akadia Information Technology

Sendmail

Survival Guide

Author: Martin Zahn

Akadia AG
Information Technology
Zieglerstrasse 34
CH-3007 Bern
Tel 031 385 30 30
Fax 031 385 30 34
E-Mail info@akadia.com
Web http://www.akadia.com

1. Electronic Mail

Die EDV-Landschaft der neunziger Jahre ist geprägt durch die Vernetzung einzelner Rechner zu Computersystemen. Dies nicht nur im lokalen Bereich (LAN's), sondern auch über die Landesgrenzen hinweg mittels interkontinentalen Computernetzen (WAN's). Am weltumspannenden *Internet* sind heute mehr als 10 Mio. Rechner angeschlossen. Viele Rechnersysteme, verteilt über die ganze Welt, bieten Netzdienste wie *anonymous ftp, gopher*, und mehr an. Mehr und mehr steht aktuelles Wissen nur noch in digitaler Form zur Verfügung, das vielen Instituten und Firmen meist gratis zur Verfügung steht. Erkannte Fehler und Verbesserungen in Computerprogrammen werden durch die Hersteller mittels *Patches* rasch an die Endanwender verteilt. Die «elektronische» Post (EMail) gewinnt dabei mehr und mehr an Bedeutung. Mitarbeiterinnen und Mitarbeiter einer Firma oder Institution sollen von der gewohnten Arbeitsumgebung, bzw. Benutzeroberfläche aus per EMail erreichbar sein.

EMail stellt ein modernes Mittel zur Kommunikation und zur Integration dezentraler Abteilungen dar. Informationen können auf diese weise rasch zwischen Projektgruppen ausgetauscht werden. Durch den Zugang zu internationalen Institutionen stehen umfangreiche Wissens-Bibliotheken zur Verfügung. Dank der elektronischen Kommunikation mit Forschungsstellen in aller Welt können Synergien genutzt werden.

Zum heutigen Zeitpunkt besteht die Hauptanwendung von Electronic Mail darin, dem Partner eine selbstverfasste Nachricht zukommen zu lassen. Der Empfänger der Nachricht braucht zum Zeitpunkt der Übertragung nicht an seinem Computer zu sein um die EMail zu empfangen, diese wird in seiner persönlichen *Mailbox* zwischengespeichert. Noch fehlende internationale Standards verhindern es, dass bald auch allgemeine Daten wie *Audio, Video, Graphik* nebst normalem Text übertragen werden können. Ein Dokumentenaustausch ist jedoch bereits heute möglich, sofern beide Partner über das gleiche Textverarbeitungsprogramm verfügen. Dokumente werden am Versandort elektronisch codiert, dann über das Netz transportiert und am Bestimmungsort wieder decodiert. Ein internationaler Standard zur Codierung bildet EDI (Electronic Data Interchange Format) in Verbindung mit dem weltweit anerkannten EMail Protokoll X.400.

Bereits vor der breiten Einführung von X.400, die in den nächsten Jahren zu erwarten ist, bildeten sich sogenannte *Defacto Standards* basierend auf RFC-822 (Request for Comment), wie SMTP (Simple Mail Transfer Protocol) und MIME (Multi Media Internet Mail Extension) heraus. Die sehr grosse Verbreitung dieser Protokolle hatte zur Folge, dass unzählige EMail Applikationen (Mailtools) auf dieser Basis entwickelt wurden und heute im Einsatz stehen. Parallel dazu wurden Computerprogramme speziell zur Weiterleitung (Routing) von EMails entwickelt und perfektioniert, sogenannte MTA's (Message Transfer Agents).

Um EMail weltweit eindeutig zu adressieren, wird in RFC-822 die Domain-Name Adressierung verwendet. Jede Domain wird bei einem nationalen NIC (Network Information Center) registriert und von einem dezidierten Rechnersystem mittels DNS (Domain Name Server) verwaltet.

1.1 Aufbau einer EMail

Eine EMail besteht wie ein «konventioneller Brief» aus drei Teilen:

- Envelope (Absender- und Empfängeradressen)
- Message-Header (From:, To:, Cc)
- Message-Body (Briefinhalt)

Das Envelope ist meist für den Benutzer nicht direkt sichtbar. Es wird vom Message Transfer Agent (MTA) dazu benutzt die Mail an den Empfänger (Recipient) weiterzuleiten. Der Message-Header ist mit einem Briefkopf vergleichbar. Er wird vom jeweiligen User Agent (UA / Mailtool) meist dem Benutzer nicht vollumfänglich dargestellt. Meistens ist jedoch möglich mittels Mailtool spezifischen Optionen den gesamten Header sichtbar zu machen. Dies ist bei Zustellungsproblemen wichtig um den Transportweg der EMail zurückverfolgen zu können. Die nachstehende Skizze zeigt den Aufbau einer EMail mit den wichtigsten Header-Einträgen.

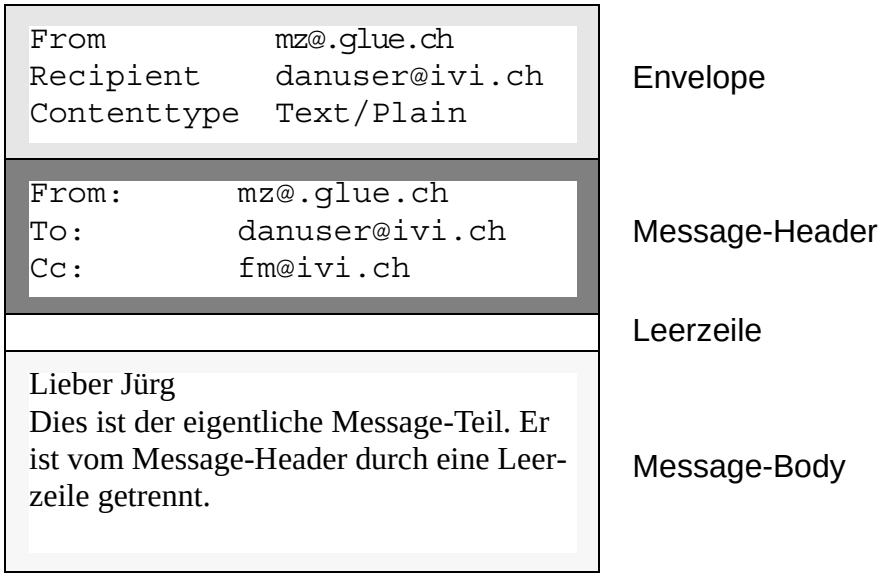


Abb. 4-1 Aufbau einer EMail gemäss RFC-822

Der Mail-Header ist vom eigentlichen Text durch eine Leerzeile getrennt.

Damit eine EMail erfolgreich von A nach B gelangen kann müssen verschiedene Voraussetzungen gewährleistet sein.

- Der Empfänger muss auf dem Zielsystem erreichbar sein. Das heisst, jeder Benutzer besitzt eine persönliche *Mailbox* auf einem ihm zugeteilten *Mailhost*.
- Die EMail-Adresse muss in einem bestimmten Format (RFC-822) vorliegen damit sie an den Bestimmungsort gelangen kann.

In einem weltweiten EMail-Verbund nimmt das Routing (Weiterleitung) der Meldung eine zentrale Bedeutung ein. Diese Aufgabe wird von vielen MTA's (Message Transfer Agent's) wahrgenommen. Das Grundprinzip der EMail Beförderung besteht im Stop and Forward Transport-Mechanismus. Das heisst, ein MTA muss den Zielhost des Empfängers nicht direkt kennen, sondern nur die Route zu diesem. Er sendet also das EMail an einen dem Ziel näher gelegenen MTA weiter. Dieser routet das EMail erneut einem MTA

weiter bis schliesslich der Zielhost des Empfänger erreicht wird. Die Anzahl «Sprünge» die dabei durchlaufen werden nennt man in der EMail Fachliteratur auch *Hops.*

Als Transport-Medium wird meistens das öffentliche Netz der staatlichen Organisationen (PTT's) genutzt. Damit eine EMail transportiert werden kann muss nebst einer physikalischen Verbindung eine Regel zwischen Sender und Empfänger bestehen wie die Meldung übermittelt werden kann. Dies ist die Aufgabe des EMail-Protokolls. Die am häufigsten eingesetzten Protokolle sind X.400 und RFC-822 (SMTP) sowie aus historischen Gründen UUCP das Unix-to-Unix-Copy Protokoll.

SMTP

EMail im lokalen Netzwerk (LAN) wird mit dem EMail Protokoll SMTP (Simple Mail Transfer Protocol) übermittelt. Dieses Protokoll ist direkt in den MTA *Sendmail* eingebaut. Das Envelope wird mittels SMTP-Kommandos *MAIL FROM:* und *RCPT TO*: von MTA zu MTA transferiert. Die beteiligten MTA's bauen eine Process zu Process Kommunikation, anschliessend erfolgt eine Synchronisation in Form einer Überprüfung der Sender und Empfänger-Adressen. Sind die Angaben korrekt, so wird die Email übermittelt. Kann der Empfänger nicht ermittelt werden, so wird der Sender sofort in Form einer Fehlermeldung darüber orientiert. SMTP ist das in einer UNIX Umgebung meist verbreitet Protokoll. Es basiert auf dem Netzwerk-Protokoll *IP* sowie dem Transport-Protokoll *TCP* allgemein bekannt unter TCP/IP.

UUCP

UUCP stammt aus der Zeit als es noch keine schnellen LAN-Verbindungen möglich waren sondern nur einfache Telefon-Leitungen zur Verfügung standen. UUCP ist also im Gegensatz zu SMTP kein Process-to-Process Protokoll sondern ein typischer Vertreter der Stop-and-Forward Methode. Die EMail gelangt über das öffentliche Telefonnetz zum nächsten MTA. Zusammen mit der EMail wird auch eine Datei mit der Instruktion wie die EMail weiterzuverarbeiten ist, übermittelt. Diese Datei wird vom MTA ausgewertet und entsprechend der Instruktion gelangt die Email zum nächsten MTA bis sie schliesslich auf dem Zielsystem ankommt. UUCP benutzt eine eigene Adressierung bekannt unter dem Begriff «Bang-Adressierung». Die Adressierung hat den erheblichen Nachteil, dass die exakte Route vom Host A zum Host B mit allen Zwischen-Hops angegeben werden muss. Diese Angaben sind heute jedoch kaum mehr möglich.

Beispiel: eniac!kanu!zephir!mzahn

Vom aktuellen Mailhost gelangt die Email zum Mailhop *eniac,* von dort weiter zum Mailhop *kanu* und von dort zum Zielhost *zephir* an den Benutzer *mzahn.* Als Transport Protokoll wird UUCP verwendet, zwischen allen Hosts.

In einer UNIX Umgebung wird als MTA sehr oft das Programm *Sendmail* eingesetzt, welches in der Lage ist sowohl RFC-822 Adressen wie auch UUCP Adressen zu verarbeiten.

X.400

X.400 ist der internationale Standard für Electronic Mail. Die Software muss vom jeweiligen UNIX Anbieter gekauft werden. X.400 basiert ebenfalls auf einer Process-to-Process Kommunikation. Der Standard UNIX MTA *Sendmail* ist auf den meisten UNIX Derivaten nicht in der Lage X.400 Mails zu verarbeiten. Als Ersatz für *Sendmail* kann der MTA *PP* (Public Domain Version vorhanden) verwendet werden, der einen wesentlich grösseren Funktionsumfang aufweist. Es ist jedoch auch möglich den Gateway-Dienst eines nationalen NIC's (Switch) für die Konvertierung RFC822 <-->X.400 zu nutzen.

1.2 EMail-Adressen

Zur Zeit sind weltweit drei Adressierungsarten bekannt: Die Domain-Adressierung nach RFC822 (Request for Comment), die UUCP Adressierung und die X.400 Adressierung. Sie unterscheiden sich in ihrem Aufbau erheblich. Um eine mit X.400 adressierte Mail an einen Rechner mit RFC-Adressierung zu senden sind Gateways nötig, welche eine Umsetzung der Adressierungsart vornehmen. Historisch bedingt, existiert noch die UUCP-Adressierung (Bang-Adressierung), welche jedoch von den meisten Rechnersystemen, welche RFC822 Adressen verarbeiten, auch verstanden werden.

1.2.1 UUCP-Adressierung («bang»-Adressierung)

Diese historische Adressierungsart zeichnet sich dadurch aus, dass der Weg, welche die EMail zum Empfänger einschlagen soll, bekannt sein muss. Jeder Knotenrechner muss in der Adresse angegeben werden.

Aufbau

```
host!user
host1!host2!user
host1!host2!hostn!user
```

Dabei ist zu beachte, dass host, host*n* **nicht** den lokalen Rechnernamen definiert. Reine «Bang»-Adressen wie `zephir!kanu!hueni` werden von **links nach rechts** gelesen.

Beispiele

`zephir!bj`	EMail an den Benutzer *bj* auf dem Rechner *zephir*.Als Transportprotokoll wird UUCP verwendet.
`bj@zephir.uucp`	Das gleiche in RFC-822 Domain-Schreibweise.
`zephir!kanu!hueni`	EMail via UUCP an Rechner *zephir* übergeben mit Bitte um Weiterleitung an den Benutzer *hueni* auf dem Rechner *kanu*.
`hueni%kanu@zephir`	EMail an den Benutzer *hueni* auf dem Rechner *kanu* erreichbar über den Rechner *zephir*. Diese Schreibweise entspricht in ihrer Funktion der Adresse *zephir!kanu!hueni*.Grundsätzlich darf nur ein «@» in einer Adresse vorkommen. Deshalb wird als zweites Zeichen «%» benutzt. Die EMail ginge also an *zephir* mit der Bitte um Weiterleitung an *hueni@kanu*. Transportprotokoll ist in diesem Fall SMTP (TCP/IP) und nicht UUCP.
`zephir!metz%iam.unibe.ch`	Dies ist eine RFC822/UUCP gemischte Adresse. EMail via Transportprotokoll UUCP an den Rechner *zephir* senden mit der Bitte um Weiterleitung an den Benutzer *metz* bekannt in der Domain *iam.unibe.ch.* Das Transportprotokoll zwischen dem Host zephir und iam.unibe.ch muss hier nicht spezifiziert werden. Diese Adressierung wird angewendet, wenn ein Rechner eine UUCP Verbindung zu einem Internet *Relayhost* besitzt, selbst also keine direkte Verbindung zum Internet hat. Der Relayhost *zephir* muss also in der Lage sein die EMail an den Benutzer *metz@iam.unibe.ch* zu senden.
`metz!zephir@iam.unibe.ch`	Die EMail wird via SMTP (TCP/IP) an die Domain *iam.unibe.ch* gesendet mit der Bitte um Weiterleitung an den Benutzer *metz* auf dem Rechner *zephir*. Die Domain *iam.unibe.ch* muss also in der Lage sein die Adresse *zephir!metz* aufzulösen. Das heisst, es muss eine UUCP Verbindung zum Rechner *zephir* bestehen.

1.2.2 RFC822 Adressierung

Die Domain-Adressierung wird im Internet verwendet. Der Weg der Email vom Sender zum Empfänger muss in der Adresse nicht mehr spezifiziert werden. Man verwendet bei dieser Adressierungsart international registrierte Domain-Namen. Dies sind logische Namen, sie haben mit der physikalischen Vernetzung der Rechner nichts gemeinsames. Die Adresse wird eingeteilt in Toplevel-, Domain-, und Subdomains. Die Toplevel-Domain's sind fest vorgegeben und können vom Benutzer nicht definiert werden. Domain's werden beim nationalen NIC (Network Information Center), z.B. Switch mittels *Domain Registration Form* registriert. Subdomain's können durch den lokalen Netzbetreiber (Bundesverwaltung, BVET, IVI) frei gewählt werden.

Aufbau

```
user@host
user@host.domain
user@[remote hosts ip-adress] (domain literal)
user@subdomain.domain.toplevel-domain
```

Beispiele

`mz@mzsun`	Benutzer *mz* im lokalen Netzwerk mit Mailbox auf Host *mzsun*. In der Regel befindet sich auch das HOME Directory des Benutzers auf diesem Rechner. Oft wird jedoch das zentrale Mail-Directory von allen Hosts am Netz mittels NFS importiert.
`charlie@hp.com`	Benutzer *charlie* bekannt in der Domain *hp.com.* In der Regel handelt es sich bei einer solchen Adresse um eine reine Domain-Adressierung. Das heisst, der verwaltende Mailhost der Domain *hp.com* wird in der Mail-Adresse nicht angegeben. Mittels DNS-Services wird der zuständige Mailhost der Domain *hp.com* dynamisch ermittelt. Dies hat den grossen Vorteil, dass die interne Struktur einer Domain nach aussen nicht bekannt sein muss, das heisst man muss nicht wissen, auf welchem Rechner sich die Mailbox des Benutzers *charlie* befindet.
`charlie@maui.hp.com`	Benutzer *charlie* in der Domain *hp.com* am Rechner *maui*. Solche Adressen sollten möglichst vermieden werden, da damit die interne Struktur nach aussen sichtbar gemacht wird. Änderungen von Rechnernamen dürfen in diesem Fall nicht mehr ohne weiteres durchgeführt werden.
`@glue.ch:fm@IVI.CH`	Benutzer *fm* in der Domain *ivi.ch* erreichbar über die Domain *glu*e.ch. In dieser Adresse ist speziell zu beachten, dass in Domain-Adressen Gross- und Kleinschreibung nicht unterschieden wird.
`fm%ivi.ch@glue.ch`	Benutzer *fm* in der Domain *ivi.ch* erreichbar über die Domain *glu*e.ch. Es ist eine andere Schreibweise für *@glue.ch:fm@IVI.CH.*
`zephir!mz@glue.ch`	Benutzer *mz* am Host *zephir* erreichbar über die Domain *glue.ch*. Hier handelt sich wieder um eine RFC-822 / UUCP gemischte Adressierung. Zum Unterschied betrachte man die Adresse *zephir!metz%iam.unibe.ch*: Die Email ginge an den Host *zephir* erreichbar über UUCP, mit der Bitte um Weiterleitung an den Benutzer *metz* in der Domain *iam.unibe.ch*

Bekannte Toplevel-Domains

com:	US-Firmen	gov:	US-Regierungseinrichtungen
mit:	US-Militär	org:	US-Sonstige Organisationen
edu:	US-Universitäten	net:	Netzwerkbetreiber
arpa:	Reste des ARPA-Netzes	uucp:	Historisches UUCP Netz
ch:	Ländercode für Schweiz		

1.2.3 X400 Adressierung

X400 (84) verwendet für das Konzept einer Adresse den Begriff O/R-Name (Originator / Recpient).

Aufbau:

Attributtyp	Wert	Bedeutung
Country-name	ch	Land des Empfängers. Benutzt werden ISO Ländercodes. Im Beispiel ist der Empfänger (Recipient) in der Schweiz.
ADMD	arcom	Administration Domain Management des Landes. Spezifiziert den offiziellen landesweiten X.400 Anbieter.
PRMD	switch	Private Domain Management Domain. Spezifiziert einen privaten X.400 Betreiber
Organistaion-name	ethz	Bezeichnung der Firma, Organisation
Organistaion-unit	inf	Bezeichnung einer Abteilung
Name	B. Jungi	Name, Vorname

Tabelle 4-1 X.400 EMail Adressierung

Die beiden Adressierungsarten RFC-822 / X.400 sind untereinander nicht kompatibel. Eine Umsetzung erfolgt meistens auf einem Adress-Konvertierungsrechner. Ein solcher Dienst wird meistens vom nationalen NIC wie Switch angeboten.

Umsetzung einer X.400 Adresse in eine RFC-822 Adresse

```
Name@organistaion-unit.organisation.pmd.admd.country
```

Beispiel:

```
Jungi.Beat@inf.ethz.switch.arcom.ch
```

oder

```
Jungi@inf.ethz.ch
```

1.2.4 RFC822 Adressierung

Die Domain-Adressierung wird im Internet verwendet. Der Weg der Email vom Sender zum Empfänger muss in der Adresse nicht mehr spezifiziert werden. Man verwendet bei dieser Adressierungsart international registrierte Domain-Namen. Dies sind logische Namen, sie haben mit der physikalischen Vernetzung der Rechner nichts gemeinsames. Die Adresse wird eingeteilt in Toplevel-, Domain-, und Subdomains. Die Toplevel-Domain's sind fest vorgegeben und können vom Benutzer nicht definiert werden. Domain's werden bei einer zuständigen Stelle (z.B. EUnet) mittels *Domain Registration Form* registriert. Subdomain's können durch den Benutzer frei gewählt werden.

Aufbau

```
user@host
user@host.domain
user@[remote hosts ip-adress] (domain literal)
user@subdomain.domain.toplevel-domain
```

Beispiele

`mz@mzsun`	(Lokale Adresse)
`charlie@hp.com`	(*com* = toplevel-domain, *hp* = domain)

Bekannte Toplevel-Domains

com:	US-Firmen	gov:	US-Regierungseinrichtungen
mit:	US-Militär	org:	US-Sonstige Organisationen
edu:	US-Universitäten	net:	Netzwerkbetreiber
arpa:	Reste des ARPA-Netzes	uucp:	Historisches UUCP Netz
ch:	Ländercode für Schweiz		

1.2.5 X400 Adressierung

X400 (84) verwendet für das Konzept einer Adresse den Begriff O/R-Name (Originator / Recpient).

Aufbau:

Attributtyp	Wert	Bedeutung
Country-name	ch	Originator / Recipient ist in der Schweiz
ADMD	arcom	Administration Domain der PTT
PRMD	switch	Betreiber des X.400 Netzes
Organistaion-name	ethz	Bezeichnung der Firma, Organisation
Organistaion-unit	inf	Bezeichnung einer Abteilung
Personal-Name	B. Jungi	Name, Vorname

Tabelle 4-2 X.400 Adressierung

Die beiden Adressierungsarten RFC-822 / X.400 sind untereinander nicht kompatibel. Eine Umsetzung erfolgt meistens auf einem Gateway-Rechner.

1.3 Ablauf des EMail Transfers

Damit eine EMail über verschiedenste Rechnersysteme und Netze hinweg transportiert werden können, muss eine klare Aufgabenteilung definiert werden. Eine erfolgreiche Übermittlung gliedert sich in die folgenden Bereiche: *Message Collection, Message Routing, Message Delivery und Message Retrieval*.

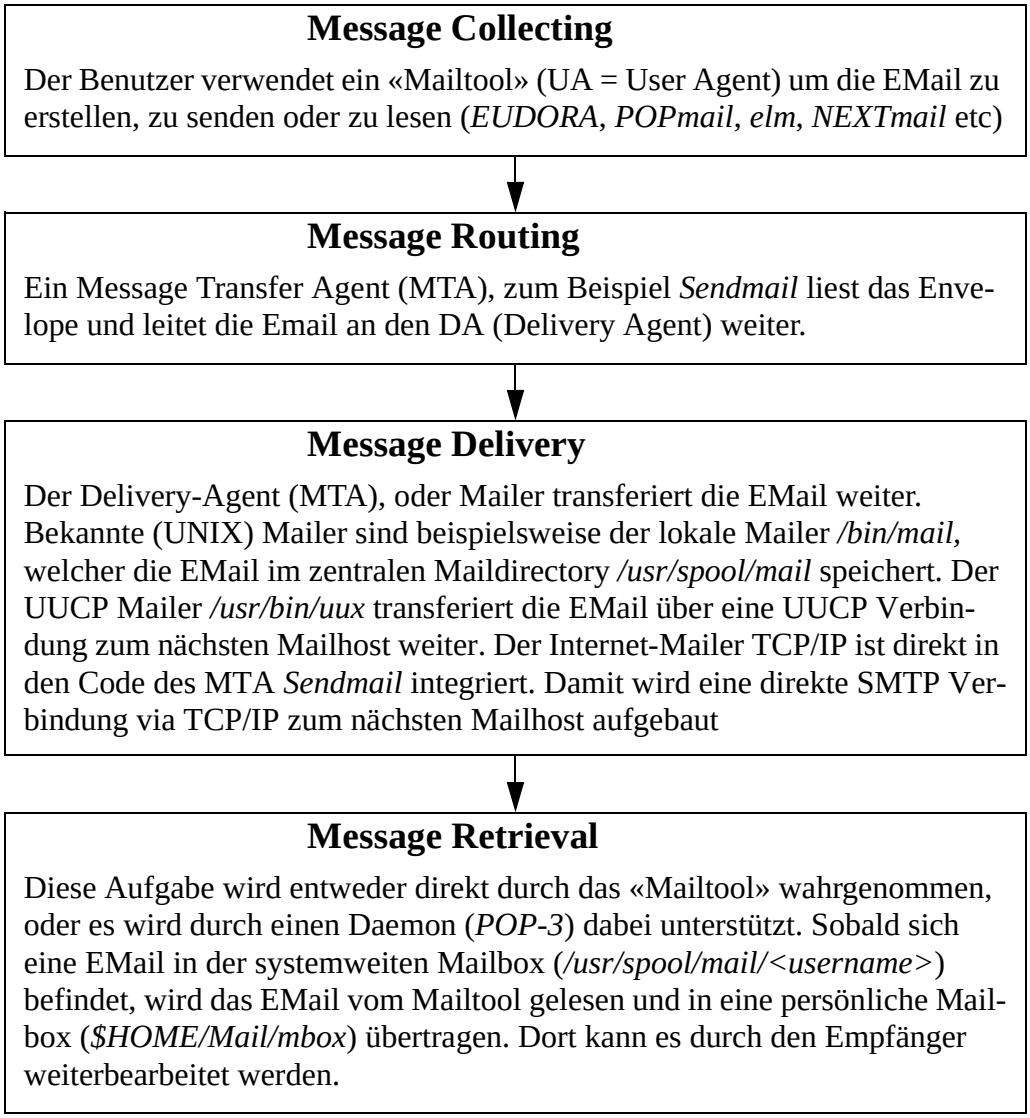


Abb. 4-2 Ablauf eines EMail-Transfers

Der Endanwender «sieht» von den *Bereichen Message-Routing* und *Message-Delivery* nichts (ausser wenn das EMail Gesamtsystem nicht korrekt funktioniert ...). Er «sieht» nur das ihm zur Verfügung stehende Mailtool wie EUDORA, POPMail, NEXTmail usw. Der EMail Verantwortliche muss jedoch die Zusammenhänge genaustens kennen, um einerseits den MTA (z.B. sendmail) korrekt zu konfigurieren und andererseits das EMail dem richtigen Mailer (Delivery Agent) zuzuleiten. All diese Aufgaben werden im Konfigurationsfile des MTA definiert. Entsprechend komplex gestaltet sich diese Aufgabe und sollte dem Spezialisten überlassen werden. Der zuverlässige und ununterbrochene Betrieb des zentralen Mailhosts ist von grösster Wichtigkeit damit EMail nicht verloren gehen oder Irrwege antreten. Grösste Aufmerksamkeit ist auch dem gefürchteten *Mail-Looping* zu schenken. *Mail-Looping* tritt oft im Zusammenhang mit der unkorrekten Vergabe von Aliasnamen auf. Eine EMail wird dann ähnlich einem Ping-Pong Spiel zwischen zwei Mailhosts hin und her gesendet, da Sender und Empfänger identisch sind.

1.4 EMail Hardware-Konfiguration

In einer EMail Umgebung unterscheidet man entsprechend ihrer Aufgabe verschiedene Rechnersysteme. Die Begriffe *Mailhost* und *Relayhost* sind typische Bezeichnungen in jeder Email-Umgebung und sollen deshalb näher umschrieben werden.

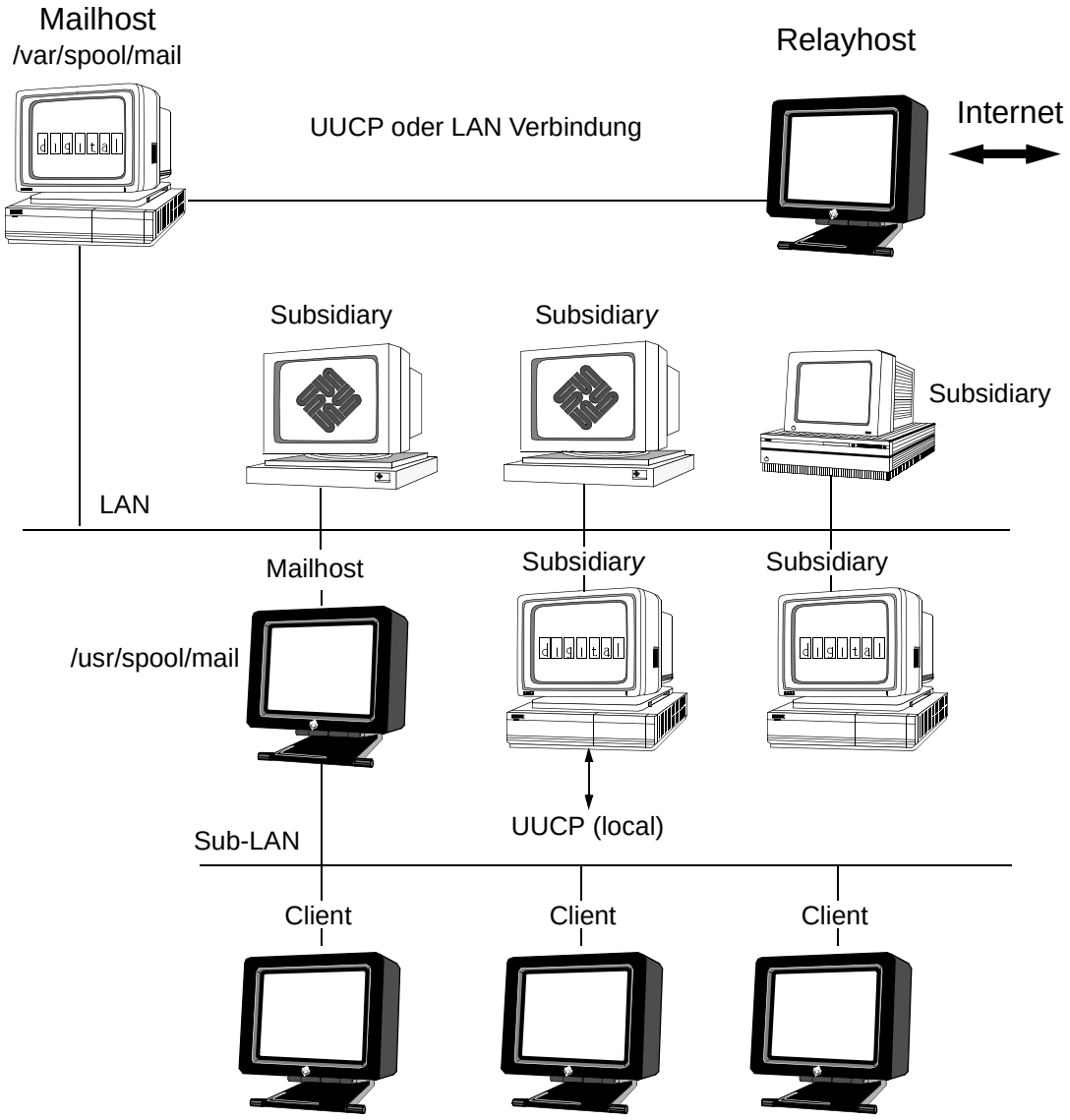


Abb. 4-3 EMail Hardware Konfiguration

1.4.1 Relayhost

Der *Relayhost* stellt die eigentliche Verbindung zur Aussenwelt her. Er «kennt» die Verbindung zu den nächst «höheren» Instanzen (z.B. Switch). Er nimmt sämtliche EMail der lokalen Domain in Empfang und sendet diese weiter. Der Relayhost betreibt seinen eigenen MTA (z.B Sendmail). Im EMail-Konzept «BVET» übernimmt der Rechner ivi.IVI.CH diese wichtige Aufgabe. Sofern der Relayhost am weltweiten Internet angeschlossen ist, muss eine Domain-Name Server (DNS / BIND) installiert werden um die lokale Domain am Internet zu verwalten. Der Relayhost «kennt» aufgrund seiner MTA Konfiguration wie ein lokaler Benutzer erreicht werden kann. Ebenso besteht vom Relayhost eine physikalische Verbindung zur nächst höheren Instanz (z.B Switch).

1.4.2 Mailhost

Der *Mailhost* verwaltet innerhalb der eigenen Domain eine Benutzergruppe (Zone) Die Aufteilung der Domain in sogenannte *Zonen* kann organisatorische oder auch räumliche Gründe haben. Jeder Benutzer einer Zone besitzt auf dem ihm zugewiesenen Mailhost eine zentrale Mailbox. Insbesondere bei räumlich und organisatorischen Einheiten empfiehlt es sich Zonen zu bilden.

Der *Mailhost* verwaltet das zentrale EMail Spooldirectory (*/var/spool/mail*). Sämtliche EMail der physikalisch zusammengehörenden Zone wird auf den Mailhost geleitet. Das Spooldirectory wird an alle *Subsidiary*-Maschinen exportiert, bzw. von diesen mit NFS gemountet. Der Mailhost betreibt immer einen eigenen MTA (*Sendmail*) mit einem eigenen Konfigurationsfile. EMail welche die eigene Domain verlassen werden an den *Relayhost* zur Weiterleitung gesendet. EMail innerhalb der eigenen Domain wird an den korrekten *Mailhost* der betreffenden Zone weitergeschickt. EMail mit einer lokalen Adressierung wird in die eigene System Mailbox (/usr/spool/mail) geleitet und dort vom jeweiligen Mailtool (EUDORA, POPMail, NEXTMail) dem Benutzer zugänglich gemacht.

1.4.3 Subsidiary Workstations (Standalone's)

Subsidiary Workstations mounten das zentrale Maildirectory ihres Zonen-Mailhosts. Sie betreiben einen eigenen MTA (Sendmail) in beschränktem Umfang. Das heisst, sie verwalten kein eigenes zentrales Maildirectory, sondern benutzen die zentrale Mailbox ihres Mailhosts. Subsidiary Workstations sind jedoch in der Lage eine EMail direkt an einen anderen Mailhost innerhalb der Domain zu senden, bzw. eine SMTP Verbindung mit diesem aufzubauen, dadurch wird der Zonen-Mailhost entlastet. Lokal ist an der Subsidiay Workstation ein User Agent (UA) vorhanden, wie beispielsweise das NEXT Mailtool. Es ist möglich, an eine Subsidiary Workstation ein Modem anzuschliessen und mittels UUCP EMails direkt zu senden / empfangen ohne diese über den Mailhost zu leiten. Subsidiary Maschinen sind also als eigentliche «Standalones-Workstations» mit eigener Disk und eigenem Betriebssystem am LAN zu betrachten. Sie betreiben einen lokalen MTA im sogenannten *Remote-Mode.* Das heisst, dass sämtliche EMail seiner Clients an den Mailhost weitergeleitet werden.(Siehe Option OR in */etc/sendmail.cf)*

1.4.4 Diskless-Clients

Diskless-Clients sind Workstations, welche «ihr» Betriebssystem von einem Diskless-Server booten.Sie verfügen lokal selbst über keine Disk. Das Management des gesamten virtuellen Memory's wird über den Discless-Server abgewickelt. Der Discless-Server kann in seiner Funktion *Mailhost* oder *Subsidiary* (Standalone) sein. Der Discless Client betreibt somit keinen eigenen MTA (Sendmail), sondern benutzt den MTA Dienst seines Diskless-Servers. Der Diskless-Server verwaltet also die EMails seiner Diskless-Clients. Er verfügt selber über eine «Mailbox» (/usr/spool/mail), welche direkt von seinen Clients verwendet wird. Typische Vertreter des Diskless-Konzepts sind HP-Workstations.

1.4.5 Dataless-Clients

Dataless-Clients sind Workstations, welche «ihr» Betriebssystem von einem NFS-Server booten.Sie verfügen lokal selbst über eine Disk. Das Management des gesamten virtuellen Memory's wird auf der lokalen Disk abgewickelt. Der NFS-Server kann in seiner Funktion *Mailhost* oder *Subsidiary* (Standalone) sein. Der Dataless Client betreibt somit keinen eigenen MTA (Sendmail), sondern benutzt den MTA Dienst seines NFS-Servers. Der NFS-Server verwaltet also die EMails seiner Dataless-Clients. Er verfügt selber über eine «Mailbox» (/var/spool/mail), welche direkt von seinen Clients verwendet wird. Typische Vertreter des Dataless-Konzepts sind SUN-Workstations.

1.5 EMail Routing

Das optimale Zusammenspiel der drei EMail-Komponenten User-Agent (Mailtool), Delivery-Agent (Mailer) und Message Transfer-Agent (EMail-Router) ermöglichen einen erfolgreichen Transport der elektronischen Meldung von Benutzer zu Benutzer. Eine zentrale Rolle spielt der Message Transfer-Agent, welcher für das korrekte Routing verantwortlich ist. Man kann folgende Aufgabenteilung vornehmen:

Mailtools (User Agent UA)

Das Mailtool stellt dem Benutzer ein möglichst einfach zu bedienendes Front-End zum Empfangen, Senden und Verwalten der EMail dar. Je nach System (DOS-PC, Windows-PC, UNIX Workstation mit X11-Oberfläche, ASCII-Terminal an UNIX-Host) werden meist unterschiedliche Tools eingesetzt. Es sind diverse Produkte auf dem Markt wie *EUDORA, POPMail, elm, mush, MH* und *xmh*. Allen Mailtools ist gemeinsam, dass sie keine Mails selber versenden, sondern diese an eine zentrale Instanz, den MTA, weiterleiten. Dabei werden die eigentlichen Messages meist via UNIX Pipe oder dem EMail Protokoll SMTP an den MTA weitergegeben.

EMail-Router (Message Transfer Agent MTA)

Der MTA übernimmt das Weiterleiten (Routing) der EMail. Dazu liest er das Envelope der EMail, interpretiert Sender- und Empfängeradresse und transferiert die EMail mit der korrekt modifizierten Adresse an den Delivery-Agent (Mailer) weiter. Der MTA muss nur seinen unmittelbar nächsten Bestimmungsort kennen. Er ist in der Lage die EMail über ein Transport-Medium (LAN. Telefonleitung) mittels fest definierten Transport-Instruktionen (Protokolle wie UUCP, SMTP) weiterzuleiten. Am Bestimmungsort übernimmt ein MTA erneut das Routing. Unter UNIX wird meist *Sendmail* als MTA eingesetzt. Sendmail wird grundsätzlich auf zwei Arten betrieben. Als Netzwerk-Daemon Process «horcht» Sendmail auf das TCP/IP Netzwerk am Port Nr. 25 auf EMail Requests von Client Sendmails. Die Übermittlung von EMail mittels SMTP basiert somit immer auf einer Client-Server Verbindung (Sockets). Nur Relayhosts und Mailhosts betreiben Sendmail als Netzwerk-Daemon, während Discless-Clients und Subsidiary-Hosts den Sendmail-Process «ihres» Mailhosts im *Remote-Mode* benutzen. Entsprechend unterschiedlich ist die Konfiguration einer Sendmail-Installation auf einem Mailhost, Relayhost, Subsidiary-Host und auf einer Discless-Workstation.

Mailer (Delivery Agent DA)

Der Mailer (Delivery-Agent) transportiert die Email physikalisch an seinen nächsten Bestimmungsort. Damit ergibt sich ein «Stop-and-Go» Ablauf oder anders ausgedrückt ein «Store-and-Forward» Prinzip. Die EMail wird ihm vom MTA über eine definierte Schnittstelle übermittelt. Der Mailer übernimmt keine Routing Funktion, sondern ist ein reiner Transport-Dienst. Die Schnittstelle zwischen MTA und Mailer ist vom Mailer selbst abhängig. Für SMTP Verbindungen via TCP/IP setzt Sendmail eine Pipe auf und kreiert mittels *fork()* einen Child Sendmail, dieser startet den Mailer mittels *exec()*. Via Pipe werden Envelope und Message der Email ausgetauscht. Anders verhält es sich beim lokalen Mailer /bin/mail und dem UUCP Mailer /bin/uux. Diese sind im Konfigurationsfile definiert. Sie werden mit dem entsprechenden Argumentvektor aufgerufen. Die nachstehenden Abbildungen zeigen den Ablauf zum Senden und Empfangen einer EMail.

1.5.1 EMails empfangen

Die folgende Übersicht zeigt eine mögliche EMail-Konfiguration eines Relayhosts zum Empfangen von EMails

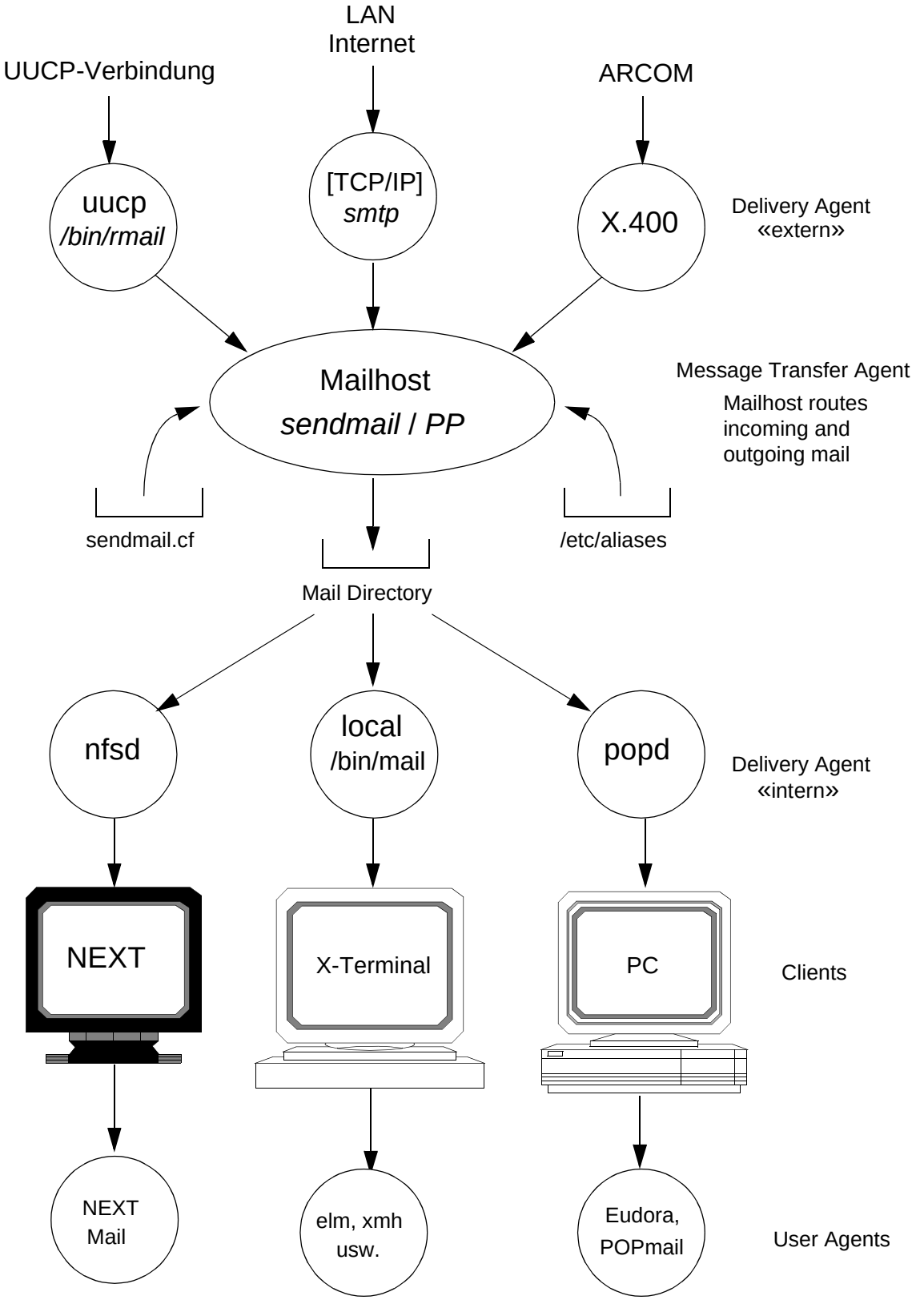


Abb. 4-4 EMail empfangen (Beispiel einer möglichen Konfiguration)

1.5.2 EMails senden

Die folgende Übersicht zeigt eine mögliche EMail-Konfiguration eines Relayhosts zum Senden von EMails

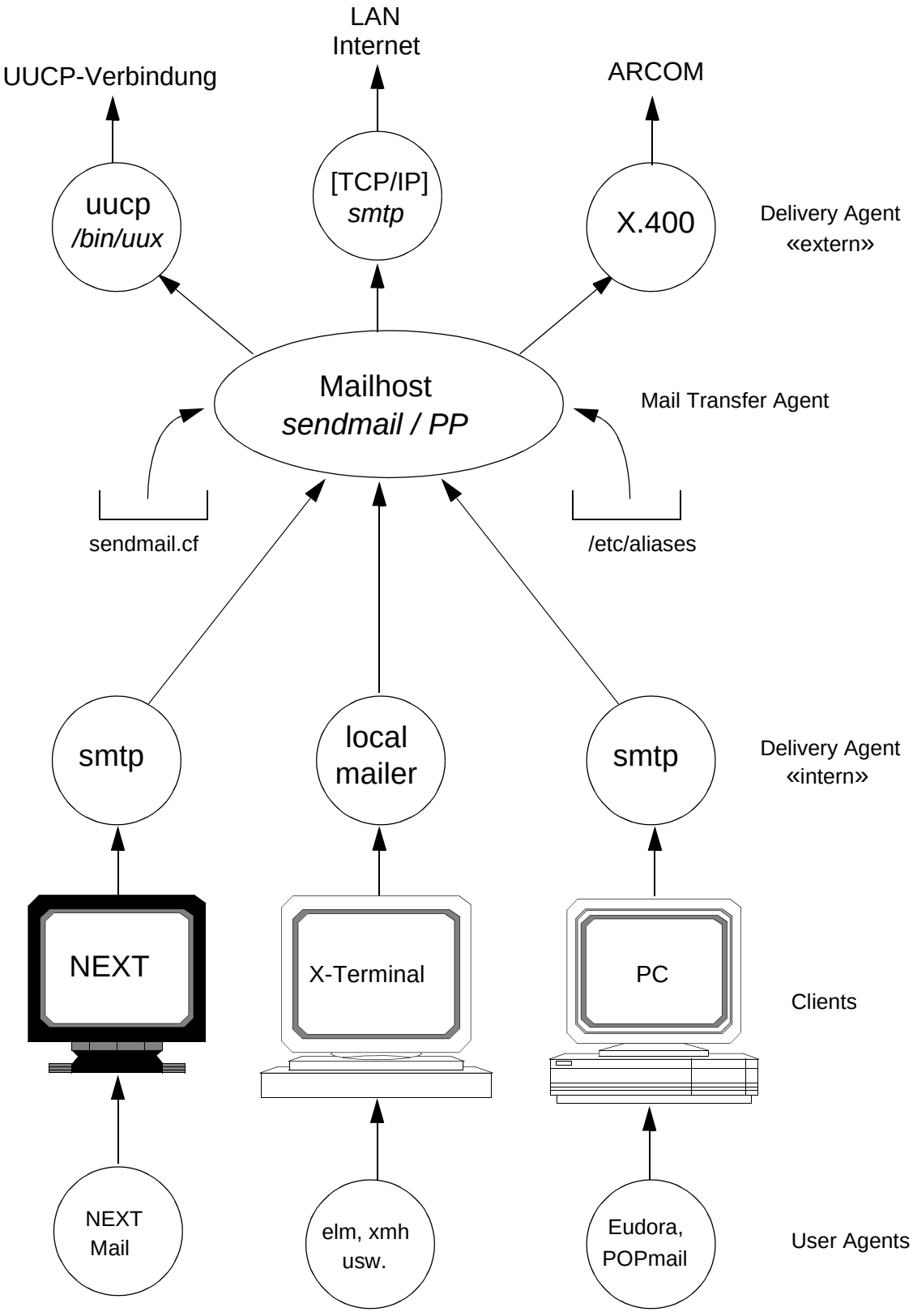


Abb. 4-5 EMails senden (Beispiel einer möglichen Konfiguration)

1.5.3 MTA Sendmail

Als zentrale Routing-Instanz wird in der «UNIX-Welt» sehr oft *Sendmail* eingesetzt. *Sendmail* liest den Header der Mail, interpretiert diesen anhand vorgegebenen Regeln, modifiziert diesen Header und leitet schliesslich die EMail mit dem modifizierten Header an den entsprechenden Mailer (Delivery Agent weiter). Sendmail ist in der Lage mehrere Mailer zu verwalten. Standarmässig sind dies UUCP und der lokale Mailer (*/bin/mail*). Einige Hersteller haben *Sendmail* erweitert, so dass auch X.400 und herstellerspezifische Mailer verwendet werden können (z.B. OpenMail von HP). *Sendmail* hat zudem einen fest integrierten Mailer um über ein TCP/IP Netzwerk EMails zu senden/empfangen. Dies ist das integrierte SMTP-Protocol (Simple Mail Transfer Protocol). Sendmail hat im Normalfall somit drei «Eingänge» und drei «Ausgänge»:

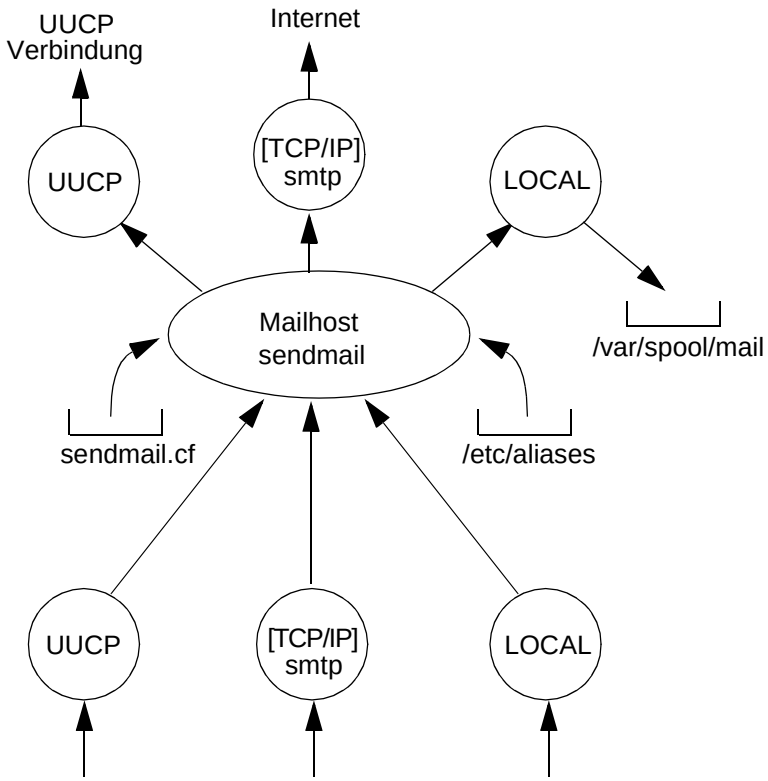


Abb. 4-6 Sendmail mit Routing-Funktion

Man beachte nochmals, dass *Sendmail* nie eine EMail selbst transportiert, sondern die EMail aufgrund seiner Adresse dem entsprechenden Delivery Agent übergibt. Erkennt *Sendmail*, dass der Empfänger lokal bekannt ist so wird die EMail dem lokalen Mailer (*/bin/mail*) übergeben. Ein Benutzer ist lokal bekannt wenn er im UNIX Konfigurationsfile */etc/passwd* bekannt ist, oder wenn ein Aliasname im EMail-Alias File */etc/aliases* erkannt wird. Ansonsten untersucht *Sendmail* die Empfängeradresse und ermittelt daraus den Mailer zum Weitertransport. Die Interpretation der Adresse wird mittels Rulesets aus der Konfigurationsdatei */etc/sendmail.cf* vollzogen. Aufgrund der grossen Vielfalt verschiedener möglichen Adressformate ist diese Aufgabe äusserst komplex. Der EMail Verantwortliche ist zudem in der Lage spezielle Rules zu erstellen um die Funktionalität von *Sendmail* zu erweitern. Dies hat zur Folge, dass das Konfigurationsfile */etc/sendmail.cf* von Hersteller zu Hersteller verschieden gestaltet ist. *Sendmail* muss in der Lage sein auch unter grosser Last sehr rasch eine *Adress-Resolution* (Herausfinden der Route) durchzuführen.

1.5.4 EMail Routing an Subsidiary-Workstations

In grösseren Workstation Umgebungen wird zur einfacheren Verwaltung der EMails ein Mailhost mit einem zentralen Mail-Directory eingerichtet. Sämtliche Subsidiary Workstations mounten mittels NFS dieses Directory. Dadurch ist ein zentraler Zugriff auf die EMails einfach möglich. Ähnlich verhält es sich mit dem Versenden von EMails von den Subsidiary-Workstations zum Mailhost. Es gibt zwei Möglichkeiten die EMails von den Subsidiary Rechnern an den Mailhost weiterzuleiten

- «Remote Sendmail Execution»
- Mailhost Forwarding mittels Aliases

«Remote Sendmail Execution»

Um EMail direkt vom lokalen Sendmail an den Mailhost zu leiten, wird dieser im «Remote-Mode» betrieben. Dazu wird der Mailhost im File in */etc/hosts* auf dem Subsidiary eingetragen. Im Sendmail Konfigurationsfile *sendmail.cf* wird der Relayhost mit dem Eintrag mailhost versehen und die Option OR gesetzt.

/etc/hosts:

```
193.5.24.67       ivi    ivi.ivi.ch    mailhost
```

/etc/sendmail.cf

```
# Remote mode - send through server if mailbox directory is mounted
OR

# major relay host
DRmailhost
CRmailhost
```

Wird Sendmail im Remote Mode betrieben, erscheinen lokale Hostnamen nicht in der From: Zeile des Headers, da sämtliche EMail über den Mailhost abgewickelt wird. Dies ist ein grosser Vorteil. Auf der Subsidiary Workstation wird kein Sendmail-Daemon gestartet. Dazu ist der Eintrag im Boot-File (/etc/rc.local) zu deaktivieren.

Beispiel eines Headers im «Remote Mode»

From: mz (Martin Zahn)
To: metz

Ablauf eines EMails Transfers im «Remote-Mode»

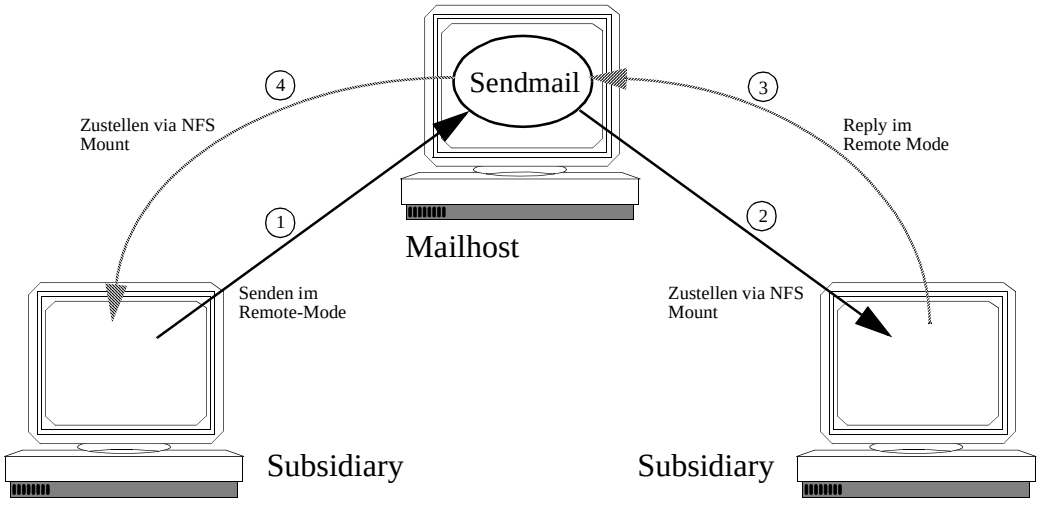


Abb. 4-7 EMail-Transfer im «Remote-Mode»

Gesamter EMail-Verkehr wird über den Mailhost abgewickelt.

Weiterleitung mittels aliases (/etc/aliases)

Kann aus einem bestimmten Grund die «Remote Mode» Variante nicht durchgeführt werden, so muss für jeden lokalen Benutzer ein Alias eingerichtet werden.

/etc/aliases:

```
root        root@eniac
mz          mz@eniac
```

Eniac ist beispielsweise der Name des Mailhosts. Nachteil dieses Variante ist, dass der lokale Hostname in der EMail Adresse erscheint, da der lokale Sendmail-Process den Hostnamen einfügt.

Beispiel eines Headers mit Alias-Namen

From: mz@mzsun (Martin Zahn)
To: metz@eniac

Zudem wird der EMail-Verkehr nicht ausschliesslich über den Mailhost abgewickelt. Das Email, welches mittels Reply zurückgeschickt wird gelangt zuerst wieder an den sendende Host und von dort erst an den Mailhost

Ablauf eines EMails Transfers mit lokalen Alias-Namen

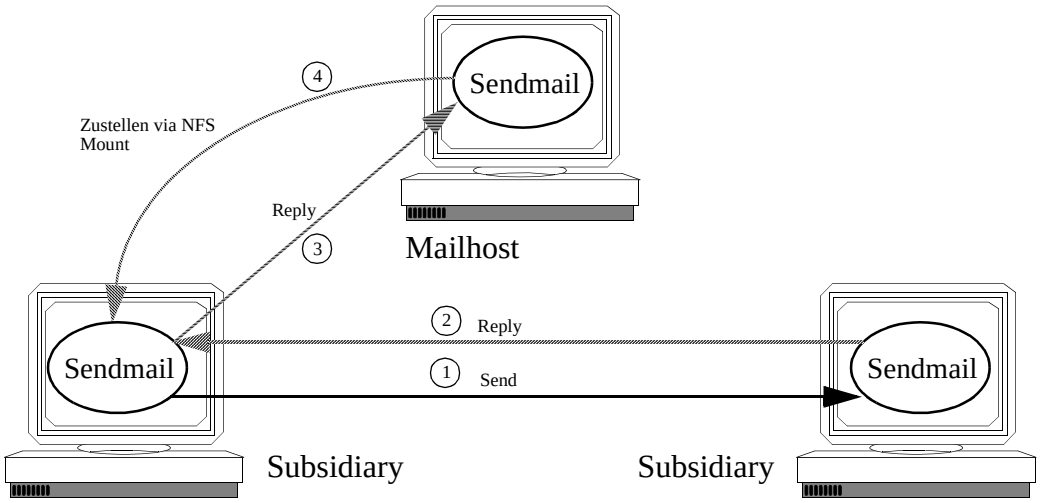


Abb. 4-8 EMail-Transfer mit lokalen Aliases

Der «Remote Mode» sollte wenn immer möglich angestrebt werden, da damit die gesamte EMail Konfiguration wesentlich vereinfacht wird. Zudem muss auf den Subsidiary Workstations kein Sendmail-Daemon (/usr/lib/sendmail -bd -q30m) aktiv sein. Dadurch spart man auch CPU Zeit.

Vorbereitete Sendmail-Konfigurationsfiles

NEXT Computer stellen bereits vorbereitete Sendmail Konfigurationsfiles zur Verfügung für die verschiedenen Anwendungszwecke

sendmail.mailhost.cf	Konfigurationsfile für den Relay-Host (z.B. ivi). Der Mayor Relay Mailer ist *ddn* für den Internet-Anschluss.Domain-Name wird an Adresse angehängt (fm@ivi.ch). Routing meist via DNS MX-Record zur übergeordneten Domain.

sendmail.subsidiary.cf	Konfigurationsfile für einen Mailhost (z.B jupiter), der über ein eigenes Maildirectory */usr/spool/mail* verfügt und dieses exportiert aber nicht am Internet direkt angeschlossen ist. Deshalb ist der Mayor Relay Mailer *ether* für SMTP Connections zum Relayhost.
sendmail.sharedsubsidiary.cf	Konfigurationsfile für eine Subsidiary-Workstation, die das zentrale EMail Spooldirectory von einem Mailhost importiert. Das Konfigurationsfile ist nicht für einen Sendmail-Daemon ausgelegt. Deshalb sollte kein Sendmail-Daemon auf dem Subsidiary-Host aktiv sein. Der Mayor Relay Mailer ist *etherl* speziell für Host ausgelegt, die das zentrale Mail-Directory mounten.

1.5.5 Mail Routing am Mailhost

Mailhosts, welche keinen direkten Zugang zum Internet besitzen, leiten EMails welche die eigene Domain verlassen an den Relayhost weiter. Dazu wird zum Relayhost eine SMTP Verbindung eröffnet oder das EMail wird via UUCP an den Relayhost zum Weitersenden geschickt. Kann der Mailhost eine EMail Adresse nicht auflösen, erkennt jedoch, dass er diese aufgrund der (korrekten) Adresse an die nächst höhere Instanz weiterleiten kann, so wird die Email mit dem richtigen Mailer an die nächst höhere Instanz, dem Relayhost weitergegeben.

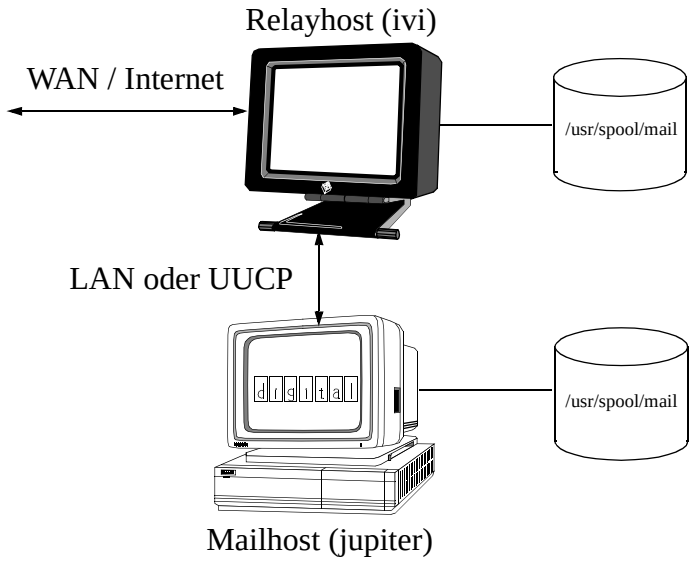


Abb. 4-9 EMail-Routing am Mailhost

Beispiel

Auf *jupiter* trifft eine Email mit der Adresse *charlie@hp.com* ein.
Der Mailhost *jupiter* erkennt, dass es sich nicht um einen Empfänger innerhalb der eigenen Domain *ivi.ch* handelt. Er wählt *ether* als Mailer und sendet die Email an den Relay-Host *ivi* weiter.

```
/usr/lib/sendmail. -v < /dev/null charlie@hp.com
charlie@hp.com... Connecting to ivi via ether...
charlie@hp.com... Sent.
```

1.5.6 EMail Routing am Relayhost

Der Relayhost ist dafür verantwortlich die Email an die fremde Domain zu senden. Sofern er die Domain kennt wird eine SMTP Verbindung über TCP/IP eröffnet und die EMail direkt an den Relayhost der fremden Domain weitergesendet. Dies ist jedoch in den meisten Fällen nicht der Fall, da der Relayhost keine physikalische Verbindung zum Empfänger besitzt. Zudem ist sehr oft die Adresse des Empfängers vorerst noch unbekannt. Hier stellt sich nun die Frage wie der Relayhost zu dieser Information kommt. Dazu steht ihm ein äusserst raffiniertes, weltweit verteiltes Informationssystem zur Verfügung, das *DNS* (Domain Name Services). Informationen zu registrierten Rechnersystemen in aller Welt können sekundenschnell mittels *Name-Server* eingeholt werden. Viele Name-Server verteilt über die gesamte Welt verwalten Domains und ihre zuständigen Rechnersysteme. Sogenannte *Root-Name-Server* kennen die Route bzw. Internet-Adresse jeder Domain. Das gesamte Internet ist hierarchisch in mehrere Domain-Ebenen gegliedert, ähnlich dem U NIX Filesystem. Die eigene Domain muss international bei einem NIC registriert werden damit sie für andere Benutzer in anderen Domains bekannt gemacht werden kann. In Sendmail-Konfigurationsfile wird die eigene Domain mit dem Macro M definiert.

```
# Dmivi.ch   <-- Primary Domain Name = boa.ch
# Cmivi.ch   <-- Alias for primary Domain Name for incoming Mail
#
Dmivi.ch
Cmivi.ch
```

Internet-Domains

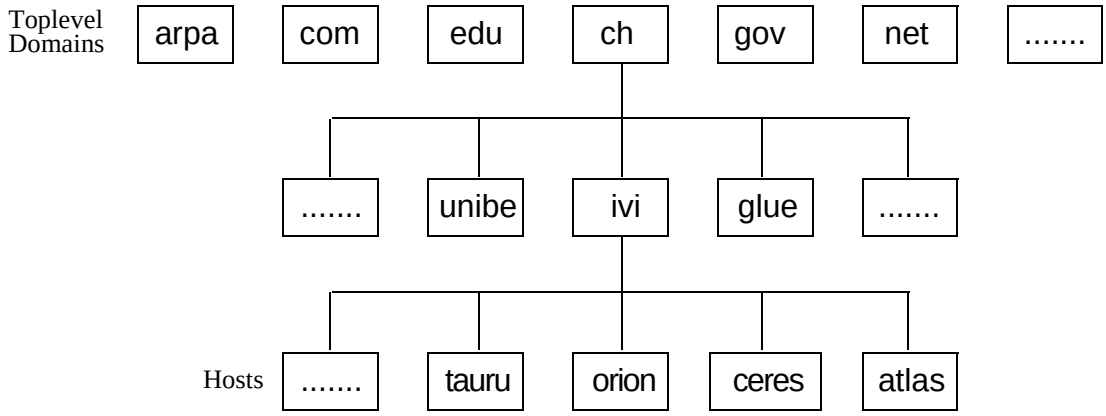


Abb. 4-10 Aufbau des Domain Name Service (Beispiel)

Die gesamte Konfiguration eines DNS Name-Servers ist nicht Bestandteil dieser Dokumentation, da DNS auch für andere Zwecke als Email benutzt wird. Es soll jedoch erläutert werden welche Bestandteile von DNS für das EMail Routing von Bedeutung sind. Im DNS Konfigurationsfile wird mittels *MX-Record* (Mail-Exchanger) ein Host am Internet definiert, der für die Weiterleitung der Email zuständig ist. Dort wird die IP-Adresse der Domain des Empfängers abgefragt. Sobald die IP-Adresse bekannt ist wird eine SMTP Verbindung aufgebaut. Das Abfragen eines DNS-Servers erfolgt mittels eines sogenannten *Resolver-Calls*. Sendmail und andere Netzwerk-Tools wie *rcp, telnet, ftp* haben die entsprechenden Resolver-Calls direkt in ihren Code gelinkt. Dazu wird meist die Library *libresolv.a* gelinkt. Zur interaktiven Abfrage des Relayhosts für eine bestimmte Domain kann auch das UNIX Utility *nslookup* benutzt werden. Das folgende Beispiel zeigt die Abfrage des Relayhosts für die Domain *glue.ch* vom Relayhost *ivi.ivi.ch* aus.

Mail-Exchanger Abfrage im DNS

```
$ nslookup
  Default Server: ivi.ivi.ch
  Address: 193.5.24.67

  > scsnms.switch.ch.
  Server: ivi.ivi.ch
  Address: 193.5.24.67

  Non-authoritative answer:
  Name: scsnms.switch.ch
  Address: 130.59.1.30

  > server scsnms.switch.ch
  ------------
  Default Server: scsnms.switch.ch
  Address: 130.59.1.30

  > set q=MX
  > glue.ch
  Server: scsnms.switch.ch
  Address: 130.59.1.30
  ------------
  glue.ch
   preference = 10, mail exchanger = chsun.chuug.ch
   ttl = 345600 (4 days)
   chsun.chuug.ch
   internet address = 146.228.10.15
   ttl = 345600 (4 days)
```

Man erkennt, dass der Mail-Exchanger (Relayhost) für die Domain *glue.ch* der Name-Server *chsun.chuug.ch* ist. Zu diesem Host wird eine SMTP-Verbindung vom Host *ivi.ivi.ch* aus aufgebaut und die Email übermittelt.

Der MX-Record Eintrag im DNS-Konfigurationsfile für den Relayhost ivi.ivi.ch hat dabei folgenden Eintrag

```
;
; Descriptions of name servers for this domain
        IN      NS      ivi.ivi.ch.
        IN      NS      scsnms.switch.ch.
;
; Mail-Exchanger for this Domain
ivi.ch.         MX      10 ivi.ivi.ch.
;
```

1.5.7 Festlegung des Major Relay Mailer

Der Mayor Relay Mailer definiert den Delivery-Agent zum Übertragen der Email an den Relayhost. Dies kann eine TCP/IP Verbindung sein über ein WAN (z.B. ISDN). Ist die physikalische Verbindung eine Modem Strecke, so kommt der Mailer UUCP in Frage. Ist dies jedoch eine LAN-Verbindung, so ist dies der in Sendmail eingebaute Mailer *ether*.

Der EMail Verantwortliche muss sich also vergewissern, wie der Relayhost erreicht werden kann und welcher Mailer dazu geeignet ist. Das Macro M im Sendmail Konfigurationsfile legt den Mailer fest.

```
###################### Major Relay Mailer: ############################
#
DMddn
```

1.6 Lokale Zustellung der EMail

Sobald eine EMail-Adresse als lokal erkannt wird untersucht *Sendmail* seine Alias-Datenbank um zu sehen, ob die EMail erneut weitergesendet werden soll. Ist dies nicht der Fall, so untersucht *Sendmail* ob der Empfänger lokal bekannt ist (*/etc/passwd* mit Home-Directory). Ist dies nicht der Fall, so wird die EMail als unzustellbar zurückgeschickt. Die folgende Abbildung zeigt den Ablauf

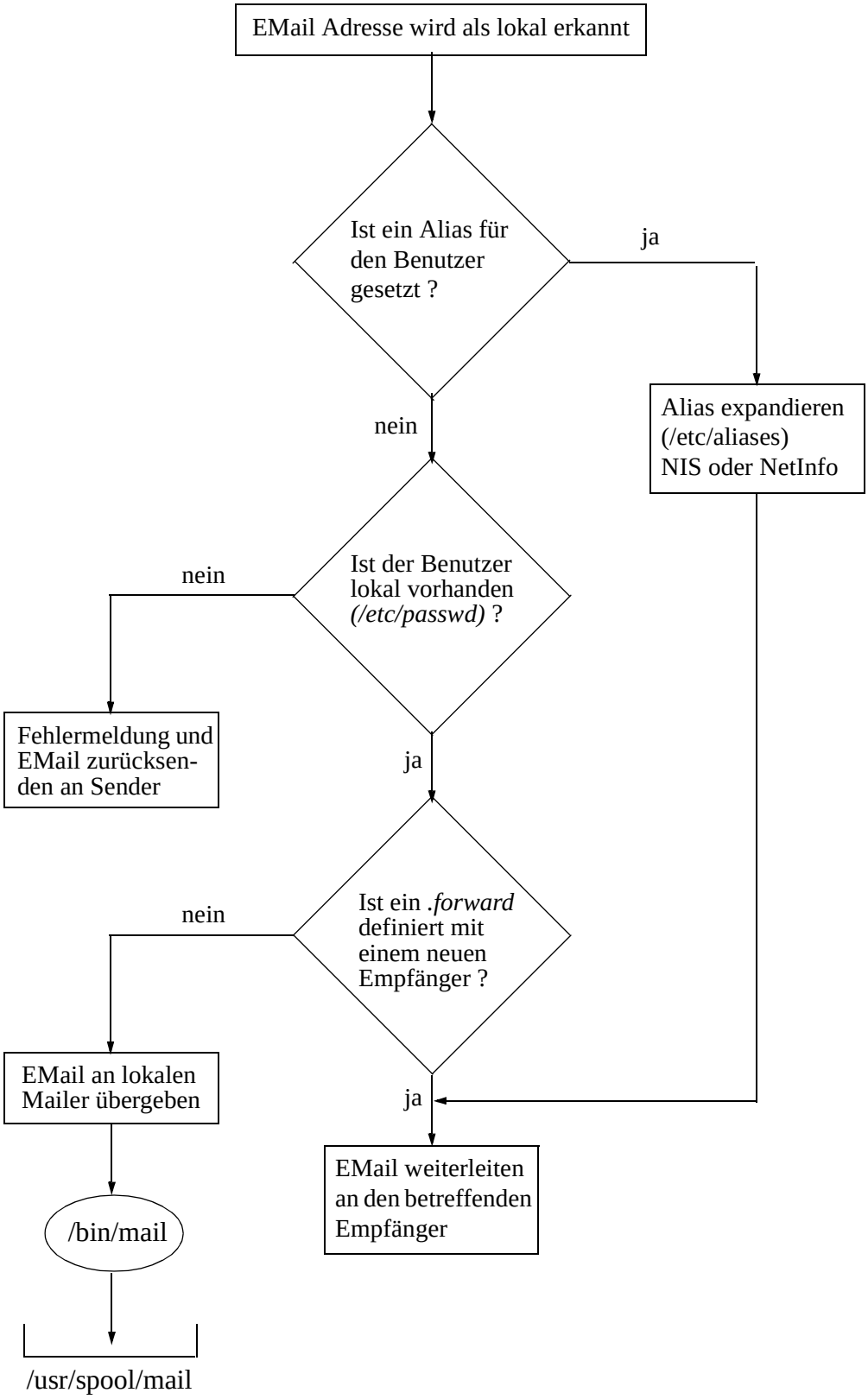


Abb. 4-11 Lokale Zustellung der EMails

1.7 Sendmail Konfiguration

1.7.1 Konfigurationsfile sendmail.cf

Das Konfigurationsfile *sendmail.cf* enthält die gesamte Information um dem MTA seine Aufgabe, das Routing der EMail zu ermöglichen. Der Kern des Files besteht aus den sogenannten Rulesets, einem Set von Regeln zur Interpretation der Adressen. Grundsätzlich liest *Sendmail* die Sender/Empfängeradresse, interpretiert diese und bereitet sie für den entsprechenden Delivery Agent = Mailer auf.

sendmail.cf gliedert sich in die folgenden Bereiche:

- Einrichtung von Makros (D), die unbedingt benötigt werden, also entweder gesetzt werden müssen oder intern von Sendmail selbst gesetzt werden.
- Klassenvereinbarungen (C), Zusammenfassung von Rechnern zu Gruppen oder Definition von Domain-Namen.
- Optionen (O) zum Definieren von Loglevel. Timeouts und diversen Hilfsfunktionen. Diese Optionen sind nicht mit den Commandline Optionen zu verwechseln.
- Headerformate (H) definieren die Headerzeilen wie from, Return-Path, Date usw.
- Adress Rewriting Rules (R und S), dies ist der eigentliche Kern von Sendmail.
- Mailer Definitionen (M) definieren wie ein Mailer angesprochen werden kann.

1.7.2 Macro-Definitionen (D)

Kleinbuchstaben sind sendmail intern reserviert. Grossbuchstaben stehen zur Verfügung.

Syntax:

Setzen eines Macros:

```
DeValue
```

Test ob Macro gesetzt ist:

Ist das Macro *x* gesetzt **$?x** so setze den Wert ein (Martin Zahn). Abschluss mit **$.**

```
$?x ($x)$.
```

If-Then-Else:

Wenn das Macro *x* gesetzt ist, verwende *text1*, else (**$|**) verwende *text2*.

```
$?x text1 $| text2 $.
```

Folgende Macro Definitionen müssen gesetzt werden:

Offizieller Domain-Name, Macro j und m:

```
Macro:      Dj$m        m wird mit dem Systemcall getdomainname() definiert]
                 $m
Ausgabe:    boa.ch      [verwendet in den Rulesets]
```

SMTP-Login Message, Macro **e**:

```
Macro       De$j Sendmail $v/$V ready at $b
                  $j          $v   $V              $b
Ausgabe:    220 ivi.ch Sendmail 4.1/SMI-4.1 ready at Wed, 15 Sep 93 15:51:24
```

$v: Version Number of Sendmail
$V: NEXT-Banner
$b: Current date in RFC 822 Format

UNIX Header Format. Macro l:

```
Macro:      Dl From $g  $d
                        $g                $d
Ausgabe:    From zephir!kanu!hueni Mon Sep 13 20:06:39 1993
```

$g: Sender Adress relative to the recipient
$d: Date in UNIX format

EMail User for Error Messages, Macro n:

```
Macro:      Dn Mailer-Daemon
```

Set of Operators (Delimiter) in Addresses, Macro o:

```
Macro:      Do .:%@!^=/[]
Ausgabe:    metz@iam.unibe.ch
```

Default Format of Sender Address, Macro q:

```
Macro:      Dq $g$?x ($x)$.
                          $g          $x
Ausgabe:    From: zephir!kanu!hueni (Hermann Hueni)
```

1.7.3 Class-Definitionen (C)

Kleinbuchstaben sind sendmail intern reserviert. Grossbuchstaben stehen zur Verfügung.

Eine Klasse wird verwendet um mehrere Werte auf eine sendmail Variable zu speichern. Oft werden Aliasnamen mittels Klasse definiert. Eine wichtige Anwendung ist die Definition der gültigen Toplevel Domains. Damit können eine Reihe von ungültigen Mails von vornherein abgeblockt werden

Gültige Toplevel Domains festlegen, Class T:

```
CT arpa com edu gov mil net org
CT us de fr jp kr nz il uk no au fi nl se ca ch my dk ar
```

Gültige Aliasnamen für Relayhost definieren, Class R:

```
CR zephir boa boa.ch zephir.boa zephir.boa.ch
```

Gültige Aliasnamen für eigene Primary Domain festlegen, Class m:

```
Cm boa.ch boa.uucp
```

1.7.4 Optionen (O)

Diese dürfen nicht mit den Kommandozeilen Optionen zu Sendmail verwechselt werden. Sie definieren globale" Variablen:

Eine besonders wichtige Option ist **OR**. Damit kann einem «lokalen» Sendmail, welcher das Maildirectory */var/spool/mail* des Mailhosts gemountet hat mitgeteilt werden, dass er alle Mails direkt an den Mailhost weiterleiten soll. Diese Betriebsart wird Sendmail Remote Mode genannt.

Remote mode - send through server if mailbox directory is mounted:

```
OR
```

Weitere Optionen:

```
# location of alias file
OA/etc/aliases
# default delivery mode (deliver in background)
Odbackground
# rebuild the alias file automagically
OD
# temporary file mode -- 0600 for secure mail, 0644 for permissive
OF0600
# default GID
Og1
# location of help file
OH/usr/lib/sendmail.hf
# log level
OL9
# default messages to old style
Oo
# Cc my postmaster on error replies I generate
OPPostmaster
# queue directory
OQ/usr/spool/mqueue
# read timeout for SMTP protocols
Or15m
# status file -- none
OS/etc/sendmail.st
# queue up everything before starting transmission, for safety
Os
# return queued mail after this long
OT3d
# default UID
Ou1
```

1.7.5 Mail Header Formate

Headerformate dürfen nicht geändert werden.

Aufbau der Header-Formate:

```
H[?mflags?]hname:htemplate
```

Kombination von bereits definierten Macros und Text. Macros werden vor dem Einfügen expandiert

Header Name

Das Headerflag definiert, ob der Header eingefügt werden soll oder nicht. Ist das Flag nicht gesetzt, so wird der Header für keinen Mailer eingesetzt. Ist das Flag jedoch gesetzt wird der Header nur eingesetzt wenn der entsprechende Mailer das gleiche Mailerflag gesetzt hat. Headerflags kontrollieren jedoch nur Outgoing Mails, nicht jedoch Incoming Mails.

Folgender Header wird beim Versenden generiert:

```
From mz  Thu Sep 16 21:24:07 1993 remote from mzsun.boa.ch
Received: by boa.ch (4.1/SMI-4.1)
        id AA01305; Thu, 16 Sep 93 21:24:07 +0200
From: mzsun.boa.ch!mz (Martin Zahn)
Message-Id: <9309161924.AA01305@boa.ch>
Subject: Test
To: zephir!bj
Date: Thu, 16 Sep 1993 21:24:06 +0200 (MET DST)
Cc: zephir!hueni
X-Mailer: ELM [version 2.4 PL21]
Mime-Version: 1.0
Content-Type: text/plain; charset=ISO-8859-1
Content-Transfer-Encoding: 8bit
Content-Length: 107
```

Entsprechende Header-Definitionen:

```
DlFrom $g  $d
HReceived: $?sfrom $s $.by $j ($v/$V)
        id $i; $b
H?F?From: $q
H?M?Message-Id: <$t.$i@$j>
```

Macros in den Header-Definitionen:

$g: Sender Adresse relativ zum Empfänger
$d: UNIX-Datum
$s: Senders Host-Name
$j: Lokaler Domainname
$i: Queue-Id
$t: Numeric Representation of current time

Welche Headereinträge erstellt werden ist abhängig vom verwendeten Mailer bzw. dessen gesetztem Mailerflag.

«mflag» in Header-Definition ◄——————► Mailer Flag

1.7.6 Mailer (Delivery Agent) Definitionen

Sendmail bereitet die Adresse für den jeweiligen Mailer auf und sendet diesem die EMail zur Weiterleitung. Die Namen der Mailer sind nicht fest vorgegeben, ausser für den lokalen Mailer *local* und den Delivery Agent welcher die Email an ein Programm sendet *prog*. Die Characteristik des Mailer wird mit *field=value* Paaren definiert. Die wichtigsten Werte für *field* sind:

• P	Path of the Mailer	P=/usr/bin/uux
• F	Sendmail Flags for this Mailer	F=lsDFMe
• S	Ruleset for Sender Address	S=10
• R	Ruleset for Recipient Address	R=20
• A	Mailer's Arguments	A=sh -c $u

Wichtigste Mailerflags (in Verbindung mit Header-Definitionen!)

- C Add @domain to addresses that do not have an @
- D Mailer wants a Date: header line.
- F Mailer wants a From: header line
- M Mailer wants a Message-Id: header line
- P Mailer wants a Return-Path: line
- U Mailer wants Unix-style From: lines
- m Mailer can send to multiple users in one transaction

Mailer-Definitionen

```
Mlocal, P=/bin/mail, F=rlsDFMmnP, S=10, R=20, A=mail -d $u
Mprog,  P=/bin/sh,   F=lsDFMeuP,  S=10, R=20, A=sh -c $u
Mether, P=[TCP], F=msDFMuCX, S=11, R=21, A=TCP $h
Muucp,  P=/usr/bin/uux, F=msDFMhuU, S=13, R=23,
        A=uux - -r -a$f $h!rmail ($u)
Mddn,   P=[TCP], F=msDFMuCX, S=22, R=22, A=TCP $h, E=\r\n
Msmartuucp, P=/usr/bin/uux, F=CmsDFMhuU, S=22, R=22,
        A=uux - -r $h!rmail ($u)
```

Der Mailer *uucp* und *smartuucp* unterscheiden sich dadurch, dass *smartuucp* die *Domain-Adressierung: user$domain* unterstützt, *uucp* jedoch nicht. (smartuucp verwendet C Mailer-Flag und andere Rulesets)

smartuucp:	uucp:
mz@mzsun.boa.ch	mzsun!mz

Kopplung Header-Definition Mailer-Flag

```
H?M?Message-Id: <$t.$i@$j>
```

```
Muucp,  P=/usr/bin/uux, F=msDFMhuU, S=13, R=23,
        A=uux - -r -a$f $h!rmail ($u)
```

Sendername relativ zu Empfänger

Das Macro $g definiert im Header den Sender relativ zum Empfänger. Beispiel: Der User *mz* sendet eine EMail an *bj@zephir* vom Rechner *mzsun* aus. Das Macro $f wird mit «*mz*» und das Macro $g wird mit «*mz@mzsun*» belegt. Das heisst der Empfänger *bj* sieht in der Headerzeile *From: mzsun.boa.ch!mz (Martin Zahn)*

1.7.7 Rewriting der Mail-Adresse

Das Aufbereiten der Mailadresse ist die Hauptaufgabe von Sendmail. Rulesets bestehen aus einer Anzahl Regeln, wie eine Sender-Adresse erhalten von einem User Agent (UA) für den entsprechenden Mailer (Delivery Agent) aufbereitet werden soll. Dazu wird der **R** Command in sendmail.cf benutzt.

R*pattern*		Transformation / Kommentar	
$*	Match zero or more tokens	$x	Expand macro x
$+	Match one or more tokens	$n	Substitute indefinite token n from LHS
$-	Match exactly one token	$>n	Call ruleset n
$=x	Match any string in Class x	$#mailer	Resolve to mailer
$~x	Match any token not in class x	$@host	Specify host (+ prefix ? ruleset return)
$%x	Match any token in yp map $x	$:user	Specify user (+ prefix rule limit)
$!x	Match any token not in yp map $x	$[host$]	Map to primary hostnam
$x	Match Macro x	${x name$}	Map name thoroug yp map $x

Pattern Matching

Input Adressen werden gegen das Pattern verglichen, trifft der Vergleich (match is found) zu, so wird die Adresse entsprechend der Transformationsregel intern neu generiert. Die neu generierte Adresse wird dem gleichen Pattern erneut vorgelegt, trifft der Vergleich erneut zu, so wird die Adresse weiter verändert. Dieser Ablauf wiederholt sich bis der Vergleich nicht mehr zutrifft.

Die Adresse wird für den Vergleich in Tokens zerlegt. Die einzelnen Tokens sind durch Operatoren (definiert im Macro $o) voneinander getrennt.

Trifft ein Vergleich zu, so werden die Address-Strings auf Variablen der Form $1, $2, $3, usw. gespeichert und der Transformationsregel zugeführt. Diese $n-Variablen werden indefinite tokens genannt.

Transforming the Address

Dies betrifft die rechte Seite des R-Commands. Literale werden unverändert eingesetzt, Macros expandiert und indefinite tokens eingesetzt.

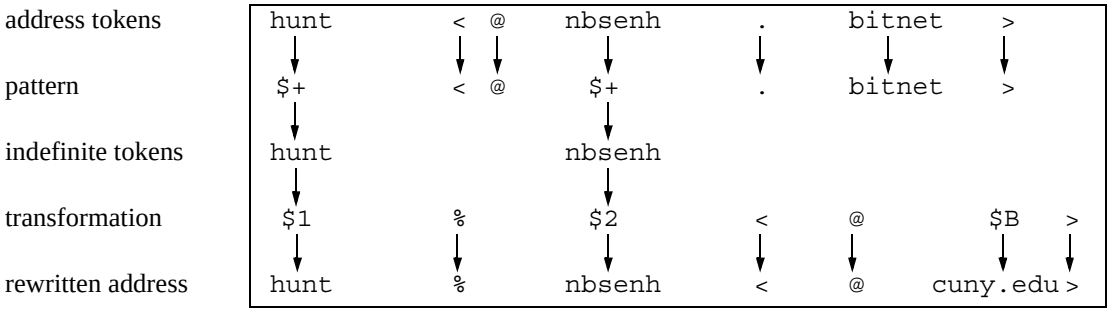
Beispiel:

Die Adresse *hunt@nbsenh.bitnet* wurde durch eine vorgängige Transformation in die folgende Form gebracht:

```
hunt<@nbsenh.bitnet>
```

Das R-Kommando für die weitere Transformation lautet:

```
R$+<@$+.bitnet>      $1%$2<@$B>      Use Bitnet relay
```



Transformation: **$[*name*$]**

Damit wird ein Host-Aliasname in den Canonical-Name gewandelt. Dazu wird *name* dem Domain Name Server BIND zur Auflösung übergeben.

Beispiel: Name Server *nuts.com*

$[mil-gw$] ergibt: almond.nuts.com

Aliasname Canonical-Name

Beispiel aus HP sendmail.cf

```
# canonicalize host and possibly recurse
R$*<@$+>$*              $:$1<@$[$2$]>$3         user@host or source route
```

Anderes Ruleset aufrufen: **$>n**

Damit wird das Ruleset n aufgerufen.

Beispiel: $>9$1%$2

Damit wird das Ruleset 9 aufgerufen. Dem Ruleset 9 wird $1%%2 übergeben zur Verarbeitung. Sobald Ruleset 9 fertig ist gibt sie die rewritten Address an die calling rule zurück. Die zurückgegebene Adresse wird erneut in der calling rule verarbeitet.

Beendigung einer Transformation: **$@**

Um mögliche rekursive Loops zu verhindern wird $@ gebraucht. Steht $@ am Anfang einer Transformation, so wird die Verarbeitung beendet und das aktuelle Transformationsergebnis wird zurückgegeben:

Beispiel:

```
# handle from:<> special case
R$*<>$*         $@@             turn into magic token
```

Verhinderung von rekursiven Loops: $:

Um mögliche rekursive Loops zu verhindern wird auch $; gebraucht. Steht $; am Anfang einer Transformation, so wird die Verarbeitung nur einmal ausgeführt.

Beispiel:

R$+@$+ **$:**$1<@$2> focus on domain

Special rewrite rule für Ruleset 0:

Ruleset 0 definiert das Triple (mailer, host, user) welches den Mailer, den Recipient Host und den Recipient User spezifiziert. Die spezielle Transformation-Syntax lautet:

$#mailer$@host$:user

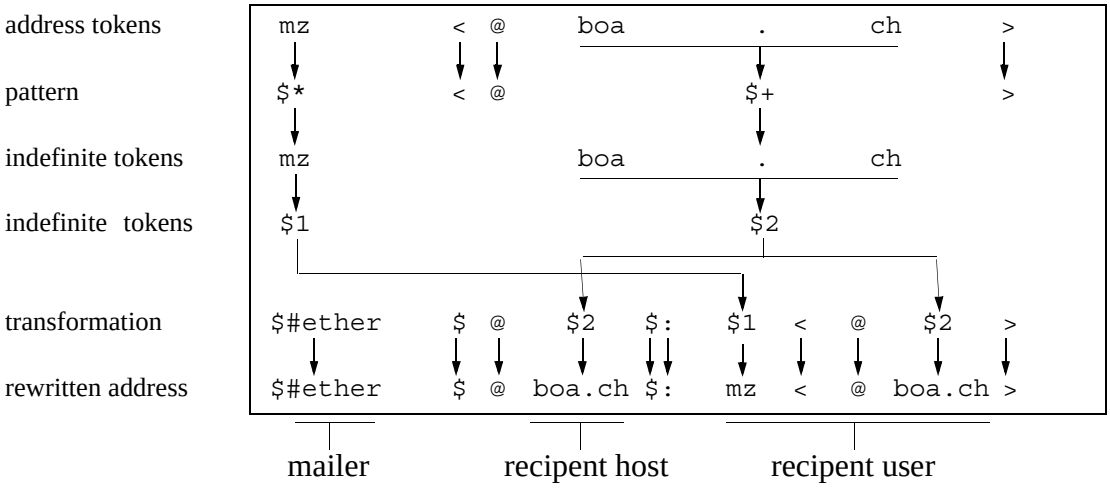
Beispiele aus SUN sendmail.cf

```
# deliver to known ethernet hosts explicitly specified in our domain
R$*<@$%y.LOCAL>$*      $#ether $@$2 $:$1<@$2>$3      user@host.sun.com

# resolve UUCP-style names
R<@$-.uucp>:$+         $#uucp  $@$1 $:$2             @host.uucp:...
R$+<@$-.uucp>          $#uucp  $@$2 $:$1             user@host.uucp
```

Ruleset 0 Beispiel:

Gegeben Adresse: mz<@boa.ch>
Rule: R$*<@$+>$*
Transformation: $#ether$@2:$1<@$2>$3



Error-Meldungen:

Im Ruleset 0 wird ausserdem das folgende Konstrukt zur Fehlerausgabe verwendet:

`$#error$:`*message*

1.7.8 Rewriting Rulesets

Eine EMail besteht aus Envelope, Message Header und Message Body. Das Envelope ist für den Benutzer meist gar nicht sichtbar, es enthält Informationen des Transportsystems über Absender (Sender) und Adressat (Recpient) der Mail. Der Header besteht aus weiteren Angaben wie From:, To: und Cc:

Die Verarbeitung dieser Angaben erfolgt durch mehrere Rules, sogenannten Rulesets. Dabei nehmen die Rules 3, 0, 1, 2 und 4 eine besondere, fest definierte Funktion ein.

Verarbeitungsfolge:

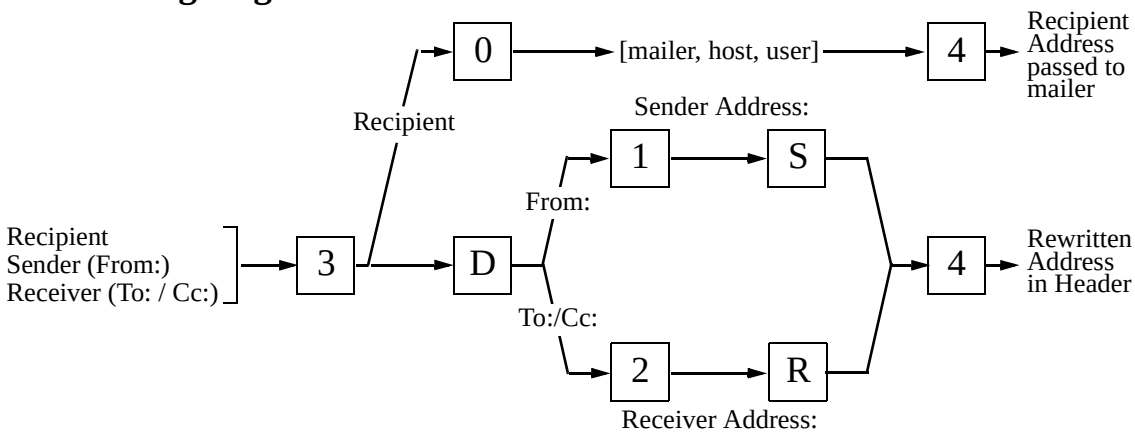


Abb. 4-12 Sendmail Rewriting Rules

3	Convertierung in canonical form (intern)	0	Resolution of mailer, host, user
D	Sender Domain addition	1	Sender Address rewriting
S	Mailer specific Sender Address rewriting	2	Receiver Address rewriting
R	Mailer specific Receiver Address rewriting	4	Convertierung in externe Form

Rulesets können als Subroutinen oder Funktionen betrachtet werden, mit der Aufgabe EMail-Adressen zu verarbeiten. Sie werden von Mailer-Definitionen, individuellen Rewrite Rules oder direkt durch Sendmail aufgerufen. Folgende Rulesets werden direkt durch Sendmail aufgerufen:

Spezielle Rulesets:

- Ruleset 3: Umfangreichstes Ruleset, auf jede Adresse appliziert. Es konvertiert eine Adresse in die Form *local-part<@host.domain>*.

  Beispiel: mz@mzsun.boa.ch wird zu mz<@mzsun.boa.ch>

- Ruleset 0: Ruleset 0 verarbeitet die Envelope Adresse. Hier wird der Mailer, der Empfänger-Host und der Empfänger-Name festgelegt.

  Beispiel: mz@mzsun.boa.ch

Mailer:	TCP/IP
Empfänger-Host:	mzsun.boa.ch
Empfänger-User:	mz<@mzsun.boa.ch>

- Ruleset 1: Alle Sender (From:) Adressen im Header werden von diesem Ruleset bearbeitet.

- Ruleset 2: Alle Empfänger (To: Cc:) Adressen im Header werden von diesem Ruleset bearbeitet.

- D Hat ein Mailer das C-Flag (Add @domain to address that do not have an @) gesetzt, (dies trifft für alle TCP/IP Mailer zu) so wird @host.domain an Adressen, die nur einen lokalen Teil haben angehängt.

  Beispiel: mz wird zu mz@mzsun.boa.ch

- S/R Mailer spezifische Verarbeitung der Sender (From:) bzw. Empfänger Adressen (To: / Cc:). Welche Rules angewendet werden steht in den Mailer-Definitionen.

  Beispiel: `Mether,P=[TCP],F=msDFMuCX,S=11,R=21,A=TCP $h`

- Ruleset 4: Zurücksetzen der internen Adress-Form

  Beispiel: mz<@mzsun.boa.ch> wird zu mz@mzsun.boa.ch

1.8 Testen der Adressen

Sendmail ermöglicht es eine Adresse zu testen. Dazu wird Sendmail im Test-Mode aufgerufen. Man hat die Möglichkeit zu beobachten, welche Rules auf eine Adresse angewendet werden und wie die Adresse durch die einzelnen Rules verändert werden.

1.8.1 Mailer, Host, User

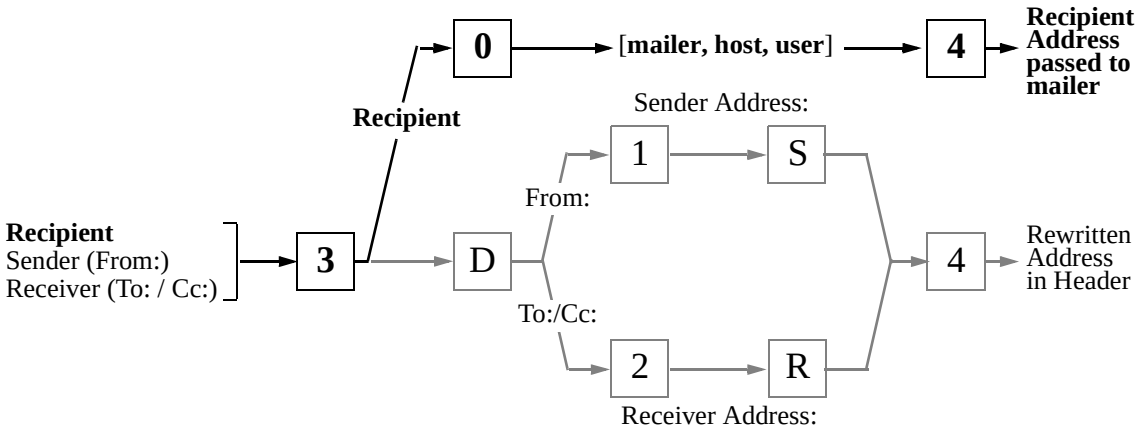


Abb. 4-13 Test der Recipient Adresse

```
$ /usr/lib/sendmail -bt -Ctest.cf
  ADDRESS TEST MODE
  Enter <ruleset> <address>
  > 0,4 zephir!bj
  rewrite: ruleset  3   input: zephir ! bj
  rewrite: ruleset  6   input: bj < @ zephir . uucp >
  rewrite: ruleset  6 returns: bj < @ zephir . uucp >
  rewrite: ruleset  3 returns: bj < @ zephir . uucp >
  rewrite: ruleset  0   input: bj < @ zephir . uucp >
  rewrite: ruleset  9   input: bj < @ zephir . uucp >
  rewrite: ruleset  9 returns: bj < @ zephir . uucp >
  rewrite: ruleset  0 returns: $# uucp $@ zephir $: bj
  rewrite: ruleset  4   input: $# uucp $@ zephir $: bj
  rewrite: ruleset  9   input: $# uucp $@ zephir $: bj
  rewrite: ruleset  9 returns: $# uucp $@ zephir $: bj
  rewrite: ruleset  4 returns: $# uucp $@ zephir $: bj
```

Zuerst wird das Ruleset 3 aufgerufen, welches die Adresse in die interne Form bj<@zephir.uucp> bringt. Ruleset 3 verwendet Ruleset 6 ($@$>6<@$1>:$2). Dann erfolgt die Verarbeitung in Ruleset 0 um dem Mailer, den Recipient-Host und den Recipient-User festzulegen. Ergebnis: $#uucp$@zephir$:bj. Ruleset 4 bringt die Adresse in die endgültige, externe Form `$#uucp$@zephir$:bj`. Zu beachten ist ferner, dass Ruleset 0 und 4 das Ruleset 9 intern verwendet. Aus dieser Information kann folgendes ermittelt werden: Als Kommunikations-Protokoll zur Übermittlung der Email wird UUCP verwendet. Der Zielhost ist *zephir*, dort wird die EMail Adresse *bj* vom *«zephir-Sendmail»* weiterverarbeitet. Man muss sich also sehr kritisch fragen, ob der Host *zephir* in der Lage ist die Adresse zu verarbeiten.

1.8.2 Sender Address Rewriting (From:)

Damit kann festgestellt werden, wie die From: Headerzeile bearbeitet wird. Je nach Mailer wird für S ein Ruleset aufgerufen. Man muss somit im Konfigurationsfile sendmail.cf nachschauen, welches Ruleset für einen bestimmten Mailer aufgerufen wird.
Beispiel: Der Mailer uucp verwendet S=13, smartuucp verwendet S=22

Sender Address Rewriting:

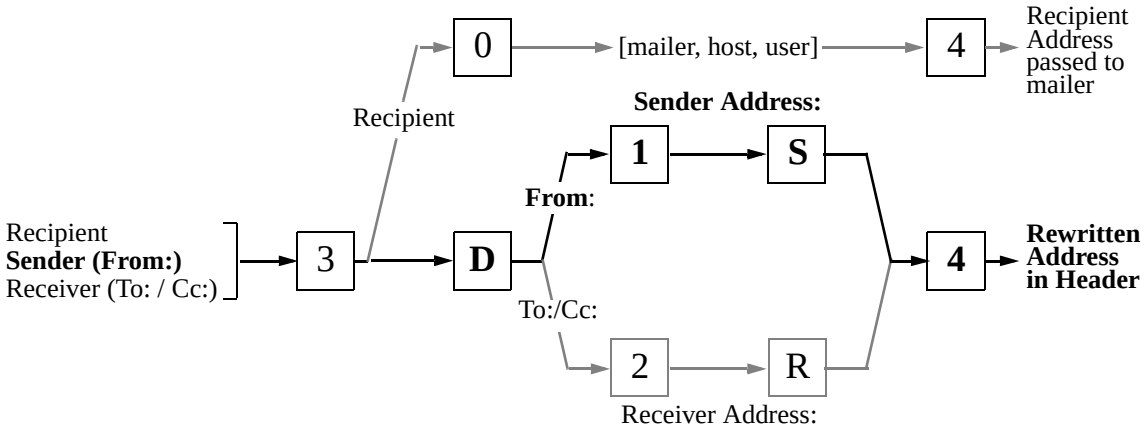


Abb. 4-14 Test der «From:» Adresse

```
$ /usr/lib/sendmail -bt -Ctest.cf
  ADDRESS TEST MODE
  Enter <ruleset> <address>
  > 1,13,4 mzsun!mz
  rewrite: ruleset  3   input: mzsun ! mz
  rewrite: ruleset  6   input: mz < @ mzsun . uucp >
  rewrite: ruleset  6 returns: mz < @ mzsun . uucp >
  rewrite: ruleset  3 returns: mz < @ mzsun . uucp >
  rewrite: ruleset  1   input: mz < @ mzsun . uucp >
  rewrite: ruleset  1 returns: mz < @ mzsun . uucp >
  rewrite: ruleset 13   input: mz < @ mzsun . uucp >
  rewrite: ruleset  5   input: mz < @ mzsun . uucp >
  rewrite: ruleset  5 returns: mzsun ! mz
  rewrite: ruleset 13 returns: mzsun . boa . ch ! mz
  rewrite: ruleset  4   input: mzsun . boa . ch ! mz
  rewrite: ruleset  9   input: mzsun . boa . ch ! mz
  rewrite: ruleset  9 returns: mzsun . boa . ch ! mz
  rewrite: ruleset  4 returns: mzsun . boa . ch ! mz
```

Damit wird die Headerzeile zu `From: mzsun.boa.ch!mz`

Das Sender-Ruleset für den UUCP Mailer ist das Ruleset-Nr 13, dies kann im Konfigurationsfile herausgelesen werden. Die Kontrolle der Sender-Adresse ist äusserst wichtig, damit der Empfänger die EMail mittels «Reply» beantworten kann. In EMail Umgebungen sind falsche Reply-Adressen der häufigste Grund für unzustellbare EMails.

1.8.3 Receiver Address Rewriting (To: / Cc:)

Für den uucp Mailer wird R=23 verwendet:

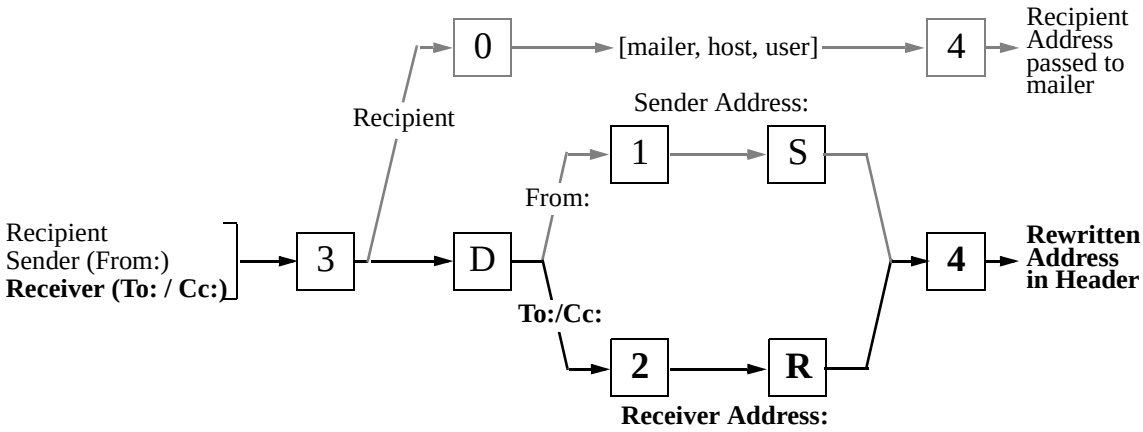


Abb. 4-15 Test der «To/Cc:» Adresse

```
$ /usr/lib/sendmail -bt -Ctest.cf
  ADDRESS TEST MODE
  Enter <ruleset> <address>
  > 2,23,4 zephir!bj
  rewrite: ruleset  3   input: zephir ! bj
  rewrite: ruleset  6   input: bj < @ zephir . uucp >
  rewrite: ruleset  6 returns: bj < @ zephir . uucp >
  rewrite: ruleset  3 returns: bj < @ zephir . uucp >
  rewrite: ruleset  1   input: bj < @ zephir . uucp >
  rewrite: ruleset  1 returns: bj < @ zephir . uucp >
  rewrite: ruleset 23   input: bj < @ zephir . uucp >
  rewrite: ruleset  5   input: bj < @ zephir . uucp >
  rewrite: ruleset  5 returns: zephir ! bj
  rewrite: ruleset 23 returns: zephir ! bj
  rewrite: ruleset  4   input: zephir ! bj
  rewrite: ruleset  9   input: zephir ! bj
  rewrite: ruleset  9 returns: zephir ! bj
  rewrite: ruleset  4 returns: zephir ! bj
```

Damit wird die Headerzeile zu: `To: zephir!bj`

Kontrolle des Headers

```
$ /usr/lib/sendmail -Ctest.cf -t -v
  To: zephir!bj
  From:mzsun!mz
  Subject: Header-Test

  Dies ist ein Header-Test
  ^D
  zephir!bj... Connecting to zephir via uucp...
  zephir!bj... Sent
```

Generierter Header

```
From root  Mon Sep 20 15:34:39 1993 remote from mzsun.boa.ch
Received: by boa.ch (4.1/SMI-4.1)
        id AA00317; Mon, 20 Sep 93 15:34:39 +0200
Date: Mon, 20 Sep 93 15:34:39 +0200
Message-Id: <9309201334.AA00317@boa.ch>
To: zephir!bj
From: mzsun.boa.ch!mz
Subject: Header-Test
```

1.8.4 Wichtigste Sendmail Kommandos

```
$ /usr/lib/sendmail -q          (Message-Queue bearbeiten)
$ /usr/lib/sendmail -bp         (Message-Queue auslisten)
$ /usr/lib/sendmail -bi         (/etc/aliases neu erstellen)
$ /usr/lib/sendmail -bz         (/etc/sendmail.fc neu erstellen)
$ /usr/lib/sendmail -bd -q1h    (Sendmail listens on TCP/IP-Port)
```

1.8.5 Testen von Mailinglists in /etc/aliases

```
$ /usr/lib/sendmail -bv -v glue
  glue... aliased to  zephir!hueni zephir!bj zephir!metz%iam.unibe.ch
  zephir!hueni zephir!bj zephir!metz%iam.unibe.ch... deliverable
```

1.8.6 Troubleshooting Sendmail

Logfile von Sendmail kontrollieren
(afxxx = Header Informationen, dfxxx= Body, lfxxx= Lock File)

```
$ /usr/spool/mqueue/syslog
```

Welche Konfigurationsfiles werden benutzt ?

```
$ grep "/[^0-9].*/" /etc/sendmail.cf
```

Stoppen des Sendmail Prozesses

```
$ kill `cat /etc/sendmail.pid`
```

Sendmail als Daemon starten

```
$ /usr/lib/sendmail -bd -q1h
```

Rebuild der Alias Listen in /etc/aliases

```
$ /usr/lib/sendmail -bi
```

Testen der Alias Listen in /etc/aliases

```
$ /usr/lib/sendmail -bv <alias>
```

Verbose Mode von Sendmail (Debbuging)

```
$ /usr/lib/sendmail -v <recipient> < /dev/null
$ mail -v <recipient> < /dev/null
```

Show Sendmail Queue

```
$ /usr/lib/sendmail -bp
```

Process Sendmail Queue

```
$ /usr/lib/sendmail -q
```

Freeze Konfigurationsfile /etc/sendmail.cf

```
$ /usr/lib/sendmail -bz
```

Test SMTP Connection

```
$ telnet mailhost zahn
  helo
  mail from: martin.zahn@akadia.com
  rcpt to: info@akadia.com
  data
  Test Mail
  .
  quit
```

1.8.7 Debug Sendmail

Sendmail im Debug Modus

```
$ /usr/lib/sendmail -d <kategorie.level> -bt
```

Show Delivery Agent (Mailer)

```
$ /usr/lib/sendmail -d0.12 -bt < /dev/null
```

Show Macros without $u, $M which will be set when mail is already delivered

```
$ /usr/lib/sendmail -d35.9 -bt
```

Show Rulesets

```
$ /usr/lib/sendmail -d21.12 -bt
```

Show internal Macros $w (Canonical Name of the Host)

```
$ /usr/lib/sendmail -d0.4 -bp

  Version 8.8.8+Sun
  Compiled with: LOG MATCHGECOS MIME7TO8 MIME8TO7 NAMED_BIND
  NDBM NETINET
  NETUNIX NIS NISPLUS QUEUE SCANF SMTP XDEBUG
  canonical name: quorum.glue.ch
  a.k.a.: quorum
  UUCP nodename: quorum
  a.k.a.: [193.72.194.29]
  UUCP nodename: quorum
  a.k.a.: [127.0.0.1]
  ============ SYSTEM IDENTITY (after readcf) ============
  (short domain name) $w = quorum
  (canonical domain name) $j = $w.$m
  (subdomain name) $m = akadia.ch
  (node name) $k = quorum
  ========================================================
  Mail queue is empty
```

MX Record testen

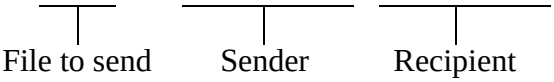
```
$ # /usr/lib/sendmail -bt
  WARNING: writable directory /var
  WARNING: writable directory /var/spool
  ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
  Enter <ruleset> <address>
  > /mx akadia.com
  getmxrr(akadia.com) returns 1 value(s):
  mail.glue.ch.
  > $=D
  akadia.com
```

1.9 Sendmail Interfaces

Sendmail kann über verschiedene Interfaces angesprochen werden:

1.9.1 Via Argument-Vektor

```
$ /usr/lib/sendmail -v < file -f mz@boa.ch zephir!bj
```




```
  zephir!bj... Connecting to zephir via uucp...
  zephir!bj... Sent
```

1.9.2 Via Pipe

Das File «mail» enthält folgenden Inhalt:

```
to: zephir!bj
cc: zephir!hueni
From: mz@mzsun
```

Dies ist der Message Body

```
$ cat mail | /usr/lib/sendmail -bm -t -v
  zephir!bj,zephir!hueni... Connecting to zephir via uucp...
  zephir!bj,zephir!hueni... Sent
```

1.9.3 Direkt via SMTP

```
$ telnet mzsun 25
  Trying 193.72.194.1 ...
  Connected to mzsun.boa.ch.
  Escape character is '^]'.
  220 boa.ch Sendmail 4.1/SMI-4.1 ready at Mon, 20 Sep 93 16:11:51 +0200
$ helo
  250 boa.ch Hello  (mzsun.boa.ch), pleased to meet you
$ mail from:<mz@mzsun>
  250 <mz@mzsun>... Sender ok
$ rcpt to:<bj@zephir.uucp>
  250 <bj@zephir.uucp>... Recipient ok
$ data
  354 Enter mail, end with . on a line by itself
  Nun kommt die eigentliche Message. Sie besteht aus
  diesen zwei Zeilen.
$ .
  250 Mail accepted
$ quit
  221 boa.ch delivering mail
  Connection closed by foreign host.
```

1.10 Versenden grosser (binärer) Dateien

Grosse binäre Files müssen vor dem Versenden in eine «Archiv-Datei» verpackt werden. Dazu eignet sich *shar(1)* sehr gut. Nach dem Verpacken kann die Archiv-Datei an einen User-Agent wie *elm(1)* übergeben werden:

Versenden[1]

```
$ shar -C bigfile | elm -s Subject user@host.domain
```

Option -C:

Compress and *uuencode* all files prior to packing. The recipient must have uudecode and uncompress in order to unpack (USE OF *UUENCODE* AND *COMPRESS* IS NOT APPRECIATED BY MANY ON THE NET).

«Auspacken» der Archiv-Datei

Message-Header entfernen

```
$ sh bigfile
```

1. shar ist nicht auf allen UNIX-Derivaten im Standardumfang enthalten. Das Tool ist jedoch auch als Public Domain Produkt erhältlich.

1.11 EMail-Delivery (Mailer)

UUCP-Mailer

Sowohl *uucp* wie auch *uux* erzeugen im Directory */var/spool/uucp/«remotename»* verschiedene Workfiles mit den notwendigen Angaben für uucico.

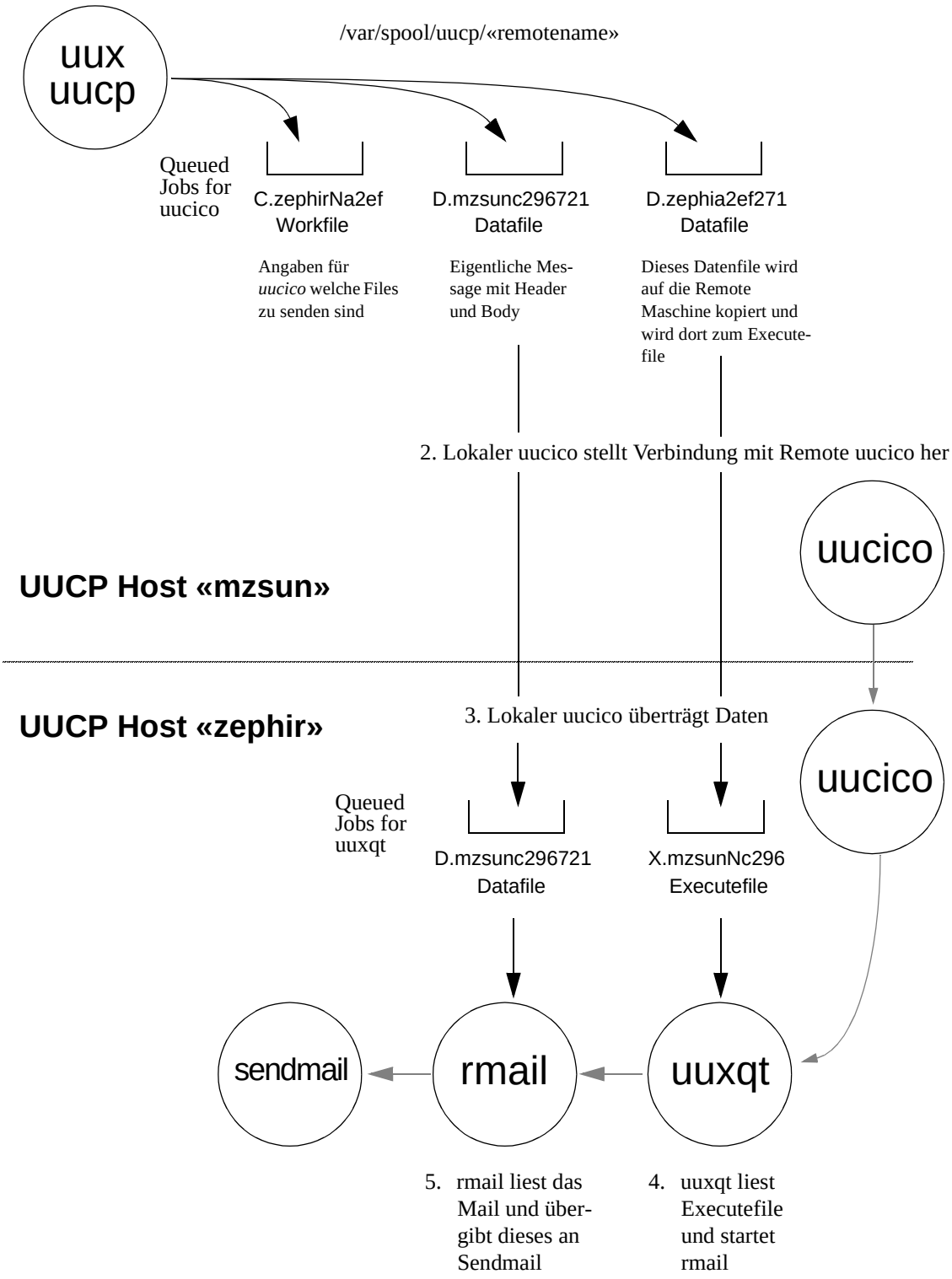


Abb. 4-16 Funktion des UUCP-Mailers

Ablauf des EMail Transfer (Beispiel)

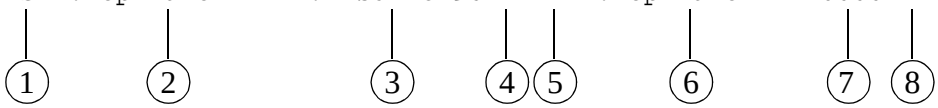
Wenn der UUCP Host *mzsun* eine EMail für den Benutzer *bj* auf dem UUCP Host *zephir* bereit hat, führt *mzsun* das Kommando *uux - zephir!rmail bj* aus (siehe *sendmail.cf*, UUCP Mailerdefinition). Gleichzeitig wird die Email via stdin an *uux* übergeben. Das Kommando *uux* auf *mzsun* liest die Email via stdin und speichert diese intern in der Workqueue für *zephir* ab (*/usr/spool/uucp/zephir*). Das Kommando *uux* startet nun sofort das Kommando *uucico*, sofern die Option -r nicht angegeben wurde. Mit *uux -r* wird die Email nicht sofort übermittelt sondern in der Queue zurückgehalten bis *uucico* explizit gestartet wird (zum Beispiel via cron oder Poll-File). Das Kommando *uucico* ist für den eigentlichen Transfer der EMail an den nächsten UUCP-Host *zephir* verantwortlich. Nachdem der automatische Loginablauf auf *zephir* erfolgreich beendet ist, wird auf *zephir* ein «Slave-uucico» gestartet, anstelle einer normalen Shell. *Uucico* auf *zephir* legt die transferierten EMails in der UUCP-Queue auf *zephir* ab. Sobald die Kommunikation zwischen den beiden uucico's beendet ist, startet *uucico* auf *zephir* das Kommando *uuxqt*. Das Kommando *uuxqt* konsultiert die Queue auf *zephir*, sieht dort die eingegangenen EMails mit dem Kommandofile-Inhalt *rmail bj*. *Uuxqt* startet dieses Kommando mit dem EMail-Inhalt als Standard-Input und legt die EMail in der lokalen EMailbox auf *zephir* ab. Ist der Benutzer nicht lokal vorhanden, so übernimmt Sendmail die Weiterleitung.

Übermittelte Files

Workfile (C.zephirNa2ef)

Das Workfile enthält Angaben für *uucico*, das die eigentliche Übertragung durchführt.

```
S D.mzsunc296271 D.mzsunc296271 mz - D.mzsunc296271 0666 mz
S D.zephia2ef271 X.mzsunNc296 mz - D.zephia2ef271 0666 mz
```



1. S: Send a file from the local system to a remote system
   R: Copy a file from the remote system to the local system
   X: Send an execution request to the remote system
2. Full pathname of the file to be sent or requested
3. Full pathname of the destination or user/file name
4. Login name of the user that requested the work
5. List of command options that the user specified for the uux command
6. Name of the associated data file in the spool directory
7. Mode bits og source file
8. login name of the user who should be notified upon completion of the job request. It is used only when the -n or -m option of *uucp* and *uux* is given.

Die zweite Zeile sagt aus, dass die Datei *D.zephia2ef271* auf die Remote Workstation kopiert werden soll und dort *X.mzsunNc296* heissen soll. Nach dem Transfer wird der Inhalt der Execute Datei *X.mzsunNc296* auf der Remote Maschine *zephir* ausgeführt.

Datafile (D.mzsunc296721)

Das Datafile enthält die Message mit dem Header und Body.

Datafile (D.zephia2ef271)

Dieses File ist auf der Source-Maschine «mzsun» noch ein Datafile. Auf der Ziel-Maschine «zephir» wird jedoch aus diesem File eine Executefile!

Datenfile wird zu Executefile:

Executefile (X.mzsunNc296)

Ein Executefile enthält das Kommando (*rmail*), welches auf der Remote Maschine ausgeführt werden soll. Unmittelbar nach dem Übertragen wird das Executefile von *uuxqt* ausgewertet.

`U mz mzsun`	Requestor's Login and System
`# return status on failure`	
`Z`	`Error status line`
`# use sh to execute`	
`e`	
`# return address for status or input return`	
`R mz`	Return Adress of requestor (uux -a)
`# job id for status reporting`	
`J zephirNa2ef`	
`F D.mzsunc296271`	File for transmission
`I D.mzsunc296271`	Standard input
`C rmail bj`	UNIX Command for **uuxqt**

Mindestens U und C müssen immer vorhanden sein. F, I Zeilen sind vorhanden wenn Files benötigt werden zur Command Execution.

1.12 PC-EMail

Durch die Verbreitung von PC's und deren Integration in ein lokales Netz (LAN) erwächst auch die Forderung jeden Benutzer via EMail zu erreichen, dies in der gewohnten PC-Umgebung unter DOS oder Microsoft-Windows. Um dies zu ermöglichen, wurden in den letzten Jahren verschiedene TCP/IP basierende PC-Mailtools am Markt oder als Public Domain Software angeboten. Da DOS standardmässig nicht netzwerkfähig ist, müssen PC's nachträglich mit Hard- und Software zur Netzwerkanbindung ausgerüstet werden. Hardwareseitig bedingt dies einen Einbau einer LAN-Karte, welche es ermöglicht die Protokolle Ethernet (IEEE802.3) oder Token-Ring (IEEE802.5) zu betreiben. Derartige LAN-Karten werden von vielen Herstellern wie 3COM, Novell, BICC, Ungermann-Bass, Western-Digital etc. angeboten. Damit war die Möglichkeit gegeben beispielsweise TCP/IP basierende Software vom PC aus zu benutzen. Eine gleichzeitige Benutzung eines andern Protokolls wie Novell's IPX oder UDP für NFS war nicht möglich. Dazu musste die Schicht 2 des OSI-Modells speziell für PC's standardisiert werden. Zur Zeit sind drei solche «Defacto-Standards» anzutreffen:

- Package-Driver Definition von FTP-Software (1986)
- NDIS (Network Driver Interface Specification) von 3COM (1988)
- ODI (Open Data Link Interface) von Novell

Gleichzeitig mit der Einführung dieser auf dem PC installierten Software wurde auf der Server Seite das PC-EMail Protokoll POP-3 (Post Office Protocol, RFC 1081) definiert. Es ist dem RFC821 Standard SMTP sehr ähnlich. Zusätzlich zum SMTP Protokoll erlaubt POP-3 die Authorisierung eines PC-Benutzers auf dem UNIX-Host. POP-3 wird als Daemon-Process auf dem UNIX-Server gestartet mit der Aufgabe eintreffende Email an den PC weiterzuleiten. Das Senden einer EMail vom PC an den UNIX-Host wird mit dem normalen SMTP-Protokoll vorgenommen.

1.12.1 Senden / Empfangen von EMail am PC (Beispiele)

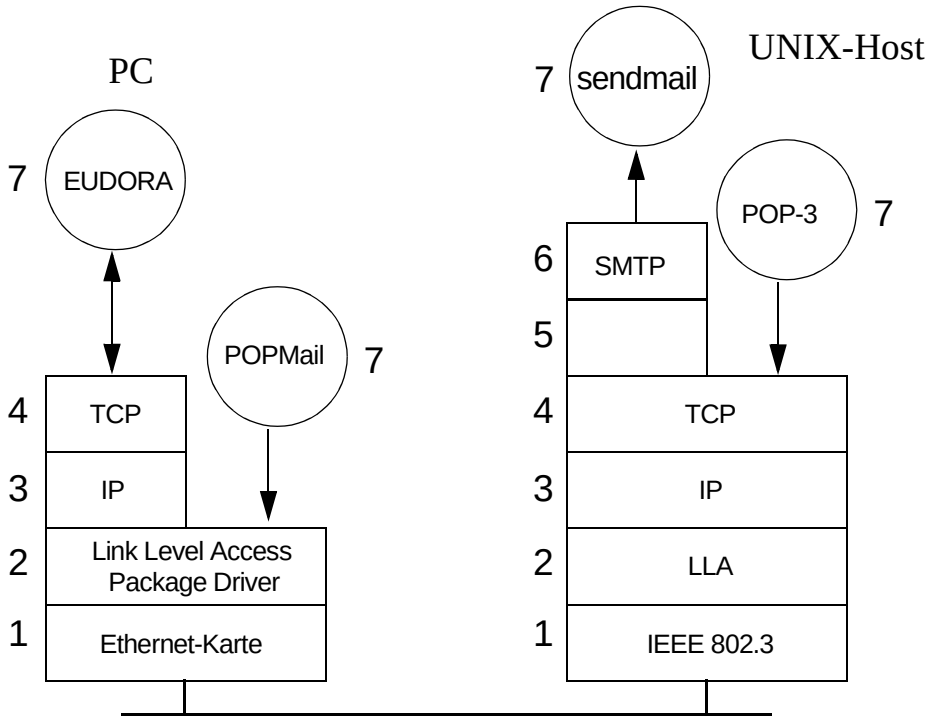


Abb. 4-17 Protokollstack einer PC-UNIX EMail Verbindung (Beispiel)

1.12.2 Funktion des Package Driver (Beispiel)

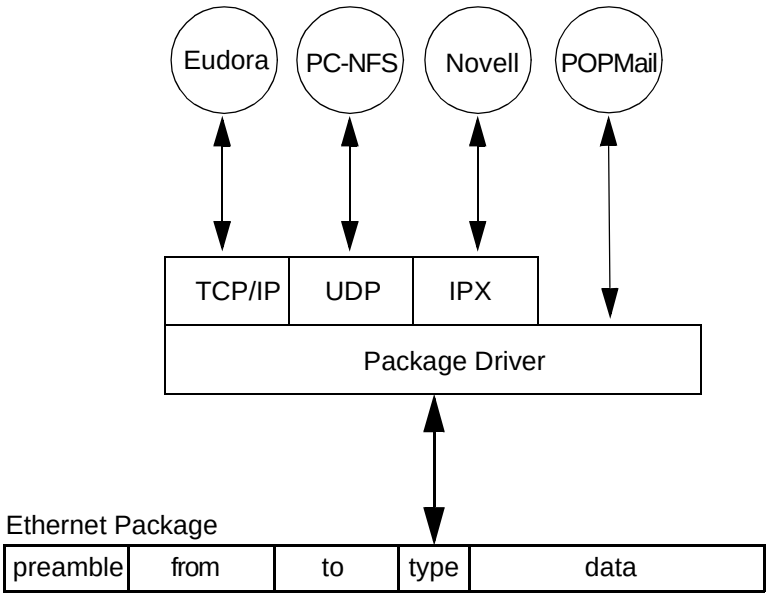


Abb. 4-18 Funktion eines Package-Driver (Beispiel)

Der Package Driver stellt die Schnittstelle zwischen der LAN-Karte und den «höher» gelegenen Protokollen dar. Er übernimmt das «Dispatching» der Ethernet-Pakete an das jeweilige Protokoll TCP/IP, UDB oder IPX. Umgekehrt leitet er Requests der Protokolle an die LAN-Karte weiter.

1.12.3 NDIS Driver

Die NDIS-Driver Spezifikation wurde von *3COM* definiert und erfüllt im Wesentlichen die gleiche Aufgabe wie ein Package Driver

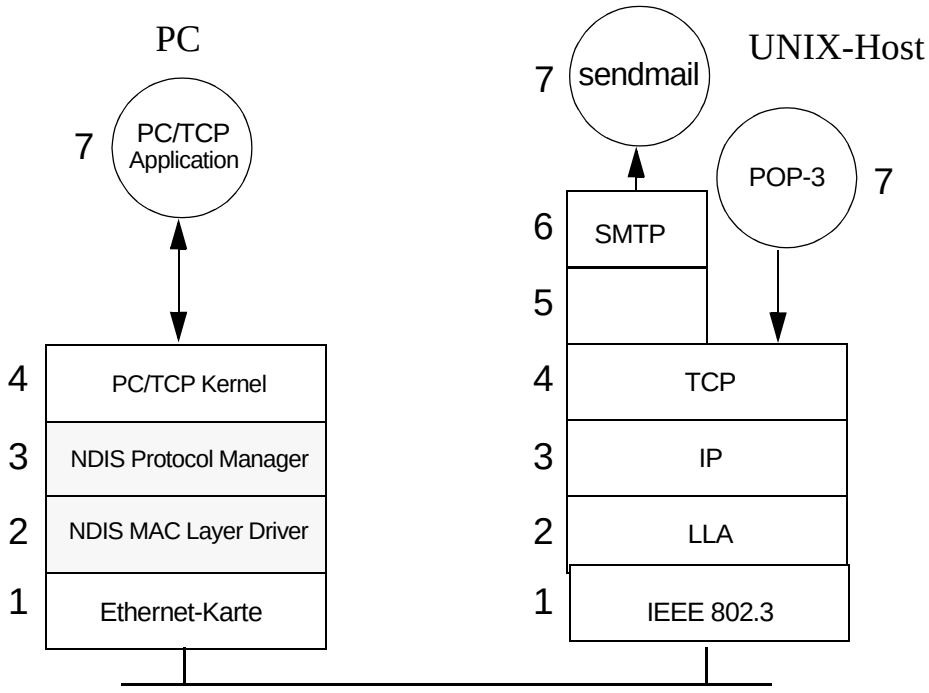


Abb. 4-19 Funktion des NDIS-Drivers (Beispiel)

1.12.4 ODI-Driver

ODI (Open Data Link Interface) ist die «Hausspezifikation» von *Novell*. Auch die ODI Schnittstelle besteht aus mindestens zwei Treibern, dem sogenannten MLI (Multiple Link Interface), das auf der Hardware aufsetzt und dem LSL (Link Support Layer), der zwischen dem MLI und dem Protokoll-Stack vermittelt.

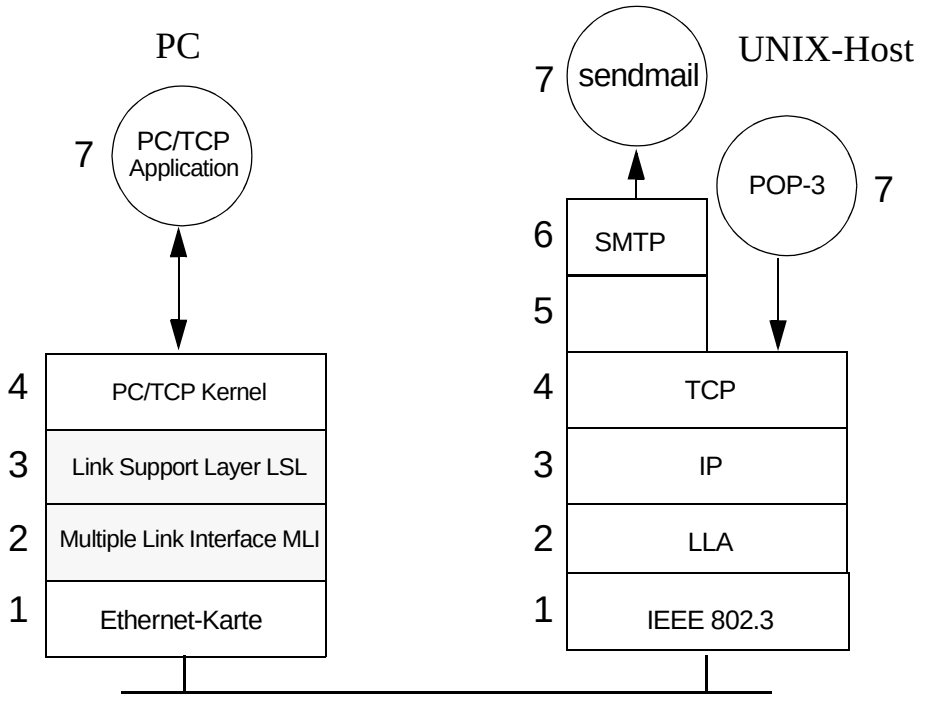


Abb. 4-20 Funktion des ODI-Drivers (Beispiel)

1.12.5 Installation / Konfiguration Package Driver

Am Beispiel einer Western-Digital Karte WE8003e[1] wird die Installation erläutert. Der Package Driver muss beim Booten des PC in das RAM geladen werden. Dies wird mit Vorteil im File *autoexec.bat* vorgenommen. Für jeden Package Driver müssen mindestens die folgenden Angaben definiert werden:

Interrupt Request Channel (IRQ)

Dies ist ein Signal von der PC-Peripherie (LAN-Karte) an die CPU des PC's um Daten mittels Interrupt Service Routine zu lesen / schreiben.

Schnittstellen Adressierung (I/O Adresse)

PC-Peripherie wird von der CPU wie eine normale RAM-Location angesprochen. Dazu wird von der PC-Peripherie (LAN-Karte) im RAM ein fest definierter Speicherbereich festgelegt, auf welche die CPU lesend / schreibend zugreifen kann.

Beide Werte sind aus der Dokumentation der LAN-Karte und des Package-Driver's ersichtlich. Für die Western-Digitalkarte und den WE8003E Package Driver gilt folgendes:

```
WE8003e -d -n -w 0x7c 7 0x280 0xd000
                                  RAM-Address
                            Base I/O Address
                       IRQ
                  Software-Interrupt
```

RAM-Buffer - Shared Memory Base Address

Der Bereich D000-DFFF ist für Page Swapping im Expanded Memory reserviert. Dieser Bereich wird ebenfalls durch ein 16-bit VGA Board benutzt.

1. WE8003e ist eine LAN-Interface-Karte der Firma Western-Digital und dient hier lediglich als Beispiel

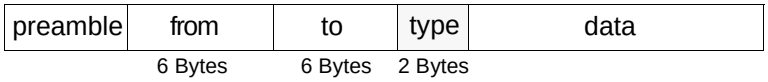
Software Interrupt

Der Software Interrupt wird benutzt um mit dem Package Driver zu kommunizieren. Der Software Interrupt muss im Bereich 0x60 ... 0x80 liegen. Gewisse Bereiche sind durch andere Packete fest reserviert. Beispielsweise benutzt die PC-TCP Software den Wert 0x61. Ein freier Bereich muss in den entsprechenden PC Handbüchern nachgeschlagen werden.

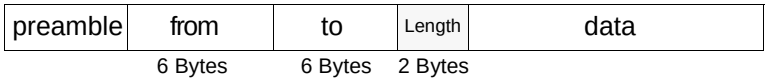
Option -n

Konvertierung von Novell-Ethernet Paketen in IEEE802.3 Pakete:

Novell Ethernet Package (IPX)



IEEE 802.3 Ethernet Package



Netware-Server und Boot PROM's verwenden ein modifiziertes Ethernet Paket (Ethernet II type 8137 encapsulation), welches anstelle des «Längenfeldes» ein «Typefeld» beinhaltet.

Option -w

In Verbindung mit Windows-3.1 verwenden. Ermöglicht Windows die Netzwerkapplikation aus dem RAM zu swappen.

Option -d

In Verbindung mit der Verwendung des Boot-PROM's. Damit wird die Initialisierung der Ethernet-Karte verzögert bis der Packet-Driver bereit ist (siehe PROMBOOT.NOT).

Eintrag in autoexec.bat

```
C:\EUDORA\WD8003e -d -n -w 0x7c 7 0x280 0xd000
C:\EUDORA\PKTADDR 0x7c 00:00:c0:3e:c0:1c
```