begriffe stehen, folgt nun eine Erklärung der Bezeichnung eines jeden Feldes:

Version In diesem Feld wird die Version des verwendeten IP angegeben. Nur wenn der Empfänger zur Handhabung der verwendeten Version fähig ist, können die Packete weiter verarbeitet werden. Momentan wird im Internet noch die Version 4 verwendet. Es steht jedoch der Wechsel zur Version 6 bevor, welche in einigen LANs bereits verwendet wird.

IHL Hier wird die Länge des gesamten Headers inkl. Options in Anzahl 32-Bit-Okletts (= 4 Bytes) angegeben. Die Angabe '5' würde demnach bedeuten, dass nur der Fixheader existiert; er also wie vorher erwähnt nur 20 Bytes lang ist.

Type of Service In diesem Bereich werden verschiedene Optionen bezüglich Priorität und Qualität des Datentransportes festgelegt. Je nach dem kann die Priorität und somit auch die Geschwindigkeit oder die Sicherheit variieren.

Packet Length ; Total Lenght Hier wird die Länge des gesamten Datagramms angegeben. Da dieses Feld nur 16 Bit lang ist, kann der Wert maximal 65535 betragen, welcher, wenn man den IHL-Wert subtrahiert die Länge des Datenteils im Datagramm ergibt. Da viele Rechner jedoch auch den Maximalwert von 65535 nicht akzeptieren, müssen die Datagramme fragmentiert werden. In diesem Falle gilt die Angabe für das Einzel-Fragment.

Identification Werden Datagramme Fragmentiert, so wird hier der Integer-Wert eingetragen, welchen das fragmentierte Datagramm bei der Nummerierung erhielt.

Flags In diesem Feld wird angegeben, ob noch ein weiteres Fragment folgt, oder ob es sich beim empfangenen um das letzte handelt. Ausserdem lässt sich angeben, ob das Fragmentieren der Packete überhaupt gestattet ist.

Fragment Offset Dieses Feld ist nötig, um eine korrekte

Wiederherstellung der fragmentierten Datagramme zu gewährleisten. Hierbei wird im Feld angegeben, an welche Stelle sich die Daten des aktuellen fragmentierten Datagramms im endgültigen Datagramm befinden sollen.

TTL / Time to Live Dieses Feld soll der Möglichkeit vorbeugen, dass Daten nicht endlos von Router zu Router im Netz kreisen. Dabei wird der Wert des Feldes bei jedem Passieren eines Routers verringert, bis er schliesslich auf 0 reduziert und das Packet vom darauffolgenden Router nicht mehr angenommen wird.

Protocol Hier wird das übergenordnete Protokoll identifiziert, an welches der IP-Layer die Daten übergeben soll. Ein Beispiel für solche Protokolle wäre etwa TCP.

Header Checksum Da bei jedem Router der TTL-Wert und ev. auch die Werte für Flag und Flag Offset geändert werden, muss die Checksumme dementsprechend angeglichen werden. Um jedoch unnötige Verzögerungen in Kauf nehmen zu müssen, beschränkt man sich hierbei auf die Checksumme des Headers.

Source IP-Adress An dieser Stelle ist die IP-Adresse des sendenden Rechners vermerkt.

Destination IP-Adress Hier ist die IP-Adresse des Empfänger-Rechners vermerkt.

Options / Padding Hier werden verschiedenste Optionen angegeben, deren Aufzählung jedoch den relativen Rahmen sprengen würde. Das Padding hier verantwortlich, das Frame mittels binärer Nullen auf 32 Bit zu vervollständigen.

ICMP (Internet Control Message Protocol) ist dafür verantwortlich, Meldungen (z.B. Fehlermeldungen) zu übermitteln. Es benutzt zwar IP als wäre es selbst ein übergeordnetes Protokoll, ist jedoch ein fester Bestandteil vom IP. Hier noch eine kleine Liste der Meldungen, welche von ICMP gesendet werden:

Fehlermeldungen:

3 Destination unreachable (Zielstation nicht erreichbar)
4 Source quench (Buffer-Ressourcen verbraucht)
5 Redirect (Pfadumleitung)
11 Time exceeded (Timer abgelaufen)
12 Parameter Problem (Parameter Problem)

Informationsmeldungen:

0 Echo reply
8 Echo request
13 Time stamp

14 Time stamp reply
15 Information request
16 Information reply
17 Adress mask request
18 Adress mask reply

IP wird dann entsprechend modifiziert, und an den Rechner zurückgesendet. Auf die einzelnen Header-Modifikationen einzugehen, wäre nun etwas übertrieben

ARP (Adress Resolutin Protocol)
In einem Netzwerk hat jeder Rechner eine durch die firmware bereits vordefinierte physikalische Adresse. Für die Kommunikation mit einem Partner-Rechner wird jedoch nicht die physikalische, sondern eine logische Adresse verwendet, welche man beispielsweise im Internet findet. Der Netzwerktreiber alleine ist nun nicht fähig, diesen Rechner über diese logische Adresse anzusprechen, da sie in keinster Weise in Verbindung mit der hardware-Adresse des Controllers steht. Es muss nun also die logische Adresse in die physikalische Adresse umgesetzt werden. Dazu dient ARP. Es setzt sich aus einem MAC-Header und dem ARP-Packet zusammen, und enhält die nötigen Daten zu logischer Quell-Protokolladresse, als auch der physikalischen Adresse

RARP (Reverse Adress Resolution Protocol) funktioniert genau in die entgegen gesetzte Richtung. Anstatt anhand der logischen Adresse die physikalische zu ermitteln, sendet hier der Host einen RARP request mit der physikalischen Adresse aus, worauf dann RARP-Server im Netz ihre eigenen Referenz-Tabellen durchsehen, und einen RARP request zurücksendet, welcher die logische Adresse an den anfragenden Host zurücksendet. Diese Methode wird eigentlich nur von Rechnern verwendet, welche kein Medium besitzen, wo sie ihre dauerhafte, logische Adresse speichern können, und somit immer nur mit der Kenntnis ihrer physikalischen Adresse ins Netz booten.

Transportschicht

Wir haben bislang wohl am ausführlichsten IP besprochen. Wer genau gelesen hat, dem viel auf, dass es keine Sicherung der Verbindung bei IP gibt. In diesem Layer kommen wir zu jenen Protokollen, welche zum Teil auch für die Sicherung verantwortlich sind. Hier eine kleine Liste, was ein Protokoll der Transportschicht gewährleisten sollte:

- Ermöglichung von Datentransfer über dedizierte Transportverbindungen
- Kontrollierter Auf- und Abbau von Verbindungen
- Verwaltung mehrer zu einem Rechner zur gleichen Zeit (Multiplexing)
- Kontrolle, Fehlererkennung und Flusssteuerung über die gesammte Verbindung
- Optimierter Datenfluss (Windowing)
- Priorisierung im Datenfluss

Wärend das ach so bekannte Protokoll namens TCP all diese Bedingungen erfüllt, treffen eben diese bei dem zweiten, recht markanten Protokoll namens UDP so gut wie gar nicht zu. Weshalb UDP trotzdem zur Transportschicht gehört, wird weiter unten erklärt

Nun kommen wir aber zu den zwei Protokollen an sich. Als erstes

werden wir TCP ziemlich ausführlich beschreiben, danach noch eher kurz gehalten das UDP, welches auch nicht so viele Optionen enthält

TCP (Transmission Control Protocol) besitzt folgende Hauptmerkmal, welche es Charakterisieren:

- Datenstrom Transfer
- Virtuelle Full-Duplex Verbindungen
- Datenflusssteuerung
- Fehlererkennungen
- Prioritätensteuerung

Eine Anwendung kann TCP zum Aufbau einer Verbindung veranlassen und übergibt die Daten. TCP teilt dann die Daten auf bzw. segmentiert sie, und versieht jedes Segment mit einem eigenen Header. Nun erfolgt die Übergabe an das Internet Protocol. Sobald IP seinen Header nach dem transport durchs Netz wieder abgestreift hat, werden die Daten wieder an TCP übergeben, welches die Packete wieder sauber zusammensetzt und auf Fehler überprüft. Nun werden die Daten noch der entsprechenden Anwendung zugewiesen, und nach dem entsprechenden Befehl der Anwendung die Verbindung aufgelöst.
Wie bereits erwähnt, sorgt TCP für Datensicherheit. Jedes TCP-Segment von Host A wird von Host B bestätigt. Das nachfolgende Packet in einer Reihe von Packeten würde also erst dann versendet, wenn das vorangehende bestätigt wurde. Dies schafft zwar die grösst mögliche Sicherheit, lässt jedoch die Netz-Performance nahezu völlig ausser Acht. Aus diesem Grund hat man einen Mittelweg gefunden. Anstatt jedes einzelne Packet zu quittieren, wird immer eine kleine Gruppe von Sendungen quittiert. Dieses Verfahren trägt einen bestimmten Namen und funktioniert folgendermassen:

Sliding Windows bzw. Windowing: Das empfangen einer bestimmten Anzahl an TCP-Segmenten, benötigt beim Empfänger auch eine entsprechende Anzahl Buffer. Sobald diese Buffer voll sind, wird ein ACK(acknowlegement) gesendet. Die Grösse des zur Verfügung stehenden Speichers kann jedoch nicht beliebig vergrössert werden, da auch nicht beliebig Ressourcen zur Verfügung stehen. Die Grösse kann jedoch mit jedem gesendeten ACK vom Empfänger der Daten neu definiert werden. Diese "Window-Size" kann sogar bis auf den Wert 0 verringert werden, doch würde dies einen Stop des Datentransfers bedeuten. Tritt die Bestätigung für ein gesendetes Segment nich innerhalb einer bestimmten Zeit ein, so wird es einfach erneut übertragen

UDP (User Datagram Protocol) ist ein ungesichertes Protokoll, welches keine der oben unter TCP erwähnten Eigenschaften besitzt. Es ist ebenso ungesichert wie die Protokolle der unter der Transportschicht liegenden Layer. Die Begründung, wesshalb UDP trotzdem zur Transportschicht gehört, lautet folgendermassen: IP kann zwar Verbindungen herstellen, jedoch keine Daten an Anwedungen weitergeben. UDP kann das, genauso wie TCP. Allerdings erwartet UDP keine Bestätigung des Empfangs. UDP ist also s.z.s. eine Anwendungsschnittstelle zu IP. Auch der UDP-Header ist sehr kurz gehalten. Er enthält Informationen zum Ursprungs-Port, Ziel-Port, Länge des Datagramms und die Checksumme der Daten des UDP-Headers.

Hier eine kleine Tabelle mit Beispielen, welche Dienste mit UDP bzw. TCP erreichbar sind:

Port:	111	53	161	69	25	21	23	80
Dienst:	SUN RPC	DNS	SNMP	TFTP	SMTP	FTP	TELNET	HTTP
Protokoll:	User Datagram Protocol (UDP)				Transfer Control Protokol (TCP)			

Anwendungsschicht

Die Sitzungs- und Darstellungsschicht lassen wir hier deswegen aus, weil sie eigentlich nicht definiert werden können. Diese Layer stellen Schichten dar, welche in der Regel einfach Services zur Verfügung stehen, welche in der über- bzw- untergeordneten Schicht angesiedelt sind. Bei der Sitzungsschicht spricht man zum Beispiel von einer TCP-Session (TCP-Sitzung), welche in ihrer Form weder in Schicht 4 noch in Schicht 5 eingeordnet werden kann. Bei der Darstellungsschicht ist das in etwa das Selbe. Die grafische Oberfläche von Xwindows stellt hier als Präsentationslevel einen integralen Bestandteil des Darstellungs-Schicht dar.
Die Anwendungsschicht ist jene Schicht, in welcher der Benutzer seine Befehle quasi direkt über eine Anwendung eingeben kann, um eine Verbindung zu einem Rechner zu öffnen, oder entsprechende Befehle zu geben. Im umgekehrten Sinne ist also die Anwendungschicht auch jene Schicht, von der eine Anwendung auf Rechner A auch seine Befehle von Rechner B erhält. In der Anwedungsschicht gibt es eine ganze Menge Protokolle, und dem sind nach oben hin auch keine Grenzen gesetzt. Es gibt auch keine klar definierte "Boarderlinie". Gewisse Anwedungen können hier auch auf andere Anwendungen aufsetzen. Dies ist beispielsweise bei einem SNMP-basierenden Tool von HP mit dem Namen "Open View" der Fall. Nun aber zur Anwendungsschicht an sich:

TELNET: Anfangs der 80-Jahre, als die Zeit des PCs erst begann, gab es noch keine echten Netzwerke. Meist standen dort Grossrechner, an die "dumme" Terminals angehängt waren. Ein Terminal konnte dann also über eine Sammlung an Kabeln mit dem Grossrechner Daten austauschen, nicht jedoch mit weiteren Terminals. Um der Anschaffung von unmengen neuer Kabel aus dem Weg zu gehen, musste eine Software-Lösung her. So kam es dann zu telnet, welches dem Benutzer die Möglichkeit gab, Daten zu editieren etc. als sitze er vor der Shell des anderen Rechners selbst. Das Öffnen einer Verbindung erfolgt direkt durch den Befehl des Benutzers an die Anwendung. Wir könnten nun hier noch alle Basisbefehle von telnet auflisten, doch würde dies wohl keinen grossen Sinn machen.

FTP (File Transfer Protocol) ist ein Dienst der es ermöglicht, innerhalb aller Betriebssysteme Daten zu pebrtragen und sie in den jeweiligen Dateiformaten abzuspeichern. Bekanntlich benutzen die meisten Betriebssysteme unterschiedliche Dateiformate. UNIX und UNIX-Klone verwenden oft NFS (Network File System), O/S2 normalerweise HPFS (High Performance File System) und DOS ausschliesslich FAT (File Allocation Table). Die Kommunikation über FTP basiert wie bei telnet auf dem Client-Server-Modell, ist jedoch um einiges komplexer. Wir wollen nicht ganz genau darauf eingehen. Folgende Punkte seien trotzdem erwähnt: Die Kommunikation ist in 5 Phasen eingeteilt:

1. Phase: Verbindungsaufbau
Hier wird vom Client die Anfrage auf Verfügbarkeit des Dienstes an den Rechner gesendet und von diesem bestätigt, User und Passwort verifiziert und die Übertragungsoptionen sowie der bzw. die Dateinamen werden übermittelt

2. Phase: Erstellung einer Datenverbindung
Hier werden die Informationen bezüglich der Ports ausgetauscht, und der eigentliche Datentransfer vorbeireitet. Nachdem dies festgelegt ist, kann der eigentliche Datentransfer beginnen

3. Phase: Datenübertragung
Die Datenübertragung erfolgt nun via FTP in der Form, wie es im entsprechenden Abschnitt bereits erklärt wurde

4. Phase: Einleitung vom Übertragungsende
Der Rechner übermittelt die letzten Daten der gesammten Datei, der Client bestätigt den Empfang dieser Dateien. Nun wird vom Rechner ein Close-Befehl an den Client gesendet, welcher den Befehl entgegennimmt und akzeptiert.

5. Phase: Übertragungsende
Der Server-Datenprozess zeigt seinem Kontrollprozess (Port 21) das Ende der Übertragung an und beendet. Der Client-DatenProzess terminiert ebenfalls, lässt den Kontrollprozess jedoch noch aktiv für weitere Transfers.

Auf die nahezu 60 unter FTP zur Verfügung stehenden Befehle möchte ich hier nicht weiter eingehen, da dies den Rahmen sprengen würde.

TFTP (Trivial File Transmission Protocol) basiert nicht wie das vorhin behandelte FTP auf TCP sondern auf UDP. Es Dient zwar auch der Übertragung von Daten, ist jedoch nicht für den Endverbraucher bestimmt. Im Grunde genommen gibt es bei diesen Transfers auch keine Passwortabfrage, sondern höchstens die Hinterlegung der Source-IP, damit man über den möglichen Befehlsumfang zu verfügen. Die wichtigsten sind hierbei connect, mode, get, put, verbose und quit.

Ich werde nun kurz erklären, wie eine Datenübertragung hier abläuft, obwohl die Verbindung ungesichert ist. TFTP basiert ebenfalls auf dem Client-Server Prinzip. Der Client sendet hierbei einen Request an den Server, dieser Bestätigt den Request und beginnt die Datenübertragung. Jeder Datensatz beträgt hierbei 512 Byte und wird vom Server bestätigt. Das Ende der Übertragung wird vom Client automatisch dann angenommen, wenn ein entgegen genommener Satz weniger als 512 Byte lang ist.

BOOTP (Boot Protocol) ist UDP basierend, und wurde eigentlich nur dazu entwickelt, um Boot-Vorgänge zu aktivieren. Dies wird nur dort nötig, wo discless-workstations betrieben werden, da diese, wie schonmal erwähnt, ihre logische Adresse nicht speichern können. Mit Booten ist hier übrigens nicht das eigentliche Booten gemeint, sondern lediglich die Übernahme wichtiger Konfigurationsdaten. Eigentlich brauchen wir darauf auch nicht genauer einzugehen, da dies heute nicht mehr von grosser Relevanz ist.

SMTP (Simple Mail Transfer Protocol) ist das im Internet wohl am meisten benutzte Protokoll. Seit frühester Zeit hat sich SMTP bereits auf UNIX-Systemen etabliert und mitlerweile auch auf normalen PCs seinen platz gefunden. Hier bedient der Anwender seine Mailsoftware und bereitet eine Nachricht vor. Schickt er dann seine Nachricht ab, so wird diese solange zwischengespeichert, bis TCP die gesammte Nachricht übertragen wurde. Auch hier stehen dem Client als auch dem Server eine Reihe von Befehlen bzw. Reaktionen zur Verfügung, welche ich nicht auflisten werde. Stattdessen werde ich hier einen kleinen Dialog zwischen Server und Client wiedergeben (wir gehen dabei immer von einer positiven Antwort seitens des Servers aus):

- Client baut eine Session zum Server auf
- Server bestätigt die Verfügbarkeit des Services, oder verweigert, da der Service nicht verfügbar ist
- Client identifiziert sich
- Server identifiziert sich
- Client übergibt den eigentlichen Befehl der einen Mail-Versand ankündigt.
- Der Server gibt sein Einverständnis
- Client übermittelt den Empfänger
- Server antwortet mit: Mailbox erreichbar bzw. Mailbox nicht erreichbar
- Client iniziiert die Datenübertragung
- Server nimmt die Daten auf, und verlangt zur Beendung den Befehl <crlf><crlf>
- Client sendet nach Beendung der Übermittlung wie vereinbart <crlf><crlf>
- Client beendet die Verbindung mit dem entsprechenden Befehl
- Server antwortet darauf mit "service closing"

Als in den 80-er Jahren noch andere Mail-Systeme eingeführt wurden, kam es zu Kompatibilitätsproblemen. Die Übergänge waren nicht einfach so zu bewältigen und so musste man sehr umständliche Konverter einsetzen. Seit 1992 Ist nun hauptsächlich MIME (Multipurpose Internet Mail Extensions) im Einsatz, welches sich nicht mehr nur auf den reinen Textversand beschränkt, sondern verschiedenste Datentypen wie Grafiken, Audiodaten u.s.w. zu versenden vermag.

RPC (Remote Procedure Calls) wird verwendet, wenn mehrere Rechner mit verschieden gestalteten Kapazitäten zur Verfügung stehen und eine Aufgabe zu erledigen ist, welche sehr grosse Systemressourcen ausschöpfen. Mit RPC können nun Teilaufgaben spezifisch an die dafür am besten qualifizierten Rechner zugewiesen werden und die verschiedenen Rechner verschmelzen s.z.s. zu einem Multi-Computer. Dieses Verfahren findet heutzutage übrigens je länger je mehr Anklang bei grossen Unternehmen. Auf die genauere Struktur wird hier nicht eingegangen.

NIS (Network Information Services) wird zur Verwaltung der Operationen, Security-Objekte und Zugriffsrechte verwendet. Dieses ursprünglich von SUN entwickelte System (damals Yellow Pages) ermöglicht die zentrale Administration dezentraler UserIDs, GroupIDs und Passwörter. Um NIS zu betreiben sind folgende Komponenten vorrausgesetzt:

NIS-Datenbank: Sie stellt quasi eine übergrosse /etc/passwd-Datei für das Netzwerk dar

NIS-Master-Server: Er verwaltet die NIS-Datenbank für die

entsprechende Domäne

NIS-Slave-Server: Enthält Sicherungskopie der Datenbank für Ausfälle

des Master-Servers

NIS-Domäne: Eine Gruppe von auf der NIS-Datenbank abgebildeten

Rechnern

NIS-Client: Rechner, welcher Daten vom Server beziehen, jedoch nicht

ändern kann

Somit hätten wären dann die wichtigsten Protokolle beschrieben, welche für den Verbindungsaufbau nötig sind. Natürlich sind es längst nicht alle. Wer fundiertere Informationen zu TCP/IP sucht, dem sei der Kauf eines entsprechenden Buches zu empfehlen. Am besten sucht man sich einmal durch Amazon.de durch und liest entsprechende Empfehlungen. Um noch tiefer in die Materie einzusteigen oder gewisse aktuelle Dinge nachzuschlagen sollte man jedoch die RFCs nachschlagen, von welchen auf http://info.internet.isi.edu/in-notes/rfc/ ein Archiv zu finden ist.