

Projekt:



X-Terminals

an der Fachhochschule Bonn-Rhein-Sieg

- Lars Ehrhardt (le)
- Markus Bläser (mb)

(Stand: 15.01.2002)

Gliederung:

- Kurzbeschreibung
- Bisherige Situation
- Zukünftige Situation
- Implementierungsdetails
- Zeitplanung
- Dokumentation
- Links zum Projekt



Kurzbeschreibung:

Ausgangsbasis:

Server	Workstations
1x Sun Ultra 5, 270 MHz UltraSPARC III Prozessor 56 MB RAM 30 GB Festplatte 24x-CD-ROM Netzwerkschnittstelle: 10/100BaseT Betriebssystem: Solaris 2.7	2x Sun JavaStation Iiep, 100MHz microSPARC Iiep 32 MB RAM - - Netzwerkschnittstelle: 10/100BaseT

Anforderung:

Nutzung der Workstations als Terminals, wünschenswert wäre eine Nutzung mit einer graphischen Oberfläche, falls machbar.

Kostenrahmen:

0,- DM :-)

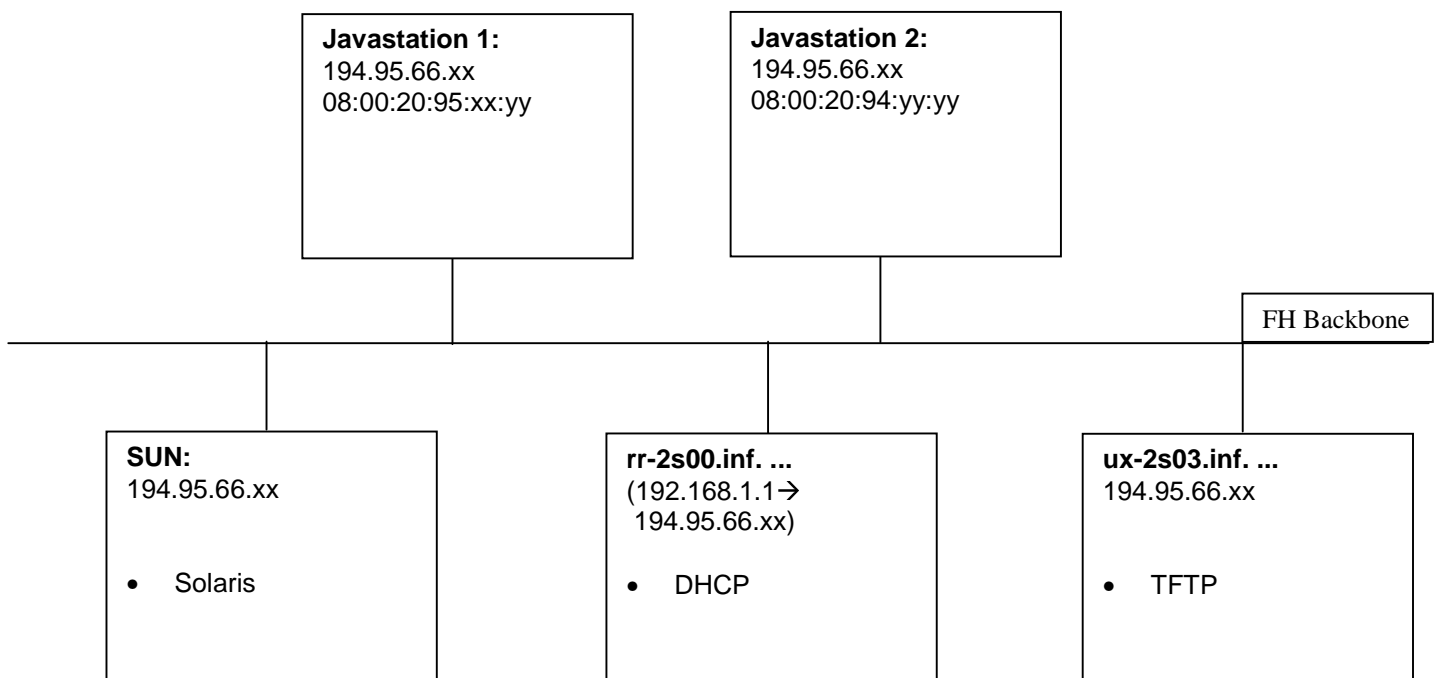
Zeitraumen:

Realisierung bis Ende Januar 2002



Bisherige Situation:

Momentaner Aufbau:



Erläuterung zur bisherigen Situation:

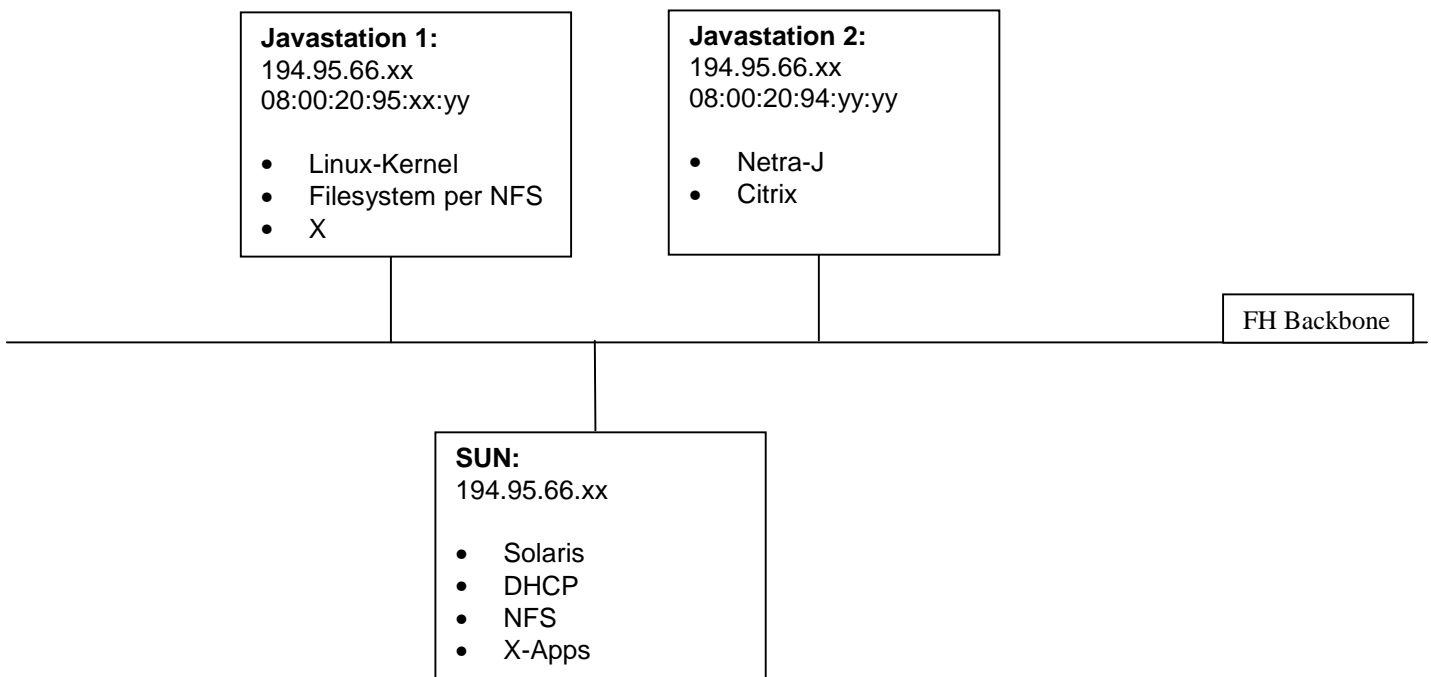
Die Javastations bekommen nach dem Selbsttest per DHCP eine IP-Adresse zugewiesen. Derzeit wird die Adressvergabe vom Server 192.168.1.1 (194.95.66.xx) geregelt. Der DHCP-Server liefert außerdem eine IP-Adresse eines TFTP-Servers (194.95.66.xx), sowie den Namen eines Bootimages zurück. Dieses Bootimage existiert aber nicht, so dass die Javastations nicht weiter booten!

Auf der Sun Ultrastation 5 ist neben einem DHCP-Server für die Javastations auch ein vollständiges Bootimage eingerichtet. Der DHCP-Server auf der SUN wird jedoch nicht von den Javastations benutzt, da der FH-weite DHCP-Server schneller antwortet.



Zukünftige Situation:

Geplanter Aufbau:



Erläuterung zur zukünftigen Situation:

Die Javastation 2 bekommt nach dem Selbsttest per DHCP eine IP-Adresse von der Sun Ultrastation zugewiesen und bootet mit einem Bootimage, welches im wesentlichen aus dem Linux-Kernel besteht. Nach dem Boot des Linux-Kernels wird per NFS ein auf der Sun Ultrastation bereitgestelltes Dateisystem benutzt. Die Javastation startet dann einen X-Server. Die Anwendungen (X-Clients) werden auf der Sun Ultrastation 5 gestartet und stellen ihr GUI auf dem X-Server der Javastation dar.



Implementierungsdetails:

Server:

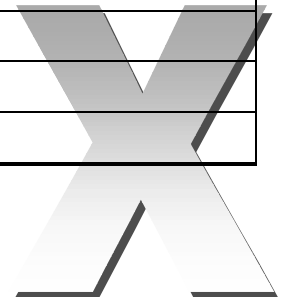
- Deaktivierung des bisherigen DHCP Servers für die 2 Javastations
- Installation und Konfiguration von DHCP auf dem Server
- Installation eines Bootimages (Linux-Kernel) auf der Sun Ultra für die Javastation
- Installation eines Dateisystems auf der Sun Ultra, welches per NFS für die JS exportiert wird
- Freigabe des X-Servers

Clients:

- Installation und Konfiguration eines X-Servers für die JS
- Konfiguration als X-Clients

Zeitplanung:

Datum	Projektziel
14.11.01	Präsentation
28.11.01	Installation von PROLL/Linux-Kernel/Boot des Linux-Kernels
05.12.01	Lauffähiges Linux-System einrichten
12.12.01	Lauffähiges Linux System einrichten/X-Konfiguration
19.12.01	X-Konfiguration
09.01.02	Testen/Optimieren
16.01.02	Testen/Optimieren
23.01.02	Präsentation



Dokumentation:

0. Abschaltung des DHCP-Servers [le]

Bisher war es so, dass die Javastation ihre IP-Konfiguration vom zentralen FH-DHCP-Server erhalten hat. Da auf der Sun Sparcstation bereits eine lauffähige (und auch für die Javastation angepasste) DHCP-Konfiguration enthalten war, wurde der zentrale DHCP-Server so umkonfiguriert, dass er auf Anfragen der Javastation nicht mehr reagiert. Der zentrale FH-DHCP-Server läuft auf einem Rechner mit Linux. Folgende Zeile wurde innerhalb der Subnet-Deklaration auf diesem Rechner hinzugefügt:

```
host js-2n01 { hardware ethernet 08:00:20:94:e1:ee; deny booting;}
```

1. Einrichten von PROLL/Anpassung des DHCP-Servers [le]

PROLL (PRom Linux Loader) ist ein Ersatz für das hauseigene Javastation-PROM. Es ist zum Laden und Initialisieren des Linux-Kernels zwingend notwendig. Folgende Konfigurationseinträge sind in der Datei /var/dhcp/dhcptab hinzugefügt worden:

```
(1) www<tab> m <tab>
:Timeserv=195.95.66.1:DNSServ=194.95.66.xx:DNSdmain=netlab.inf.fh-bonn-
rhein-sieg.de:NISdmain=inf.fh-bonn-rhein-sieg.de:NISservs=194.95.66.xx:
(2) SUNW.Linux.Krups <tab> m <tab>
:Include=www:Rootpath=/tftpboot:BootFile=C25F426C:BootSrvA=194.95.66.xx:TFT
PsrvN=194.95.66.xx:
(3) js-2n01 <tab> m <tab> :LeaseTim=-1:Include=SUNW.Linux.Krups:
```

Wichtig sind hier insbesondere die Einträge Rootpath=/tftpboot und Bootfile=C25F426C. Der Rootpath gibt das Basisverzeichnis für das Bootfile an. Der Dateiname des Bootfiles ergibt sich aus der Konvertierung der IP-Adresse in das Hex-Format. Also das Bootfile C25F426C steht für die IP-Adresse 194.95.66.xx. Weitere Informationen zum Format der Datei dhcptab findet man in der dazugehörigen dhcptab man-page¹.

In das Verzeichnis /tftpboot müssen einige Dateien kopiert, sowie Links erstellt werden.

1. Das PROLL-Image. Dieses bekommt man z.B. auf der Seite des Kernel-Hackers Peter Zaitcev². Zum Zeitpunkt der Erstellung dieses Dokumentes war die Version 1.3 aktuell.
2. Einen Linux-Kernel. Fertig kompilierte Linux-Kernel findet man auf der JavaStation-HowTo Seite³.
3. Ein symbolischer Link proll.krups.ID13 → HEXIP
4. Ein symbolischer Link vmlinux.aout → HEXIP.PROL
5. Ein symbolischer Link Pfad_des_NFS-Linuxfilesystem → IP-Adresse des Clients

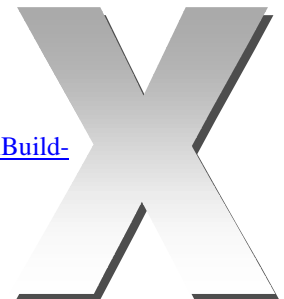
Das Verzeichnis /tftpboot sieht jetzt folgendermaßen aus:

```
lrwxrwxrwx 1 netlab other 37 Nov 28 13:59 194.95.66.xx-> /export/linux-fs
lrwxrwxrwx 1 netlab staff 11 Nov 28 13:59 C25F426C -> proll.krups
lrwxrwxrwx 1 netlab other 12 Nov 28 13:59 C25F426C.PROL -> vmlinux.aout
lrwxrwxrwx 1 netlab other 8 Nov 28 13:59 C25F426C.SUN4M -> C25F426C
lrwxrwxrwx 1 netlab other 16 Nov 28 13:59 proll.krups -> proll.krups.ID13
-rw-rw-r-- 1 netlab other 89680 Nov 26 00:18 proll.krups.ID13
-rw-r--r-- 1 netlab other 1577056 Nov 26 00:18 vmlinux.aout
```

¹ <http://www-biocomp.doit.wisc.edu/cgi-bin/man-cgi?dhcptab+4>

² <http://people.redhat.com/zaitcev/linux/>

³ <http://www.linuxdoc.org/HOWTO/JavaStation-HOWTO/kernelbuildchapter.html#KernelBuild-KernelSamplesSection>



Außerdem war zum Booten des PROLL-Image eine NFS-Freigabe auf der Sun Sparcstation notwendig. In der Datei /etc/dfs/sharetab ist zum funktionierenden Betrieb folgende Zeile hinzugefügt worden:

```
/tftpboot          -   nfs      ro=js-2n01          (TFTPBoot-Freigabe)
```

→ Der kompilierte Kernel 2.4.2 auf der HowTo-Seite hat zwar erfolgreich gebootet, allerdings mit dem Nebeneffekt, dass der DHCP-Server nach der BootP-Konfiguration die vergebene IP-Adresse als ungültig markiert hat. Somit konnte die Javastation nur einmal gebootet werden. In der Datei /var/dhcp/194.95.66.xx findet sich dann folgender Eintrag:

```
01080020952647  07 194.95.66.xx  194.95.66.xx  -1  js-2n01 #JavaStation
```

Ändert man den Eintrag der zweiten Spalte (07) auf 03, so wird die IP-Adresse wieder vergeben!

2. Einrichten eines Dateisystems [le]

Zum vollständigen Betrieb der Javastation unter Linux ist außerdem ein Dateisystem notwendig. Zum ersten Probieren eignet sich dazu ein fertiger Snapshot eines funktionierenden Dateisystems, welches man z.B. auf der UltraLinux.org⁴-Seite bekommt. Die Archive wurden in das Verzeichnis /export/linux-fs/ entpackt. Im Schritt 1.5 wurde bereits ein symbolischer Link von der IP-Adresse zu diesem Verzeichnis erstellt, so dass der Linux-Kernel später auf dieses Verzeichnis per NFS zugreifen kann. Dieses Verzeichnis muss ebenfalls über NFS freigegeben werden. In der Datei /etc/dfs/sharetab wurde dazu folgende Zeile hinzugefügt werden:

```
/export/linux-fs  -   nfs      rw=js-2n01,root=js-2n01  Home Directory
```

- Zum Erfolgreichen Einbinden eines Dateisystems per NFS ist die Option no_root_squash zu setzen. Unter Solaris ist das Equivalent dazu die Option root=clientname. Die Option no_root_squash bewirkt, dass der root der Clients die Dateien auf dem Server auch mit root-Rechten ablegen kann. Ohne diesen Eintrag würden alle Dateien, die der root auf dem Client lesen oder schreiben will, mit dem user nobody und der Gruppe nogroup behandelt werden, was schon während des Bootens zu Problemen mit den Rechten führen wird. (Normalerweise erhält man nach dem Booten nicht einmal einen Prompt, um sich auf dem System einzuloggen, da die Datei /var/run/utmp nicht mit den notwendigen Rechten ausgestattet ist.)
- Das Entpacken des Snapshots muss unter root Rechten erfolgen, da ansonsten nicht die korrekten Rechte und Besitzer der Dateien extrahiert werden. Außerdem kann nur der root User die Dateien unterhalb von /dev entpacken.

Nachdem das Dateisystem entpackt wurde, sind in einigen Dateien noch Änderungen notwendig. Die wichtigsten Änderungen betreffen die Dateien /etc/fstab, wo folgender Eintrag vorgenommen wurde:

```
194.95.66.xx:/export/linux-fs  /          nfs defaults  1 1
```

und die Datei /etc/sysconfig/network-scripts/ifcfg-eth0 mit den Angaben.

```
DEVICE=eth0
IPADDR=194.95.66.xx
NETMASK=255.255.255.0
NETWORK=194.95.66.0
BROADCAST=194.95.66.255
ONBOOT=yes
```

Angepaßt wurden außerdem die Dateien /etc/sysconfig/network und /etc/hosts mit den entsprechenden Netzwerkeinträgen.

⁴ <http://www.ultralinux.org/js/>



3. Zusammenfassung [le]

Nach dem Einschalten einer Javastation passiert nach diesen Änderungen folgendes:

- 1.) Es wird ein Broadcast ans Netzwerk gesendet, worauf sich der DHCP-Server der Sun Sparcstation meldet. Der DHCP-Server liefert u.a. eine IP-Adresse für die Javastation und den Namen eines Bootimages, welches mittels NFS übertragen wird.
- 2.) Nach dem Laden des Bootimages sucht PROLL nach der Datei HEXIP.prol. Diese Datei wird per TFTP übertragen.
- 3.) Die Datei HEXIP.prol ist ein Link auf das Kernel-Image, d.h. in Schritt 2 wird das Linux-Kernel-Image übertragen und mit speziellen Bootoptionen (Übergabe des NFS-Root-Directories, etc.) gestartet.
- 4.) Der Linux-Kernel übernimmt das System. Es werden Gerätetreiber initialisiert, nochmals mittels BOOTP Daten übertragen und per NFS wird das Dateisystem eingebunden.
- 5.) Das Programm /sbin/init übernimmt das Starten wichtiger Skripts, bindet die Laufwerke ein, etc. Genauere Informationen hierzu findet man im From-PowerUp-To-Bash-Prompt-HOWTO⁵!

Anmerkungen:

- ◆ *Der Snapshot des Dateisystems ist ziemlich alt, aus Sicherheits- und Wartungsgründen sollte man überlegen, ob man nicht Sparc-Debian installiert*

Anpassungen:

- ◆ *Für SSH mussten noch zwei Anpassungen gemacht werden:*

1. Erzeugung eines Host-Keys

2. Die Datei /etc/pam.d/sshd existiert nicht,

bash-2.03\$ cat sshd

##PAM-1.0

auth required /lib/security/pam_pwdb.so shadow nodelay

auth required /lib/security/pam_nologin.so

account required /lib/security/pam_pwdb.so

password required /lib/security/pam_cracklib.so

password required /lib/security/pam_pwdb.so shadow nullok use_authok

session required /lib/security/pam_pwdb.so

session required /lib/security/pam_limits.so

- ◆ *Da der Rechner über eine offizielle IP-Adresse im Internet ansprechbar ist, wurden die Dienste telnet, ftp, etc. in der Datei /etc/inetd.conf deaktiviert.*

⁵ <http://www.linuxdoc.org/HOWTO/From-PowerUp-To-Bash-Prompt-HOWTO.html>



4. Installation und Konfiguration von X [mb]

4.1 X-Konzept⁶

4.1.1 X-Überblick

Zunächst eine Namensklärung. Das Window-System, das die graphische Oberfläche unserer Rechner in der Informatikhalle bereitstellt, heißt X oder X11, nicht *X-Windows* (Originaltext des MIT: *It is a window system called X, not a system called X Windows*)! Die Bezeichnung X-Windows ist fälschlicherweise sehr verbreitet und wird leider auch im größten Teil der UNIX-Literatur verwendet. Im vorliegenden Handbuch soll dagegen der korrekte Name der Software, X oder X11, benutzt werden.

X ist eine Entwicklung des MIT (Massachusetts Institute of Technology) im Rahmen des *Project Athena*. Es handelt sich um eine Benutzeroberfläche, die es erleichtern soll, die Fähigkeiten von UNIX-Workstations voll zu nutzen. Da UNIX ein Multitasking-System ist, -- es kann also mehrere Programme gleichzeitig abarbeiten -- sollten die *ablaufenden Programme* (im folgenden als Prozesse bezeichnet) auch übersichtlich angeordnet werden können. Jedem Prozess wird deshalb die Möglichkeit gegeben, Bildschirmausgaben auf einem Teil des Monitors anzuzeigen. Der Benutzer selbst kann darüber entscheiden, ob er die Information auch wirklich angezeigt haben will. X wird kostenlos im Quellcode vom MIT abgegeben.

Die dieser Beschreibung zugrundeliegenden Versionen sind X Release 11 Version 4 und 5.

X ist im wesentlichen in zwei Teile gegliedert, die miteinander durch ein festgelegtes Protokoll kommunizieren. Der eine Teil ist der *X-Server*, die anderen Teile sind die *X-Clients*.

4.1.2 X-Server

Der X-Server ist das Programm, das alle Bildschirmausgaben übernimmt und alle Eingaben von der Tastatur und der Maus verarbeitet. Daher ist ein Teil des X-Servers sehr an die Hardware des Rechners gebunden (Farb- oder Schwarzweiß-Bildschirm, Art der Tastatur, Anzahl der Maustasten, Bildschirmgröße ...). Ein Programm, das etwas auf dem Bildschirm ausgeben will, schickt einen diesbezüglichen Auftrag an den X-Server, der daraufhin eine Linie zeichnet, einen Text ausgibt oder tut, was immer das Programm von ihm verlangt. In der anderen Richtung gibt der X-Server Meldungen an die X-Clients, wann immer der Benutzer eine Eingabe getätigt hat, sei es das Bewegen der Maus, das Drücken einer Maustaste oder eine Eingabe über die Tastatur. Die Programme können dann entscheiden, was sie mit dieser Eingabe anfangen und wie (oder ob überhaupt) sie darauf reagieren. Vorteil dieser Konfiguration ist, dass nur der X-Server über die Möglichkeiten der vorhandenen Hardware informiert sein muß. Die Clients können diese Information vom Server erfragen, wenn sie sie brauchen, müssen sich ansonsten aber nicht darum kümmern.

4.1.3 X-Client

Jedes Programm, das auf einem X-Bildschirm ein Fenster darstellen will, ist ein X-Client. Die Bezeichnung *Client*, zu Deutsch *Kunde* kommt daher, dass der X-Client dem Server gegenüber als Kunde auftritt: Er bittet den Server, gewisse Aufgaben (eben das Zeichnen des Fensters) für ihn zu übernehmen. X-Clients, die auf Eurer Benutzeroberfläche standardmäßig gestartet werden, sind zum Beispiel xterm, xrn usw, aber auch der Window-Manager ctwm.

⁶ Vgl Unix-Einführung der Fakultät für Informatik der TU München



4.2 X-Protokoll

4.2.1 Einführung⁷

Die Verbindung zwischen Client und Server läuft über das sogenannte *X-Protokoll* ab. Dieses Protokoll kann entweder innerhalb eines Rechners oder über eine Netzwerkverbindung abgewickelt werden. Daraus folgt, dass X-Client und X-Server nicht zwangsläufig auf der gleichen Maschine laufen müssen, Voraussetzung ist lediglich, dass zwischen beiden eine Netzverbindung existiert. Der X-Client kann beim Start festlegen, auf welchem Server er seine Fenster aufbauen will. Voraussetzung dafür ist jedoch, dass ihm der gewünschte Server auch die Berechtigung erteilt. Dies kann vom Benutzer des Terminals, das durch den Server verwaltet wird, festgelegt werden.

4.2.2 Message Types⁸

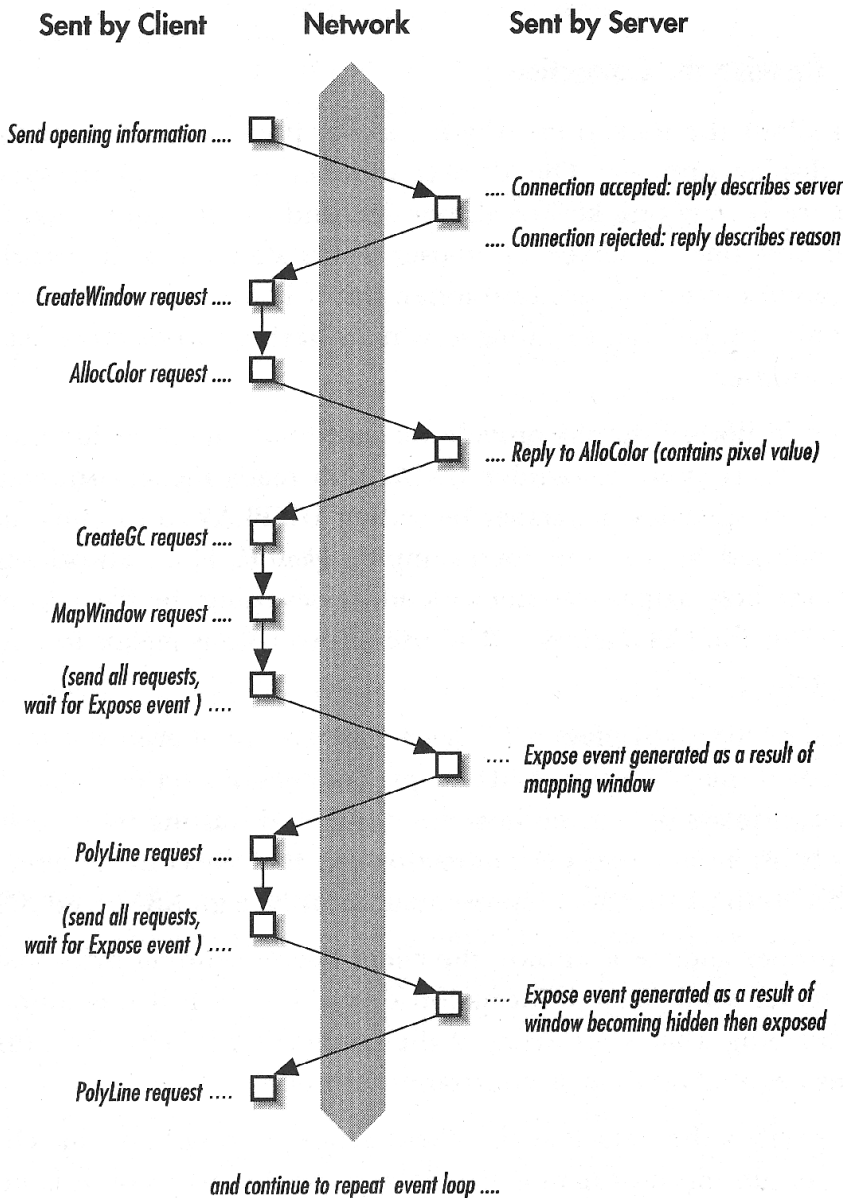
Im X-Protokoll sind 4 Typen spezifiziert:

- **Request**
Ein Request wird von einem Client generiert und an einen Server gesendet. Ein Request kann verschiedenste Informationen übertragen. Einige Beispiele:
Spezifikationen zum zeichnen einer Linie, verändern der Farbe einer Zelle in einem colormap oder anfrage der aktuellen Fenstergröße. Die Länge eines Request beträgt immer ein vielfaches von 4 Bytes.
- **Reply**
Ein Reply wird vom Server zum Client als antwort auf einen Request gesendet. Aber nicht alle Requests werden mit einem Reply beantwortet, nur diejenigen, die nach Informationen fragen. Replies haben eine Mindestlänge von 32 Bytes und sind ebenfalls immer vielfache von 4 Bytes.
- **Event**
Ein Event wird vom Server zum Client gesendet und beinhaltet Informationen über device-actions oder Seiteneffekte vorangegangener Requests. Die Informationen eines Events können sehr unterschiedlich sein, da der Client hauptsächlich mittels Events seine Informationen erhält. Alle Events werden in einer 32-Bit Struktur gespeichert um das Verarbeiten zu vereinfachen.
- **Error**
Ein Error ähnlich einem Event, aber er wird auf anderer Weise Verarbeitet. Error werden zu eine Fehler-Behandlungs-Routine gesendet, welche sich in der client-programming-library befindet. Errors haben die gleiche Größe wie Events.

⁷ Vgl Unix-Einführung der Fakultät für Informatik der TU München



4.3 Sample X-Session⁹



- Client öffnet Verbindung zum Server und übersendet Information über sich
- Server sendet seine Beschreibung zurück oder verweigert die Verbindung
- Client sendet Anfrage zum Öffnen eines Fensters
- Client möchte Farbe allokiieren
- Server antwortet mit einer Beschreibung der allokierten Farbe
- Client möchte grafischen Kontext wissen (z.B. für spätere Zeichenoperationen)
- Client möchte erzeugtes Fenster auf dem Bildschirm darstellen
- Client wartet auf **expose** Ereignis.
- Server sendet **expose** Ereignis (Fenster wurde dargestellt)
- Client möchte eine Linie zeichnen
- Loop back um auf ein **expose** Ereignis zu warten

⁹ X Protokoll Reference Manual, volume one



Web-Seiten zum Thema:

Zentrale Dokumente:

<http://www.linuxdoc.org/HOWTO/JavaStation-HOWTO/>

<http://people.redhat.com/zaitcev/linux/>

<http://www.ultralinux.org/js/> (Javastation Images)

Link zu Ultralinux (gute Übersicht) über Linux auf der Sparc:

<http://www.ultralinux.org/>

<http://www.linuxdoc.org/HOWTO/SPARC-HOWTO.html>

<http://lists2.suse.com/archive/suse-sparc/>

Verwendung alter Rechner als X-Terminals:

<http://www.menet.umn.edu/~kaszeta/unix/xterminal/>

<http://www.ibiblio.org/pub/Linux/docs/HOWTO/unmaintained/mini/Xterminal>

<http://www.linuxdoc.org/HOWTO/mini/Remote-X-Apps-9.html>

<http://pangea.stanford.edu/computerinfo/unix/xterminal/>

Solaris2 FAQ:

<http://www.science.uva.nl/pub/solaris/solaris2/>

Links zum Linux Terminal Server Project:

<http://www.ltsp.org/index.php>

<http://www.newspapersystems.com/top/press/workstations.html>

Information über den micro-Sparc:

<http://www.sun.com/microelectronics/microSPARC-IIep/>

Informationen über JavaStation und Nachfolgeprojekte:

[http://www.sun.com/javastation/;\\$sessionid\\$UJDY0IAAAD0XLAMTA1FU4GQ](http://www.sun.com/javastation/;$sessionid$UJDY0IAAAD0XLAMTA1FU4GQ)

