Das ARPANET

Ein technischer Einblick

Inhaltsverzeichnis

„Es ist der Vorläufer des heutigen Internets[...]“[1] wird in vielen Quellen behauptet, teilweise ohne über die Gemeinsamkeiten und Unterschiede zwischen ihm und dem *World Wide Web* nachzudenken. Doch zweifellos war es wohl eines der erstaunlichsten Netzwerke, mit dem meiner Meinung nach geringsten technischen Aufwand:

DAS ARPANET

Meine Arbeit möchte ich diesem Netzwerk widmen. Einige Schüler trafen in der Vergangenheit auch diese Wahl, jedoch waren die originalen Programmcodes des ARPANETs lange unter Verschluß und die recherchierten Informationen unvollständig oder sogar häufig falsch.

Am 14.02.2005 hat das **A**merican **N**ational **S**tandards **I**nstitute (ein Normungsinstitut) erstmals die ursprünglichen Quellcodes offengelegt. Mit Hilfe dieser Quellen, an denen sich damals ausschließlich Programmierer beteiligten, möchte ich so manchen Gerüchten aus Büchern begegnen. Viele Buchautoren werden sich auch in der Zukunft nicht mit den Quelltexten beschäftigen. Sie sind in der Programmiersprache B+(71) geschrieben und nur sehr schwer zu verstehen beziehungsweise auf heutige Rechner zu transferieren. Den groben Inhalt einiger Quelltextauszüge erkläre ich in meiner Arbeit und versuche die Gedankengänge der Programmierer darzulegen. Vor allem beziehe ich mich auf die Quelltexte des Programmierers Oliver Hasten, da ihm der überwiegende Teil der Softwareentwicklung zu verdanken ist.
Zu Beginn meiner Jahresarbeit möchte ich auf eine Verbindung eingehen, die in vielen Büchern zur Entstehungsgeschichte des heutigen Internets auf den ersten Seiten erwähnt wird:

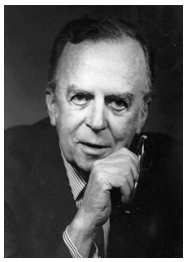
"Das ARPANET – eine Technologie..."[2]
Der Begriff „Technologie" ist sehr vielseitig definiert. Zum ersten Mal trat er 1769 auf und wurde als „Wissenschaft, welche die Verarbeitung der Naturalien lehrt"[3] definiert. Das ARPANET ist demnach eine Wissenschaft? Nein, es diente wohl eher als Werkzeug für das wissenschaftliche Arbeiten. Programmierer schlossen sich zu einer Art Gemeinschaft namens ARPA (**A**dvanced **R**esearch **P**rojects **A**gencies) zusammen und entwickelten ein Hilfsmittel um Forschungsdaten auszutauschen. Nur durch das ARPANET konnten die Universitäten in den Vereinigten Staaten verbunden werden. Außerdem hat das ARPANET absolut nichts mit Naturalien zu tun. Der ursprüngliche Technologiebegriff trifft demnach nicht mehr auf das ARPANET zu wohl aber die Definition „Technology is a tool for further works[...]"[4].

[1] Hauben, M.; Hauben, R. (1997): Netizens: On the History and Impact of Usenet and the Internet. Los Alamitos, CA: IEEE Computer Society Press. , Seite 1, Zeile 12
[2] Prof. Teuber: „Die Erfindung des Internets“, Seite 1, Z.3
[3] Definition von J. Beckmann
[4] Definition von Peter H. Salus: „Casting the Net“ (S. 1, Z. 14)

Die Technologie ARPANET ist, wie die letzten drei Buchstaben schon andeuten, ein Netzwerk. Die Idee zur Entwicklung ist aber nicht erst in den sechziger Jahren entstanden, wie es viele Buchautoren behaupten. Schon etwas früher spielten Menschen mit dem Gedanken einer Vernetzung:
Im Mai 1945 endete der Zweite Weltkrieg in Europa. Genau in dieser Zeit entwickelte sich die erste Idee. Die USA sind aus dem Zweiten Weltkrieg als Sieger hervorgegangen mit der Einstellung: „Wir-schaffen-das" [5]. Schon damals wußten die Menschen von der „verändernden Kraft des Zweiten Weltkrieges"[6]. Diese Kraft machte auch vor Universitäten nicht halt.
Im Jahre 1943 wurde der amerikanische programmgesteuerte Rechner **Computing Maschine 1 (CM 1)** in Betrieb genommen. Die Bausteine waren Relais. Davon gab es 800 im Rechenwerk, sowie 1.800 Relais im Speicherwerk. Die Taktfrequenz betrug etwa 5-10 Hertz. Der ca. eine Tonne schwere Rechner diente überwiegend ballistischen Berechnungen.

Vergleichbar war sie mit Zuses Z3, welche Zuse aber bereits 1941 in Deutschland vorgestellte.

Ärgerlich war zur damaligen Zeit eine fehlende Datenübertragung der Forschungs- und Rechenergebnisse. Sie konnten entweder auf einem Blatt Papier oder telefonisch weitergegeben werden. Beide Varianten waren entweder langwierig oder sogar unsicher.
Der Programmierer Dr. J.C.R. Licklider, dem heutzutage fälschlicherweise die Anerkennung an der Netzentwicklung „zugeschanzt" wird, versuchte damals zwei Rechner über einen parallelen Anschluß zu verbinden. (Heute werden noch einige Drucker noch über diesen betrieben. (LPT1))



Solange die Kabellänge zehn Meter nicht überschritt, kamen Stromimpulse beim empfangenden Rechner an. Aber lt. Steve Carr war Lickliders Bauweise fast schon idiotisch[7], denn auch damals kannten Programmierer die Abhängigkeit der Stromspannung von der Kabellänge eines parallelen Anschlußkabels: Wurde ein Kabel länger als zehn Meter hergestellt, waren die gesendeten Stromimpulse zu schwach und zu fehlerbehaftet, um Sie zu interpretieren. Diese physikalische Einschränkung beobachten heute noch Anwender, wenn ihre Drucker zu weit vom Computer entfernt stehen. Es kam, wie es kommen mußte, Licklider verwarf dieses Projekt nach zahlreichen Fehlschlägen.

[5] David Kennedy: http://www6.dw-world.de/de/1950.php
[6] David Kennedy: http://www6.dw-world.de/de/1950.php (unterer Seitenabschnitt)
[7] Quellcode Vornotizen von Steve Carr: „Introduction", 1. Absatz, 1961

Die erfolglosen Versuche mit der parallelen Schnittstelle dauerten insgesamt bis 1950. Danach schliefen Lickliders Unterlagen in einer Schublade des Verteidigungsministeriums den Schlaf des Vergessenen.

Die Zeit verstrich, die Computer wurden etwas leichter und kompakter und an der Westküste Amerikas schlossen sich Programmierer 1961 zu einer Projektgemeinschaft **ARPA** (Advanced Projects Research Agency) zusammen.
Was bisher nicht bekannt war: Der angebliche Leiter Dr. J.C.R. Licklider war gar nicht an der Gemeinschaft beteiligt. Obwohl seine Teilnahme eingeplant war, zog er es vor weiterhin mit der alten Programmiersprache BCPL zu experimentieren:
„In November 16th 1961 Licklider refused his membership."[8]
Beteiligt waren lt. Aufzeichnungen folgende Programmierer: „[...]present were **Oliver Hasten** and **Steve Carr** from **University of Utah**, **Stephen Crocker** from **University of California (LA)**, **Jeff Rulifson** from **Stanford Research Institute** and **Ron Stoughton** from **UCSB**[i]."[9]

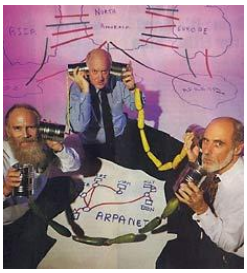
Hier sind die Programmierer von links nach rechts abgebildet. **Oliver Hasten** steht auf der linken Seite im Vordergrund, hinter ihm **Steve Carr**, Der erste Programmierer in der vorderen Sitzreihe links ist **Stephen Crocker**, daneben sitzt **Jeff Rulifson** und ganz rechts **Ron Stoughton**.
Die anderen Programmierer im Hintergrund waren nur aushilfsweise an der Programmierung beteiligt.

Die Programmierung erfolgte mit s.g. Terminalrechnern. Solch ein Rechner hatte eine Taktfrequenz von 600 Hz – 900 Hz und kostete ca. 80.000$. Sein Stromverbrauch war mit einem heutigen Blockhaus in Berlin zu vergleichen. Der wie ein Kühlschrank aussehende Rechner bestand aus sechs Einzelrechnern:

Die eigentliche Aufgabe des Teams war es allerdings nicht, wie in vielen Quellen angegeben, ein bombensicheres Netzwerk aufzubauen: „created by the United

[8] Quellcode Vornotizen von Oliver Hasten: „Members of ARPA:" (letzter Absatz)
[9] Quellcode Vornotizen von Oliver Hasten: „Members of ARPA:"

States Defense Advanced Research Project Agency (ARPA)"[10]
Sie sollten die damalige Programmiersprache BCPL in B (Hochsprache) weiterentwickeln. Die Programmiersprache B sollte es ermöglichen mit wenig Zeitaufwand eine Schlachtsimulation „durchzuspielen" und zwar in Kombination mit ballistischen Berechnungen: „We are supposed to invent a high level language for ballistic simulations in accordance to military rendering."[11]
Die Programmierer arbeiteten in verschiedenen Universitäten, die weit von einander entfernt lagen (Universität von Utah und Universitäten in Kalifornien). Möglichst schnell austauschen konnten sie Informationen nur telefonisch.
Die einzelnen Teilarbeiten diktierten sich die Programmierer per Telefon, die Codes tippten sie dann ab und fügten diese erneut zu einem Programm zusammen.
Darüber macht sich heute noch das **A**merican **N**ational **S**tandards **I**nstitute mit folgender Karikatur lustig:



Sehr mühsam war es auf diese Art und Weise, die neue Programmiersprache fertigzustellen, bis die Programmierer eine Idee der Vernetzung in Betracht zogen. Die Idee dazu kam übrigens von der University of Utah[12] (von Oliver Hasten & Steve Carr) und nicht von einer Universität an der Ostküste. Es wurde versucht, drei Universitäten miteinander zu vernetzen, das Netz sollte **ARPANET** heißen. Die Entwicklung war technisch so primitiv:
„A climate of pure research surrounded the entire history of the ARPANET."[13]

"Am Anfang war das Wort..." und die Wörter hießen bei der Entwicklung „HALLO WORLD". Es war die erste Zeichenfolge, die sie durch das ARPANET schickten und seit diesem Ereignis ist die Ausgabe „HALLO WELT" bzw. „HALLO WORLD" bei Programmierneulingen Tradition geworden. Das erste Programm gibt meistens genau den oben genannten String[ii] auf den Bildschirm aus.

Warum sich der Programmierer Oliver Hasten zur damaligen Zeit genau diese zwei Wörter ausgedacht hat, kann ich seinen Kommentaren zum Quellcode nicht entnehmen. Wahrscheinlich ahnte nicht einmal er, welche Bedeutung ihnen

[10] http://www.webopedia.com/TERM/A/ARPANET.html, Zeile 2
[11] Notizen des Stanford Research Institutes: „Sources of ARPA", 2. Kommentarabschnitt
[12] Chronologische Übersicht der Entwicklungsstufen des ARPANETs (University of Utah)
[13] Peter H. Salus: „Casting the Net: From ARPANET to INTERNET and Beyond" (S. 3 / Zeile 7)

zukommen sollte. Sein Problem bestand darin, diese beiden Wörter von einem Computer zum anderen zu senden.
Bevor er jedoch die Zeichen transferieren konnte, mußte Hasten sie dem Computer erst einmal in der Binärsprache (od. Dualsprache) verständlich machen, denn es ließen sich auch zur damaligen Zeit nur die Zustände **0** oder **1** auf einem Speichermedium festhalten.
Die Lösung war ein Zeichensatz, eine Verknüpfung eines jeden Zeichens mit einer Zahl in Binärform. Für ein Zeichen wurden 6 Bit Arbeitsspeicher benötigt [iii]. In diesen 6 Bit hätten sie 64 mögliche Zeichen speichern können. Der Zeichensatz bestand jedoch nur aus 40 Zeichen. Das Resultat: Speicherverschwendung
Wurde die Zeichenkette „HALLO WORLD" im Arbeitsspeicher abgelegt, sah das so aus:

H	A	L	L	O		W	O	R	L	D	\0

Jedes Zeichen benötigte 6 Bit, das Zeichen für die Terminierung[iv] (‚\0') wird angehangen, um dem Computer mitzuteilen, wo die Zeichenkette endet, denn die eigentliche Länge des Strings wurde damals nicht im Arbeitsspeicher abgelegt. Heute ist das Speichern der Länge auch nur in der Programmiersprache BASIC üblich. Der obere String benötigt 12 Zeichen x 6 Bit = 72 Bit (9 Byte) Arbeitsspeicher. Im Speicher wird die Zeichenkette schematisch so abgelegt:
0x2FF0

0	1	0	0	1	0	0	0	1	0	1	1	0	1	0	1	1	0	0	1	0	1	1	0	0	1	1	0	0	1	1	0	1	0	0	1
1	0	0	0	0	1	0	1	1	0	0	1	0	1	1	1	0	0	0	1	0	1	1	0	0	0	1	1	1	0	0	0	0	0	0	0

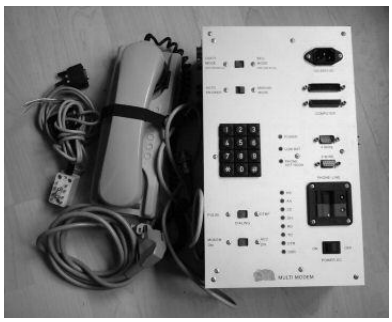
Anhand der Terminierung (rot dargestellt) weiß der Computer, wo die Zeichenkette endet. In einer Variable vom Datentyp **char**[v] legt das Programm die Anfangsadresse der Zeichenkette im Speicher ab, in diesem Fall den hexadezimalen Wert **0x2FF0**. Hasten speicherte die Zeichenkette in einer Variablen namens „str": „Let's call it str"[14]. Der Befehl dazu sah folgendermaßen aus:

```
char *str=new char[11+1]
strcpy(str, "HALLO WORLD\0");
```

In der ersten Zeile wird eine Variable deklariert, das heißt im Arbeitsspeicher werden genau 12 x 6 Bit für eine Zeichenkette reserviert. Die Variable *str* assoziiert die Rechenmaschine mit der Anfangsadresse des Strings.

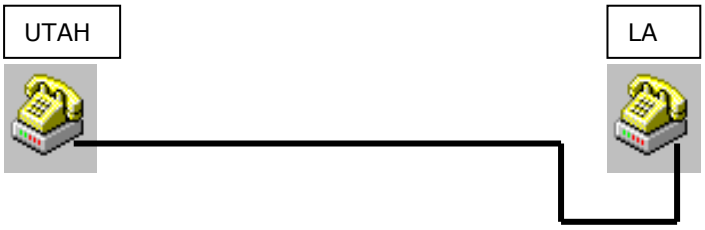
[14] Oliver Hasten, Quelltextkommentar zur Deklaration der Variablen „str"

In der zweiten Zeile kopiert der Rechner in den reservierten Speicher die Zeichenkette „HELLO WORLD". Jetzt entspricht der Speicherausschnitt von der Anfangsadresse bis zur (Anfangsadresse+11) dem Ausschnitt des oberen Schemas. Diesen Speicherabschnitt wollte Hasten einem zweiten Rechner senden.
In den USA war das Telefonnetz in den 60er Jahren an der Ost- und Westküste gut bis sehr gut ausgebaut. Es verband wichtige Städte miteinander. Der einfachste Weg wäre gewesen, so Hasten, die andere Universität anzurufen und die beiden Worte mitzuteilen (Er nannte es „Information by call"[15].) Jedoch hätte wieder eine Person das Wort abtippen müssen. Aber „how is it possible to render the word automatically?"[16] Die Antwort war gar nicht so schwer:
Wenn der Mensch anrufen kann, um sich mit anderen Wissenschaftlern zu unterhalten, warum soll sich dann der Computer nicht auch mit einer anderen Rechenmaschine unterhalten können?
Durch die Telefonleitung konnte man Töne senden. Hasten mußte nur noch eine Möglichkeit finden, die Buchstaben in Töne umzuwandeln. Folgendes überlegte er sich:
Es würden drei Töne ausreichen, um eine Zeichenkette zu übertragen: „The machine needs three different tones for submitting strings."[17] Die Töne sollten sich in der Tonhöhe sehr stark unterscheiden, aber nicht in der Lautstärke und der Intensität. Aus diesen Überlegungen wählte Hasten den höchstmöglichen, den tiefstmöglichen und einen Zwischenton für die Übertragung von Zahlen und Zeichen aus. Die Töne dürfen dabei für jeden Leitungstyp individuell sein. Er behielt das Binärsystem bei. Aus jeder 1 wurde ein hoher Ton und aus jeder 0 ein tiefer ton. Der Zwischenton sollte die Übertragung abschließen (terminieren).[18]
Für die Übertragung bedienten sich die Programmierer eines einfachen Prinzips: Ein Lautsprecher, ein Mikrophon, ein kleiner Speicher und ein „Identifier" zum Erkennen, ob es sich um einen hohen, einen tiefen oder einen Zwischenton handelt. Das Gerät nannte er Modulator, da es die Zustände 0, 1, - in die jeweiligen Töne umwandeln (modulieren) und die Töne in die mathematischen Zustände zurückverwandeln (demodulieren) konnte:



Die Verbindung wurde folgendermaßen hergestellt: Ein Terminal befindet sich in Utah, das andere in Los Angeles. Zunächst wählt eine Person in Utah mit dem Telefon die Nummer in LA. In LA wird abgenommen. Eine Verbindung ist aufgebaut..

[15] Quellcode Vornotizen von Oliver Hasten: „Sending mathodes"
[16] Quellcode Vornotizen von Oliver Hasten: „Sending mathodes"
[17] Quellcode Vornotizen von Oliver Hasten: „Sending mathodes"
[18] Anhang: Zeichensatz (T → tiefer Ton, H → hoher Ton, Z → Zwischenton)



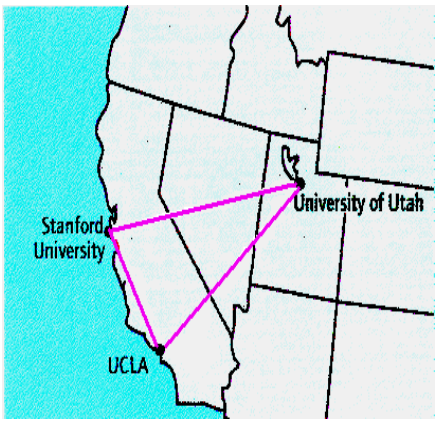
Beide Personen legen jeweils ihre Telefonhörer auf den Modulator. (Im oberen Bild liegt bereits ein Telefonhörer auf den Modulator.). Jetzt senden sich beide Modulatoren den tiefsten, den höchsten und den mathematischen Zwischenton zu und speichern ihn. Die Modulatoren sind wiederum mit dem Computer verbunden und zwar durch einen parallelen Anschluß mit kurzem Kabel (Im Bild befinden sich die beiden parallelen Anschlüsse rechts oben unterhalb des Stromanschlusses.) Über einen Anschluß wurden Daten zum Computer gesendet, über den anderen wurde empfangen.

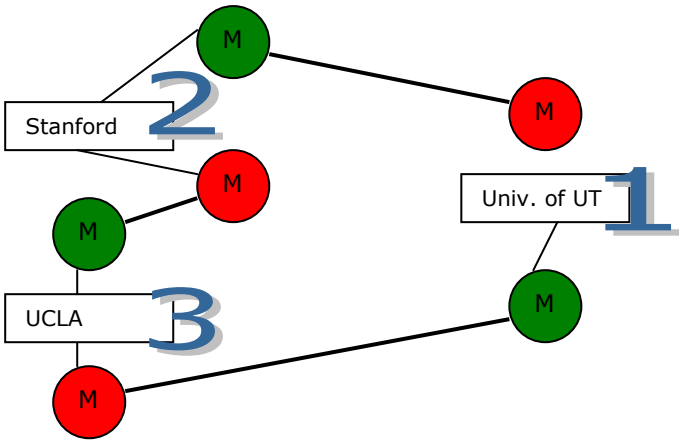
Der Modulator bekommt in Form von Stromimpulsen die Daten im Binärformat, wandelt Sie in Töne um und gibt diese Töne durch die Telefonleitung an den zweiten Modulator durch. Der Modulator in Utah „spricht" praktisch mit dem Modulator in LA. Dabei muss vorher geregelt werden, welcher Modulator „spricht" und welcher hört.

Die Programmierer wählten dafür die Einteilung „master"[19] und „slave"[20]. Diese Einteilung wird heute noch auf Festplatten angewandt. Wurde das Gerät als „master" konfiguriert, sendet es den ersten String nach der Verbindung. Das „slave"-Gerät hört zunächst und empfängt den gesendeten String vom „master". Wurde der Sendevorgang vom „master" durch den mathematischen Zwischenton beendet, überträgt das „slave"-Gerät die empfangenen Töne als Stromimpulse auf den Rechner. Der Rechner kann die nun empfangenen Daten im Arbeitsspeicher ablegen. Jetzt kann der „slave" Daten an den „master" senden, da der „master" nun in den Hörmodus übergegangen ist. Auf diese Art und Weise konnten Informationen zwischen zwei Rechner gesendet werden. Ein Modulator befindet sich dabei im Hörmodus der andere im Sendemodus. Durch die Dialektik der beiden Modi sendeten sich beide Rechner die Daten abwechselnd zu. Da die Töne auch bei großen Entfernungen deutlich übermittelt wurden, konnte die Projektorganisation ARPA die Universitäten von Utah und LA 1969 erstmals verbinden.

Bei zwei Rechnern funktionierte das Verfahren, doch was passierte, wenn es mehr Rechner wurden. In der unteren Abbildung möchte ich dies verdeutlichen.

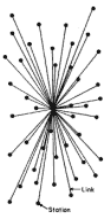
[19] Quellcode Vornotizen von Oliver Hasten: „device configuration"
[20] Quellcode Vornotizen von Oliver Hasten: „device configuration"





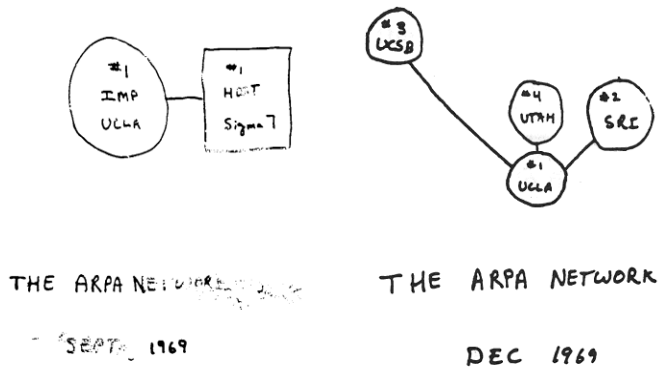
Jeder Rechner benötigte zwei Modulatoren (M). Ein Modulator hört (grün) und ein zweiter sendet (rot). Gesendet wird entgegen dem Uhrzeigersinn. Möchten die Programmierer der Universität von Utah der Universität von Kalifornien, Los Angeles die Zeichenfolge „HALLO WORLD\0" senden, dann sendet der Computer in Utah zunächst „3HALLO WORLD\0" an den Rechner der Stanford University. Dieser erkennt an der drei (3) als erstes Zeichen, dass die Information für UCLA bestimmt ist und sendet Sie an ihren Zielort weiter. Die Computer werden bei dem Netzwerk entgegen dem Uhrzeigersinn durchnumeriert. Ein direktes Senden von Informationen ist im obigen Beispiel nur von Rechner 1 zu 2, 2 zu 3 und 3 zu 1 möglich. Befinden sich mehr als 9 Rechner in solch einem Netzwerk müssen schon die ersten zwei Zeichen eines Strings für die Zielindizierung (Empfängernummer) verwendet werden. Je mehr Rechner ein solches Netzwerk aufspannen, desto weiter werden die Datenwege und desto länger dauert der Sendevorgang. Außerdem ist dieses Verfahren unsicher, da die Daten erst durch Fremdrechner transferiert werden. Fällt ein Rechner aus, fällt das gesamte Netzwerk aus.
Ein solches Netzwerk bezeichnet der Programmierer als dezentral. Es gibt keinen Computer im Zentrum, bei dem sich alle Rechner einwählen.

Ganz anders ist das heutige Internet aufgebaut. Alle Nutzer wählen sich bei Ihrem Provider ein und es fällt auch nicht gleich das ganze Internet aus, wenn Nachbars Rechner abstürzt. Ein zentrales Netzwerk sieht z.B.: so aus:



Im Zentrum befindet sich der s.g. Einwahlknoten. Die Rechner, die sich bei ihm einwählen, nennen Programmierer Clientrechner. Fällt ein Clientrechner aus, bleibt

das Netzwerk weiterhin bestehen. Dieser Vorteil brachte den Programmierer Jeff Rulifson zum grübeln: „Network expandation means a centeral network?!"[21]
Es kam, wie es kommen mußte. Am 08.06.1969 entwickelten die Programmierer die zentrale Version des ARPANETs. Die Versandmethode blieb allerdings gleich. Der Server bekam insgesamt 400 Modulatoren zugewiesen, also 800 parallele Anschlußmöglichkeiten. Das sind 80 große Anschlußleisten mit jeweils 10 Anschlüssen (5 x Eingang & 5x Ausgang).
Hardwaretechnisch war das zwar kein großes Problem, doch wie sollte der Server im Zentrum wissen, an welchen Rechner Daten zu senden sind?
Jeder Rechner benötigte eine eindeutige Nummer, welche vom Rechner im Zentrum (Server) zugewiesen wird. Der Rechner der UCLA befand sich im Zentrum und bekam die Identifikationsnummer 1. Der Rechner der SRI wurde mit 2 indiziert. LXSG bekam die ID 3 und die University Of Utah die Nummer 4.



Die drei Universitäten wurden Clients genannt und den Rechner der UCLA gaben die Programmierer den Namen Server. Nachdem sich die drei Rechner eingewählt hatten und der Server jedem Rechner eine ID zugewiesen hatte, spannten Sie die s.g. location auf.
Eine location ist der Ort um ein Netzwerkverbindung. Sie wird auch als zweite Verbindungsebene bezeichnet.
Die Programmierer betrachten ab diesem Zeitpunkt nicht mehr die physikalischen Verbindungen. Alle Rechner sahen Sie unvernetzt in einer Fläche und ab jetzt konnten immer zwei Rechner beliebig verbunden werden. Dabei ignorierten sie die Hardware und den Server im Mittelpunkt. (Dieses vereinfachte Prinzip soll die Denkweise verdeutlichen. Die Programmierer beachteten natürlich zweitrangig trotzdem die physikalische Verbindungsstruktur.)
Bei Softwareverbindungen der zweiten Ebene zwischen zwei Rechnern war jeweils ein Rechner Server und der andere Client.

[21] Jef Rulifson: Notizen zur Verbindungstestreihe 1969

Die Verbindung zweier Rechner über erfolgte softwaretechnisch über Sockets. Ein Socket ist ein virtueller Kommunikationssendepunkt, eine abstrakte Schnittstelle zur Kommunikation zwischen zwei Computern. Die Programmierer definierten eine Schablone für Verbindungen, die Klasse CSocket:

```
class CSocket
{
    UINT Connect(){...}
    UINT Send(char *pstr){...}
    UINT Receive(char *pstr, int nSize)
    {...}

    int NumberOfClient;
    int NumberOfServer;

    ...
};
```[22]

```
class Schaf
{
      Sprechen(){Ausgabe: Määäääh;}
      Laufen(ZAHL Schritte)
         {//Schaf läuft}

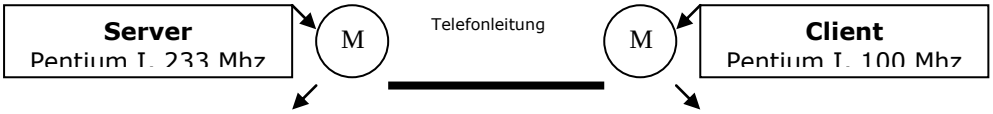
      ZAHL Größe;
      ZAHL Alter;
      ZAHL Gewicht;
};
```

Das Klassensystem möchte ich zunächst zum besseren Verständnis an der Klasse „Schaf" erklären. Eine Klasse ist, wie schon erwähnt, eine Schablone, die Gemeinsamkeiten bestimmter Objekte erfaßt. Jedes Schaf hat z.B.: eine Größe, ein Alter und ein Gewicht. Die einzelnen Werte sind jedoch individuell und werden durch Zahlen (Variablen) repräsentiert.
Jedes Schaf kann bestimmte Dinge tun z.B.: „sprechen" und laufen. Was ein Objekt kann, wird als eine Funktion erfaßt. Eine Funktion kann Parameter haben, die wiederum Auswirkungen auf das Funktionsverhalten haben. Bei der Funktion *laufen* bestimmt der Parameter „Schritte" die Anzahl der Schritte, die ein Schaf laufen soll.
Aus einer solchen Schablone lassen sich nun beliebig viele Schafe erzeugen. Analog funktioniert dies mit Sockets. Jeder Socket enthält die Nummer des Clienten und die Kennnummer des Servers (2. Ebene), mit dem er verbunden ist. Dadurch weiß er, an welchen Rechner Informationen gesendet werden können. Bildlich gesehen ist ein Socket nichts weiter als ein virtueller Modulator, der eine Verbindung zwischen zwei beliebigen Rechnern in der Softwareebene aufspannt. Dadurch können in einem zentralen Netzwerk die Rechenmaschinen ohne Zwischencomputer verbunden werden.

Mit meinen technischen Möglichkeiten möchte ich nun einen Teil des ARPANETs in Miniaturform wieder aufleben lassen, also eine hardwaretechnische und softwaretechnische Verbindung zwischen zwei Verbindungsrechnern simulieren. Leider habe ich nur noch zwei funktionsfähige Modulatoren. Deshalb starte ich die

[22] Quellcode: Die Klasse CSocket, 1969

Simulation mit zwei Rechnern. Folgender Versuchsaufbau:



Ich habe die Telfonhörer auf die Modulatoren gelegt (1. Ebene). Die drei Töne tauschten sich die Modulatoren aus und wiesen sich IDs zu. Der Server wurde mit ID 0 und der Client mit ID 1 indiziert. Auf Seiten des Servers erzeuge ich nun einen Socket (2. Ebene), der auf einen Clienten wartet:

```
Advanced Research Projects Agencies
Terminal Client Connection


terminal > server_sock
terminal > Socket launched
terminal > Socket threaded
terminal > Socket launched as server 0
terminal > ARPANET is waiting for client...
```

Ich verbinde zum Test meinen Clientrechner mit dem Server (Clientbildschirm rechts, Serverbildschirm links):

```
Advanced Research Projects Agencies
Terminal Client Connection


terminal > server_sock
terminal > Socket launched
terminal > Socket threaded
terminal > Socket launched as server 0
terminal > ARPANET is waiting for client...
terminal > Client connection accepted, client ID 1
```

```
Advanced Research Projects Agencies
Terminal Server Connection


terminal > client_sock
terminal > Winsock launched
terminal > Socket created!
terminal > Connected with server 0. this: 1

Insert characters [max 256]:
```

Der Server akzeptiert den Clienten. Ich könnte theoretisch weitere Verbindungen aufspannen, wenn mir mehr Modulatoren zur Verfügung stehen würden. Jetzt bietet mir der Clientrechner an eine Zeichenfolge zu senden. Ich sende zum Test die Zeichenfolge „HALLO WORLD“:

```
Advanced Research Projects Agencies
Terminal Client Connection


terminal > server_socket
terminal > Socket launched
terminal > Socket threaded
terminal > Socket launched as server 0
terminal > ARPANET is waiting for client...
terminal > Client connection accepted, client ID 1
Client sends: HALLO WORLD

Insert characters [max 256]: HELLO WORLD from Server
```

```
Advanced Research Projects Agencies
Terminal Server Connection


terminal > client_socket
terminal > Winsock launched
terminal > Socket created!
terminal > Connected with server 0. this: 1

Insert characters [max 256]: HALLO WORLD

Server response: HELLO WORLD from Server

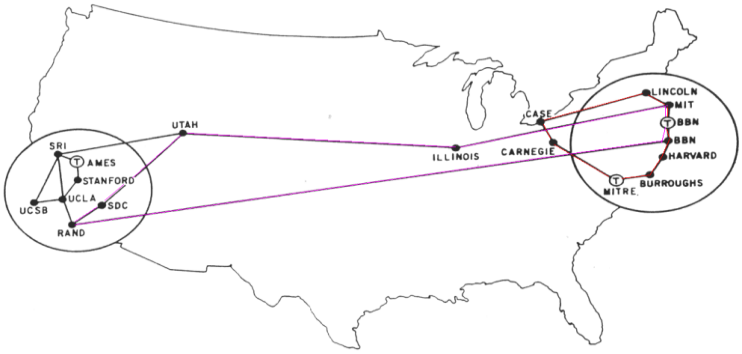
Insert characters [max 256]: _
```

Der String kommt beim Server an (links). Jetzt kann ich einen String zum Clienten Senden. Auch das funktioniert ohne Probleme. Und an diesem einfachen Versuch ist noch mal deutlich zu sehen: Das Senden funktioniert immer abwechselnd nach dem Motto: Reden kann immer nur ein Modulator („Only one device can send strings“[23])

[23] Quellcode Vornotizen von Oliver Hasten: „Sending mathodes“

Der Sendevorgang von „HALLO WORLD" dauerte 48 Sekunden. Es hätte damals ca. 38 Minuten gedauert, die Forum-Startseite von vonwolkenstein.de zu laden.

Es schien zur damaligen Zeit der Durchbruch geschafft. Informationen konnten ausgetauscht werden. Dieses Netzwerk begeisterte auch andere Universitäten an der Ostküste. Die Begeisterung führte zu einer unkontrollierten Ausbreitung bis 1971:

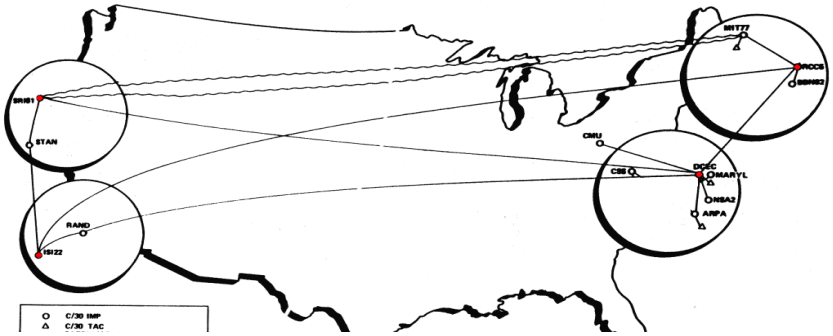


MAP 4 September 1971

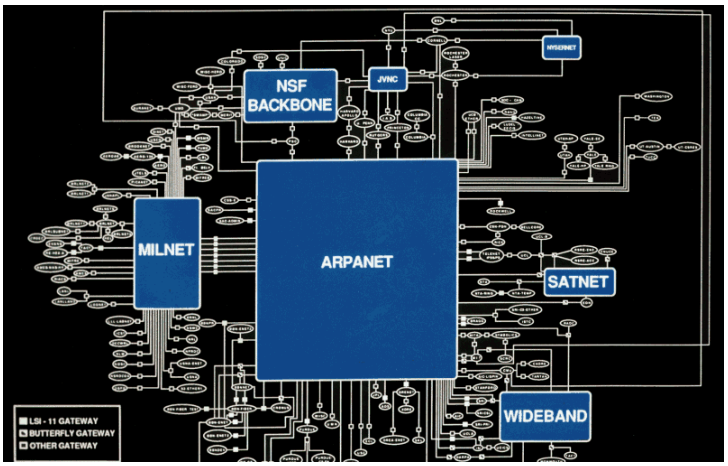
Doch irgend etwas stimmte nicht. Beim Betrachten der oberen Karte sollte doch etwas auffallen: „A centeral network and where is the center?"[24]
Gute Frage, da zur damaligen Zeit jegliche Normungen fehlten, stellte jede Universität an der Ost- und Westküste ihre eigene Verbindung zum ARPANET her. Es entstand ungewollt ein dezentrales Netzwerk. Die Datenwege waren unheimlich lang. Von Utah nach Illinois dauerte der Sendevorgang von „HALLO WORLD" schon über fünf Stunden.
Erneut mußten sich die Programmierer etwas einfallen lassen. Sie erstellten Knotenpunkte, welche zentrale Netzwerke darstellten. Die Verbindungen zwischen diesen Knoten wurde durch Modulatoren hergestellt: Vier Knotenpunkte in Amerika sorgten für eine teilweise zentrale Verbindung. Es ist allerdings falsch, das ARPANET mit dem heutigen Internet zu vergleichen. Das Internet ist nämlich hingegen ein vollständig genormtes zentrales Netzwerk. In Deutschland befindet sich dieses Zentrum in Frankfurt (DENIC). Alle Provider sind mit ihm verbunden.

[24] Licklider: „Reaction to the ARPANET" (Developers Zone)

ARPANET Geographic Map, 31 October 1989



Das Militär selbst hatte im Prinzip mit dem ARPANET nichts zu tun. Erst 1989 wurde das MIL-net für militärische Zwecke entwickelt, das ARPAnet blieb für zivile Bereiche nutzbar. Das MIL-net wurde an das ARPANET angeschlossen. Aufgrund der umgekehrten Sendereihenfolge von Strings im MIL-net (Bei „HALLO WORLD" wurde zuerst das ‚D' gesendet.) benötigten die Programmierer ein Gateway für die Umwandlung der Daten zwischen ARPANET und MIL-net. Ein Gateway ist ein Computer, welcher die Sendereihenfolge umkehrt.



Das MIL-net muß demnach getrennt vom eigentlichen Netzwerk ARPANET betrachtet werden.

Das ARPANET war also eine semidezentrales (zum Teil dezentrale) Netzwerktechnologie, welche durch die Projektorganisation **A**vanced **R**esearch **P**rojects **A**gencies entwickelt wurde. Diese Projektorganisation entwickelte es als Hilfsmittel für den Datenaustausch zwischen Universitäten in Kalifornien und der Universität in Utah. Die Entwicklung begann 1961 mit einem dezentralen Netzwerk. Später versuchten Oliver Hasten und Steve Carr das bestehende Netzwerk in ein zentrales Netzwerk umzuwandeln. Sie benutzten auf der Hardwareebene Modulatoren und das bestehende Telefonnetz. Auf der Softwareebene kamen Sockets (virtuelle Kommunikationssendepunkte) zum Einsatz. Die zentrale

Netzwerkidee scheiterte im großen Rahmen und das ARPANET wurde 1989 semidezentral. Ein solches Netzwerk ist mit dem heutigen zentralen Internet nicht zu vergleichen. Neben dem ARPANET schlossen sich weitere Projektorganisationen zu Netzen zusammen. Ein Beispiel hierfür ist das Military Network, welches vom US-Verteidigungsministerium in Auftrag gegeben wurde.
Bleibt nur noch die Frage, was eigentlich aus ARPA wurde. Die Gemeinschaft entwickelte neben und mit dem ARPANET die Programmiersprache B++ und wurde später aus Kostengründen aufgelöst.
Ohne Zweifel war ihre Arbeit von größter Bedeutung für die Entwicklung weiterer Programmiersprachen und Netzwerke. Denn: "The ARPANET was only one attempt. Hallo world, learn and try it better."[25]
Haben wirklich zukünftige Programmierer etwas gelernt? Darüber läßt sich streiten. Aber einen Punkt haben sie mit Sicherheit beachtet:
Das heutige Internet ist zentral.

Erklärungen

[i] University of California, Santa Barbara
[ii] *In einer Programmiersprache: Aneinanderreihung von Zeichen, die terminiert wurde*
[iii] Anhang: Zeichensatz
[iv] Beendigung eines Strings
[v] Datentyp in den Programmiersprachen BCPL, B, C, C++, C#, welcher die Anfangsadresse eines Speicherabschnitts speichern kann. Dieser Speicherabschnitt enthält Zeichen.

[25] Oliver Hasten: „A Stepp further", Zukunftsreport des Entwicklers